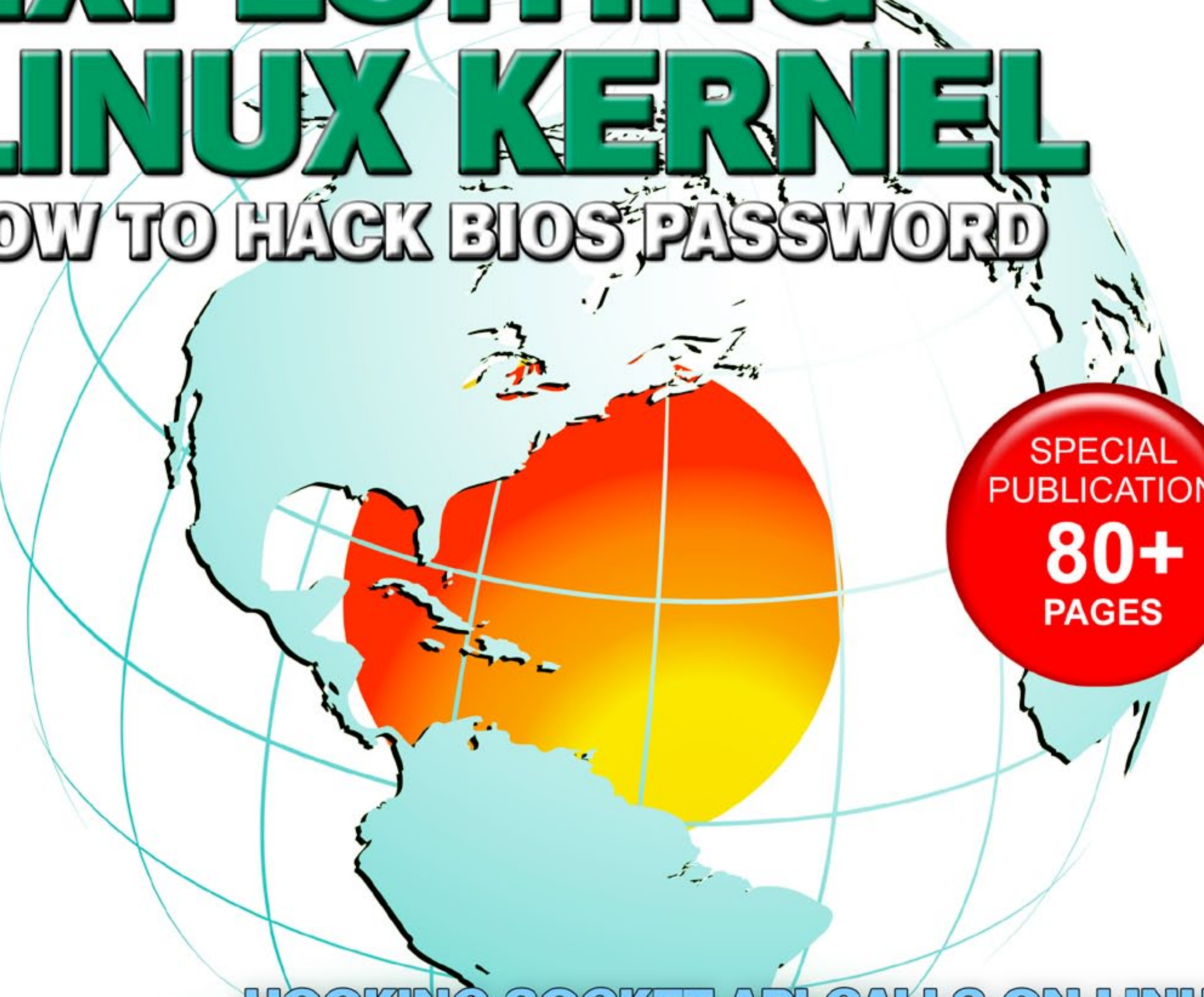
HAKING
ON DEMAND

Vol.1 No.8
Issue 08/2012(8) ISSN: 1733-7186

EXPLOITING LINUX KERNEL

HOW TO HACK BIOS PASSWORD



SPECIAL PUBLICATION
80+
PAGES

HOOKING SOCKET API CALLS ON LINUX

LINUX KERNEL EXPLOIT : IS THE STORM OVER?

SECURITY POLICY DEVELOPMENT IN TRUSTED BSD MAC FRAMEWORK

CRACKING BIOS PASSWORD IN KERNEL

PLUS 2 ARTICLES ON CYBERSECURITY

Source HDD
Target HDD
Atola
Busy
Power
Power On
Power Off
Power

CRACK HACK FORUM

CHF is regarded as one of the best online hacking communinity with over 76k+ members.

CHF was created by a renowned hacker and web specialist named ProVirus.

-CHF-

- CHF has over 2k+ tutorials teaching you the very art of hacking from the very basic to the most advanced level.
- Has a special forum for cracked premium accounts worth thousands of dollars.
- The VIP section is filled with the tools and tutorials unseen elsewhere making the section unique.

Join CHF NOW!!!

www.CrackHackForum.com

Greetings to: Srinuboy, Terrorbyte, Rain112, Hacker4life, Rynaldo, Mschoudhry, fakhru

Hakin9 ON DEMAND

08/2012 (08)

Hakin9 ON DEMAND team

Editor in Chief: Ewa Dudzic
ewa.dudzic@hakin9.org

Managing Editor: Ewa Duranc
ewa.duranc@hakin9.org

Paweł Plocki
pawel.plocki@hakin9.org

Jakub Walczak
jakub.walczak@hakin9.org

Editorial Advisory Board: Board: Arsen Darakdjian, Scott Paddock, Matthew Holley, Derek Thomas

Proofreaders: Ewa Duranc, Jakub Walczak, Arsen Darakdjian, Scott Paddock, Derek Thomas

Special Thanks to the Beta testers and Proofreaders who helped us with this issue. Without their assistance there would not be a Hakin9 magazine.

Senior Consultant/Publisher: Paweł Marciniak

CEO: Ewa Dudzic
ewa.dudzic@hakin9.org

Production Director: Andrzej Kuca
andrzej.kuca@hakin9.org

Art Director: Ireneusz Pogroszewski
ireneusz.pogroszewski@hakin9.org

DTP: Ireneusz Pogroszewski

Marketing Director: *Pawel Plocki*
pawel.plocki@software.com.pl

Publisher: Software Press Sp. z o.o. SK
02-682 Warszawa, ul. Bokserska 1
Phone: 1 917 338 3631
www.hakin9.org/en

Whilst every effort has been made to ensure the high quality of the magazine, the editors make no warranty, express or implied, concerning the results of content usage.
All trade marks presented in the magazine were used only for informative purposes.

To create graphs and diagrams we used smartdraw.com program by SmartDraw

Mathematical formulas created by Design Science MathType™

DISCLAIMER!
The techniques described in our articles may only be used in private, local networks. The editors hold no responsibility for misuse of the presented techniques or consequent data loss.

Dear Hakin9 Readers,

This month's issue is devoted to Linux Kernel Hacking. As always, we have prepared for you a set of interesting articles provided by experts in Kernel exploiting. Johnatan Levin will teach you about hooking socket API calls, Amr Thabet will walk you through the process of creating your own security tool, and Massimiliano Sembiante disputes on Android OS kernel exploit. In addition, just as the last time, we have managed to bring you something extra – William F. Slater III and CISSP Terrance J. Stachowski write about cyber-security (respectively on integration of cyberwarfare and cyberdeterrence strategies in the US, and conflict between the U.S. Government and Chinese telecommunications company Huawei). As a bonus, Bitdefender's Bogdan Botezatu presents his research on Windows 8's resistance to malware.

Hakin9's editorial team would like to give special thanks to the authors, betatesters, proofreaders and our editor in chief, Ewa Dudzic.

We hope that you will enjoy reading this issue!

Ewa Duranc, Paweł Płocki, Jakub Walczak & the Hakin9 Team.

Good Read for Thanksgiving!

With Hakin9, Thanksgiving lasts longer! Especially for our Readers BSD Magazine prepared a nice 2 for 1 offer. Now you can have The Best of BSD 2011 and Last Year of BSD Security for the price of one. Together, it gives 380 pages dedicated to BSD for $36,90!

Just buy one issue at http://stackmag.org/ and contact editors@bsdmag.org with "BSD Thanksgiving" in the subject of the message and your username. You will get the free access to the second issue!

The offer is valid till the end of November.

CONTENTS

DEFENCE PATTERN

STEP BY STEP WITH KERNEL TOOLKIT TUTORIALS

KERNEL USAGE IN MAC OS-X AND WINDOWS 8

SECURITY IN THE ENTERPRISE AREA

Kernel Security

In computing, the kernel is the main component of most computer operating systems; it is a bridge between applications and the actual data processing done at the hardware level. The kernel's responsibilities include managing the system's resources (the communication between hardware and software components). [1]

Usually as a basic component of an operating system, a kernel can provide the lowest-level abstraction layer for the resources (especially processors and I/O devices) that application software must control to perform its function. It typically makes these facilities available to application processes through inter-process communication mechanisms and system calls.

Operating system tasks are done differently by different kernels, depending on their design and implementation. While monolithic kernels execute all the operating system code in the same address space to increase the performance of the system, microkernels run most of the operating system services in user space as servers, aiming to improve maintainability and modularity of the operating system [1]. A range of possibilities exists between these two extremes (Figure 1).

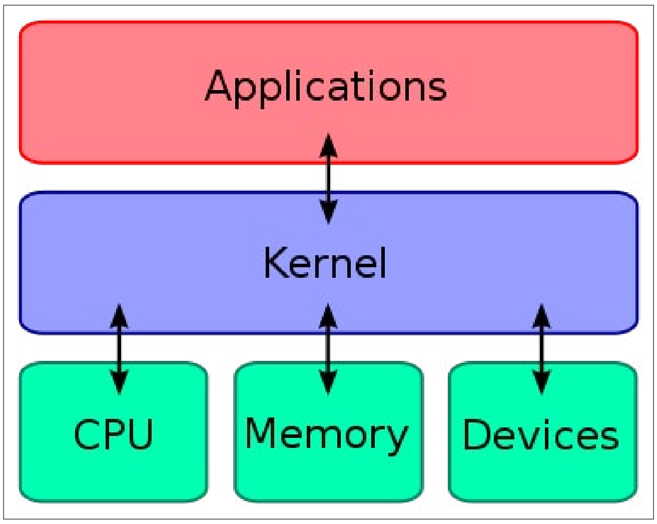


Figure 1. *Kernel's role in a computer*

Kernel Security

This paper introduces concepts of the security kernels as well as two examples of them: Kernelized Security Operating System and Honeywell Secure Communications Processor. The security kernel is a methodology that provides the functionality of the operating system and good internal security in multiuser systems. They are especially useful in organizations where different users are trusted on different levels. Also, security kernels are able to co-operate over networks which is very important nowadays.

Principles of Security Kernels

The security kernels can be divided into two categories: actual security kernels and *trusted computing bases* (TCB). The security kernel is defined as an isolated portion of a computer system that is designed to enforce the security policy of the system. A TCB is defined as the totality of hardware and software protection mechanisms responsible for enforcing the security policy of a given system. The difference is small and often security kernels and TCBs are viewed as synonymous. However, the little difference is that security kernels involve an isolated portion of a system architecture for security functions but in TCBs, security functions may be spread throughout various portions of a system. These are discussed in detail later on this chapter.

The security kernels cannot guarantee full protection. They are as efficient as the chosen policy which is discussed. Additionally, they cannot protect system from authorized, careless users. For example, users select quite often easily guessed passwords or write them down which makes the intruder's work relatively easy.

Reference Monitor

In the security kernel approach, a very important concept is the reference monitor which is an abstract notation adopted from the models of Butler Lampson. The reference monitor provides an underlying security theory for conceptualizing the idea of protection. In a reference monitor all active entities such as people or computer processes reference to passive entities such as documents or segments of memory using a set of correct access authorizations. Every reference to passive entities or change of authorizations must go through the reference monitor. The access control information is stored into a database and important security events are stored into the audit file (Figure 2).

Security Kernel

Figure 3 illustrates a general purpose operating system with on-line, interactive users. The kernel provides a relatively small and simple subset of operating system functions. The kernel primitives are the interfaces of this subset to the rest of the operating system (supervisor mode). The supervisor primitives provide the general-purpose operating system functions used by the applications.

Usually, an operating system consists of several functional areas such as process management, I/O control, and file system management. Some of the functions are security relevant and they must be placed into the kernel. The rules of policy model, discussed in Section 3, help to identify security relevant functions. Some of the parts of the operating system must be in the kernel because the model requires that these resources are virtual and that their location be hidden from untrusted software. The functions that provide useful common utilities do not manage anything shared among users and those that address denial of service are outside the scope of the security policy and can generally be in the supervisor.

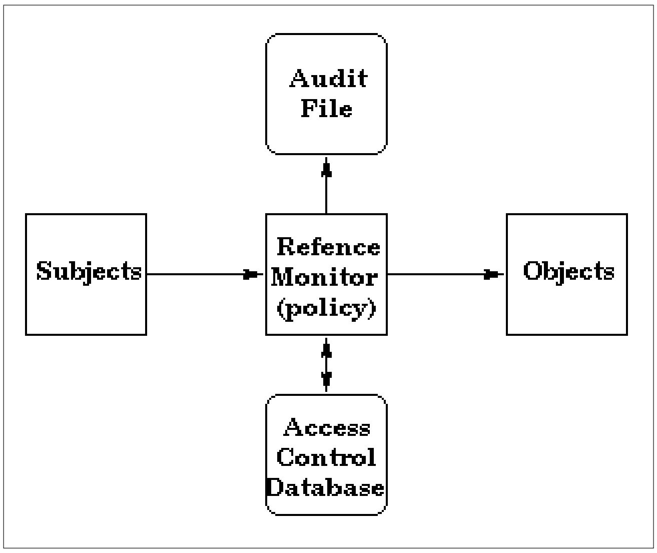


Figure 2. *Reference monitor*

Often systems require a security policy that is more specifically tailored to their needs than those defined by the basic security model. This tailored policy is generally exercised on a limited basis for infrequent operations and may apply only under special circumstances or to a special class of users. If this extended policy is implemented into the kernel, usually a set of interfaces that can be invoked by only certain trusted subjects is provided. Trusted subjects have some internal identifier, e.g., a privilege indicator. When a running program has such privileges, it may be able to perform actions not permitted by the access checks built into normal kernel functions.

Trusted subjects may be needed to perform system maintenance such as access policy controlling for untrusted subjects. Sometimes, normal users invoke certain trusted subjects to perform security sensitive functions. For example, since the basic security model does not allow an untrusted subject to lower the access class of information, the occasional need for downgrading a segment that a user accidentally over-classifies is satisfied by providing a trusted subject for the user. Trusted subjects are often implemented as asynchronous processes, called trusted processes, or as extensions of the kernel itself, called trusted functions. Regardless of the implementation technique, trusted subjects must adhere to the same engineering principles as the kernel if the security policy is to be correctly implemented. Other than the implementation tech-

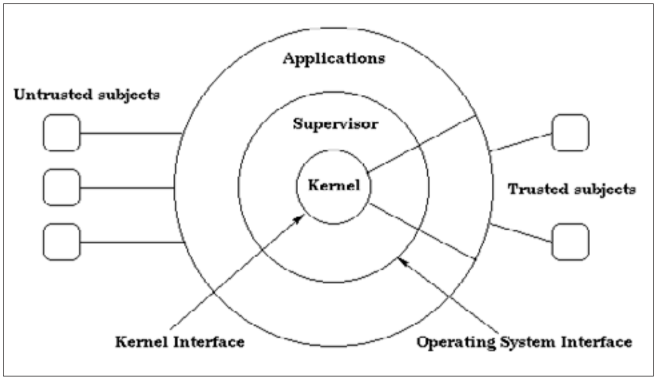


Figure 3. *Structure of kernel-based operating system*

nique, the only difference is the specific security policy enforced.

Defend against kernel malware

Kernel malware, commonly known as rootkits, are malicious applications that run in the kernel of the OS with absolute rights to system resources. End user devices infected with this type of application are open to undetectable processes that can steal data, collect PII, and otherwise control the system regardless of the presence of any anti-virus or personal firewall software.

How kernel malware works

According to Kimmo Kasslin at F-Secure, there are two types of kernel malware infections in Microsoft Windows environments: full-kernel and semi-kernel ("Kernel Malware: The Attack from Within", 2006). Before jumping into a description of each, it's important to review how Windows memory is managed from a system protection perspective (Figure 4).

Windows applications run in one of two modes: kernel mode or user mode. Kernel mode applications perform tasks such as accessing hardware resources on behalf of a user application. These applications typically have privileged access to system resources. Because of this, user applications are run in user mode to protect the integrity of the operating system. User mode applications, like word processors and Internet browsers, are unable to directly access hardware or protected OS services. Rather, they must make calls to kernel libraries or drivers that ensure resource requests are executed on behalf of the user applications. This separation of processing tasks is enforced at the hardware level. Kernel malware circumvents this abstraction of privileges by running in kernel mode with direct access to all system services. In other words, it has complete control of the infected system. One attack vector is the installation of a malicious driver.

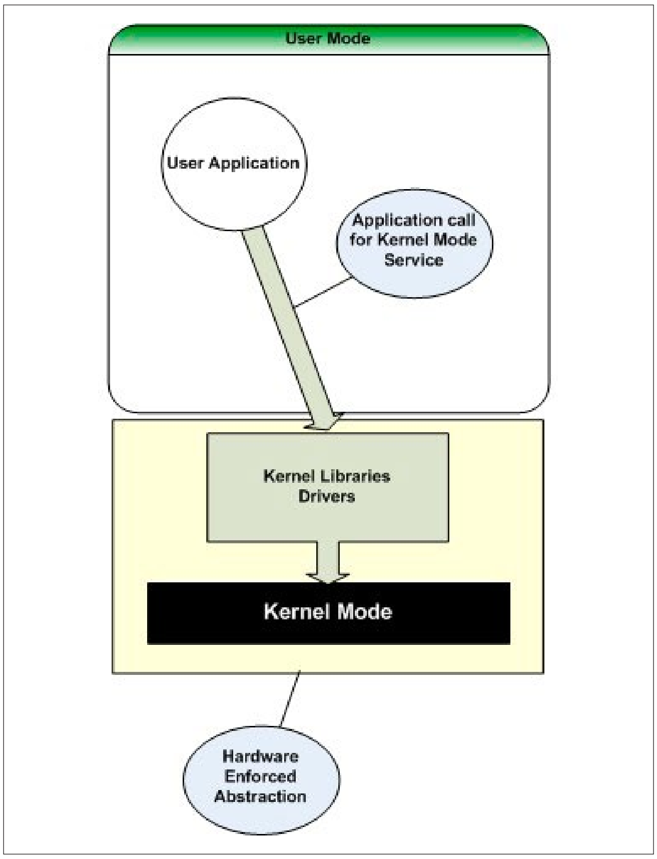

User Mode
User Application
Application call for Kernel Mode Service
Kernel Libraries Drivers
Kernel Mode
Hardware Enforced Abstraction

Figure 4. *Kernel malware work mode*

Malware running in full-kernel mode performs all tasks within the kernel layer. Although it might need a little help from the user to get installed, once operational it performs its assigned tasks without further user intervention.

Semi-kernel mode malware runs in both user mode and kernel mode. One method of deployment consists of placing a .dll or .exe in user mode with access to a kernel mode driver.

According to Kasslin, there is a rise in popularity of kernel malware that coincides with the move of cyber criminals to a hacking-for-profit model. The advantage to criminals is that kernel malware is usually undetectable when using standard antivirus and antispyware applications.

Mounting a defense

The first line of defense is denying the local administrator access to PC users. If an attacker can't take advantage of user privileges to install kernel-based software, the level of effort required to compromise the PC might be high enough to encourage him to find a softer target. In addition, management should ensure user awareness of the dangers of clicking on unknown links and consenting to the installation of unauthorized software.

Another important control is the implementation of a personal firewall on all workstations. This can help prevent self-propagating infections from spreading. It should be coupled with a strong patch management process. Patching helps eliminate software flaws that can be used to inject malicious kernel code.

Also, consider prohibiting the installation of any unsigned drivers. Installation of malicious drivers is a favorite method of placing kernel malware on target systems.

KERNEL SECURITY IN WINDOWS

Today you will come to know about how to secure the kernel by implementing some security level in windows for which you can use these given main security points below:

- Put Password on Power On Password in the system.
- Put Password on BIOS.
- Deep-Freezer

BIOS PASSWORD SETUP (For Kernel Security)

BIOS History

In IBM PC compatible computers, the *Basic Input/output System (BIOS)*, also known as the *system BIOS* or *ROM BIOS* is the *de facto* standard defining a firmware interface. The name originated from the Basic Input Output System used in the CP/M operating system (released in 1976), where the BIOS was loaded from disk, with only a small boot loader program stored in read-only memory.

The BIOS software is built into the PC, and is the first code run by a PC when powered on ('boot firmware'). When the PC starts up, the first job for the BIOS is the power-on self-test, which initializes and identifies system devices such as the CPU, RAM, video display card, keyboard and mouse, hard disk drive, optical disc drive and other hardware. The BIOS then locates boot loader software held on a peripheral device (designated as a 'boot device'), such as a hard disk or a CD/DVD, and loads and executes that software, giving it control of the PC.[2] This process is known as booting, or booting up, which is short for bootstrapping.

A BIOS has a *user interface* (UI), typically a menu system accessed by pressing a certain key on the keyboard when the PC starts. In the BIOS UI, a user can:

- configure hardware
- set the system clock
- enable or disable system components
- select which devices are eligible to be a potential boot device
- set various password prompts, such as a password for securing access to the BIOS user interface functions itself and preventing malicious users from booting the system from unauthorized peripheral devices.

The role of the BIOS has changed over time. As of 2011, the BIOS is being replaced by the more complex *Extensible Firmware Interface* (EFI) in many new machines, but BIOS remains in widespread use. EFI booting has been supported in only Microsoft Windows versions supporting GPT [2], the Linux kernel 2.6.1 and later, and Mac OS X on Intel-based Macs [2]. However, the distinction between BIOS and EFI is rarely made in terminology by the average computer user, making BIOS a catch-all term for both systems.

The first BIOS virus was CIH, whose name matches the initials of its creator, Chen IngHau. CIH was also called the "Chernobyl Virus," because its payload date was 1999-04-26, the 13th anniversary of the Chernobyl accident.

CIH appeared in mid-1998 and became active in April 1999. It was able to erase flash ROM BIOS content. Often, infected computers could no longer boot, and people had to remove the flash ROM IC from the motherboard and reprogram it. CIH targeted the then-widespread Intel i430TX motherboard chipset. The then-widespread Windows 9x operating systems allowed direct hardware access to all programs.

Modern systems are not vulnerable to CIH because of a variety of chipsets being used which are incompatible with the Intel i430TX chipset, and also other flash ROM IC types. There is also extra protection from accidental BIOS rewrites in the form of boot blocks which are protected from accidental overwrite or dual and quad BIOS equipped systems which may, in the event of a crash, use a backup BIOS. Also, all modern operating systems such as Linux, OS X, Windows NT-based Windows OS like Windows 2000, Windows XP and newer, do not allow user-mode programs to have direct hardware access. As a result, as of 2008, CIH has become essentially harmless, at worst causing annoyance by infecting executable files and from antivirus software. Other BIOS viruses remain possible, however; [2] since most Windows home users without Windows Vista/7's UAC run all applications with administrative privileges, a modern CIH-like virus could in principle still gain access to hardware without first using an exploit. The operating system OpenBSD prevents all users from having this access and the grsecurity patch for the linux *kernel* also prevents this direct hardware access by default, the difference being an attacker requiring a much more difficult *kernel* level exploit or reboot of the machine.

What is BIOS?

A BIOS password will make sure you need to enter a password when you make changes to the BIOS settings.

The most basic BIOS password will prevent people from making changes to your BIOS settings. In the BIOS you can define the boot order of hardware. In practice this means you tell the computer that you want him to look for bootable data in a certain order (e.g. Floppy -> DVD -> HDD). After your computer is first installed you probably want to only allow it to boot from you hard disk and disallow to boot from USB or DVD. Removing these from the boot order will also speed up the boot process of your computer since he will not be checking these devices for bootable media.

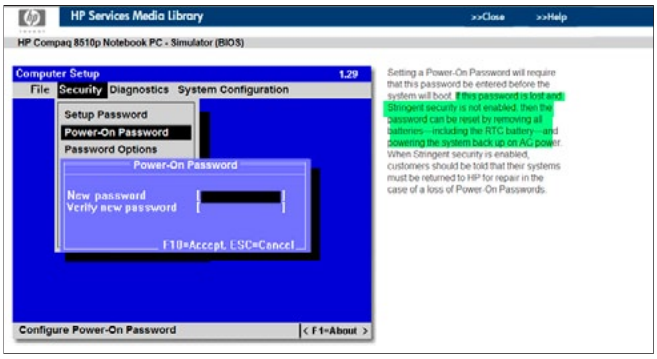
Normally you will not have a need to boot regularly from DVD or USB since you could just as well install these operating systems on virtual systems. Once your PC has been properly set up the only reason to boot from other media would be in case of restoring a failing computer (e.g. Windows Rescue Disks) or if your computer has a failing hard disk. When this is the case you just go to the BIOS settings, change the boot order to include DVD or USB, enter the password and reboot.

Preventing changes to the boot order and removing DVD and USB (and floppy or anything else than hard disk) from the boot order will make sure your computer boots the OS as you have it installed and not something else. An attacker would need physical access in order to put some kind of media in your computer (e.g. A DVD disk).

There are some BIOS manufacturers that also allow putting a password in the BIOS that is needed or simply booting the computer, you need to enter this password every time you boot the computer no matter what media you boot it from. You could compare this to the login screen you might have to log into the OS after booting.

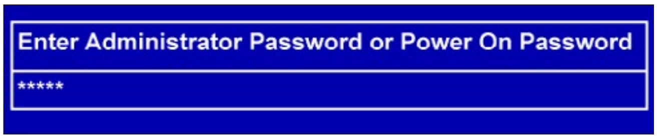
When do we need this?

I would advice that a BIOS password be set for all laptops because it enhances the security level on Kernel Layer so that any individual can bypass it easily. Laptops are designed to be carried and are often left alone (e.g. in your hotel room) in places where you have little control over the people that have access. I would also advice to do it on all computers that are in public places or places where there is little or no control on who has access or where lots of people have access (e.g. workplaces).



Setting a Power-On Password will require that this password be entered before the system will boot. If this password is lost and Stringent security is not enabled, then the password can be reset by removing all batteries—including the RTC battery—and powering the system back up on AC power. When Stringent security is enabled, customers should be told that their systems must be returned to HP for repair in the case of a loss of Power-On Passwords.

Figure 5. *Configure Power-On Password*



Enter Administrator Password or Power On Password

*****

Figure 6. *Power-On Password Login window*

Setting up Power-On Password

Before the Windows 7 (or any other OS) operating system loads, the computer goes through a brief procedure known as the Power-On Self-Test. This function makes an inspection of any changes made to the hardware installed on your computer. The Basic Input Output System stores the settings pertaining to such hardware. In addition, you may set a user password that takes place before said inspection, which also prevents the operating system from loading.

Step 1

Turn on your Windows 7 computer. Access the BIOS screen by pressing the appropriate keyboard key, which generally varies by motherboard make and model.

Step 2

Go to the BIOS' "Security" or "Privacy" section. While the layout presented greatly depends by its Motherboard manufacturer, all information and settings are generally similar (Figure 5).

Step 3

Enter the desired Power-On password, and re-type it into the confirmation field if necessary.

Step 4

Exit the BIOS menu through its respective "Save and Exit" function, typically executed by pressing "F10" on your keyboard (Figure 6).

Setting up BIOS Password

Your computers BIOS is the first program that is run when your computer starts. You can tell the BIOS to ask for a password when it starts, thus restricting access to your computer.

To enter the BIOS setup program, sometimes called CMOS setup:

Turn on or reboot your computer. Than press F8 and one screen will display a series of diagnostics and a memory check.

A message like "Hit the <DEL> key to enter the BIOS setup program" will appear.

When you do hit DEL at the right time [1] you'll see a menu screen something like this: Figure 7.

Note
Some BIOS versions use a graphical type menu with icons (a GUI) or have a text interface that appears different to the one shown, the principle however is exactly the same.

As you can see there are two options that relate to passwords, Supervisor Password and User Password, these relate to controlling access to the BIOS Setup Program and the Machine Boot respectively.

Note that not all BIOS's have this password feature; your bios may not have it in which case you won't be able to restrict access to your computer in this way.

Select SUPERVISOR PASSWORD and you'll be prompted to enter a password:

You should now enter a password of up to eight characters (most BIOS's are limited to eight characters unfortunately). I recommend you use the full eight but take care that you choose something you'll not forget.

The BIOS will then prompt you to confirm the password, just type the same thing again (Figure 8).

Now you'll want to set your system to ask for that password every time it boots, so select the BIOS FEATURES SETUP option, to see a menu something like this:
Fairly obviously, it's the Password Check option we're interested in, so select it and change the setting to ALWAYS.

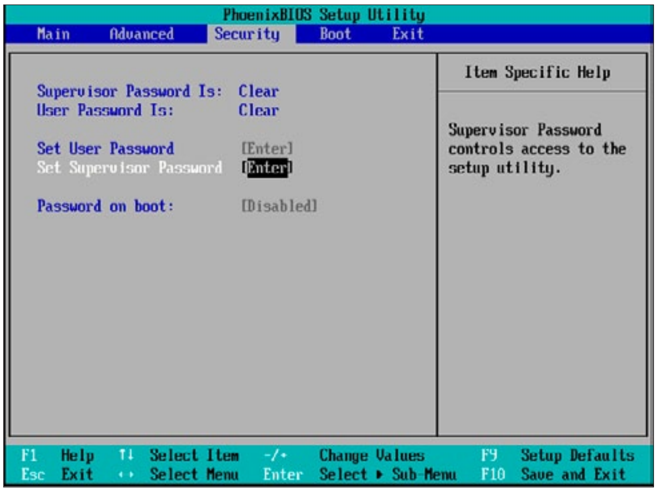
Now navigate back to the main menu and select SAVE & EXIT SETUP. Your machine will then reboot and you'll be prompted for the password (Figure 9).

Each and every time you boot you'll be asked for password you chose (Figure 10).

Please note that this method of restricting access to your computer is not completely full proof, there are ways around it. But it will stop or at least delay the majority of casual attempts to get access.

If you forget your BIOS password, consult your motherboard manual or if you don't have one, consult the website of the BIOS manufacturer.

It's not always the DEL key some BIOSs use F2 or F10 or another key combination, check your motherboard manual.



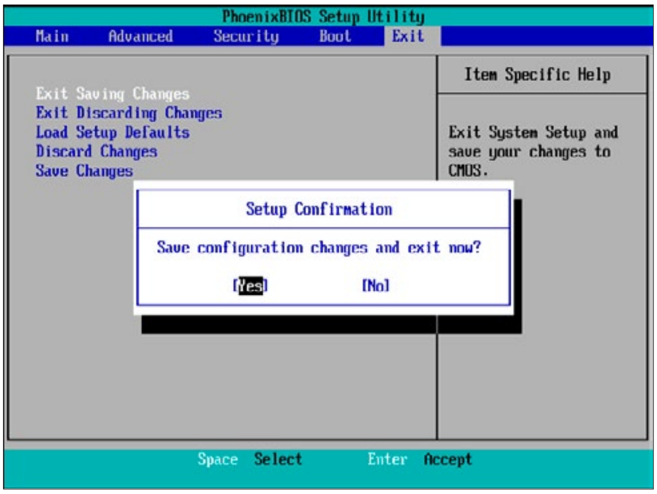
```
                         PhoenixBIOS Setup Utility
  Main     Advanced     Security     Boot     Exit

  Supervisor Password Is:   Clear               Item Specific Help
  User Password Is:         Clear
                                                Supervisor Password
  Set User Password         [Enter]             controls access to the
  Set Supervisor Password   [Enter]             setup utility.

  Password on boot:         [Disabled]

  F1   Help   ↑↓  Select Item   -/+    Change Values       F9   Setup Defaults
  Esc  Exit   ↔   Select Menu   Enter  Select ▸ Sub-Menu   F10  Save and Exit
```

Figure 7. *BIOS Setup*



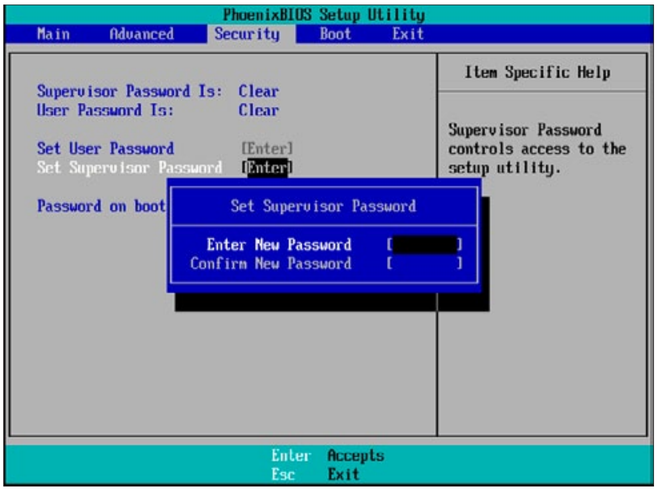
```
                         PhoenixBIOS Setup Utility
  Main     Advanced     Security     Boot     Exit

  Exit Saving Changes                           Item Specific Help
  Exit Discarding Changes
  Load Setup Defaults                           Exit System Setup and
  Discard Changes                               save your changes to
  Save Changes                                  CMOS.

              Setup Confirmation
      Save configuration changes and exit now?
              [Yes]            [No]

                  Space  Select          Enter  Accept
```

Figure 9. *Saving changes and exit*



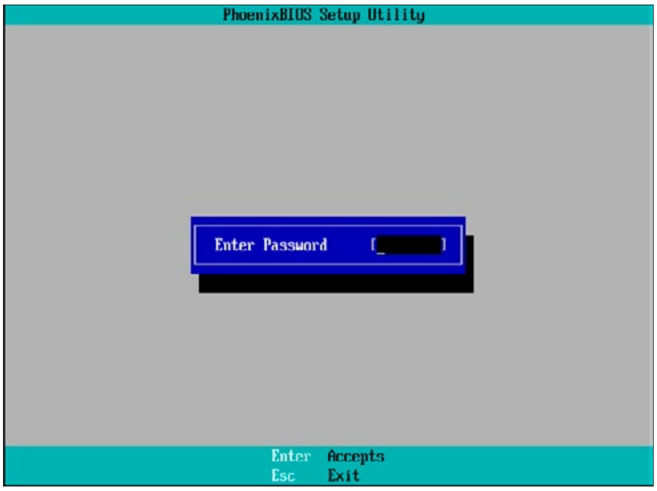
```
                         PhoenixBIOS Setup Utility
  Main     Advanced     Security     Boot     Exit

  Supervisor Password Is:   Clear               Item Specific Help
  User Password Is:         Clear
                                                Supervisor Password
  Set User Password         [Enter]             controls access to the
  Set Supervisor Password   [Enter]             setup utility.

  Password on boot
              Set Supervisor Password
      Enter New Password      [        ]
      Confirm New Password    [        ]

                          Enter  Accepts
                          Esc    Exit
```

Figure 8. *Set password on Supervisor Password*



```
                         PhoenixBIOS Setup Utility

                    Enter Password     [_       ]

                          Enter  Accepts
                          Esc    Exit
```

Figure 10. *BIOS Login Screen*

Ways to defeat BIOS passwords?

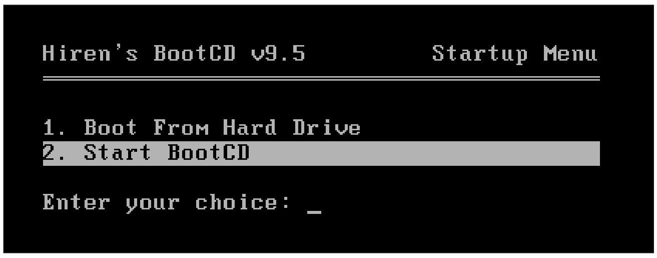
There are many ways to defeat a BIOS password:

- remove the CMOS battery to clear the password
- reset the jumpers for the BIOS to clear the password
- try one of the master BIOS passwords to bypass the user placed BIIOS password
- use a BIOS password cracking utility

As you notice all these actions will require physical access to the computer and in case of removing the battery or using the jumpers on the motherboard to clear the password the attacker would even need to open your computer. You cannot update a BIOS remotely on normal computer hardware, you can on some servers but to my knowledge that requires extra hardware to be installed on the said server. Most of these actions also take time (the exception being the master or generic passwords that manufacturers put in as back doors)

Conclusion

A BIOS password is a good investment since it takes very little time or knowledge to set up and it might stop an attacker since he will need a certain amount of time to get past the BIOS password (opening the computer or looking at the manufacturer and then trying possible master passwords takes time). Also if the BIOS password is cleared it will be visible to you that an attack on your system has happened and you can take appropriate action.

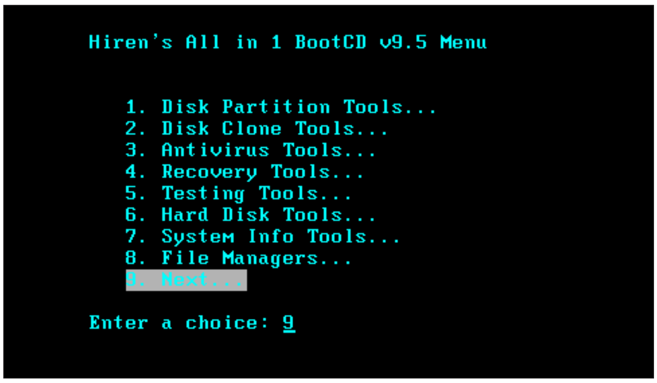


```
Hiren's BootCD v9.5           Startup Menu

1. Boot From Hard Drive
2. Start BootCD

Enter your choice: _
```

Figure 11. *Start Hiren Live CD tool*



```
Hiren's All in 1 BootCD v9.5 Menu

   1. Disk Partition Tools...
   2. Disk Clone Tools...
   3. Antivirus Tools...
   4. Recovery Tools...
   5. Testing Tools...
   6. Hard Disk Tools...
   7. System Info Tools...
   8. File Managers...
   9. Next...

Enter a choice: 9
```

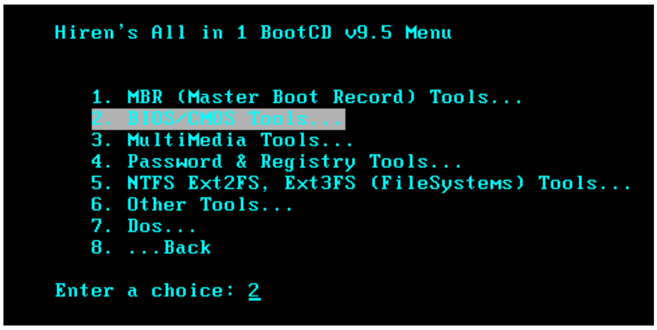
Figure 12. *Select option 9 for next*

If your BIOS allows you to set a password that is required for booting the computer I would certainly use that option, it will take an extra step to log in (you need to enter the password) but it does add an extra hurdle and more time for an attacker to gain access to your computer.

I firmly believe security should be layered and there should always be more than one level of protection on each functionality of your computer. Therefore, a BIOS password will always be a good investment. The fastest possible way I see an attacker getting past this security measure would include him knowing you use a BIOS password, a first investigation on what the manufacturer of your BIOS is, a search for the manufacturer master password(s) (it is possible that there are none for your BIOS) and then he needs physical access to your computer to (re-)boot, change the BIOS settings (using the password), reboot from other media, do his evil stuff, reboot and change the BIOS settings back to the original settings, reboot and put the computer back in the state it was when he found it (probably powered down). I think this would take 10 minutes at the least.

Hiren Live CD Tool – A way to hack BIOS Password

Hiren is a Live CD Tool by which you can crack BIOS Password. I am telling you some steps by

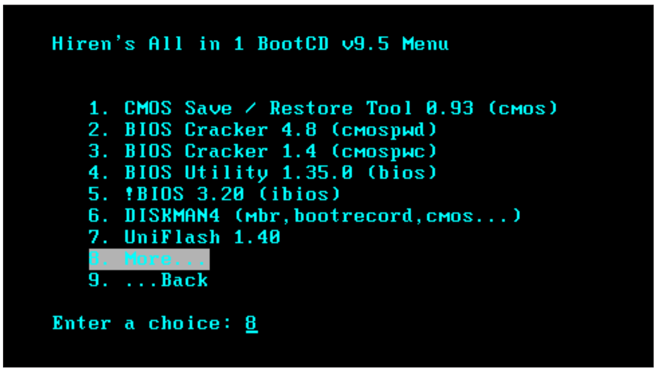


```
Hiren's All in 1 BootCD v9.5 Menu

   1. MBR (Master Boot Record) Tools...
   2. BIOS/CMOS Tools...
   3. MultiMedia Tools...
   4. Password & Registry Tools...
   5. NTFS Ext2FS, Ext3FS (FileSystems) Tools...
   6. Other Tools...
   7. Dos...
   8. ...Back

Enter a choice: 2
```

Figure 13. *Select option 2 for BIOS/CMOS Tools*



```
Hiren's All in 1 BootCD v9.5 Menu

   1. CMOS Save / Restore Tool 0.93 (cmos)
   2. BIOS Cracker 4.8 (cmospwd)
   3. BIOS Cracker 1.4 (cmospwc)
   4. BIOS Utility 1.35.0 (bios)
   5. !BIOS 3.20 (ibios)
   6. DISKMAN4 (mbr,bootrecord,cmos...)
   7. UniFlash 1.40
   8. More...
   9. ...Back

Enter a choice: 8
```

Figure 14. *Select option 8 for More*

which you can crack it by using it. For cracking your system's BIOS password you are supposed to follow these given steps which you will find in pictures step by step.

Step 1

Put Hiren Live CD tool into CD-Rom and reboot your system.

Step 2

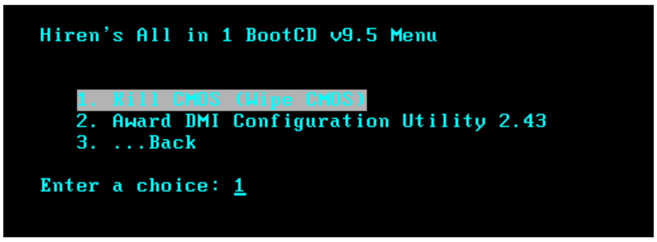
You have to choose 2nd option *Start BootCD* (Figurte 11).

Step 3

Choose 9th option for *next* (Figure 12).

Step 4

Choose 2nd option for *BIOS/CMOS Tools….* (Figure 13).

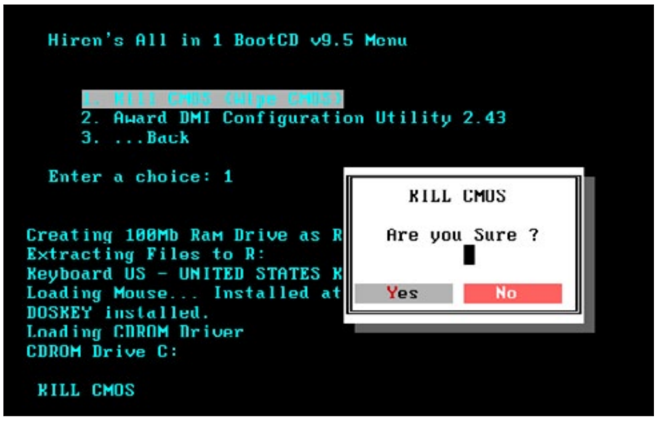


```
Hiren's All in 1 BootCD v9.5 Menu


     1. Kill CMOS (Wipe CMOS)
     2. Award DMI Configuration Utility 2.43
     3. ...Back

Enter a choice: 1
```

Figure 15. *Select option 1 for Killing CMOS Information*



```
Hiren's All in 1 BootCD v9.5 Menu


     1. Kill CMOS (Wipe CMOS)
     2. Award DMI Configuration Utility 2.43
     3. ...Back

Enter a choice: 1

Creating 100Mb Ram Drive as R
Extracting Files to R:
Keyboard US - UNITED STATES K
Loading Mouse... Installed at
DOSKEY installed.
Loading CDROM Driver
CDROM Drive C:

KILL CMOS
```

KILL CMOS

Are you Sure ?

Figure 16. *Final process to kill CMOS information*

Figure 17. *Deep-Freezer Introduction*

Step 5

Choose 8th option for *More…..* (Figure 14).

Step 6

Choose 1st option for *Kill CMOS (Wipe CMOS)* (Figure 15).

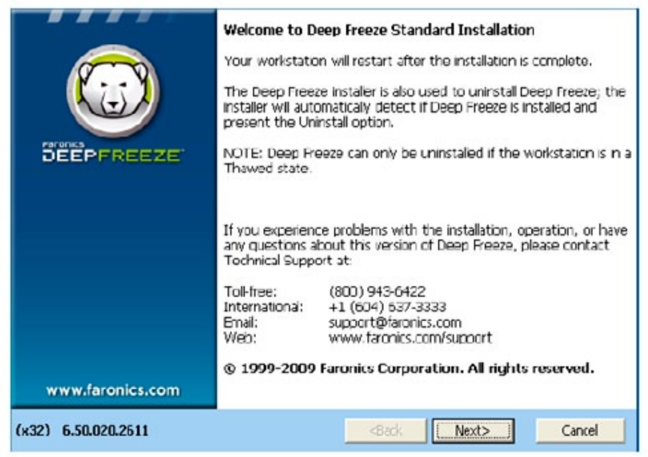
Step 7

Select *yes* for cracking your BIOS password (Figure 16).

You will be successful in cracking the BIOS Password by following these steps.

Deep-Freezer Tool

Deep Freeze works on *Kernel Layer* and helps eliminate computer damage and downtime by making computer configurations indestructible. Once Deep Freeze is installed on a computer, any changes made to the computer – regardless of whether they are accidental or malicious – are never permanent. DeepFreeze provides immediate immunity from many of the problems that plague computers today – inevitable config-



Welcome to Deep Freeze Standard Installation

Your workstaton will restart after the installation is complete.

The Deep Freeze Installer is also used to uninstall Deep Freeze; the installer will automaticaly detect if Deep Freeze is installed and present the Uninstall option.

NOTE: Deep Freeze can only be uninstalled if the workstation is in a Thawed state.

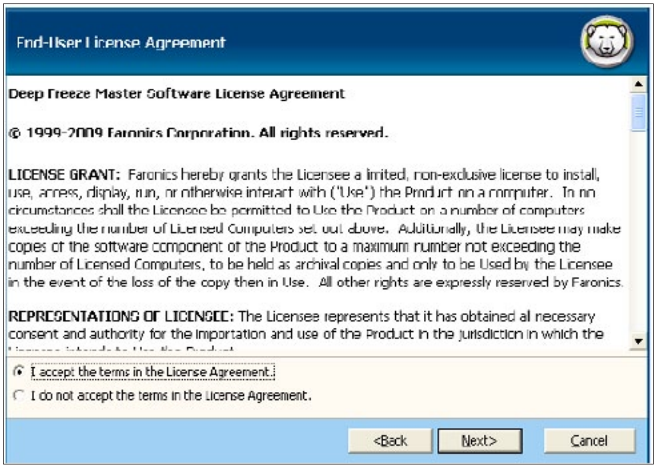
If you experience problems with the installation, operation, or have any questions about this version of Deep Freeze, please contact Technical Support at:

Toll-free:	(800) 943-6422
International:	+1 (604) 637-3333
Email:	support@faronics.com
Web:	www.faronics.com/support

© 1999-2009 Faronics Corporation. All rights reserved.

(x32) 6.50.020.2611

Figure 18. *Deep-Freeze installation step 1*



End-User License Agreement

Deep Freeze Master Software License Agreement

© 1999-2009 Faronics Corporation. All rights reserved.

LICENSE GRANT: Faronics hereby grants the Licensee a limited, non-exclusive license to install, use, access, display, run, or otherwise interact with ('Use') the Product on a computer. In no circumstances shall the Licensee be permitted to Use the Product on a number of computers exceeding the number of Licensed Computers set out above. Additionally, the Licensee may make copies of the software component of the Product to a maximum number not exceeding the number of Licensed Computers, to be held as archival copies and only to be Used by the Licensee in the event of the loss of the copy then in Use. All other rights are expressly reserved by Faronics.

REPRESENTATIONS OF LICENSEE: The Licensee represents that it has obtained all necessary consent and authority for the importation and use of the Product in the jurisdiction in which the

● I accept the terms in the License Agreement.
○ I do not accept the terms in the License Agreement.

Figure 19. *Accept License Agreement for installing purpose*

uration drift, accidental system misconfiguration, malicious software activity, and incidental system degradation.

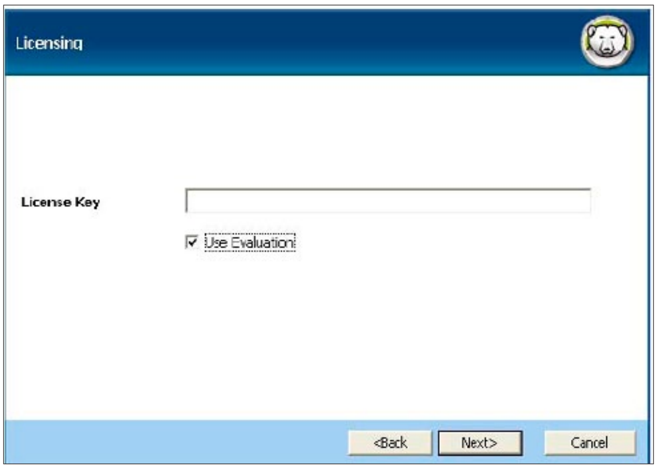
System Requirements

Deep Freeze protects the computers that are set to boot from the hard drive. Configure the CMOS to boot from the hard drive only. The CMOS must be password protected to prevent unauthorized changes. Deep Freeze protects the *Master Boot Record* (MBR) when the computer is frozen.

Attended Install

Complete the following steps to perform an attended install.

- Double-click *DFStd.exe* to begin the installation process (Figure 18).
- Click *Next*. Click *I agree to the terms in the License Agreement*. Click *Next* (Figure 19).
- Enter the License Key or select the *Use Evaluation* check box to install Deep Freeze in Evaluation mode (Figure 20).

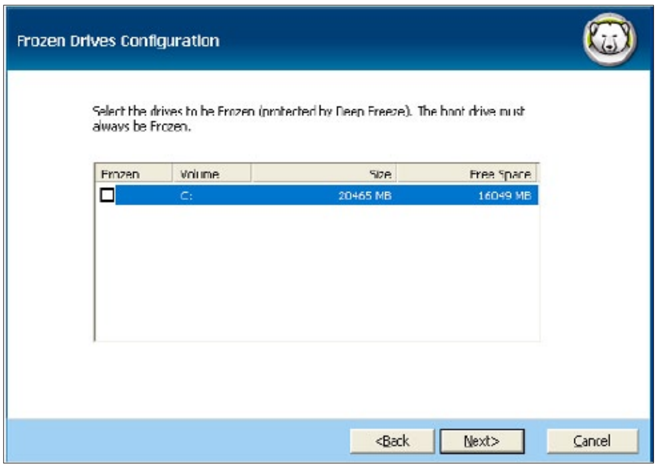


Licensing

License Key:

☑ Use Evaluation

Figure 20. *Put License Key or use Evaluation for Demo mode*



Frozen Drives Configuration

Select the drives to be Frozen (protected by Deep Freeze). The boot drive must always be Frozen.

Frozen	Volume	Size	Free Space
□	C:	20465 MB	16049 MB

Figure 21. *Select Drive for implementing Deep-Freezer effect*

- Choose the drives to Freeze from the displayed list. Click *Next* (Figure 21).
- Click *Install* to begin the installation.

The computer restarts immediately after the installation is complete.

Install Using Imaging

Deep Freeze has been designed to work with all major imaging and desktop management software. Use either an Attended Install or the Silent Install to install Deep Freeze on a master image.

Deep Freeze must be prepared for deployment before finalizing a master image. To prepare the master image for deployment complete the following steps:

- Restart the computer into a *Thawed* state.
- Launch Deep Freeze using the keyboard shortcut *CTRL+SHIFT+ALT+F6*. Alternatively, press
- *SHIFT* and double-click the Deep Freeze icon in the System Tray.
- Enter the password and click *OK*.
- Click *Set Flag* in the *Status* tab.
- The message *The flag has been set successfully. Do you want to reboot your computer now?*

Is displayed. Click Yes to reboot the computer immediately. Click No to reboot the computer later.

After imaging, the computers require an additional restart for Deep Freeze to correctly detect the changes in disk configuration. If the computers are imaged in an unattended mode, steps should be taken to ensure the computers are restarted to allow the configuration to update.

KERNEL SECURITY IN LINUX

Kernel Security through password protect GRUB ENTERIES

Boot loader is a software code that runs before the Operating System and helps in loading the Operating System. Boot loaders usually contain several ways to boot the *Operating System kernel* and also contain commands for trouble-shooting or passing some values to the *kernel* while booting.

When a computer with Red Hat Enterprise Linux is powered on, the Red Hat Enterprise Linux Operating System is loaded into memory and started by a boot loader. A boot loader program is located on the system's primary hard drive and the boot loader has the responsibility of loading the *Linux kernel* with its required necessary files into the computer's memory.

Red Hat enterprise editions for different hardware architecture use different boot loaders. The following table shows the different boot loaders for different hardware platforms (Table 1).

The GRUB (GNU GRand Unified Boot loader) is the default boot loader for AMD32, AMD64, Intel x86 and Intel EMT64T based hardware platforms. GRUB (GNU GRand Unified Boot loader) enables the selection of the installed operating system at boot time. GRUB also allows the user to pass arguments to the *kernel* while booting.

Linux Booting Levels

1. The Stage 1 or primary boot loader is read into memory by the BIOS from the Master Boot Record (MBR). The primary boot loader exists on less than 512 bytes of disk space within the MBR and is capable of loading either the Stage 1.5 or Stage 2 boot loader.
2. The Stage 1.5 boot loader is read into memory by the Stage 1 boot loader.
3. The Stage 2 or secondary boot loader is read into memory. The secondary boot loader displays the GRUB menu and command environment. This interface allows the user to select which kernel or operating system to boot, pass arguments to the kernel, or look at system parameters.
4. The secondary boot loader reads the operating system or kernel as well as the contents of `/boot/sysroot/` into memory. Once GRUB determines which operating system or kernel to start, it loads it into memory and transfers control of the machine to that operating system.
5. init program is initiated and it will read the inittab file (`/etc/inittab`) and set up the appropriate run level.

Table 1. *A sample grub.cof file*

Architecture	Boot Loaders
AMD32, AMD64,	GRUB
Intel x86, EMT64T	GRUB
Intel Itanium	ELILO
IBM eServer System i	OS/400
IBM eServer System p	YABOOT
IBM System z	z/IPL

Working with grub.conf configuration file

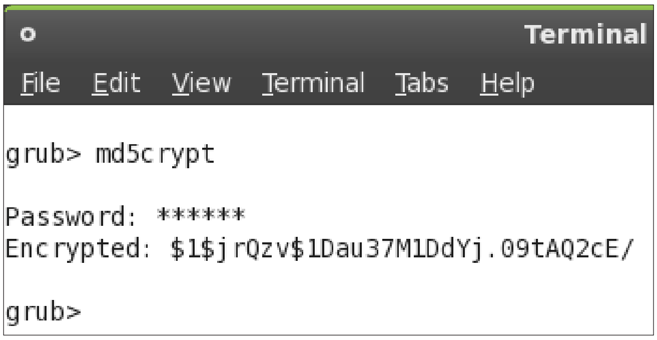
A sample grub.cof file is shown Listing 1. The lines beginning with a `#` are comments.

Listing 1. *grub.cof Shellcode*

```
### Beginning of grub.conf ###
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/sda2
#          initrd /initrd-version.img
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
#####First Operating System#####
title Red Hat Enterprise Linux Server (2.6.18-8.el5)
        root (hd0,0)
        kernel /vmlinuz-2.6.18-8.el5 ro root=LABEL=/ rhgb quiet
        initrd /initrd-2.6.18-8.el5.img
#####Second Operating System#####
titleRedHat Operating System 2
        root(hd1,0)
        kernel /vmlinuz-2.6.18-8.el5 ro root=/dev/sdb2 rhgb quiet
        initrd /initrd-2.6.18-8.el5.img
### End of grub.conf ###
```

The *grub.conf* configuration file is explained in detail below.

- The *default=0* directive points to the first stanza, which is the default Operating System to boot.
- The *timeout=5* directive specifies the time, in seconds, for GRUB to automatically boots the default operating system.
- The *splashimage* directive locates the graphical GRUB screen.
- The *hidden menu* directive means that the GRUB options are hidden.
  A stanza begins with a title, (the text to be displayed in boot menu for selecting the Operating System) and the next three lines specify the location of the /boot directory, the kernel, and the initial RAM disk (The initial RAM disk (initrd) is an initial root file system that is mounted prior to when the real root file system is available), respectively.
- *root (hd0,0)* – Specifies the boot directory is in first hard disk, first Partition.
- *kernel /vmlinuz-2.6.18-8.el5 ro root=LABEL=/ rhgb quiet* – Specifies the kernel location which is inside the /boot folder. This location is related to the root(hd0,0) statement. The “ro” option specifies the kernel should be opened as read only to protect it from any accidental writes from the initial RAM disk and “rhgb” enables the Red Hat Graphical boot option.
- *initrd /initrd-2.6.18-8.el5.img* – Initial RAM disk.

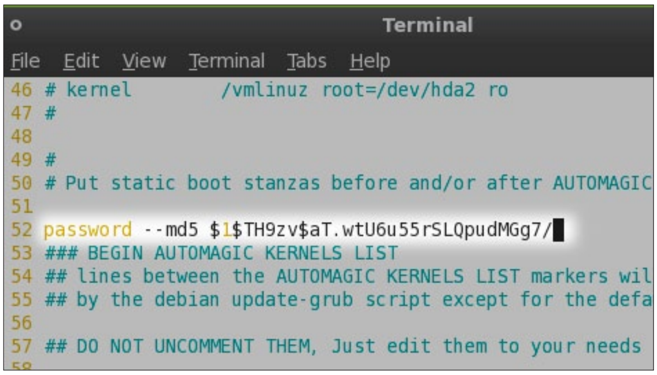


```
grub> md5crypt

Password: ******
Encrypted: $1$jrQzv$1Dau37M1DdYj.09tAQ2cE/

grub>
```

Figure 22. *Putting password on GRUB*



```
# kernel        /vmlinuz root=/dev/hda2 ro
#

#
# Put static boot stanzas before and/or after AUTOMAGIC

password --md5 $1$TH9zv$aT.wtU6u55rSLQpudMGg7/
### BEGIN AUTOMAGIC KERNELS LIST
## lines between the AUTOMAGIC KERNELS LIST markers wil
## by the debian update-grub script except for the defa

## DO NOT UNCOMMENT THEM, Just edit them to your needs
```

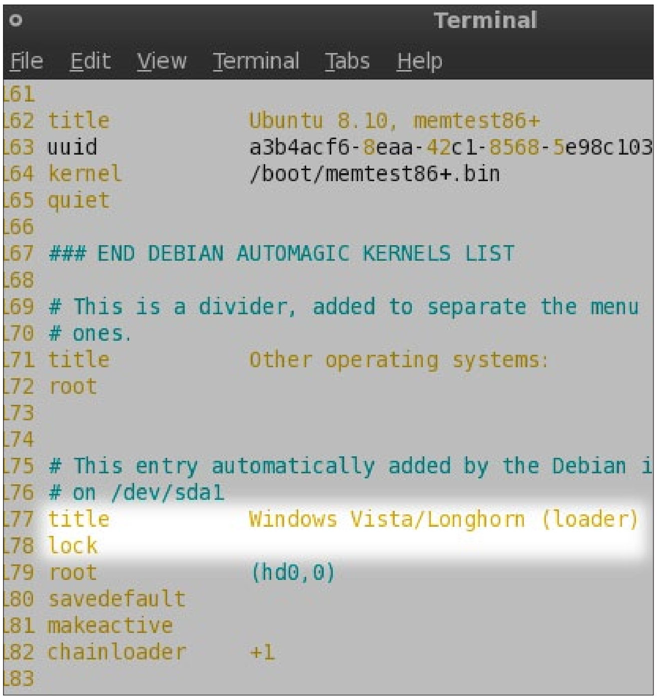
Figure 23. *Making password backup*

Setting up GRUB password in Linux

GRUB security features allow you to lock down the editing of boot options accessed by pressing the ‘e’ key and they allow you to password protect selected or all boot entries.

Follow the steps below to see how to password protect GRUB entries:

- Fire up the terminal. Type *grub* and press enter. The prompt would change to something like ‘grub>’.
- Enter `md5crypt` at the GRUB prompt. Type in the password when prompted for and press enter. The command will return you password encrypted as an md5 hash. You will need this so make a note of it or copy to the clipboard (Figure 22).
- Now we need to edit the `/boot/grub/menu.lst` file. You are advised to make a backup of the file before editing it in case something goes wrong (Figure 23).
- Enter the line `password –md5 <the copied md5 string from step 3>` before the line that reads: “BEGIN AUTOMAGIC KERNEL LIST” (actually it just needs to come before any of the boot menu entries, so you can write it anywhere as long as it is before them).



```

title           Ubuntu 8.10, memtest86+
uuid            a3b4acf6-8eaa-42c1-8568-5e98c103
kernel          /boot/memtest86+.bin
quiet

### END DEBIAN AUTOMAGIC KERNELS LIST

# This is a divider, added to separate the menu
# ones.
title           Other operating systems:
root


# This entry automatically added by the Debian i
# on /dev/sda1
title           Windows Vista/Longhorn (loader)
lock
root            (hd0,0)
savedefault
makeactive
chainloader     +1

```

Figure 24. *Securing kernel Layer through grub.conf file*

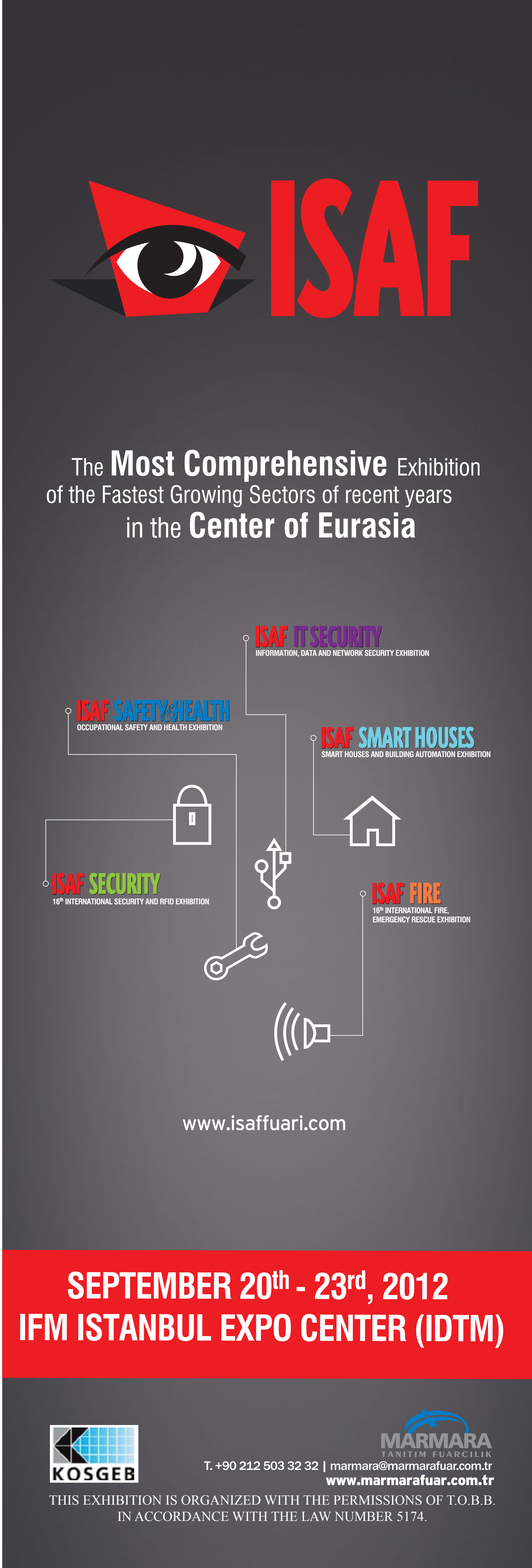
On the Web

[1] *http://en.wikipedia.org/wiki/Kernel_(computing)*
[2] *http://en.wikipedia.org/wiki/BIOS*

- If you save the file at this moment without any further edits you would have locked down interactive editing in GRUB. The administrator or in this case you would have to press ‘p’ key and enter the correct password to access these advanced options.
- If in addition you want to lock down specific menu entries so that anyone without the knowledge of the correct password cannot boot into that operating system you should add the word *lock* all by itself on a separate line just after the title specification for each entry in the menu (Figure 24).
- The next time anyone tries to select the locked menu entry he/she will be required to enter a password before he/she can boot into the corresponding operating system.
- To lock the recovery mode entries it is best to change the line `lockalternative=false` to `lockalternative=true`. This will lock down all future recovery mode entries as well even if you update the kernel.

VIKAS KUMAR | ETHICAL HACKER | SPEAKER

VIKAS KUMAR (ISHAN) is one of the leading computer security experts available in India. VIKAS KUMAR born on 26 July 1990 in a town called Meerut, UP (India). VIKAS KUMAR started his Group “hackers4u” on Facebook in year 2010 and in two years he bangs the World Wide Web with good computer ethical hacking articles and going to launch the website on Cyber Security & Ethical Hacking and working with an Anti-Hacking Community “I-hackers4u”. The 22 year old guy has the capability to compete with the best people in the business so called” Ethical Hacking”. Workshops and Seminars: VIKAS KUMAR has trained more than 1350 people from all around the world, from countries like Thailand, Australia, Canada, Ghana, United States, South Africa, China, Malaysia, Singapore, Omen, Yemen, Indonesia, Korea, Iran and etc. www.cyber-hunt.com | Blog: www.cyber-hunt2012.blogspot.com | LinkedIn Profile: https://www.linkedin.com/profile/view?id=71569482&trk=tab_pro | Facebook: https://www.facebook.com/hackers4u | BackTrack Fan Club Page: https://www.facebook.com/pages/Cyber-Hunt-BackTrack-Fan-Club/395372283859684?ref=tn_tnmn | Facebook Page: https://www.facebook.com/vikas7852?ref=tn_tnmn | Email ID: vikas_ind2008@yahoo.in | cyberhunt2012@gmail.com


ISAF
The Most Comprehensive Exhibition
of the Fastest Growing Sectors of recent years
in the Center of Eurasia
ISAF IT SECURITY
INFORMATION, DATA AND NETWORK SECURITY EXHIBITION
ISAF SAFETY&HEALTH
OCCUPATIONAL SAFETY AND HEALTH EXHIBITION
ISAF SMART HOUSES
SMART HOUSES AND BUILDING AUTOMATION EXHIBITION
ISAF SECURITY
16th INTERNATIONAL SECURITY AND RFID EXHIBITION
ISAF FIRE
16th INTERNATIONAL FIRE, EMERGENCY RESCUE EXHIBITION
www.isaffuari.com
SEPTEMBER 20th - 23rd, 2012
IFM ISTANBUL EXPO CENTER (IDTM)
KOSGEB
MARMARA
TANITIM FUARCILIK
T. +90 212 503 32 32 | marmara@marmarafuar.com.tr
www.marmarafuar.com.tr
THIS EXHIBITION IS ORGANIZED WITH THE PERMISSIONS OF T.O.B.B.
IN ACCORDANCE WITH THE LAW NUMBER 5174.

Configure And Build

Your Own Secure Linux Kernel

One of the best ways to get a feeling for the Linux kernel internals and security features is to configure its settings and then compile it. Most GNU/Linux users and administrators use kernels configured and provided by the community (free and open source distributions) or corporate sponsors (e.g. Red Hat Enterprise Linux, SUSE Linux Enterprise, Canonical – Ubuntu).

One of the best ways to get a feeling for the Linux kernel internals and security features is to configure its settings and then compile it. Most GNU/Linux users and administrators use kernels configured and provided by the community (free and open source distributions) or corporate sponsors (e.g. Red Hat Enterprise Linux, SUSE Linux Enterprise, Canonical – Ubuntu).

The goal of the article is to give you an idea of how to configure a kernel with customized and/or fewer features, which will reduce the chances of an attacker breaking into your systems. For that purpose we will be using 32-bit Ubuntu 12.04 LTS (Long Term Support) distribution.

Lab configuration:

- 32-bit Ubuntu 12.04 LTS virtual machine running on:
- VMware Fusion 5.0.1 installed on Mac OS X 10.7.4 (2.66 GHz Intel Core i7 MacBook Pro with 8 GB RAM).

The Stock Kernel (aka Vanilla or Mainline or Linus Linux Kernel)

The *stock kernel (aka vanilla or mainline or Linux Linux kernel)* is the generic kernel developed and maintained by the *kernel.org* (The Linux Kernel Archives) repository. *Kernel.org* developers associated with *The Linux Kernel Organization* constantly keep adding features to the kernel, including security improvements and patches.

For the purpose of this article, we will recompile the stock kernel. Linus Torvalds began work on Linux in April 1991 and announced it on August 25 1991, in a message to the `comp.os.minix` Usenet newsgroup. Linus is one of the Linux Kernel Organization's board members and the owner of Linux Registered Trademark. In his blog, he explains that his life isn't glamorous and that these days he usually "writes code in the mail reader – mostly telling people 'do it like this' rather than actually writing real code":

```
http://torvalds-family.blogspot.ca/2011/02/pearls-
            before-swine.html
```

Linus Explains Linux Trademark Issues:

```
http://slashdot.org/story/00/01/19/0828245/linus-
            explains-linux-trademark-issues
https://lkml.org/lkml/2005/8/20/95
```

The current maintainers of the `-stable` branch of the Linux kernel are Greg Kroah-Hartman and Chris Wright. Kernels that are close to release but not yet ready are called Release Candidate and have `-rc` suffix.

Kernel Version Numbering

The Linux kernel has gone through three numbering systems. Linus announced the most recent one on May 29, 2011, when he moved it from the release 2.6.39 to version 3.0, to celebrate the 20th anniversary of Linux. The new system uses time-based release practice by incrementing the second number every 6-7 weeks when new features are introduced to the kernel, while the `-stable` team can use the third number for their versioning, for example to designate security and bug fixes.

Distribution-Specific Kernels

Distribution developers customize their Linux kernel to include security and other updates. They thoroughly test them and regularly release updated kernel versions. This makes job of Linux server administrators, especially in most enterprise environments, easier because kernel has already been tested and approved. Consequently, that is the preferred method of keeping your production system kernels updated. However, if you want to customize the kernel, for example, to further change its security features, you can recompile it by starting with either a distribution-provided kernel or with the stock Linux kernel.

To print currently loaded kernel release and version, use the `uname` command with `-r` and -v options: Listing 1.

To get more details about the current kernel, use the `apt-cache` command: Listing 2.

In this case, the current kernel is for 32-bit “systems with more than 4GB RAM”.

Preparation

Since the kernel compilation is a significant project, it’s recommended not to do it on production systems. Instead, it’s advisable to always first test it on your developmental/staging systems.

In addition, make sure to make backups of the current and previous kernel configuration. That is, copy all kernel configurations to a safe location: Listing 3.

Stock Kernel Security Updates

The Linux Kernel developers communicate via The Linux Kernel Mailing List:

- *https://lkml.org/*
- *http://www.tux.org/lkml/*

This is a high-traffic list with the average of around 400 messages per day so you might choose other sources for Linux kernel related news, including security related updates:

- Kernel Coverage at LWN.net (weekly news) *http://lwn.net/Kernel/*
- LinuxSecurity.com – The Community’s Center for Security *http://www.linuxsecurity.com/*

Distribution-Specific Kernel Security Updates

The command to update packages on Red Hat and Red Hat derived systems is `yum update`.

`yum update` includes the latest Linux kernel versions. When updating production systems, system administrators often exclude kernel updates until they test them on test systems first. To ignore kernel updates, use the `yum’s  --exclude=kernel-*` option: Listing 4.

On Ubuntu and Ubuntu based systems, the command to update packages is

Listing 1. *Print the currently loaded kernel release and version*

```
uname -r
3.2.0-31-generic-pae

uname -v
#50-Ubuntu SMP Fri Sep 7 16:39:45 UTC 2012
```

Listing 2. *Obtain information about the current kernel*

```
apt-cache search 3.2.0-31-generic-pae | grep
                  image


linux-image-3.2.0-31-generic-pae - Linux
                  kernel image for version
                  3.2.0 on 32 bit x86 SMP



apt-cache show linux-image-3.2.0-31-generic-
                  pae

Package: linux-image-3.2.0-31-generic-pae
Priority: optional
Section: kernel
Installed-Size: 110438
Maintainer: Ubuntu Kernel Team <kernel-team@
                  lists.ubuntu.com>
Architecture: i386
Source: linux
Version: 3.2.0-31.50
. . .
. . .
Description-en: Linux kernel image for version
                  3.2.0 on 32 bit x86 SMP
This package contains the Linux kernel image
                  for version 3.2.0 on 32 bit
                  x86 SMP.
.
Geared toward 32 bit desktop or server systems
                  with more than 4GB RAM.

Origin: Ubuntu
Supported: 5y
```

```
apt-get update
```

This will download lists of new available packages from all repositories to find out whether any of the packages needs update. To actually upgrade the system, use `apt-get upgrade`, which will download and install actual packages. Thus, to install all packages, including kernel, run the both

Listing 3. *Backup kernel configurations*

```
ls -lh /usr/src/linux*/.config

-rw-r--r-- 1 root root 144K Apr 10 19:20 /usr/src/linux-headers-3.2.0-23-generic-pae/.config
-rw-r--r-- 1 root root 144K Sep  7 10:53 /usr/src/linux-headers-3.2.0-31-generic-pae/.config

cp /usr/src/linux-headers-3.2.0-23-generic-pae/.config ~/lin-hdr-3.2.0-23.config
cp /usr/src/linux-headers-3.2.0-31-generic-pae/.config ~/lin-hdr-3.2.0-31.config



ls -lh /boot | grep config

-rw-r--r-- 1 root root 144K Apr 10 19:17 config-3.2.0-23-generic-pae
-rw-r--r-- 1 root root 144K Sep  7 10:50 config-3.2.0-31-generic-pae


cp /boot/config-3.2.0-23-generic-pae ~/config-3.2.0-23-generic-pae.backup
cp /boot/config-3.2.0-31-generic-pae ~/config-3.2.0-31-generic-pae.backup
```

Listing 4. *Exclude kernel updates on Red Hat based systems*

```
yum update --exclude=kernel-*
```

Listing 5. *Update all packages, including kernel, on Ubuntu based systems*

```
apt-get update && apt-get upgrade
```

Listing 6. *Ensure that the system has necessary development software*

```
apt-get install fakeroot kernel-package build-essential ccache libncurses5 libncurses5-dev
```

Listing 7. *Download the most recent kernel source*

```
wget http://www.kernel.org/pub/linux/kernel/v3.0/linux-3.5.4.tar.xz
```

Listing 8. *Download the corresponding PGP signature*

```
wget http://www.kernel.org/pub/linux/kernel/v3.0/linux-3.5.4.tar.sign
```

Listing 9. *Decompress the stock kernel source and verify the .tar archive against the signature*

```
unxz linux-3.5.4.tar.xz

gpg --verify linux-3.5.4.tar.sign linux-3.5.4

gpg: linux-3.5.4: read error: Is a directory
gpg: Signature made Fri 14 Sep 2012 03:28:50 PM PDT using RSA key ID 6092693E
gpg: Can't check signature: public key not found
```

`apt-get update` and `apt-get upgrade` commands: Listing 5.

Kernel Development Software

We need to make sure that the system has C language libraries and compilers, kernel header files and related development tools (Listing 6).

Good Practice

After testing it and making sure that new kernel

Listing 10. *gpg --verify output when run first time*

```
gpg: Signature made Fri 14 Sep 2012 03:28:50 PM PDT using RSA key ID 6092693E
gpg: Can’t check signature: public key not found
```

Listing 11. *Download the public key from the PGP keyserver in order to verify the signature*

```
gpg --recv-keys 6092693E
gpg: requesting key 6092693E from hkp server keys.gnupg.net
gpg: key 6092693E: public key “Greg Kroah-Hartman (Linux kernel stable release signing key) <greg@
                 kroah.com>” imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)

gpg --verify linux-3.5.4.tar.sign
gpg: Signature made Fri 14 Sep 2012 03:28:50 PM PDT using RSA key ID 6092693E
gpg: Good signature from “Greg Kroah-Hartman (Linux kernel stable release signing key) <greg@kroah.
                 com>”
gpg: WARNING: This key is not certified with a trusted signature!
gpg:  There is no indication that the signature belongs to the owner.
Primary key fingerprint: 647F 2865 4894 E3BD 4571  99BE 38DB BDC8 6092 693E
```

Listing 12. *Linux kernel releases PGP signatures page (https://kernel.org/signature.html) quote about verifying the owner of the key used to sign the kernel source archive*

```
Notice the WARNING: This key is not certified with a trusted signature! You will now need to verify
that the key used to sign the archive really does belong to the owner (in our example,
Greg Kroah-Hartman). There are several ways you can do this:

1. Use the Kernel.org web of trust (https://kernel.org/signature.html#kernel-org-web-of-trust).
This will require that you first locate the members of kernel.org in your area and sign their keys.
Short of meeting the actual owner of the PGP key in real life, this is your best option to verify
the validity of a PGP key signature.
2. Review the list of signatures on the developer’s key by using “gpg --list-sigs”. Email as many
people who have signed the key as possible, preferably at different organizations (or at least
different domains). Ask them to confirm that they have signed the key in question. You should attach
at best marginal trust to the responses you receive in this manner (if you receive any).
```

Listing 13. *Move and extract the kernel source to /usr/src directory*

```
mv linux-3.5.4.tar /usr/src

cd /usr/src

tar -xf linux-3.5.4.tar
```

works, you should remove kernel development software from a production system.

Download The Stock Linux Kernel and Verify Its Digital Signature

Visit the Linux Kernel Archive and check which release is the latest: *https://kernel.org*.

As of time of this writing (September 22, 2012), the latest release is *3.5.4*. Download the latest release and the corresponding PGP signature (Listing 7 – 10). Note the “key ID” and download this key from the key servers (Listing 11). At this point you can follow instructions from the Linux kernel releases PGP signatures page: Listing 12.

Listing 14. *Content of the /usr/src directory after extracting the kernel source*

```
ls -lh

drwxrwxr-x 23 root root 4.0K Sep 14 15:28 linux-3.5.4
drwxr-xr-x 24 root root 4.0K Apr 23 04:37 linux-headers-3.2.0-23
drwxr-xr-x  7 root root 4.0K Apr 23 04:37 linux-headers-3.2.0-23-generic-pae
drwxr-xr-x 24 root root 4.0K Sep 22 15:38 linux-headers-3.2.0-31
drwxr-xr-x  7 root root 4.0K Sep 22 15:38 linux-headers-3.2.0-31-generic-pae
```

Listing 15. *Collect the currently loaded kernel configuration*

```
cd /usr/src/linux-3.5.4

cp -vi /boot/config-`uname -r` .config

`/boot/config-3.2.0-31-generic-pae’ -> `.config’
```

Listing 16. *Making oldconfig*

```
/usr/src/linux-3.5.4# make oldconfig

  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  SHIPPED scripts/kconfig/zconf.tab.c
  SHIPPED scripts/kconfig/zconf.lex.c
  SHIPPED scripts/kconfig/zconf.hash.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
scripts/kconfig/conf --oldconfig Kconfig
*
* Restart config...
*
*
* General setup
*
. . .
. . .
. . .

#
# configuration written to .config
#
```

Listing 17. *Getting familiar with make utility*

```
make help | less
```

Listing 18. *Build the kernel*

```
make -j2
```

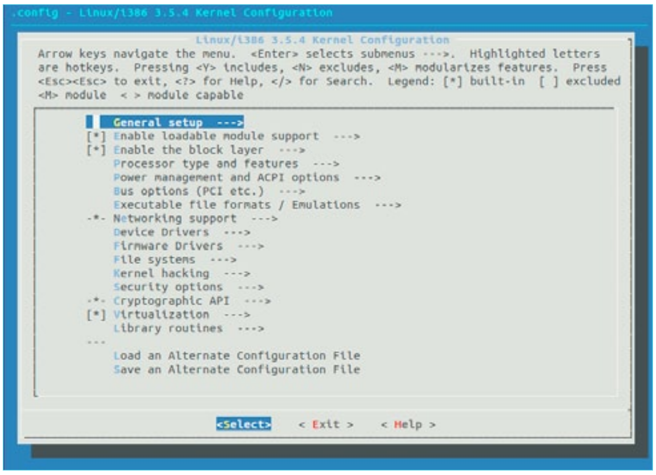
Listing 19. *Output of the make command*

```
SYSHDR  arch/x86/syscalls/../include/generated/
                  asm/unistd_32.h
HOSTCC  arch/x86/tools/relocs

. . .
. . .
```

Configure and Build the New Kernel

Extract the kernel source to `/usr/src` directory. In our case, ensure that previously decompressed `.xz` archive is moved and extracted in `/usr/src` directory: Listing 13 and Listing 14. Since we will be starting with Ubuntu-specific kernel, we will need to collect the currently loaded kernel configuration from the `/boot` directory and copy it to the extracted `linux` directory with the source code for the kernel that we want to compile (Listing 15).



```
.config - Linux/i386 3.5.4 Kernel Configuration

                    Linux/i386 3.5.4 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters
  are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press
  <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded
  <M> module  < > module capable

          General setup  --->
      [*] Enable loadable module support  --->
      [*] Enable the block layer  --->
          Processor type and features  --->
          Power management and ACPI options  --->
          Bus options (PCI etc.)  --->
          Executable file formats / Emulations  --->
      -*- Networking support  --->
          Device Drivers  --->
          Firmware Drivers  --->
          File systems  --->
          Kernel hacking  --->
          Security options  --->
      -*- Cryptographic API  --->
      [*] Virtualization  --->
          Library routines  --->
      ---
          Load an Alternate Configuration File
          Save an Alternate Configuration File
```

Figure 1. *Make menuconfig kernel configuration tool*

Note

From this point on almost everything will be done in the directory with the kernel source code, which is, in our case:

```
/usr/src/linux-3.5.4
```

Next, use `make oldconfig`, which uses this previous `.config` file to answer all questions that it can, only interactively presenting the new features that are not already answered in that file.

Table 1. *Kernel Sections for Configuration*

General setup	How the kernel talks to the BIOS, whether to support PCI or PCMCIA, ... Options whether the system is embedded, kernel compression modes, etc. Except the "Automatically append version information to the version string" option (see the Tip! box below the table) and, if you would like to enable it, the "Enable access to .config through /proc/config.gz" option, most likely you will not change any options in this section.
Enable loadable module support	The default options load appropriate modules as needed. This keeps the basic kernel small.
Processor type and features	Options regarding the architecture that will be running the kernel.
Power Management and ACPI Options	Power management options related to hibernation and suspend to RAM and standby. Since during hibernation the content of RAM is written to disk, it could present a security risk so you might want to disable hibernation in the kernel.
Bus options (PCI etc.)	Support for PCI cards is typically enabled in this part of the kernel configuration. If you are not planning to use PCI, PCMCIA cards, you can disable them here.
Executable file formats/ emulations	Executable Linkable Format (ELF)
Networking support	Extensive network options, relating to wired and wireless networking, IrDA (infrared), Bluetooth, WiMAX Wireless Broadband, Amateur Radio, etc. Also includes options related to the kinds of network packets that the kernel can work with: DHCP, the Bootstrap Protocol (BOOTP), and the Reverse Address Resolution Protocol (RARP).
Device Drivers	You can experiment with disabling some of the devices and that way reduce number of available devices in case of a malicious attack. The list is really big and some of the devices are: Memory Technology Devices (flash cards, RAM chips, and similar), SCSI devices support, SATA (Serial ATA) and PATA (Parallel ATA) drivers, RAID and LVM, Macintosh device drivers, ISDN support, etc.
Firmware Drivers	Firmware drivers support: BIOS, EFI Variable Support via sysfs, iSCSI Boot Firmware Table Attributes, Google Firmware Drivers.
File systems	Enable or disable ext3, ext4, Reiserfs, FUSE, CD-ROM/DVD Filesystems, DOS/FAT/NT Filesystems, Pseudo (/proc, sysfs), Network File System (NFS), Quota support.
Kernel hacking	Mostly for kernel developers. Includes advanced kernel debugging options. Not for production kernels because it adds additional routines, which increases the kernel size and slows performance.
Security options	Enable or disable and adjust different security models: Security Enhanced Linux (SELinux), TOMOYO Linux, AppArmor, Yama.
Cryptographic API	Support for cryptography hardware, cryptographic and cipher algorithms.
Virtualization	Enable or disable virtualization modules for the Kernel-based Virtual Machine (KVM).
Library routines	CRC functions and decompression support.

Good Practice

It’s recommended to always run `make oldconfig` before building a kernel (Listing 16).

This will present you with a series of interactive questions. If you are not sure about some of the options, leave the default values.

It could be useful if you get familiar with the `make` utility: Listing 17.

After that, use `make menuconfig` kernel configuration tool. It’s a menu-driven user-interface, which allows you to choose the features of the Linux kernel that will be compiled (Figure 1).

This is where the real fun begins! You will have to go through trial and error process until you configure a more secure kernel but as long as you have system and kernel backups, you are safe to experiment.

Refer to Table 1 for an overview of the options.

Tip!

Under the “General Setup” menu, navigate to the “Automatically append version information to the version string” option and enable it. This will add a suffix to the name of the kernel, so that you can distinguish it from the stock kernel.

After completing the configuration, it’s time to start compiling the kernel. If you have multiple CPU cores, you can use the `-jN` option, which specifies the number of jobs (commands) to run simultaneously. Usually you can spawn one or two jobs per core. For example, if you have two cores, you could run (Listing 18 and Listing 19):

```
make -j4
```

Be patient… On my system, it took 32 minutes for the compilation to complete.

Listing 20. *Ensure that the new kernel will have access to modules*

```
SYSHDR  arch/x86/syscalls/../include/generated/asm/unistd_32.h
HOSTCC  arch/x86/tools/relocs

. . .
. . .
```

Listing 21. *Completing the kernel installation*

```
make install
```

Listing 22. *Output of the make install*

```
make install

sh /usr/src/linux-3.5.4/arch/x86/boot/install.sh 3.5.4 arch/x86/boot/bzImage \
      System.map “/boot”
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 3.5.4 /boot/vmlinuz-3.5.4
update-initramfs: Generating /boot/initrd.img-3.5.4
run-parts: executing /etc/kernel/postinst.d/pm-utils 3.5.4 /boot/vmlinuz-3.5.4
run-parts: executing /etc/kernel/postinst.d/update-notifier 3.5.4 /boot/vmlinuz-3.5.4
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 3.5.4 /boot/vmlinuz-3.5.4
Generating grub.cfg ...
Found linux image: /boot/vmlinuz-3.5.4
Found initrd image: /boot/initrd.img-3.5.4
Found linux image: /boot/vmlinuz-3.2.0-31-generic-pae
Found initrd image: /boot/initrd.img-3.2.0-31-generic-pae
Found linux image: /boot/vmlinuz-3.2.0-23-generic-pae
Found initrd image: /boot/initrd.img-3.2.0-23-generic-pae
Found memtest86+ image: /boot/memtest86+.bin
done
```

The build process will create the file `System.map` in the root of the kernel source tree, which is in our case `/usr/src/linux-3.5.4`. This file is used during debugging – it contains a symbol lookup table, which maps kernel symbols to their start addresses.

Install The Kernel

After we built the kernel, we can install it. To set up the new kernel and initial RAM disk in the `/boot` directory, and to make necessary changes to the bootloader (in our case GRUB – the Grand Unified Bootloader): Listing 21 and Listing 22.

In order to ensure that Linux will have access to modules that are compatible with the new kernel, install modules by running `make modules_install`: Listing 20.

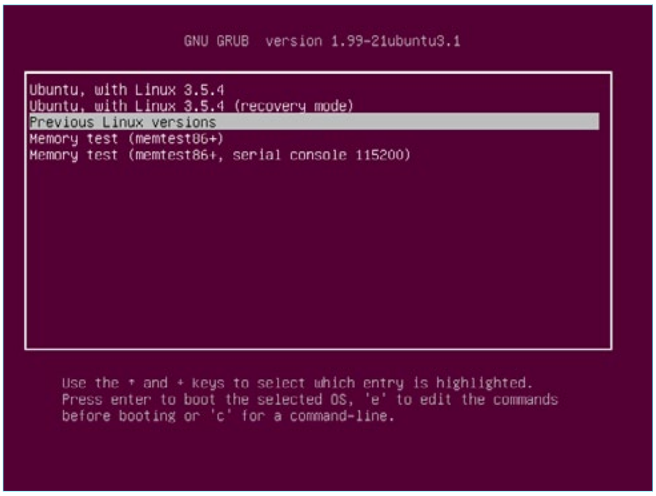
This will install all the compiled modules to `/lib/modules`.

We are done! Reboot the system and check the new kernel!

Troubleshooting – Restore to Previous Kernel Version

If you are experiencing problems with the new kernel, boot into the previous kernel. Note that with the new GRUB2, boot menu does not appear by default. To show the menu, you need to hold down the Shift key in order to display the menu during boot. In some cases, pressing the Esc key may also display the menu (Figure 2 and Figure 3).

If you need to permanently change the boot order, modify the `/etc/default/grub` file.

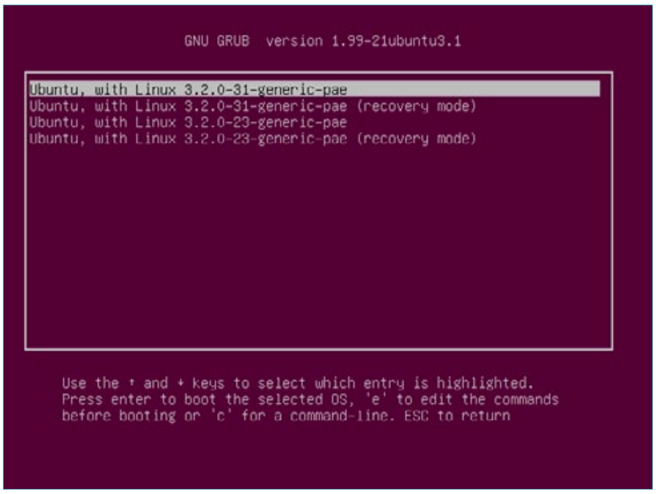


```
GNU GRUB  version 1.99-21ubuntu3.1

Ubuntu, with Linux 3.5.4
Ubuntu, with Linux 3.5.4 (recovery mode)
Previous Linux versions
Memory test (memtest86+)
Memory test (memtest86+, serial console 115200)

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the commands
before booting or 'c' for a command-line.
```

Figure 2. *GRUB2 boot menu*



```
GNU GRUB  version 1.99-21ubuntu3.1

Ubuntu, with Linux 3.2.0-31-generic-pae
Ubuntu, with Linux 3.2.0-31-generic-pae (recovery mode)
Ubuntu, with Linux 3.2.0-23-generic-pae
Ubuntu, with Linux 3.2.0-23-generic-pae (recovery mode)

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the commands
before booting or 'c' for a command-line. ESC to return
```

Figure 3. *GRUB2 menu – Previous Linux versions option*

Listing 23. *Update grub.cfg after modifying /etc/default/grub file*

```
update-grub

Generating grub.cfg ...
Found linux image: /boot/vmlinuz-3.5.4
Found initrd image: /boot/initrd.img-3.5.4
Found linux image: /boot/vmlinuz-3.2.0-31-generic-pae
Found initrd image: /boot/initrd.img-3.2.0-31-generic-pae
Found linux image: /boot/vmlinuz-3.2.0-23-generic-pae
Found initrd image: /boot/initrd.img-3.2.0-23-generic-pae
Found memtest86+ image: /boot/memtest86+.bin
done
```

Listing 24. *Disable ICMP broadcast echo activity by adding the following line to /etc/sysctl.conf file*

```
net.ipv4.icmp_echo_ignore_broadcasts = 1
```

Listing 25. *Ignore ICMP spoofed messages by adding the following line to /etc/sysctl.conf file*

```
net.ipv4.icmp_ignore_bogus_error_responses = 1
```

For example, if you want to change the order and revert the default to the third menu entry, which in our case is 'Ubuntu, with Linux 3.2.0-31-generic-pae', we would change GRUB_DEFAULT line in the /etc/default/grub file from

```
GRUB_DEFAULT=0
```

to

```
GRUB_DEFAULT=2
```

Tip!
GRUB2 menu item entries start with 0 so the third menu item is numbered 2.

After changing the /etc/default/grub file, you have to run update-grub to update /boot/grub/grub.cfg file: Listing 23.

Hardening the Kernel via the /proc/ Filesystem

In addition to making the kernel more secure by configuring and compiling it, you can additionally harden it by using the dynamic kernel options in the /proc/ directory filesystem.

Since the /proc/ directory filesystem is dynamic, changes will not persist after a reboot. To make permanent changes, you need to add the necessary options to the /etc/sysctl.conf file. This file controls sysctl values (sysctl is an interface for modifying kernel parameters at runtime).

Disable ICMP Broadcast Echo Activity

To prevent a Smurf attack, configure your system to not reply to broadcasts by adding the following line in /etc/sysctl.conf:

```
net.ipv4.icmp_echo_ignore_broadcasts = 1
```

This doesn't prevent responses to targeted ping commands. It only affects broadcasts.

Protect from Bogus ICMP Messages

To ignore ICMP spoofed messages that do not comply to standards, enable this option:

Listing 26. *Activate TCP Cookie by adding the following line to /etc/sysctl.conf file*

```
net.ipv4.tcp_syncookies = 1
```

Listing 27. *Enable Reverse Path Filtering by adding the following two lines to /etc/sysctl.conf file*

```
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1
```

Listing 28. *Prevent traffic between networks by adding the following three lines to /etc/sysctl.conf file*

```
net.ipv4.ip_forward = 0
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0
```

Listing 29. *Prevent MITM attacks by adding the following line to /etc/sysctl.conf file*

```
net.ipv4.conf.all.accept_redirects = 0
```

Listing 30. *Disable source routing by adding the following two lines to /etc/sysctl.conf file*

```
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0
```

Listing 31. *Enable Martian packets logging by adding the following two lines to /etc/sysctl.conf file*

```
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.default.log_martians = 1
```

```
  /proc/sys/net/ipv4/icmp_ignore_bogus_error_
                  responses
```

Make the System More Resistant to SYN Floods

In the SYN flood attack, the targeted machine is flooded with TCP segments requesting connections. These TCP segments have only the SYN bit set in the TCP header but not the ACK bit. You can help prevent dropped connections by enabling TCP SYN Cookie. With TCP SYN Cookie, the kernel does not allocate the TCP buffers unless the server's ACK/SYN packet gets an ACK back, confirming that the request was legitimate.

To enable TCP Cookie, enable this option:

```
/proc/sys/net/ipv4/tcp_syncookies
```

Enable Reverse Path Filtering

To prevent source address spoofing, enable Reverse Path Filtering. When enabled, it compares packets against the routing table and drops a packet for which the best route for the source IP address does not use the same interface that the packet was received on.

To activate reverse path filtering, enable these options:

```
/proc/sys/net/ipv4/conf/all/rp_filter
/proc/sys/net/ipv4/conf/default/rp_filter
```

This will configure RPF for default and all networks.

Disable Traffic Between Networks

Most systems are not set up to be routers. To deactivate a system as a router, configure it via these three options:

```
/proc/sys/net/ipv4/ip_forward
/proc/sys/net/ipv4/conf/all/send_redirects
/proc/sys/net/ipv4/conf/default/send_redirects
```

This will disable traffic between networks for default and all networks.

Prevent MITM (Man-In-The-Middle) Attacks

Do not accept ICMP redirects (Listing 29).

Prevent Source Routing

Disabling source routing prevents network users from specifying the route a packet takes to a destination. This way you maintain control over how packets are sent (Listing 30).

This will disable traffic source routing for default and all networks.

Resources

- The Linux Kernel Organization: The Primary Site for the Linux Kernel Source *https://kernel.org*
- Latest Stable Kernel (3.5.4) – As of Sep 24, 2012: *http://www.kernel.org/pub/linux/kernel/v3.0/linux-3.5.4.tar.xz*
- Linux Kernel Releases PGP Signatures *http://www.kernel.org/signature.html, https://kernel.org/signature.html*
- The Linux kernel mailing list *https://lkml.org/*
- Linux 3.6-rc6 (Release Candidate) Released, Final Is Coming "Soonish" *https://www.linux.com/news/software/linux-kernel/636007-linux-36-rc6-released-final-is-coming-qsoonishq*
- Linus Announcing the New Version Numbering – Linux Kernel Version 3.0 *https://lkml.org/lkml/2011/5/29/204*
- Linux Kernel Releases PGP Signatures – To Verify the Integrity of the Linux Kernel Source Code *https://kernel.org/signature.html*
- The Linux Advisory Watch Security Newsletter *http://www.linuxsecurity.com/*

Log and Drop Martian Packets

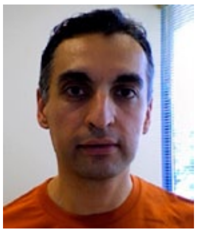
Packets with addresses that should not be possible (for which the host does not have a route back to the source IP address) are known as Martian packets (Listing 31).

This will enable Martian packets logging for default and all networks.

Summary

The developers of major Linux distributions update kernels on a regular basis. If you want to secure your kernel even more it's not such a daunting task to configure it and compile it by yourself. In addition, for some distributions it takes some time for the latest kernel updates to be incorporated and to make this waiting period shorter, you can build your own kernel. By going through this process you will get familiar with the kernel internals and learn which of its configuration settings are related to security. The gained knowledge will help you tremendously in making your systems and networks more secure.

DUSKO PIJETLOVIC



Dusko Pijetlovic is an IT Manager and Sr. Systems Administrator in Vancouver, Canada and holds a M.Sc. in Mechanical Engineering and Diploma of Technology in Computer Systems Technology. He is a proponent of GNU/Linux and Free and Open Source Software, with a passion for security, solving problems and helping organization members perform their jobs with excellence and efficiency.

ca.linkedin.com/in/duskopijetlovic

http://www.xing.com/profile/Dusko_Pijetlovic

SRDF: Write Your Own Security Tool

Do you see writing a security tool in windows is hard?
Do you have a great idea but you can't implement it?
Do you have a good malware analysis tool and you don't need it to become a plugin in OllyDbg or IDA Pro?
So, Security Research and Development Framework is for you.

This is a free open source Development Framework created to support writing security tools and malware analysis tools. And to convert the security researches and ideas from the theoretical approach to the practical implementation.

This development framework created mainly to support the malware field to create malware analysis tools and anti-virus tools easily without reinventing the wheel and inspire the innovative minds to write their researches on this field and implement them using SRDF.

Introduction

In the last several years, the malware black market grows widely. The statistics shows that the number of new viruses increased from 300,000 viruses to millions and millions nowadays.
The complexity of malware attacks also increased from small amateur viruses to stuxnet, duqu and flame.

The malware field is searching for new technologies and researches, searching for united community can withstand against these attacks. And that's why SRDF

The SRDF is not and will not be developed by one person or a team. It will be developed by a big community tries to share their knowledge and tools inside this Framework

SRDF still not finished … and it will not be finished as it's a community based framework developed by the contributors. We just begin the idea.

The SRDF is divided into 2 parts: User-Mode and Kernel-Mode. And we will describe each one in the next section.

The Features

Before talking about SRDF Design and structure, I want to give you what you will gain from SRDF and what it could add to your project. In User-Mode part, SRDF gives you many helpful tools … and they are:

- Assembler and Disassembler
- x86 Emulator
- Debugger
- PE Analyzer
- Process Analyzer (Loaded DLLs, Memory Maps …)
- MD5, SSDeep and Wildlist Scanner (YARA)
- API Hooker and Process Injection
- Backend Database, XML Serializer
- And many more

In the Kernel-Mode part, it tries to make it easy to write your own filter device driver (not with WDF and callbacks) and gives an easy, object oriented (as much as we can) development framework with these features:

- Object-oriented and easy to use development framework
- Easy IRP dispatching mechanism
- SSDT Hooker
- Layered Devices Filtering
- TDI Firewall
- File and Registry Manager
- Kernel Mode easy to use internet sockets
- Filesystem Filter

Still the Kernel-Mode in progress and many features will be added in the near future.

Let's now see the design:

The User-Mode Part

The Design: Figure 1.

Infrastructure

This includes the essential elements of any development framework and it's not related to security like: string, hash, list, serializer, database, registry manipulation, sockets and so on.

We decided to create this part rather than depending on any development framework to make this framework independent from any other development frameworks and to be portable on any development framework.

Targets

This is the beginning of the SRDF. This part is simply the Target from your security tool. What do you want to secure or secure from. And it includes Files (PE Files and others), Processes and Packets.

Libraries

That's the security tools that the SRDF support. And it's divided into two namespaces: malware and network.

Malware includes the assemblers and disassemblers, emulator, debugger, API Hooker, Yara Scanner (wildcard scanner) file recursive scanner and other tools.

Network includes User-Mode capturing and Firewall.

Core (The Application Interface)

The Core includes the Logging system and the back-end Database.

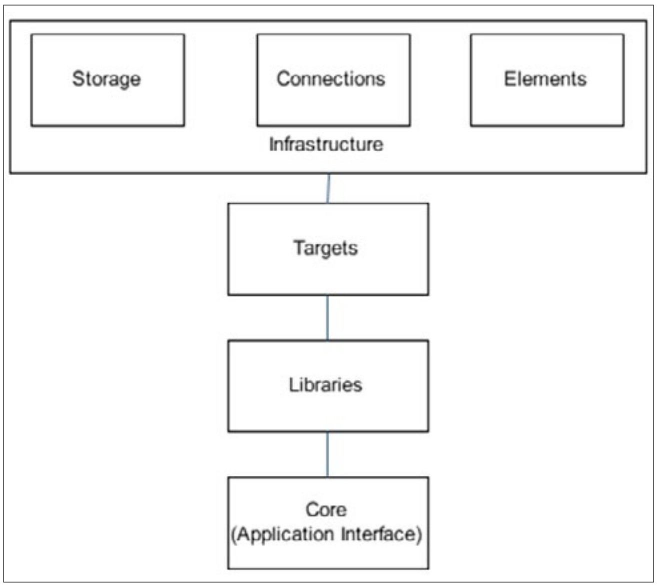


Figure 1. *Core Scheme*

And also, it's the Application Interface. Like cConsoleApp … and you can inherit from it to create your own User-Interface.

We wish this part to be expanded to include more user interfaces and management systems.

The Infrastructure

Elements

It's divided into three namespaces:

- String: it contains the string class, encoded string, hash and list
- Code: it contains the NativeCode class and StoredProcedure … and they represents the shellcode and the code that stored in database. Like a virus detection routines inside an Antivirus
- XML: and it contains the XML Encoder and the Serializer.

Connections

It's divided into three namespaces:

- Internet: and it contains the internet communication protocols like sockets, HTTP Sockets and so on.
- IPC: and it contains the Inter-Process Communication protocol
- User-Mode to Kernel-Mode Communication: and it contains the communication protocol to communicate to the kernel-mode part of the SRDF

Storage

It's divided into three namespaces:

- Databases: and it contains the Database class and SQLiteDB and so on.
- Files: and contains the File writing and logging classes
- Registry: and it contains the registry read and write

The Targets

Files

This namespace describes the File Formats of The Files that could contain malicious code like: Executable Files (PE and ELF) and Document Files (PDF, Docx …) and so on.

Until now it contains The PE Files parser.

Process

And it includes one class only named cProcess. And, this class describes a running process and parses its PEB and gives you the important information about the process and its memory map. And support injecting code and create a remote thread.

Packets

And it includes classes that describe an internet packets captured on the wire or generated for an attack.

Libraries

It contains two namespaces:

Malware

This namespace contains the scanning, Hooking and emulation libraries and contains Pokas Emulator wrapper class, Yara wrapper class (wildcard scanner), a debugger and contains a directory recursive scanner and other tools. And also, it contains the x86 assembler and disassembler (using Pokas Emulator Assembler) and allow to contain other assemblers and for other platforms.

Network

This namespace should contain the User-Mode Packet capture and firewall. And should contain the Winpcap Packet capturing and firewall system. It also should include Application Layer parsers for FTP, HTTP, IRC and all known protocols and include Pcap Reader and writer.

The Core

And the core includes the cApp class that contains the back-end database and logging and the User-Interface such as `cConsoleApp`.

The Kernel-Mode

The Kernel-Mode Goals

The Goals of the kernel-Mode development Framework are:

- Easy to create a Kernel-Mode security tool
- Support OOP using the native device driver programming APIs
- Support detaching between devices in IRPs
- Easy to use files, registry and so on
- Create a User-Mode/Kernel-Mode communication protocol
- Designed only for hooking and security tools.

The Kernel-Mode SRDF is designed on native device driver programming APIs and independent

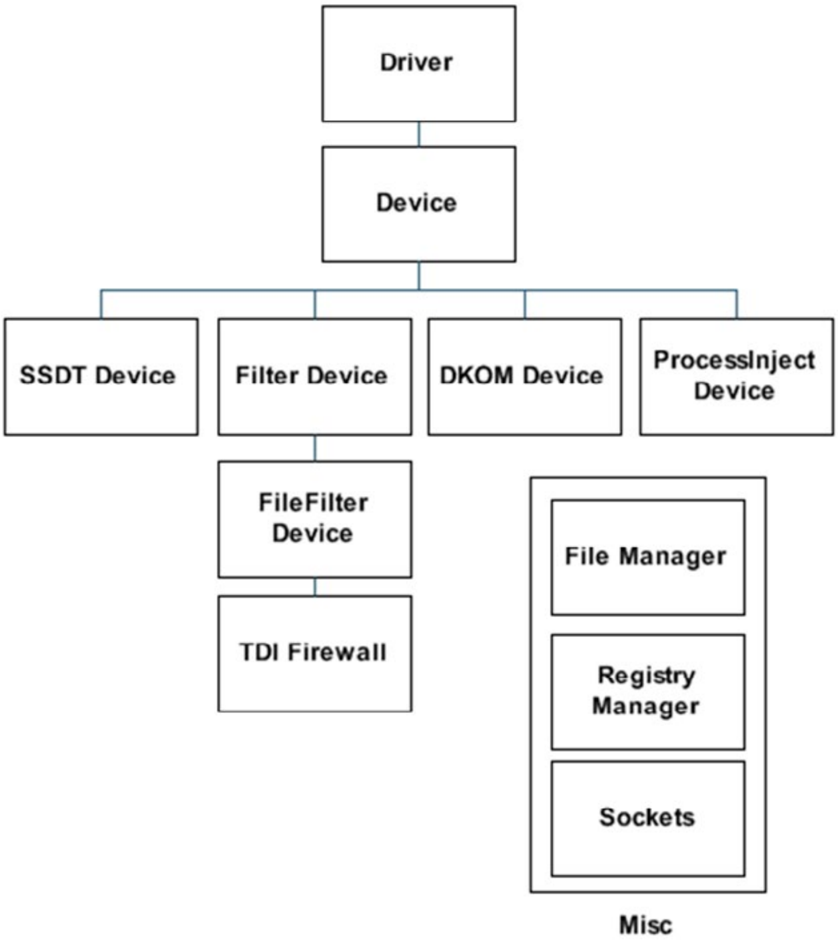


Figure 2. *Kernel-Mode SRDF*

from the WDF (windows drivers foundation). Now we will describe the design of Framework and then we will go through the IRP dispatching mechanism in the KM-SRDF. The Design: Figure 2.

Driver
It's the core management system that dispatching the IRPs to the devices and manage the devices.

Device
It represents a device object and it contains the IRP dispatching between the control device object and the filtering device objects and includes attaching and detaching from a devices chain and all necessary functions for a device object.

SSDT Device
This class is inherited from device class and it's created for SSDT Hooking.

Filter Device
This class created for attaching to a chain and filtering the inputs and the outputs of the IRPs.

File Filter Device
This class is inherited from Filter Device and it's created for filtering the File system I/O request packets (IRPs) or monitoring file operations.

TDI Firewall
This class is inherited from Filter Device and it's created for filtering the internet packets and connections and the processes that tries to connect to the internet.

DKOM Device
This class created to provide a generic way to work with opaque structures in windows without worrying about windows version and subversion (under construction).

Process Device
This class provides a way to inject code or modify the memory of a process from the kernel-mode.

File/Registry Managers
They are tools created to support writing files and working with registry easily without worrying about IRQL.

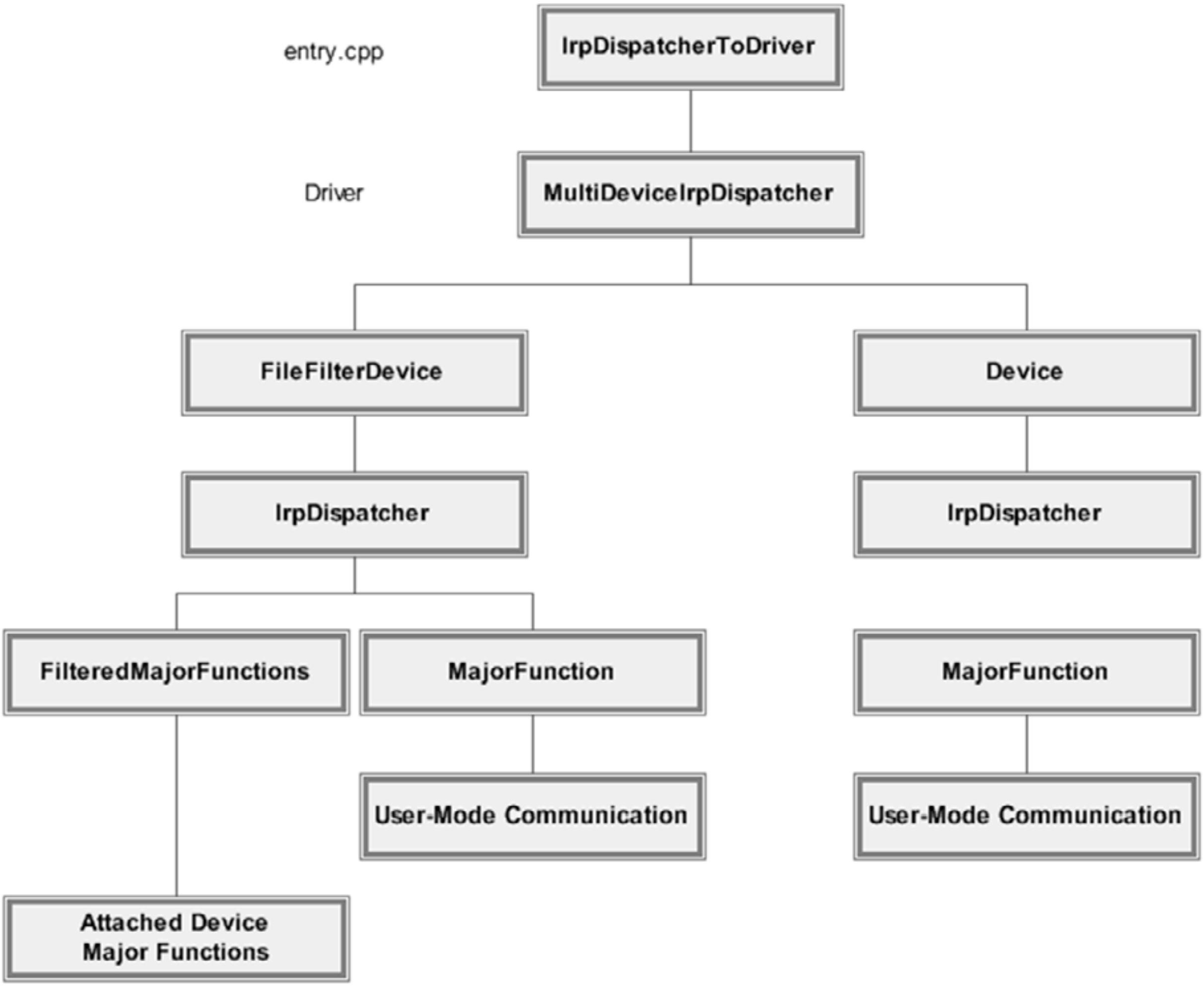


Figure 3. *Kernel-Implemention Scheme*

Sockets
It's an easy interface to connect to the internet using the TDI interface.

The IRP Dispatching

- The IRP dispatching begins from the entry.cpp and it dispatch the IRP to the Driver
- The driver checks the device object and dispatch the IRP to the related device
- The device sends the IRP to the User-Mode communication object to work with it as it's sent to the control device object
- If it's a FileFilter Device, the device dispatches the IRP based on the device object to the Attached Device Objects or to the control device object and the user-mode communication

Source Code
http://code.google.com/p/srdf/.

Join Us

Do you get benefit from this framework and you need to give something back?
Do you want to add something to your CV?
Do you want to meet smart developers and join a big community?
Do you want to learn new things?

Here is place … join the development community, meet new smart people and have fun.

To do List

- Antivirus:
  - XRAY Tool
  - Heuristics Analysis
  - Behavior-based Detection Tools.
  - More File Formats (PDF, apk, …)
  - OpenSBI and other Virus Classification File Formats
  - Sandboxing Mechanism.
    - Using API/ SSDT Hooking
    - Emulation Based on Pokas Emulator.
  - Update System with Flexible Mechanism
- Malware Analysis:
  - SSDT Hooking for (Processes, Files, Registry and Sockets System Calls)
  - API Hooking (for the same as above)
  - Improvement in Pokas Emulator, Assembler and Disassembler
  - Packet Capturing Tool and Emulated IRC and HTTP Connection (Server emulate the replies to the malware and log the data)
  - Recursive Disassembler
  - More APIs Emulation in Pokas x86 Emulator
  - Support more Instructions (All FPU instructions, All general purpose instructions and support mmx and 3dnow)
  - Support idb (IDA Pro Database) to read it and use its analysis
- Unpackers: I'm aiming to create a database for all static unpacking codes for the mostly common unpackers and I hope it could be updated by the community
- Integrations:
  - Integration into IDA Pro Plugin Interface … and in (Debugger Menu)
  - OllyDbg Plugin Interface
  - Ollyscript Executer on cDebugger
  - Metasploit Integeration (in Meterpreter Post Exploitation
  - Python, Ruby, Delphi Header files and cTypes for SRDF.dll
- Network:
  - Support NDIS, kernel sockets and more new libraries
  - Process Analyzer in Kernel-Mode
  - Packet Capturing Library
  - More Debugging and Bug fixing
- Others:
  - We need to build website.
  - We need activities for learning.
  - We need more documentations and tutorials
  - We need more helpful tools and applications based on SRDF

Conclusion

This development framework will support the anti-malware technologies to grow and support implementing researches in the malware field more to withstand against the new attacks nowadays.

The framework is based on community and we aim to create a big community for it. We didn't finished the framework … we just begin.

AMR THABET
I'm Amr Thabet. I'm a Malware Researcher at Q-CERT. I began in this field from nearly 4 years. I'm the Author of Pokas x86 Emulator. I gave a talk in Cairo Security Camp 2010 and the University of Sydney.

Hooking Socket API calls on Linux

The socket APIs of BSD are the de-facto standard for network programming. Unlike operating at the packet level, which requires dealing with issues such as fragmentation, duplicates, and stream assembly, sockets provide the logical abstraction of a connection endpoint. From this endpoint, either messages (UDP) or logical streams (TCP) may be sent and received.

The same easy abstraction provided to programmers proves useful for hackers. By hijacking socket calls, a hacker can gain unparalleled views into an application, inspecting and even injecting data to its flows. This article describes the various methods of socket level hijacking found in Linux.

User-mode Hijacking

The easiest way to hook calls at the socket level is, undoubtedly, to perform simple user-mode hooking of functions. Two methods come to mind here:

I) Using LD_PRELOAD, you can override (technically, "interpose") any function (not just socket calls!) with your own implementation. All it takes is crafting your own library, as shown in Listing 1.

As you can see, the wrapper is a simple function, which prints out the values of the arguments to `socket()`, then calls on the real socket. Naturally, we are not going to implement the (surprisingly complex) logic of socket(2). Rather, we call on dlsym(3) to find us the address of the real socket(2), and pass the call to it.

Note two things about this approach:

- This method will not interpose calls to the function performed by other libraries, loaded after yours. In other words, it is guaranteed to get only the calls of the executable proper.
- `LD _ PRELOAD` will not work with setuid binaries, unless you are already root.

Incidentally, a similar approach (albeit with a different environment variable, `DYLD _ INSERT _ LIBRARIES`) will work in OS X and iOS. You can also specify a linker attribute of "interpose" to create a special interpose section. An example can be found in [OSX].

II) The second approach is using the ptrace(2) APIs. A tracer can use ptrace(2) calls, much like gdb(1) does, to find the address of the call in memory, and interpose it during runtime. This has the huge advantage of catching all calls, not just those of the executable. The disadvantage, however, is in having to either start the traced process, or react fast enough to intercept the calls before they are being made.

Kernel-mode Hijacking

While user-mode libraries provide the socket functions to applications, they do little in effect but wrap corresponding system calls. These system calls, like many others, are provided by the kernel – and are therefore fair game for interception by a rootkit.

Due to the idiosyncrasies involved with the various protocols, socket system calls normally find and make use of the underlying protocol structure, which supports the socket operations (Listing 2).

As you can see, there is a one to one correlation between the user mode APIs and the kernel implementation. Specifically, the familiar socket descriptor (an int file descriptor in user mode) is a struct sock*.

Linux exposes these protocol structures, which are – surprisingly – global. You can verify the protocol symbols are exported using by grep(1)ing /proc/kallsyms. The second grep(1) call isolates those symbols which are global data structures ('D' in nm(1) notation) (Listing 3).

Intercepting a socket call now becomes no more difficult than any standard function hook. Namely, it merely involves saving a pointer to the call being hooked, then wrapping it with a prolog and epilog.

Building the bare bones module skeleton around this code, yields Listing 4.

Inspecting Socket Data

Note

The methods shown here *cannot* be used to intercept SSL-encrypted data, because SSL-encryption occurs in user mode, prior to queuing the data on the send buffer list, or after `recv()` returns the

Listing 1. *Hooking socket()*

```
#include <stdio.h>
#include <dlfcn.h>

void _init (void)
{
        // You can perform some initialization
                tasks, if required,
        // in _init (and likewise, teardown in _
                fini, although bear
        // in mind not all processes unload
                libraries in an orderly
        // fashion)

        //   printf(“My library has been
                loaded\n”);
}

typedef int (socket_impl) (int, int , int);

int socket (int af, int socktype, int proto)
{
        void *handle_to_real_libc;
        socket_impl     *real_socket;
        int rc;

        // prolog: simply print the arguments
                here

        printf (“IN MY SOCKET: Socket of type %d
                created\n”, socktype);

        // Now do the magic: The -11 is also
                known as “RTLD_NEXT”

        real_socket = (void *) dlsym((void *)
                -11, “socket”);

        if (!real_socket)
               {

                fprintf(stderr,”Ah...\n”);
                      return -1;

            }
                rc = (real_socket)(af, socktype,proto));

                // optional epilog

                return (rc);


}



#
# Compile, but not yet link our library. Note
                  the -fPIC, for position inde-
                  pendent code
# (required in some 64-bit cases)
#
morpheus@Forge$ cc trojan.c -c -o trojan -fPIC
#
# Now link
#
morpheus@Forge$ ld -shared trojan -soname lib-
                  trojan.so.1 -o libtrojan.so.1
#
# Now inject
morpheus@Forge$ LD_PRELOAD=$PWD/libtrojan.so.1
                  telnet 127.0.0.1 22
IN MY SOCKET: socket of type 2050 created
                  (SOCK_DGRAM | SOCK_CLOEXEC)
IN MY SOCKET: socket of type 2 created
                  (SOCK_DGRAM)
Trying 127.0.0.1
IN MY SOCKET: socket of type 1 created
                  (SOCK_STREAM)
Connected to 127.0.0.1.
Escape character is ‘^]’.
SSH-2.0-OpenSSH_5.9....
```

receive buffers. Nonetheless, the same does not hold for IPSec encryption, which occurs in kernel mode prior to actually sending the socket buffer – and hence these methods will, in fact, intercept plaintext data from IPSec connections.

Hooking `connect()` operations is useful for intercepting when connections are established. But sometimes the intrepid hacker would like to sift through the packet data itself. The problem is that the receive path is entirely different from the send path. The reception is performed by `tcp_protocol`'s "handler", as shown in the Listing 5.

But .. what's that? "static"? "const"? Indeed, we have a small problem: The `tcp_protocol` is defined as static (that is, not exported for other modules use) and – worse – in read only memory. A quick glance in `/proc/kallsyms` corroborates this: Note the lowercase (static) and "r" (read-only).

```
root@Forge:~# grep tcp_protocol /proc/kallsyms
ffffffff815c13a0 r tcp_protocol
```

Listing 2. *The struct proto definition, from <net/sock.h> in the Linux kernel headers*

```
  struct proto {
        void                    (*close)(struct sock *sk,
                                        long timeout);
        int                     (*connect)(struct sock *sk,
                                        struct sockaddr *uaddr,
                                        int addr_len);
        int                     (*disconnect)(struct sock *sk, int flags);
        struct sock *           (*accept) (struct sock *sk, int flags, int *err);

        int                     (*ioctl)(struct sock *sk, int cmd,
                                         unsigned long arg);
        int                     (*init)(struct sock *sk);
        void                    (*destroy)(struct sock *sk);
        void                    (*shutdown)(struct sock *sk, int how);
        int                     (*setsockopt)(struct sock *sk, int level,
                                        int optname, char __user *optval,
                                        unsigned int optlen);
        ...
        int                     (*sendmsg)(struct kiocb *iocb, struct sock *sk,
                                           struct msghdr *msg, size_t len);
        int                     (*recvmsg)(struct kiocb *iocb, struct sock *sk,
                                           struct msghdr *msg,
                                        size_t len, int noblock, int flags,
                                        int *addr_len);
        ...

}
```

Listing 3. *Verifying protocol symbols are available to be hooked*

```
root@Forge:~# cat /proc/kallsyms | grep prot$ | grep D
ffffffff817d8c00 D selinux_checkreqprot
ffffffff8180ec40 D tcp_prot
ffffffff8180f280 D raw_prot
ffffffff8180fc20 D udp_prot
ffffffff8180fec0 D udplite_prot
ffffffff818149c0 D udpv6_prot
ffffffff81814c80 D udplitev6_prot
ffffffff81814f60 D rawv6_prot
ffffffff81815c40 D tcpv6_prot
```

Listing 4. *Sample module to hook TCP connect() operations*

```
#include <linux/module.h> /* Module related stuff */
#include <linux/sched.h> /* for current , for_each_process ... */
#include <net/sock.h>    /* for struct proto */
#include <linux/in.h>   /* for struct sockaddr_in */

MODULE_LICENSE("GPL"); /* Can use GPL-only symbols, and we're open source.. */
MODULE_AUTHOR("J@Technologeeks.com");
MODULE_DESCRIPTION("This is a test module, demonstrating TCP protocol hooking");

extern struct proto tcp_prot; // For TCPv4. Use tcpv6_prot for TCPv6

typedef int (*connect_t)
  (struct sock *sk, struct sockaddr *uaddr, int addr_len);

connect_t old_connect;

int my_connect (struct sock    *sk,
               struct sockaddr *uaddr,
               int addr_len)
{

  // Cast to sockaddr_in is safe, because we hooked tcp_prot
  // For IPv6, you'll need a my_connect6, which casts to a
  // sockaddr_in6 structure, instead
  //
  struct sockaddr_in *sinaddr = (struct sockaddr_in *)uaddr;

  char *addrbytes = (char *) &(sinaddr->sin_addr.s_addr);

  printk("Attempting to connect to %d.%d.%d.%d port: %hd\n",
                addrbytes[0],
                addrbytes[1],
                addrbytes[2],
                addrbytes[3],
                ntohs(sinaddr->sin_port));

   // Example: reroute port 2410 connections to port 22
  if (sinaddr->sin_port == htons(2410))
        sinaddr->sin_port = htons(22);

  // Example: block port 80 connections
  if (sinaddr->sin_port == htons(80)) return (-EPERM);

  // Example: block a particular black-listed IP
  {
    unsigned long blacklistedIP = inet_addr("127.0.0.1");
    if (sinaddr->sin_addr.s_addr == blacklistedIP) return (-EPERM);
  }

 // Example: block a particular process by inspecting "current"...

 /*
  * pass thru to original, old_connect
  * We could also add an epilog if we wanted to
  * save the original connect's return value
  */

  return((old_connect)(sk, uaddr, addr_len));


} // end my_connect


static int __init my_module_entry(void)
{
 old_connect = tcp_prot.connect;
 tcp_prot.connect = my_connect;
 return(0); // Otherwise insmod/create_module fails

}

static void my_module_exit(void)
{
   // Restore original connect - otherwise the
   // next TCP connection will cause an oops!
   tcp_prot.connect = old_connect;
}

/* Best saved for last */
module_init(my_module_entry);
module_exit(my_module_exit);
```

Getting past the "static" part can be done in a simple, though unelegant way – simply pass the address to the module (or programmatically access the kernel symbols, for that matter). The "read-only" part, however, will cause a kernel "oops" if we don't do something about it:

Listing 5. *Using tcp_protocol's „handler"*

```
static const struct net_protocol tcp_protocol = {
    .early_demux    =       tcp_v4_early_demux,
    .handler        =       tcp_v4_rcv,
    .err_handler    =       tcp_v4_err,
    .gso_send_check =      tcp_v4_gso_send_check,
    .gso_segment    =       tcp_tso_segment,
    .gro_receive    =       tcp4_gro_receive,
    .gro_complete   =       tcp4_gro_complete,
    .no_policy      =       1,
    .netns_ok       =       1,
};
```

Listing 6. *Setting a jprobe on the TCP handler*

```
int my_handler(struct sk_buff *skb)
{
        int i = 0;

        printk(“GOT SKB %hd\n”, skb-
                 >protocol);

        char *dataPtr = skb->data;

        // Inspect frame data: The skb->data
                 points to the Ethernet header
        // IP Header is at skb->data + (sizeof
                 eth_header).
        // TCP Header is at skb->data +
                 (sizeof eth_header) +
                 sizeof(tcphdr);

        for (i = 0 ; i < skb->data_len; i++)
        {
               printk(“%02x “,skb->data[i]);

        }
        printk(“\n”);


        jprobe_return();
        //return (old_handler)(skb);

}
```

References

[OSX] *http://www.newosxbook.com/src.jl?tree=listings&file=4--5-interpose.c*

[2] kprobes, *http://www.kernel.org/doc/ols/2006/ols2006v2--pages-109-124.pdf*

[3] kprobes, RedHat Magazine

```
[31534.076886] BUG: unable to handle kernel paging
 request at ffffffff815c13a0
[31534.076891] IP: [<ffffffffa01dc070>]
 my_module_entry+0x70/0x97 [module]
[31534.076898] PGD 1003067 PUD 1007063 PMD 2f433063
PTE 800000000015c1161
[31534.076901] Oops: 0003 [#1] SMP
 <..snip..>
```

Remember, though – we are already in kernel mode. The OS is our oyster. We can therefore use one of two workarounds:

- Make the memory writable, and *then* set our value
- Set a kprobe or jprobe on the handler function (Listing 6).

Matching the data to the socket is a simple matter. Interesting sockets can be tagged during the `connect()` and `accept()` hooking, shown above.

Conclusion

This article focused on the myriad ways by means of which a simple protocol-level hook can assist a hacker in getting a better insight into the network traffic of a system. By staying at the socket level, the complicated packet handling logic involving fragmentation, reassembly and ordering can be reused, providing a view nearly identical to what the application sees – albeit much more powerful.

There are other methods for hooking and intercepting network data on Linux. These include, but are not limited to, Netfilter hooks, BRfilter hooks, and BPF. Those striving to get packet and frame-level control, are encouraged to investigate them further. For now, they are left out, possibly for some future article.

JOHNNY LEVIN

Register right now to Save more!

MIWA

Mobile Internet World & Awards2012

5-7 December 2012| Beijing Marriott Hotel Northeast

The Dawn of a New Mobile Era

Conference Highlights:

- **500+** Attendees
- **80+** One-on-one meetings
- **40+** Speakers
- **30+** Operators
- **20+** Hours networking
- **5** Special Awards

OUR VISION:

Mobile Internet World & Awards (MIWA) is the largest and most elite global mobile Internet summit. The novel state of the mobile Internet industry will prove to be a stimulating point of discussion, and will make **MIWA** one of the most exciting and engaging events,with enterprises from a wide array of industries ranging from services, software, games, end–user device producers, telecom operators, government representatives, industrial organizations, academic elites, investors, media, totally reaching over **250** representatives participating to discuss the developmental changes of the global mobile Internet industry.

Organizer	Co-organizer	Endorser	Associate Sponsor	Presentation Sponsor	Co-located with
IITA	CDMC Refresh Your Work!	SVC WIRELESS	DOLBY.	one2many Reaching your audience	DP Asia Digital Publishing Summit & Awards

Media support

上方网 sfw.cn yesky.com 天极网 移動通信 MOBILE COMMUNICATIONS 懒汉互联 PHONE WORLD THE SMART SENSE TechWeek europe Hakin9 IT SECURITY MAGAZINE

AND…YOU CAN' T AFFORD TO MISS!
Specialized and Interactive Workshop-October 31st 2012

Four kinds of the host mode
Marketing : Elim Weng
☎ +86 21 6840 7631
🖷 +86 21 6840 7633
http://www.cdmc.org.cn/mi2012/
✉ mi@cdmc.org.cn

Linux Kernel Exploit

Android OS – The storm is over?

Writing an article on Linux Kernel Exploitation is always a challenge. During the last decade the Linux Kernel has been constantly under the spotlight for a number of issues including vulnerabilities, controversial design, and structural aspects.

The security of Linux Kernel has been covered at many levels. Discussing the latest penetration technique or the latest bug report is always a useful exercise, but makes it seem that defense strategy is exclusively based on a reactive approach rather than a preventive approach.

During the last few years, many attacks have been developed to hit commercial platforms for obvious economic reasons.

Smartphones and portable devices are the preferred target, Android and Apple IOS2 being two of the most popular mobile operating systems.

Both platforms derive from a Unix/Linux system (even though Google engineer Patrick Brady stated unambiguously that "Android is not Linux"). In fact, the kernel and the Command Line Interface are solidly based on the most common Linux environment.

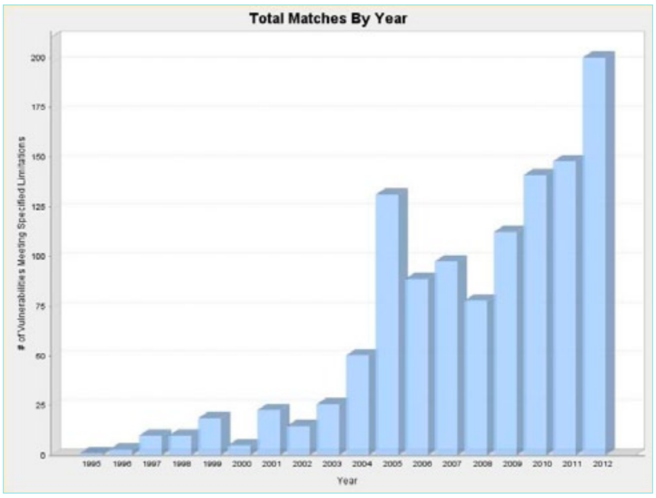


Figure 1. *Total Matches Diagram*

Some Statistics

Let's have a look at the statistics. Querying the US National Vulnerability Database on Linux Kernel vulnerabilities, including Software Flaws – CVE (Figure 1), and looking at the last 12 years, we can observe and analyze the trend of the attacks against Linux systems.

Reading these figures, we can see that vulnerabilities are progressively increasing, mostly because of the wider use of Linux on mobile and portable devices.

According to Kaspersky Lab, "28 percent of all mobile devices attacked by malware, in the second and third quarter of 2012, were running Gingerbread (Android) and 91 percent of all Android malware detected in the last 14 days of September were on mobile devices, running either Gin-

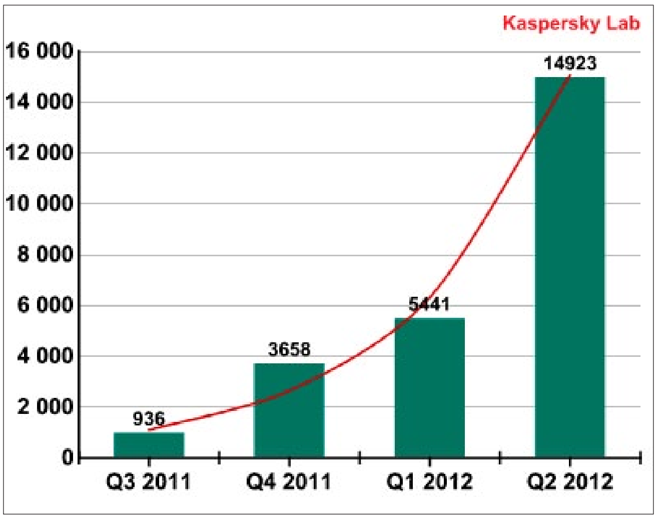


Figure 2. *Malware Attack Overview*

gerbread or Ice Cream Sandwich (Android).” Perhaps, not many users were upgrading their mobile or Firmware.

Another study on malicious code targeting Android platforms, reports an increment of about 14,900 new cases between Q1 and Q2 of 2012 (Figure 2).

Statistics show the threat evolution. Recently, the attacks have been propagating faster and broader, thus, we should expect an accordingly huge amount of new attacks targeting Android devices in the near future.

History

Android Inc. was founded in 2003 at Palo Alto, California. The company was acquired by Google in 2005 (for a very low price), planning to develop a new mobile device based on a flexible and upgradable Linux-Based System. An initial release of the Android OS was made in September 2008, launched to be the first market competitor for Apple Iphone iOS.

Android can be considered as an “open” project, as it is partially disclosed and constantly under development by a community made of around 1339 contributors (the entire kernel project can be found at: *https://android.googlesource.com/*).

Android has stolen a huge portion of the market from all of the Linux based systems, firing up a long and controversial debate between Google and Linux founders, such as Linux Torvalds and Greg Kroah-Hartman.

Kernel Security Analysis

The Android Kernel has been elaborated and based on Linux Kernel 2.6/3.x. using C, C++, and Java as main programming languages (Unix/Linux Kernel is generally written in Assembly, a Low-Level language code, and C).

By design, the aim is to protect the kernel from the rest of the running programs/processes and propose at least 2 execution operational modes: the privileged mode (full access) and the unprivileged mode (access to a subset of instructions).

In the kernel space, the execution of the code runs full privileges thus, when it comes to set up virtual memory, it is mandatory to ensure separation between Kernel and User-Land in order to limit potential misbehaving and malicious activity.

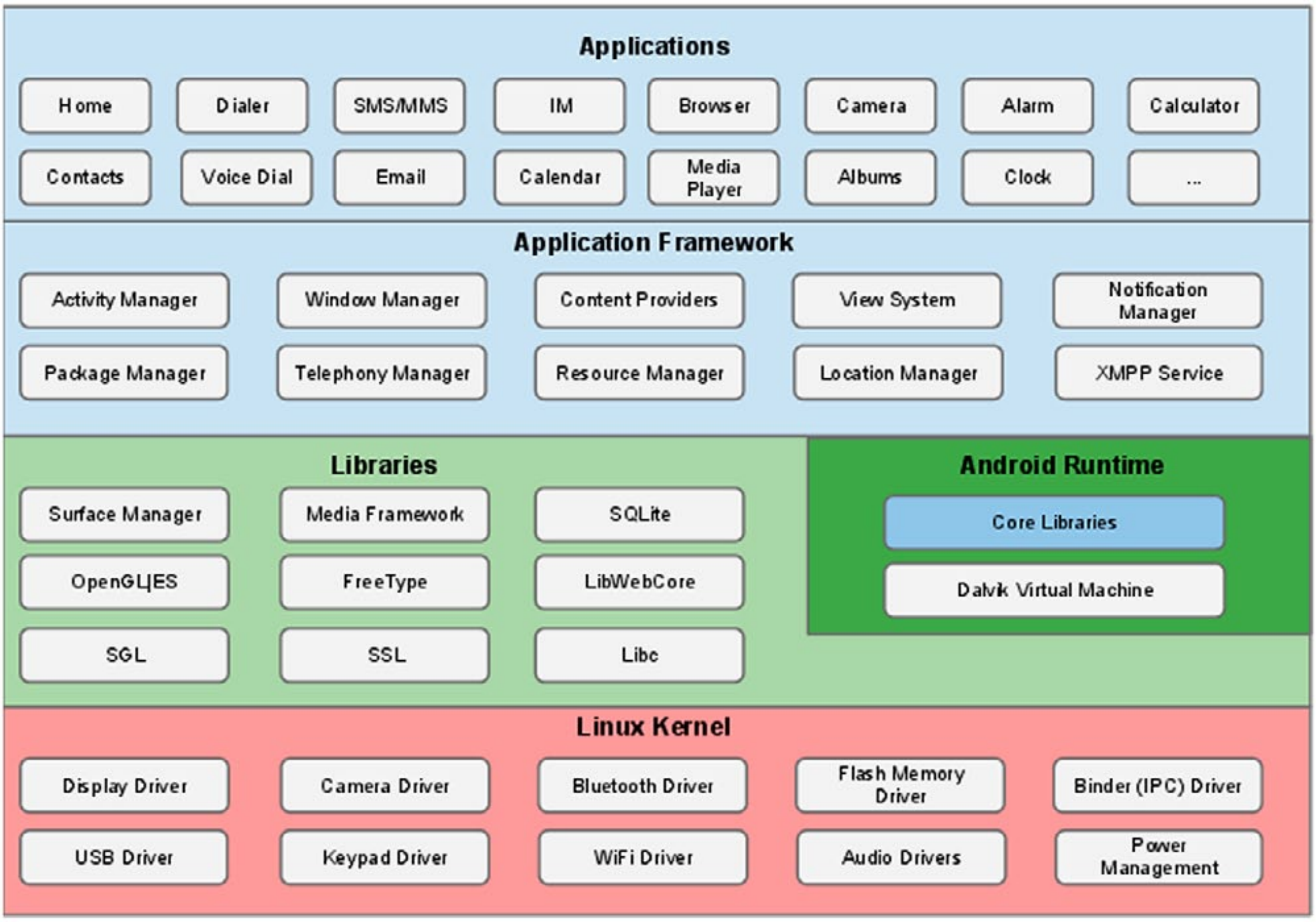


Figure 3. *Kernel Application Scheme*

Android kernel provides the same secure architecture as the other Linux kernels, included a secure IPC (Inter-process communication) and a Sandbox that aim to limit rogue code execution able to harm the system.

The Figure 3 shows Android system stack highlighting separation between protected and unprivileged areas.

Kernel Attacks

Nowadays, the exploitation of Android kernel vulnerabilities seems to be a very lucrative activity.

The most common exploitation attacks have been mitigated using countermeasures and patches including security mechanisms to protect kernel memory corruptions such as NULL page mappings, stack and heap corruptions, and so on. Although, while using unsigned/uncertified .apk packages, most of these mitigations seem to be nullified.

Attackers have developed many ways to bypass kernel and system protections. They can count on a number of entry points such as: IOCTL, FS code, Network, sys_calls etc. Unfortunately, security protection based on Sandbox may be bypassed and compromised by using known vulnerabilities,memory managements, and allocation. NULL pointers and stack overflow in pre-processors can still be a valid attack vector.

Programming "bad-practices," such as the usage of vulnerable functions, may lead to bugs and memory corruption due to, among others, incorrect or insufficient input validation and race conditions.

Pitfalls lurk in many aspects of programming and implementation of the system.

Attempting to set the specific condition directly in kernel-land may result in kernel panic, error messages, as well as in system reboot. Thus, launching an attack from Kernel-Land to User-Land may avoid dangerous conditions and open the door to a more "silent" intrusion. Shellcode and Malware (Rootkit for instance…) are two of the most frequent attacks.

Rootkit is a malicious application that attempts to exploit kernel vulnerabilities and run the code with higher privileges on the target machine.

This "bad code" may reside in the kernel at Ring 0 manipulating and replacing a portion of the code, acting as a device driver or as a loadable module. When this malicious programs gain unrestricted access, they can become very hard to detect and remove.

It is not simple to write a Rootkit, but once installed, this program can hide and prevent detection by using a variety of stealth measures to survive, propagate, compromise, and subvert all the operating system functions.

The obfuscation mechanism is based on a complex and polymorphic behavior. Once the Rootkit is enabled, it will try disabling the logging system (Android logs are available under the directory `/dev/log`), or blocking process monitoring and detection applications, as well as preventing commands such as "ps" to display specific active processes.

The Kernel structure

The system call is the way the kernel communicates with all of the processes. Thus, if you want to change the behavior of the kernel in interesting ways, this is the place to do it.

First, the program sets up arguments for the system call. One of the arguments is the system call number. After the arguments are all set up, the program executes the "system call" instruction. This instruction causes an exception (an event that causes the processor to jump to a new address and start executing the code there).

The instructions at the new address save the program's state, figure out which system call you want, call the function in the kernel that implements that system call, restores program state, and returns control back to the user program.

When an exception is generated on the ARM processor, the execution is forced to a fixed memory offset that corresponds to the thrown exception. These fixed offsets are called the exception vectors.

Kernel Hooking Technique Through sys_call_table

Rootkit's attempts to modify parts of the OS by installing compromised drivers or Kernel modules using various techniques. By using LKM (Loadable Kernel Module), it can both add and remove a new code without a kernel compile.

Good examples of Rootkit Kernel Hooking methodologies for Android ARM platform are available at: *http://www.phrack.com/issues.html?issue=68&id=6*.

As we can see, one of the ways described to hook Android Kernel based on ARM processor is to search and get the `sys _ call _ table` address in `vector _ swi` (the software interrupt part of the exception vector table) handle. Then, find nop code (NOP or NOOP is short for No Operation, is an assembly language instruction – See URL Ref. 10) around and store the address of compromised `sys _ call _ table`. Now, we get the `sys _ call _ table` handle code from the offset in which compromised `sys _ call _ table` resides, and hooking starts.

Other Ways to Exploit Kernel

The privilege escalation is another critical threat for the kernel.

A simple exploit can be developed by "misusing" the class/function WRITE_EXTERNAL_STORAGE (for example on an SD Card) to obtain higher privileges, and attempting to propagate these privileges onto other files located in the conquered location (have a look at URL Ref. 11).

Shellcode is a sequence of instructions injected at runtime and coded using a low level language, such as Assembly, to gain privileges and run commands.

An interesting example is "CVE-2010-1119 Found by Ralf Philipp Weinmann & Vincenzo Iozzo on Android 2.1."

This exploit is based on Android/ARM and has been developed following the same kind of memory exploitation techniques used for many other systems. The vulnerability allows remote attackers to execute an arbitrary code, cause a denial of service, or read the SMS database and other data via vectors related to "attribute manipulation."

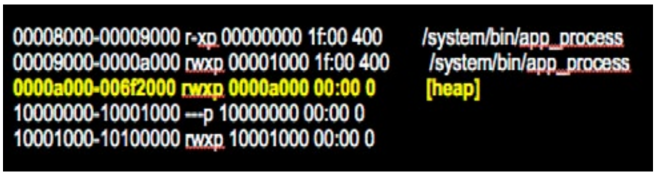
The attack takes advantage of a weakness in *Address Space Layout Randomization* (ASLR). The ASLR is a feature introduced to randomly arrange the positions of key data areas, usually including libraries, heap, and stack, in a process's address space.

It is possible to verify if the Address Space randomization is active in the kernel by using a terminal using Unix commands as follows:

- Make sure you are in the correct location `# pwd ---- >/proc/sys/kernel`
- Display the value inside the file randomize_va_space `# cat randomize _ va _ space ---> 1`

The `randomize _ va _ space` variable can have the values 0 (do not randomize), 1 (randomize stack and vdso page and mmap), and 2 (also randomize brk base address).

Unfortunately, the randomization has been applied only to the stack, leaving the heap executable and unprotected (Figure 4).



```
00008000-00009000 r-xp 00000000 1f:00 400     /system/bin/app_process
00009000-0000a000 rwxp 00001000 1f:00 400      /system/bin/app_process
0000a000-006f2000 rwxp 0000a000 00:00 0       [heap]
10000000-10001000 ---p 10000000 00:00 0
10001000-10100000 rwxp 10001000 00:00 0
```

Figure 4. *Randomization Console*

A simple shellcode used to corrupt the system such as:

```
"\x02\x20\x42\xe0\x1c\x30\x8f\xe2\x04\x30\x8d\xe5\
x08\x20\x8d\xe5\x13\x02\xa0\xe1\x07\x20\xc3\xe5\
x04\x30\x8f\xe2\x04\x10\x8d\xe2\x01\x20\xc3\xe5\
x0b\xff\x90\xef/bin/sh"
```

Prevention strategy attempts

Due to the high number of attacks targeting Android system devices, in February 2012, Google decided to develop new protection measures. A server-side security service called "Bouncer" aimed to provide an automatic malware scanning functionality for all the Android downloadable software in the Marketplace.

Bouncer scans for malicious code and looks for dangerous applications. All the software preventively runs on Google's cloud service, simulating the behavior and scanning for potential threats.

Unfortunately, Bouncer can be easily bypassed as presented at BlackHat by Nicholas J. Percoco Sean Schulte on the research called: "Adventures in BouncerLand Failures of Automated Malware Detection within Mobile Application Markets."

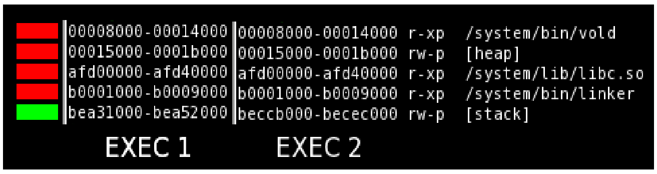
The research has proven how applications that use Javascript bridge cannot be part of trusted environments, because the functionality may allow to bypass automated or manual review process. With this method, an application may turn into malicious even after it has been certified as non-dangerous.

In the past, some of the preventive security mechanisms to protect Android kernel didn't quite fulfill the expectations. Although, recently, we have finally seen some improvements and some good results were reported.

As discussed, ASLR (*Address Space Layout Randomization*) is a memory allocation mitigation control to protect applications and the system (this is protection mechanism is common for many Linux systems).

ASLR aims to mitigate the memory corruption attack by randomizing the memory allocation.

To be fully effective, this technique must be applied to all of the memory areas such as: Exec, Linker, Heap, Stack, etc. A single mapped exe-



```
🟥 |00008000-00014000 |00008000-00014000 r-xp  /system/bin/vold
🟥 |00015000-0001b000 |00015000-0001b000 rw-p  [heap]
🟥 |afd00000-afd40000 |afd00000-afd40000 r-xp  /system/lib/libc.so
🟥 |b0001000-b0009000 |b0001000-b0009000 r-xp  /system/bin/linker
🟩 |bea31000-bea52000 |beccb000-becec000 rw-p  [stack]
        EXEC 1            EXEC 2
```

Figure 5. *Randomization Console II*

cutable static location may be sufficient to build a ROP payload and nullify the mitigation approach.

Android provided this feature since vesion 4.0 within fs/binfmt_elf.c invoked executing ELF binaries.

Dumping a process, for example `/proc/pid/maps`, is possible to verify the randomization process.

The image below (Figure 5) shows that memory areas for Kernel Ice Cream 4.0 are NOT all correctly randomized. Heap and linker are loaded at the same memory location in the address space.

In Android JellyBean 4.1, ASLR mitigation has been effectively improved and randomization properly applied as visible in the image below. (Figure 6).

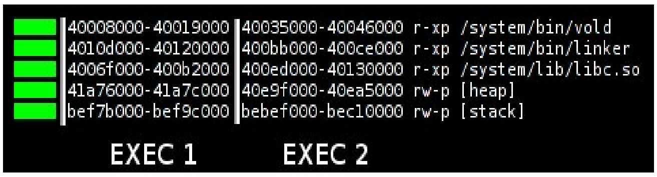
JellyBeans 4.1 brings a number of updates increasing the level of security and enabling features such as:

- immediate binding,
- dmesg_restrict enabled (avoid leaking kernel addresses),
- kptr_restrict enabled (avoid leaking kernel addresses),
- PIE (Position Independent Executable) support read-only relocations.

JellyBeans 4.1 also provides a strengthened Sandbox and a correct UID Isolation, fixing a number of bugs. Any data stored by an application will receive the correct application's user ID. For the creation of a new file with functions `getSharedPreferences(String, int)`, `openFileOutput(String, int)`, or `openOrCreateDatabase(String, int, SQLiteDatabase.CursorFactory)` will be possible to use `MODE_WORLD_READABLE` and/or `MODE_WORLD_WRITEABLE` flags.

When the flags are enabled, the file is still owned by your application, but the global read and/or write permissions are appropriately set, and any other application will detect it ensuring the correct permissions and limiting potential privilege escalations.

ASLR together with DEP (Data Execution Prevention, a mitigation control intended to prevent an application or service from executing code from a non-executable memory region), should increase the overall level of security, forcing the attacker to overcome the obstacles. However, the robustness and resiliency of these 2 mitigations is untested against future attacks.



```
40008000-40019000 | 40035000-40046000 r-xp /system/bin/vold
4010d000-40120000 | 400bb000-400ce000 r-xp /system/bin/linker
4006f000-400b2000 | 400ed000-40130000 r-xp /system/lib/libc.so
41a76000-41a7c000 | 40e9f000-40ea5000 rw-p [heap]
bef7b000-bef9c000 | bebef000-bec10000 rw-p [stack]
```

Figure 6. *Randomization Console III*

Appendix

Materials/Documents/URL's

- *http://web.nvd.nist.gov/view/vuln/statistics*
- *http://www.kaspersky.com/about/news/press/2012/Android_Under_Attack__Malware_Levels_for_Googles_OS_Rise_Threefold_in_Q2_2012*
- *https://android.googlesource.com/*
- *http://media.blackhat.com/bh-us-12/Briefings/Percoco/BH_US_12_Percoco_Adventures_in_Bouncerland_WP.pdf*
- *http://jon.oberheide.org/files/summercon12-bouncer.pdf*
- *https://blog.duosecurity.com/*
- *http://source.android.com/tech/security/index.html#system-and-kernel-level-security*
- *http://lxr.free-electrons.com/source/arch/arm/kernel/entry--common.S?v=2.6.31;a=arm*
- *http://comments.gmane.org/gmane.linux.ports.arm.general/8351*
- *http://en.wikipedia.org/wiki/NOP*
- *http://marakana.com/expert/aleksandar_gargenta,1.html*

Conclusion

Market strategy WINS over security. So far, a system that aims to support the highest number of applications to obtain the larger user satisfaction cannot simultaneously ensure the same higher level of security. Is the storm over?

SEMBIANTE MASSIMILIANO

M.S.c. Computer Security

Employed at UBS Bank as IT Security and Risk Specialist. Collaborating as Research Engineer at R.I.F.E.C. (Research Institute of Forensic and E-Crimes) focusing on: New Virus, Malware Analysis and reverse, Digital Forensic, Sandbox bypass, Shellcoding, Testing Overflows and Exploitation, Code corruption, Testing unexpected behavior, Privilege Escalation, Cryptography, Cryptanalysis, Data infection analysis, new attack vectors, approaches including new tactics and strategies. Defeating protections, intrusion methodologies, polymorphic and intelligent masquerading. Antivirus adaptation and detection avoidance. Development of Tools and scripts.

Contacts

WebSite: www.rifec.com

Email: msembiante@rifec.com

Mobile: +41 79 71 53 205

IMF
INTERNATIONAL MANAGEMENT FORUM

IT Security Courses and Trainings

IMF Academy is specialised in providing business information by means of distance learning courses and trainings. Below you find an overview of our IT security courses and trainings.

Certified ISO27005 Risk Manager

Learn the Best Practices in Information Security Risk Management with ISO 27005 and become Certified ISO 27005 Risk Manager with this 3-day training!

CompTIA Cloud Essentials Professional

This 2-day Cloud Computing in-company training will qualify you for the vendor-neutral international CompTIA Cloud Essentials Professional (CEP) certificate.

Cloud Security (CCSK)

2-day training preparing you for the Certificate of Cloud Security Knowledge (CCSK), the industry's first vendor-independent cloud security certification from the Cloud Security Alliance (CSA).

e-Security

Learn in 9 lessons how to create and implement a best-practice e-security policy!

Information Security Management

Improve every aspect of your information security!

SABSA Foundation

The 5-day SABSA Foundation training provides a thorough coverage of the knowlegde required for the SABSA Foundation level certificate.

SABSA Advanced

The SABSA Advanced trainings will qualify you for the SABSA Practitioner certificate in Risk Assurance & Governance, Service Excellence and/or Architectural Design. You will be awarded with the title SABSA Chartered Practitioner (SCP).

TOGAF 9 and ArchiMate Foundation

After completing this absolutely unique distance learning course and passing the necessary exams, you will receive the TOGAF 9 Foundation (Level 1) and ArchiMate Foundation certificate.

TOGAF

CSA cloud security alliance · CCSK · CompTIA CLOUD ESSENTIALS · ArchiMate® · SABSA

For more information or to request the brochure please visit our website:
http://www.imfacademy.com/partner/hakin9

IMF Academy
info@imfacademy.com
Tel: +31 (0)40 246 02 20
Fax: +31 (0)40 246 00 17

SysFS Little Tutorial

and How to Use it to Trigger Kernel Faults

A few months ago, I got curious about a strange and kind of obscure bug in the Linux Kernel.

This bug threw a *divide by zero* error (see Listing 1) when the Kernel tried to update the average load on a group of CPUs. According to the bug report [0] in the Kernel's bugzilla, and my own experience, this bug presented itself randomly in time. Sometimes it would appear after 6 months uptime, on others, it would show after just 2-3 months, or even after just 4 days. It wasn't even bound to a given machine architecture or type of processor because it had been reported in Intel and AMD processors and even in Amazon EC2 instances. Then, my curiosity kicked in. I needed to learn more about this bug.

Initial approach

Since this bug was random, it was not the best idea to wait for it to occur to see if my so-called solutions worked. Also, even if I could wait for the bug to happen, nothing much could be done after it showed because the Kernel panic (see Listing 1) would just lock the box up.

I needed a mechanism to trigger the fault whenever I wanted, reliably and fast, and most importantly, that the mechanism gave me some kind of control on how to trigger it.

The initial approach was to write up a really simple Kernel module that would export a given symbol, say `crash _ it _ now`, and upon loading this new module, the error would be triggered. But this approach, although simple and effective, since I could now trigger the error whenever I wanted; lacked flexibility. I could trigger the error from only one location in the source files I was studying (initially `kernel/sched _ fair.c`, as hinted by the Kernel panic message), and every time I wanted to check other places the bug might have come from, I would have to go through a cycle of Kernel compilation that, after the fourth time, was not fun at all.

So, another solution was needed to hunt this bug down. I thought that if I could modify some amount of source files, those I suspected might be involved in the division error, compile only once and then,

Listing 1. *Snippet of the Kernel panic stack trace*

```
[...]
Code: 48 f7 f6 49 c1 ee 07 83 7d cc 00 74 1c 48 8b 55 d0 4c 89 a5
RIP  [<ffffffff8008bb03>] find_busiest_group+0x23a/0x621
 RSP <ffff81083616bdb8>
 <0>Kernel panic - not syncing: Fatal exception
 <0>divide error: 0000 [3] SMP
last sysfs file:
[...]
```

somehow, control from *userspace* which modification to activate on a given time, then everything would be more elegant and effective, and the bug hunting would be much, much more fun.

Then SysFS came. This subsystem would allow me to “talk” to my modified Kernel and would allow me to activate my “traps” individually, and at any given moment. I had previously enumerated all the “traps” I had set, and to activate a particular one would be just as easy as writing to a SysFS file the number of the “trap”.

After digging some of the intricacies of SysFS from the Kernel documentation and some other places, I came to the conclusion that you have to be not an expert, but pretty familiar with the Linux Kernel and its vocabulary to write even a short module that exposes a SysFS entry. After messing around with it, I wanted to write about it, and hopefully, make it a little bit simpler to grasp.

What is SysFS?

Mochel and Murphy, in the kernel documentation, define *sysfs* as:

> *... a ram-based filesystem initially based on ramfs. It provides a means to export kernel data structures, their attributes, and the linkages between them to userspace.*

So, sysfs is a mechanism that exposes information about data structures from the kernel in the form of a hierarchical filesystem. The data structures that are mentioned in the above definition for sysfs are called *kobjects*. These *kobjects* are represented in this filesystem as directories, their attributes are represented as files and the relationships between *kobjects* are represented as symlinks.

Avoiding some complexities, when a kobject is registered with sysfs, a new directory is created under `/sys`. Corresponding files will be created inside the directory, according to the attributes assigned to the kobject. These files represent an opportunity for userspace utilities to interact with the kernel space, either by reading from them to obtain specific information from a device, or by writing to them to modify a driver’s behavior. Sysfs files are generally regular files (a.k.a ASCII files), although it is possible to define binary attributes.

There are several kobjects that already export attributes through sysfs. For instance, it is easy to know the current CPU frequency for a given CPU, just by reading from a sysfs file: `/sys/devices/system/cpu/cpuN/cpufreq/cpuinfo_cur_freq`, where `cpuN` is the number of the CPU we want to query. As another example, we can also interact with the power subsystem, by reading/writing to sysfs files under `/sys/power`. Specifically, we could send a machine into a ‘Suspend-to-Disk’ or ‘Suspend-to-RAM’ state, just by writing to `/sys/power/state`.

A simple kernel module

We are now going to delve into the details of building a kernel module, which exposes a sysfs interface. The module we are going to work with, is going be be called `t_sysfs_base`, which will create the following structures in the sysfs hierarchy:

```
/sys
|-- t_sysfs
   |-- level
```

The file level is going to be a regular file, with read-write access so we can query and adjust its value from *userspace*.

Let's start with simple kernel module as a base. There is a chance you have bumped into a bare minimum kernel module before, I’ll show it here anyways.

Listing 2 shows a minimal kernel module. Lines 1 and 2 show the relevant include files to create

Listing 2. *t_sysfs_base.c: Bare minimum kernel module*

```
 1 #include <linux/module.h>
 2 #include <linux/kernel.h>
 3
 4 #define MODVERSION “0.1”
 5
 6 static int __init t_sysfs_base_module_
                  init(void)
 7 {
 8    printk(KERN_DEBUG “INIT CALLED\n”);
 9    return 0;
10 }
11
12 static void __exit t_sysfs_base_module_
                  exit(void)
13 {
14    printk(KERN_DEBUG “EXIT CALLED\n”);
15 }
16
17 module_init(t_sysfs_base_module_init);
18 module_exit(t_sysfs_base_module_exit);
19
20 MODULE_LICENSE(“GPL”);
21 MODULE_AUTHOR(“Jesus Rivero <neurogeek@
                  gentoo.org>”);
22 MODULE_VERSION(MODVERSION);
```

this simple module. Line 4 defines a variable to hold the version of the module. Then we see two functions, t_sysfs_base_module_init and t_sysfs_base_module_exit, in lines 6 and 12 respectively. These functions are to be called from the kernel when we load and unload the module. In our case, we just print debug messages to dmesg.

As you might have guessed by now, the call to module_init (at line 17) establishes which will be the function to initialize the module and module_exit establishes the function to call when destroying the module, so cleanup tasks can be executed before unloading it.

The rest of the calls in the file, are macro helpers to set metadata about the module, like the module's license, author and version.

To build and install the module, we need a Makefile. Listing 3 provides such a Makefile to build t_sysfs_base as an external kernel module.

This Makefile says that we want to build a module from the file t_sysfs_base.c (line 1), using the make tools from the current kernel. The install make target will install the t_sysfs_base.ko kernel module under /lib/modules/<current_kernel>/extra directory. The extra directory is just a convenient location to install externally built modules.

Once we have our module and the Makefile to build it, let's test it:

```
$ make && make install
$ depmod -a
$ modprobe t_sysfs_base
```

After these 3 steps, and if everything went alright, the t_sysfs_base module should be loaded. Running dmesg, you should see a 'INIT CALLED' message. To test the exit function, unload the module:

```
$ rmmod t_sysfs_base
```

Then, you should see a 'EXIT CALLED' message in dmesg. After executing make to build the module, you should see a lot of files in your work directory, such as Module.symvers, modules.order, some intermediate files and, of course, t_sysfs_base.ko which is the actual module we just built. We are going to go back to some of these files in a bit.

We now have a brand new kernel module, but our module is useless, except for educational purposes. Let's add more interesting stuff, say, sysfs stuff.

Adding sysfs goodness

In order to make our module interact with the sysfs subsystem, we need to add several things to our initial kernel module. We need the following things:

- A kobject, with a name and, optionally, attributes.
- Functions to provide callbacks to sysfs operations.
- Register the kobject.

Let's start with the kobject itself. What we are going to do, is go bit by bit and then put everything together (I will leave the line numbers when going over small bits of code, so you can map them to the full code shown in Listing 4). According to Listing 2, a kobject is an object with a *struct kobject* and enclosing ktype. Let's build a kobject:

```
15 struct kobject *t_sysfs_kobj;
```

The ktype of our kobject will be:

```
71 static struct kobj_type t_sysfs_type = {
72    .sysfs_ops = &t_sysfs_ops,
73    .default_attrs = t_sysfs_attrs,
74 };
```

This particular ktype has references to sysfs operations and to the attributes of the kobject. Since sys-

Listing 3. *Makefile: Makefile to build t_sysfs_base as an external module*

```
1 obj-m         += t_sysfs_base.o
2
3
4 all:
5  KBUILD_NOPEDANTIC=1 make -C /lib/modules/`uname -r`/build M=`pwd`
6
7 clean:
8  KBUILD_NOPEDANTIC=1 make -C /lib/modules/`uname -r`/build M=`pwd` clean
9
10 install:
11    install -D -m 755 t_sysfs_base.ko /lib/modules/`uname -r`/extra/t_sysfs_base.ko
```

fs operations depend on the kobject attributes, let's continue with the definition of the attributes. Defining attributes for a kobject is a three-step process. First, we need to create the underlying structure of the attributes. In our case, we will call this t_sysfs_attr:

```
21 struct t_sysfs_attr {
22   struct attribute attr;
23   int value;
24 };
```

This struct embeds an attr variable of type struct attribute, which is defined in linux/sysfs.h and that will allow us to later associate t_sysfs_attr to our kobject. The variable value, is going to be used to store the values written to the sysfs file for our kobject. Now, we are going to add a specific attribute called level. This variable will have t_sysfs_attr as a type and will, eventually, make sysfs create a regular file with a given name, some place under /sys. The code for the attribute level, is as follows:

```
28 static struct t_sysfs_attr level = {
29   .attr.name="level",
30   .attr.mode = 0644,
31   .value = 0,
32 };
```

The attr variable in the t_sysfs_attr type, gives us a way to set some interesting information about our attribute. First, we can specify a name for the file to be created, which in our case will be the same name of the attribute (level). We can also specify a file mode and an initial value. In our example, we have an initial value of 0 (given by .value=0, from line 23) and a file mode of 0644, which will allow us to read from and write to the sysfs file.

To close the kobject attribute step, we need to list all the attributes we have defined for our kobject in an array of struct attributes, which we'll then feed to the ktype (see line 73). This array can be built in the following way:

```
36 static struct attribute *t_sysfs_attrs[] = {
37   &level.attr,
38   NULL
39 };
```

Here, we create a NULL-terminated array, listing the addresses of the attributes we defined for our kobject. In our case, we only have one, level, but if you want more attributes, you can follow the instructions for level, and properly list them in t_sysfs_attrs.

Now that we took care of the attributes, let's now take care of the sysfs operations. We can define two operations, store and show. As you might have guessed, store let's us grab the value being written to the sysfs file (e.g. echo 1 > /sys/<...>/level), while the show is an operation to show the current value of the attribute (e.g. cat /sys/<...>/level).

Our show operation will be called show_level and will be the following:

```
43 static ssize_t show_level(struct kobject *kobj,
   struct attribute *s_attr,
44      char *buf)
45 {
46   struct t_sysfs_attr *a = container_of(s_attr,
     struct t_sysfs_attr, attr);
47   return scnprintf(buf, PAGE_SIZE, "%d\n",
     a->value);
48 }
```

And our store operation will be called store_level and will be as follows:

```
52 static ssize_t store_level(struct kobject
  *kobj, struct attribute *s_attr,
53      const char *buf, size_t len)
54 {
55   struct t_sysfs_attr *a = container_of(s_attr,
     struct t_sysfs_attr, attr);
56   sscanf(buf, "%d", &a->value);
57   t_sysfs_level = a->value;
58 
59   return sizeof(int);
60 }
```

Note that these are the expected signatures for both sysfs operations. Sysfs has the task to call these when some processes reads or writes to the level file. *kobj is a pointer to a t_sysfs_kobj structure, *s_attr is a pointer to the actual attribute being accessed, while *buf* and *len* are pointer for return values.

To access the underlying attribute, we need to use, in both store and show, the container_of function, defined at linux/kernel.h. This function allows us to extract the actual t_sysfs_attr, so we can access the value variable we defined in line 31.

To close the sysfs operations section, we have to create a sysfs_ops type, that sets up the store and show functions accordingly:

```
63 static struct sysfs_ops t_sysfs_ops = {
64   .show = show_level,
65   .store = store_level,
66 };
```

Listing 4a. *t_sysfs_base.c: Complete module source code*

```
#include <linux/sysfs.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/slab.h>

// Module version information.
#define MODVERSION "0.0.1"

/**
 * Global variable to store t_sysfs_level and
 * t_sysfs KObject
 */
static int t_sysfs_level;
struct kobject *t_sysfs_kobj;

/**
 * Define attributes for t_sysfs KObject.
 * struct attribute is defined in linux/sysfs.h
 */
struct t_sysfs_attr {
  struct attribute attr;
  int value;
};

// Actual specification of the attribute, like
// initial value, file name and permissions.
static struct t_sysfs_attr level = {
   .attr.name="level",
   .attr.mode = 0644,
   .value = 0,
};

// NULL terminated array of t_sysfs_attr's. If we
// had more than one, they all would be listed here.
static struct attribute *t_sysfs_attrs[] = {
   &level.attr,
   NULL
};

// Function for sysfs. Shows stored value of level. This will be called
// when a reading operation is executed on a sysfs file.
static ssize_t show_level(struct kobject *kobj, struct attribute *s_attr,
         char *buf)
{
   struct t_sysfs_attr *a = container_of(s_attr, struct t_sysfs_attr, attr);
   return scnprintf(buf, PAGE_SIZE, "%d\n", a->value);
}

// Function for sysfs. Stores a new value for attribute level. This will be
called
// when writing to a sysfs file.
static ssize_t store_level(struct kobject *kobj, struct attribute *s_attr,
        const char *buf, size_t len)
{
   struct t_sysfs_attr *a = container_of(s_attr, struct t_sysfs_attr, attr);
   sscanf(buf, "%d", &a->value);
```

Listing 4b. *t_sysfs_base.c: Complete module source code*

```
  t _ sysfs _ level = a->value;
  printk("Setting value to %d\n", t_sysfs_level);
  return sizeof(int);
}

// Define what functions will be called for .show and .store operations.
static struct sysfs_ops t_sysfs_ops = {
  .show = show_level,
  .store = store_level,
};

// The type of the KObject. All KObjects have a type. The type defines
// what the attrubutes of the kobject are, and what operations are defined
// for those attributes, via sysfs.
static struct kobj_type t_sysfs_type = {
  .sysfs_ops = &t_sysfs_ops,
  .default_attrs = t_sysfs_attrs,
};

// Module initialization routine
static int __init t_sysfs_module_init(void)
{
  int err = -1;
  t_sysfs_kobj = kzalloc(sizeof(*t_sysfs_kobj), GFP_KERNEL);
  if (t_sysfs_kobj) {
        kobject_init(t_sysfs_kobj, &t_sysfs_type);
        if (kobject_add(t_sysfs_kobj, NULL, "%s", "t_sysfs_base")) {
              err = -1;
              printk("Could not add SysFS module\n");
              kobject_put(t_sysfs_kobj);
              t_sysfs_kobj = NULL;
        }
        err = 0;
  }
  return err;
}

// Module exit routine
static void __exit t_sysfs_module_exit(void)
{
  if (t_sysfs_kobj) {
        kobject_put(t_sysfs_kobj);
        kfree(t_sysfs_kobj);
  }
}

//Specify module_init and module_exit functions.
module_init(t_sysfs_module_init);
module_exit(t_sysfs_module_exit);

//Module information and exported symbols.
EXPORT_SYMBOL_GPL(t_sysfs_level);
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Jesus Rivero <neurogeek@gentoo.org>");
MODULE_VERSION(MODVERSION);
```

You will recognize `t_sysfs_ops` from the ktype definition at line 72.

Our kobject definition, with attributes and sysfs operations is now complete. Now we need to initialize it and test everything. And what ia a better place to initialize our kobject than in our module_init function?

```
82        kobject_init(t_sysfs_kobj, &t_sysfs_type);
```

Line 82 initializes a kobject. Set up the kobject we defined at line 15 and associate it with the ktype we defined at line 71. One more thing before we can continue: we need to add the kobject to the kobject hierarchy, so the kernel can track it. This is done here:

```
83        if (kobject_add(t_sysfs_kobj, NULL, "%s",
          "t_sysfs_base")) {
```

The NULL in the second argument to `kobject_add`, means that this particular kobject does not have a parent kobject. Sysfs-wise, this means that this kobject sysfs directory will be shown directly inside `/sys`, like this:

```
- /sys/t_sysfs_base
```

The 4th argument to `kobject_add`, specifies the name of the kobject and it is also the name of the kobject related directory in sysfs.

It would be easy enough to give our kobject a parent, we just need a reference to the kobject we want and substitute NULL with the kobject symbols. As an example, let's give `t_sysfs_base` a parent. Let's suppose we want `t_sysfs_base` to show inside `/sys/fs`. This would mean that fs has to be `t_sysfs_base` parent. In order to do this, we need fs' kobject symbol and use it instead of NULL in the call to `kobject_add`.

The symbol for fs kobject is `fs_kobj` and is declared in `linux/fs.h`.

So, include `<linux/fs.h>` and replace NULL with `fs_kobj` in `kobject_add`, compile, install and modprobe your new module. You will note that `t_sysfs_base` sysfs entry will be under `/sys/fs`.

Now that we have the complete source code for the module, compile it and install it following the steps given above, and load it using modprobe:

```
$ modprobe t_sysfs_base
```

We should be able to read abd write to `/sys/t_sysfs_base/level`, and the initial value for the file should be 0.

What happens now?

Tying my initial problem with what we have talked about sysfs, I could use the value of `/sys/t_sysfs_base/level` to activate a given "trap" in my modified kernel. So how can we read the value I wrote to the level file from other locations or kernel modules?

The answer is simple. We could use a variable to store the current value of the level attribute and then export the variable as a kernel symbol, so we can access it from different parts of the kernel ecosystem.

The initial step is to declare the variable, which we do in line 14 of Listing 4. Then, we add a little modification to our `store_level sysfs` operation to also store the value written to the level file, to the variable `t_sysfs_level`. Now, we need to export the variable using the `EXPORT_SYMBOL_GPL` macro (we use the GPL macro since we specified our module is GPL with the `MODULE_LICENSE("GPL")` call).

After these modifications to the code, if we call make once again in our working directory, we can now see that the file `Module.symvers` is no longer empty. This is because of the call to the EXPORT macro, the exported symbol (`t_sysfs_level`) is listed there now.

If we want to use `t_sysfs_level` from outside the module where it was defined in, we need one more thing. We need to create an include file so the type of the variable is known to the callers. Let's create a file called `t_sysfs_base.h` in your working directory, with the following contents:

```
#ifndef T_SYSFS_BASE_H
#define T_SYSFS_BASE_H
extern int t_sysfs_level;
#endif
```

As an example on how to access this variable from other modules, we are going to actually create a new module to access `t_sysfs_level` from it. In order to do this, let's use the same bare kernel module from Listing 2 and name it `t_test_read_mod.c`. Replace the module_init function contents with the following:

```
// put this at the beggining
#include "t_sysfs_base.h"

// now replace printk(KERN_DEBUG "INIT CALLED");
   in the module init function with
// the following
require_module("t_sysfs_base");
printk(KERN_DEBUG "t_sysfs_level is: $d\n");
```

Modify the Makefile for the bare module to reflect the changes made here (basically, the name of the module). Once done, and before calling make to build our new module, we need to copy `Module.symvers` from `t_sysfs_base` to the current working directory, so the reference to the `t_sysfs_level` variable can be picked up by make and MODPOST.

Once you copied `Module.symvers` over, we can build and install our new module. The new module, `t_test_read_mod`, will have a dependency on `t_sysfs_base`.

Testing the whole thing

Now let's test our modules. For starters, load the first module:

```
$ modprobe t_sysfs_base
```

Now modify the value for level, under `/sys/t_sysfs/level` with:

```
$ echo 10 > /sys/t_sysfs/level
```

This will set our kobject level attribute value to 10. You can verify the value stored by `cat'ing` the file. If w `"t_sysfs_level is:e`, now load the `t_test_read_mod` module, we will see:

```
t_sysfs_level is: 10
```

printed out in dmesg.

Conclusion

Debugging the Linux Kernel can be a really complex task, so we better use all the tools available to make it a little easier. In this case, we used the sysfs subsystem the create a mechanism to facilitate the activation of custom kernel "traps" to make a bug appear at any given time.

Sysfs is a subsystem to export information about kernel objects, their attributes and relationships, so processes from userspace can interact with them. In this article we learned how to expose certain attributes from a kobject using sysfs operations and how to read and access those values from other parts of the kernel, including other externally built modules.

I hope this little guide helps you advance in your progress and will also help you understand the amazing world of the Linux Kernel better.

Happy Hacking

JESUS RIVERO

15 Percent of Malware

Still Compatible with Windows 8, Test Reveals

The introduction of Windows 8 marks an important milestone in more than 30 years of operating system development for US vendor Microsoft. The new operating system boasts a major overhaul in terms of visuals with the introduction of the Advanced UI, as well as massive changes of the security subsystems that ship with Windows 8.

A couple of days after the official release we took Windows 8 to a spin to determine how much of the malware that runs on Windows 7 also affects the new operating system.

Testing Methodology

Step 1

In order to carry the test, we used three identical physical machines running stock configurations of Windows 7 Windows 8 without AV and Windows 8 with Windows Defender respectively, booted from a network server.

Step 2

After running a malicious sample and assessing whether the computer has been compromised or not, the system is rebooted to a clean operating system and testing resumes. It is assumed that the piece of malware has successfully infected the PC when it has spawned its own process and kept that process running until reboot.

Step 3 – Testing on Windows 7, Windows 8 and Windows 8 with Windows Defender

The malware test on Windows 8 was carried in two steps, as follows:

- In order to ensure that both Windows 7 and Windows 8 environments are on par, we disabled the anti-malware solution that ships by default with Windows 8 in the first test.
- The second test was a real-life scenario, with Windows 7 versus Windows 8 + Windows Defender.

Step 4

The malicious sample set was built of 380 samples of the most popular 100 families of malware in the past six months, as reported by the Bitdefender Real-Time Virus Reporting System. These samples were hosted on an internal FTP repository and copied to the machine after booting it up.

Step 5

After running the sample in the selected environment, the python script emails a detailed report with the process differences between the original system and the infected one.

All samples have been run in vanilla Windows environments with User Account Control set to ON. If any of the sample's operations resulted in UAC prompts, we consider that the sample *did not achieve its goal* and the computer was not harmed.

Test Results

The comparative tests between Windows 7 and Windows 8 with the bundled antivirus disabled were not notable at all. From a pool of 385 samples, 234 e-threats have run successfully, created one or more processes and achieved persistence until reboot. 138 samples could not be started on the machine on various reasons – this means that the machine's state did not change from the initial, clean boot. 6 e-threats (generic Trojans) executed but crashed before performing any change to the PC, and 7 others have launched but their payload was blocked by UAC.

The situation is much brighter when Windows 8 is paired with an antivirus. The test ran on Win-

Table 1. *Running Windows 8 with the default security software - analysis*

	Windows 7	Windows 8	Windows 8 + Defender
Total samples	385	385	385
Successfully run	262	234	61

dows 8 with Windows Defender activated revealed that only 61 samples were able to infect the PC, while 322 were immediately deleted on copy by Windows Defender. Two other samples that bypassed Windows Defender crashed on execution and were blocked by User Account Control, respectively. However, even when running Windows 8 with the default security software, 61 pieces of malware still managed to subvert the system.

Accuracy of the Test

The comparative test has been carried in a controlled environment using samples picked by popularity criteria and run in a pristine environment with no additional software installed. In the case of exploits that rely on the presence of third party applications such as Adobe Reader or Microsoft Word, failure to infect is guaranteed. Therefore, the rates of successful infection might slightly be higher than presented in this article.

Conclusions

When it comes to user-mode malware, in the absence of a security solution Windows 8 makes little difference in terms of safety as compared to Windows 7. While it may be true that security subsystems such as ELAM and SafeBoot may block rootkit-based threats, out of 100 families of malware, only three samples were built around rootkits.

Always remember that, although Microsoft has made huge leaps in improving the overall security of their newest operating system, it's the antivirus that ultimately makes the difference.

BOGDAN BOTEZATU

Bogdan Botezatu has been working with Bitdefender for about 5 years in the e-threat analysis field. With a strong background in computer networking, Bogdan has activated as system administrator at the Alexandru Ioan Cuza University of Iași, where he supervised the implementation of IT functions in the educational sector for non-IT-related specialties. He is the author of "Malware History", an overview of the most notable developments in the malware landscape.

Security Policy Development in

Trusted BSD MAC Framework

Trusted Operating Systems are the "Next Level" of system security. They offer both new security features and high assurance of successful implementation. Trusted systems differ from secure systems in many principles.

Trusted Systems established the concept of "ranking" systems within different degrees of trustworthiness. In such systems, users decide on trustworthiness and make a judgment based on systems. Operating systems have to implement security policies and different mechanisms are used to enforce such policies. There are various operating system security policies such as MLS and Biba policies. In this article we will describe the overall process of developing and applying different security policies within FreeBSD kernel under the TrustedBSD MAC Security Framework.

Introduction

A kernel is a central component of an operating system. It acts as an interface between the user applications and the hardware. The sole aim of the kernel is to manage the communication between the software (user level applications) and the hardware (CPU, disk memory, etc). The main tasks of the kernel are: Process Management, Device Management, Memory Management, Interrupt Handling, I/O Communication, and File System. New kernel structures arose consisting of several modules classified into base-kernel modules and dynamic "pluggable" kernel modules. The main advantage of dynamic kernel modules is the ability to be attached through run-time. The dynamic kernel module behavior differs from the base kernel module from the flexibility point of view. In base kernel, all modules should be persistent by the kernel compilation time while it can be available only at the run-time in the dynamic kernel module case. Furthermore, the security and performance tradeoffs are formulating the best approach to use in kernel compilation. On the other hand, different security models and policies were implemented in OS kernels to add many security features to the "plain" kernels. From operating system point of view, security policies are the restrictions that administrators would like to apply while mechanisms are the procedures used to enforce such policies. There are various operating system security policies, such as MLS, and Biba policies.

Security Policies Problems

The variety and non-standardization in the security policies introduced some conflicts between implementations of these policies. Different vendor implementations and security policies intentions caused big headache for the security developers. In addition, the frequent change in user requirements and the need for policies customization were pushing to find a new security methodology to cover these issues. In the past, kernels were adapted to one security model at most. Furthermore, we had to include the security policy in the kernel configuration file before compiling the kernel and this inflexibility in embedding security policies in the kernel was a huge trouble. The need for policies customization and dynamic attachment for different policies led the researchers to introduce new concept of OS security techniques. This technique focuses on implementing an intermediate layer between the security models and the kernel services. This intermediate layer offers policy composition manipulation and customization besides adding the capability to attach different security policies at the run-time. One of the most common security frameworks implementing the mentioned points is MAC framework. The MAC framework provides a set of wrappers for usage by different policies' vendors. The following figure describes the overall architecture of MAC framework: Figure 1.

MAC Security Framework in detail

Two of the most significant new security mechanisms are file system *Access Control Lists* (ACLs) and *Mandatory Access Control* (MAC) facilities. Mandatory Access Control allows new access control modules to be loaded, implementing new security policies. Some provide protections of a narrow subset of the system, hardening a particular service. Others provide comprehensive labeled security across all subjects and objects. The mandatory part of the definition comes from the fact that the enforcement of the controls is done by administrators and the system, and is not left up to the discretion of users as is done with discretionary access control (DAC, the standard file and System V IPC permissions on FreeBSD). One of the most common implementation of the MAC Security is the TrustedBSD MAC Framework. The TrustedBSD MAC framework provides a mechanism to allow the compile-time or run-time extension of the kernel access control model. New system policies may be implemented as kernel modules and linked to the kernel; if multiple policy modules are present, their results will be composed. The MAC Framework provides a variety of access control infrastructure services to assist policy writers, including support for transient and persistent policy-agnostic object security labels.

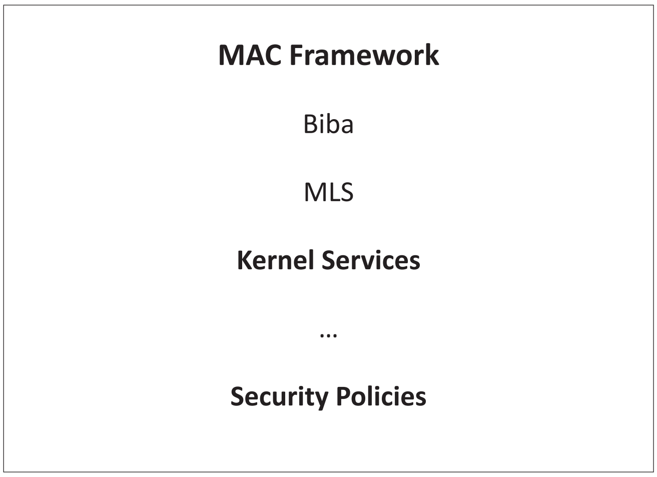


Figure 1. *MAC Framework Overall Architecture*

Detailed Architecture

MAC Framework Interfaces for Kernel Services

The MAC Framework presents a set of entry points to selected kernel services, permitting the services to provide event notification to the MAC framework, and providing the ability for the MAC Framework to maintain a security label within kernel objects maintained by the kernel services. The interface used by FreeBSD kernel services to communicate with the MAC Framework is defined in `sys/mac.h`. This includes the APIs for all entry points from the kernel services. In addition, `sys/label.h` defines struct label, a data structure used to store policy-agnostic label data in kernel objects. This structure is embedded into many kernel service structures.

Framework Kernel Service Entry Points

Modifications have been made to kernel services to invoke MAC Framework entry points. These modifications affect object initialization, association/creation, and destruction, as well as in common paths requiring access control at high levels in the kernel. With layered services, it is often necessary to defer access control decisions until enough information is available.

Framework Implementation

Entry point implementations, label primitives, policy registration, and user/kernel APIs are centralized in `kern _ mac.c`.

Framework Interface for Policies

The MAC Framework provides several interfaces to security policy implementations, including interfaces for policy management, label storage, process label management, object life cycle, access control, and system life cycle. Extensions implement arbitrary subsets of the available interfaces, allowing implementers to select the events and services that are relevant to a particular policy. Interfaces common to the framework and policies and defined in sys/mac policy.h. Definitions include entry point and registration interfaces, as well as common access methods for MAC Framework services.

Policy Implementations

Each policy is represented by one kernel module, discouraging inter-dependency. Typical policies are implemented in a single C file, but complex policies are implemented over many files. Interfaces to User Processes. Interfaces for user processes are defined in `sys/mac.h`, implemented in libc, and may be dynamically linked into any applications.

Main Idea

The main idea behind the TrustedBSD MAC framework is labeling different kernel objects to provide the ability to track them. A label is a security attribute which can be applied to files, directories, or other items in the system. It could be considered a confidentiality stamp; when a label is placed on a file it describes the security properties for that specific file and will only permit access by files, users, resources, etc. with a similar security setting. The meaning and interpretation of label values depends on the policy configuration: while some policies might treat a label as representing the integrity or secrecy of an object, other policies might use labels to hold rules for access. The following table shows the basic kernel elements that are labeled for kernel elements: Table 1.

Now, let's move to the practical part. We will describe how to use TrustedBSD MAC Framework to secure FreeBSD systems.

Adding MAC support to the kernel

Kernels should have MAC support to give the flexibility of implementing and composing security policies. To achieve this, please add mac option to the default kernel configuration file `/usr/src/sys/conf/GENERIC`

Table 1. *Labeled Objects*

Structure	Description
struct ucred	Process credential
struct vnode	VFS node
struct socket	BSD IPC socket
struct pipe	IPC pipe
struct mbuf	In-flight datagram
struct mount	File system mount
struct ifnet	Network interface
struct devfs_dirent	Devfs entry
struct ipq	IP fragment queue
struct bpf_desc	BPF packet sniff device

```
options MAC
```

Next, recompile the kernel to rephrase the kernel for MAC support.

- Change to the `/usr/src` directory:

```
# cd /usr/src
```

- Compile the kernel:

```
# make buildkernel KERNCONF=MYKERNEL
```

- Install the new kernel:

```
# make installkernel KERNCONF=MYKERNEL
```

Embedding MAC Policy

Security policies are either linked directly into the kernel, or compiled into loadable kernel modules that may be loaded at boot, or dynamically using the module loading system calls at runtime.

Policy Declaration

Modules may be declared using the `MAC_POLICY_SET()` macro, which names the policy, provides a reference to the MAC entry point vector, provides load-time flags determining how the policy framework should handle the policy, and optionally requests the allocation of label state by the framework (Listing 1).

The MAC policy entry point vector, `mac_policy_ops` in this example, associates functions defined in the module with specific entry points. Of specific interest during module registration are the `.mpo_destroy` and `.mpo_init` entry points. `.mpo_init` will be invoked once a policy is successfully registered with the module framework but prior to any other entry points becoming active. This permits the policy to perform any pol-

Listing 1. *The MAC Policy*

```
static struct mac_policy_ops mac_policy_ops =
{
        .mpo_destroy = mac_policy_destroy,
        .mpo_init = mac_policy_init,
        .mpo_init_bpfdesc_label = mac_policy_init_bpfdesc_label,
        .mpo_init_cred_label = mac_policy_init_label,
/* ... */
        .mpo_check_vnode_setutimes = mac_policy_check_vnode_setutimes,
        .mpo_check_vnode_stat = mac_policy_check_vnode_stat,
        .mpo_check_vnode_write = mac_policy_check_vnode_write,
};
```

icy-specific allocation and initialization, such as initialization of any data or locks. `.mpo _ destroy` will be invoked when a policy module is unloaded to permit releasing of any allocated memory and destruction of locks. Currently, these two entry points are invoked with the MAC policy list mutex held to prevent any other entry points from being invoked: this will be changed, but in the meantime, policies should be careful about what kernel primitives they invoke so as to avoid lock ordering or sleeping problems.

The policy declaration's module name field exists so that the module may be uniquely identified for the purposes of module dependencies. An appropriate string should be selected. The full string name of the policy is displayed to the user via kernel log during loading and unloading events, and also exported when providing status information to user processes.

Label Configuration

Virtually all aspects of label policy module configuration will be performed using the base system utilities. These commands provide a simple interface for object or subject configuration or the manipulation and verification of the configuration.

All configuration may be done by the use of the `setfmac(8)` and `setpmac(8)` utilities. The `setfmac` command is used to set MAC labels on system objects while the `setpmac` command is used to set the labels on system subjects. Observe:

```
# setfmac biba/high test
```

If no errors occurred with the command above, a prompt will be returned. The only time these commands are not quiescent is when an error occurred; similarly to the `chmod(1)` and `chown(8)` commands. In some cases this error may be a `“Permission  denied”` and is usually obtained when the label is being set or modified on an object which is restricted. The system administrator may use the following commands to overcome this:

Table 2. *MAC Labels*

Policy	Description
mac_biba	Hierarchal fixed-label integrity
mac_ bsdextended	“File system firewall” using existing credentials/permissions
mac_ifoff	Interface silencing
mac_lomac	Hierarchal floating-label integrity
mac_mls	Multi-Level Security with compartments
mac_none	Prototype stub policy
mac_partition	Inter-process visibility policy based on process partition labels
mac_seeotheruids	Inter-process visibility policy based on existing credentials.
mac_test	MAC Framework invariant tests
sebsd	Port of the SELinux/FLASK/TE

```
# setfmac biba/high test
“Permission denied”
# setpmac biba/low setfmac biba/high test
# getfmac test
test: biba/high
```

Two types of labels are available: singlelabel and multilabel. By default, all the labels are singlelabel items .The `multilabel` option will permit each subject or object to have its own independent MAC label in place of the standard `singlelabel` option which will allow only one label throughout the partition. The `multilabel` and `singlelabel` options are required only for the policies which implement the labeling feature, including the Biba, Lomac, MLS and SEBSD policies. The multilabel is required when we have different policies implemented to set a policy for each labeled object. To set multilabel on the filesystem:

```
# tunefs -l enable /
```

Policy Configuration

The following table shows set of predefined policies for TrustedBSD MAC for usage instead of creating new ones (Table 2).

Case Study

In this example, we will simulate typical MAC Security scenario using Nagios, one of the most common application in the field of IT infrastructure monitoring. Before beginning this process, the `multilabel` option must be set on each file system. Not doing this will result in errors. While at it, ensure that the `net-mngt/nagios-plugins`, `net-mngt/nagios`, and `www/apache22` ports are all installed, configured, and that theyare working correctly.

Create an insecure User Class

Begin the procedure by adding the following user class to the `/etc/login.conf` file: Listing 2.

Finally, add the following line to the default user class:

```
:label=biba/high:
```

Once this is completed, the following command must be issued to rebuild the database:

```
# cap_mkdb /etc/login.conf
```

Boot Configuration

Next, we have to adapt boot configurations to load the MAC policies at boot time. Add the following lines to `/boot/loader.conf` so the required modules will load during system initialization:

```
mac_biba_load="YES"
mac_seeotheruids_load="YES"
```

Configure Users

Set the `root` user to the default class using:

```
# pw usermod root -L default
```

All user accounts that are not `root` or system users will now require a login class. Otherwise, users will be refused access to common commands such as `vi`. The following `sh` script should do the trick:

```
# for x in `awk -F: '($3 >= 1001) && ($3 != 65534)
                  { print $1 }' \
/etc/passwd`; do pw usermod $x -L default; done;
```

Drop the `nagios` and `www` users into the insecure class:

```
# pw usermod nagios -L insecure
# pw usermod www -L insecure
```

Create the Contexts File

A contexts file should now be created; the following example file should be placed in `/etc/policy.contexts` (Listing 3).

This policy will enforce security by setting restrictions on the flow of information. In this specific configuration, users, root and others, should never be

Listing 2. *Create the Contexts File - Introduction*

```
insecure:\
:copyright=/etc/COPYRIGHT:\
:welcome=/etc/motd:\
:setenv=MAIL=/var/mail/$,BLOCKSIZE=K:\
:path=~/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/
                  local/sbin:/usr/local/bin
:manpath=/usr/share/man /usr/local/man:\
:nologin=/usr/sbin/nologin:\
:cputime=1h30m:\
:datasize=8M:\
:vmemoryuse=100M:\
:stacksize=2M:\
:memorylocked=4M:\
:memoryuse=8M:\
:filesize=8M:\
:coredumpsize=8M:\
:openfiles=24:\
:maxproc=32:\
:priority=0:\
:requirehome:\
:passwordtime=91d:\
:umask=022:\
:ignoretime@:\
:label=biba/10(10-10):
```

Listing 3. *Create the Contexts File*

```
# This is the default BIBA policy for this
                  system.
# System:
/var/run                          biba/equal
/var/run/*                        biba/equal

/dev                              biba/equal
/dev/*                            biba/equal

/var           biba/equal
/var/spool                        biba/equal
/var/spool/*                      biba/equal

/var/log                          biba/equal
/var/log/*                        biba/equal

/tmp           biba/equal
/tmp/*            biba/equal
/var/tmp       biba/equal
/var/tmp/*        biba/equal

/var/spool/mqueue    biba/equal
/var/spool/clientmqueue    biba/equal

# For Nagios:
/usr/local/etc/nagios
/usr/local/etc/nagios/*           biba/10

/var/spool/nagios                 biba/10
/var/spool/nagios/*               biba/10

# For apache
/usr/local/etc/apache             biba/10
/usr/local/etc/apache/*           biba/10
```

allowed to access *Nagios*. Configuration files and processes that are a part of *Nagios* will be completely self contained or jailed.

This file may now be read into our system by issuing the following command:

```
# setfsmac -ef /etc/policy.contexts /
# setfsmac -ef /etc/policy.contexts /
```

Note

The above file system layout may be different depending on environment; however, it must be run on every single file system.

The `/etc/mac.conf` file requires the following modifications in the main section:

```
default_labels file ?biba
default_labels ifnet ?biba
default_labels process ?biba
default_labels socket ?biba
```

Enable Networking

Add the following line to `/boot/loader.conf`:

```
security.mac.biba.trust_all_interfaces=1
```

And the following to the network card configuration stored in `rc.conf`. If the primary Internet configuration is done via DHCP, this may need to be configured manually after every system boot:

```
maclabel biba/equal
```

Testing the Configuration

Ensure that the web server and *Nagios* will not be started on system initialization, and reboot. Ensure the `root` user cannot access any of the files in the *Nagios* configuration directory. If root can issue an `ls` command on `/var/spool/nagios`, then something is wrong. Otherwise a "permission denied" error should be returned.

If all seems well, *Nagios*, *Apache*, and *Sendmail* can now be started in a way fitting of the security policy. The following commands will make this happen:

```
# cd /etc/mail && make stop && \
setpmac biba/equal make start && setpmac biba/10\
(10-10\) apachectl start && \
setpmac biba/10\(10-10\) /usr/local/etc/rc.d/
nagios.sh forcestart
```

Finally, check the log files or error messages to make sure everything is fine. Use the sysctl utility to disable the `mac_biba` security policy module enforcement and try starting everything again, like normally.

References

- FreeBSD Handbook *http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/*
- TrustedBSD MAC Project *http://www.trustedbsd.org/mac.html*

Related Work

One of the most common projects running for securing Linux systems is the Security Enhanced Linux which is a set of patches to the Linux kernel and some utilities to incorporate a strong, flexible *mandatory access control* (MAC) architecture into the major subsystems of the kernel.

MOHAMED FARAG

Mohamed Farag is post graduate student at Maharishi University of Management in USA. In the first six months of 2012, Mohamed worked as teaching assistant in Maharishi University of Management. Also, he worked as instructor in Ain Shams University in Egypt during the year 2011. In 2010, Mohamed received Google Summer of Code award and was honored by the scientific community in Menoufia University in Egypt. In addition, Mohamed received "The Best Programming Project" award in Egyptian Universities Summit in 2010 and the same award in 2011. Mohamed has been an active contributor in FreeBSD community since May, 2010 and has led ArabBSD project since June, 2011. In 2012, Mohamed was selected to join the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering "ICST", International Association of Computer Science and Information Technology "IACSIT", Computer Science Teachers Association "CSTA ACM" and Academy & Industry Research Collaboration Center "AIRCC". In September 2012, Mohamed was selected to be reviewer for International Journal of Computer Science and Information Technology "IJCSIT" and program committee in AIRCC. Mohamed has published one paper in IJCSIT in August, 2012.

Integration of

Cyberwarfare and Cyberdeterrence Strategies into the U.S. CONOPS Plan to Maximize Responsible Control and Effectiveness by the U. S. National Command Authorities

This paper deals with issues related to the present situation of lack of a clearly defined national policy on the use of cyberweapons and cyberdeterrence, as well as the urgent present need to include strategies and tactics for cyberwarfare and cyberdeterrence into the national CONOPS Plan, which is the national strategic war plan for the United States.

One of the main disadvantages of the hyperconnected world of the 21st century is the very real danger that countries, organizations, and people who use networked computer resources connected to the Internet face because they are at risk of cyberattacks that could result in one or more cyber threat dangers such as denial of service, espionage, theft of confidential data, destruction of data, and/or destruction of systems and services. As a result of these cyber threats, the national leaders and military of most modern countries have now recognized the potential for cyberattacks and cyberwar is very real and many are hoping to counter these threats with modern technological tools using strategies and tactics under a framework of cyberdeterrence, with which they can deter the potential attacks associated with cyberwarfare.

Nature of the Threat

During my studies prior to and as a student in this DET 630 – Cyberwarfare and Cyberdeterrence course at Bellevue University, it occurred to me that considering the rapid evolution of the potentially destructive capabilities of cyberweapons and the complex nature of cyberdeterrence in the 21st century, it is now a critical priority to integrate the cyberwarfare and cyberdeterrence plans into the CONOPS plan. Indeed, if the strategic battleground of the 21st century has now expanded to include cyberspace, and the U.S. has in the last five years ramped up major military commands, training, personnel, and capabilities to support cyberwarfare and cyberdeterrence capabilities, the inclusion of these capabilities should now be a critical priority of the Obama administration if has not already happened.

How large a problem is this for the United States?

Without the integration of cyberwarfare and cyberdeterrence technologies, strategies, and tactics into the CONOPS Plan, the national command authorities run a grave risk of conducting a poorly planned offensive cyberwarfare operation that could precipitate a global crisis, impair relationships with its allies, and potentially unleash a whole host of unintended negative and potentially catastrophic consequences. In non-military terms, at least four notable cyberspace events caused widespread damages via the Internet because of the rapid speed of their propagation, and their apparently ruthless and indiscriminant selection of vulnerable targets. They are 1) the Robert Morris worm (U.S. origin, 1988); 2) the ILOVEYOU worm (Philippines origin, 2000); the Code Red worm (U.S. origin, 2001); and the SQL Slammer worm (U.S. origin, 2003). If not executed with great care and forethought, a cyberweapons could potentially unleash even greater damage on intended targets and possible on unintended targets that were connected via the Internet.

Other Not So Obvious Challenges for Cyberweapons and Cyberdeterrence

The cyberspace threat and vulnerability landscape is notable in that it is continually dynamic

◀ ▶

and shifting. Those who are responsible for protecting assets in cyberspace have many more challenges on their hands than their military counterparts who utilize weapons like guns, explosives, artillery, missiles, etc. For example, there are by some estimates over 350 new types of malware that are manufactured each month. There are also monthly patch updates to most Microsoft software and operating systems, and phenomena such as evil hackers and zero-day exploits are apparently never ending. Therefore, the inclusion of cyberweapons and cyberdeterrence capabilities into the CONOPS Plan would require more frequent, rigorous, complex, and integrated testing to ensure that it was always effective and up to date. In the dynamic world of cyberspace with its constantly shifting landscape of new capabilities, threats and vulnerabilities, the coordination of the constant refresh and testing of a CONOPS Plan that integrated these cyberwarfare and cyberdeterrence capabilities would be no small feat. In addition, constant intelligence gathering and reconnaissance would need to be performed on suspected enemies to ensure that our cyberweapons and cyberdeterrence capabilities would be in constant state of being able to deliver the intended effects for which they were designed.

Is it a problem for other countries?

The careful planning and integration of cyberweapons and cyberdeterrence is likely a challenge for every country with these capabilities. For example, much is already known about our potential adversaries, such as Russia, China and North Korea, but what is perhaps less understood is the degree to which they have been successful in integrating cyberwarfare and cyberdeterrence capabilities into their own national war plans. Nevertheless, due to the previous extensive experience of Russia and the U.S. with strategic war planning, it is more likely that each of these countries stand the greatest chance of making integrating cyberwarfare and cyberdeterrence capabilities into their respective war plans. Yet, as recently as June 2009, it was clear that the U.S. and Russia were unable to agree on a treaty that would create the terms under which cyberwarfare operations could and would be conducted (Markoff and Kramer, 2009).

Is it problematic for these countries in the same ways or is there variation? What kind?

Every country that is modern enough to have organizations, people, and assets that are connected to computers and the Internet faces similar challenges of planning and managing cyberweapons and cyberdeterrence, and the poorer the country, the more significant the challenges. For example, when a small group of hackers from Manila in the Philippines unleashed the ILOVEYOU worm on the Internet in 2000, it caused over $2 billion in damages to computer data throughout the world. Agents from the FBI went to Manila to track down these people and investigate how and why the ILOVEYOU worm catastrophe occurred. To their surprise, they learned that each of these hackers who were involved could successfully escape prosecution because there were no laws in the Philippines with which to prosecute them. So actually most countries lack the technological and legal frameworks with which to successfully build a coordinated effort to manage the weapons and strategies of cyberwarfare and cyberdeterrence, despite the fact that most now embrace cyberspace with all the positive economic benefits it offers for commerce and communications.

What are the consequences to the U.S. and others if this threat is left unchecked?

As stated earlier, without the careful integration of cyberwarfare and cyberdeterrence technologies, strategies, and tactics into the CONOPS Plan, the national command authorities run a grave risk of launching a poorly planned offensive cyberwarfare operation that could precipitate a global crisis, impair relationships with its allies, and potentially unleash a whole host of unintended negative and potentially catastrophic consequences.

What consequences has the threat already produced on American/global society?

The absence of well-defined cyberwarfare and cyberdeterrence strategies and tactics in the CONOPS Plan has already produced some situations that have either damaged America's image abroad, or that could imperil its image and have far more negative consequences. For example, operates such as Stuxnet, Flame, Duque, etc., might have either been better planned or possibly not executed at all if cyberwarfare and cyberdeterrence strategies and tactics were defined in the CONOPS Plan. Also, the news media indicated during the revolution in Libya that resulted in the fall of Qaddafi, cyberwarfare operations were considered by the Obama administration. The negative reactions and repercussions on the world stage might have far outweighed any short term advantages that could have resulted from a successful set of cyberattacks against Libyan infrastructure assets that were attached to computer networks. Again, a comprehensive CONOPS

Plan that included well-defined cyberwarfare and cyberdeterrence strategies and tactics could have prevented such possible cyberattacks from even being considered, and it could have prevented the news of the possible consideration being publicized in the press (Schmitt, E. and Shanker, T., 2011). Without such restraint and well-planned deliberate actions, the U.S. runs the risk of appearing like the well-equipped cyber bully on the world stage, and an adversary who is willing to unleash weapons that can and will do crippling damage to an opponent, using technologies that are rapid, decisive, and not well-understood by those for whom they are intended. A similar effect and world reaction might be if U.S. Army infantry troops were equipped with laser rifles that emitted deadly laser blasts with pinpoint precision across several hundred yards.

The Rapid Evolution of Cyberthreats

As predicted in the Technolytics chart below, cyberweapons have rapidly evolved over time.

Since Stuxnet was released in 2010, countries and the general public are now aware of some of the offensive, strategic and destructive capabilities and potential of cyberweapons (Gelton, T., 2011).

The changes that produced Stuxnet and other recent, more modern cyberweapons were a national resolve to excel in the cyberwarfare area, coupled with excellent reconnaissance on desired targets, and partnering with computer scientists in Israel. The political consequences are not well understood yet, except to say that the U.S. and Israel are probably less trusted and suspected of even greater future capabilities, as well as having the will to use them. Again, having well-planned cyberwarfare and cyberdeterrence strategies and tactics defined in the CONOPS Plan might indeed, restrain such possibly reckless decisions as to unleash cyberweapon attacks without what the world might consider the correct provocation.

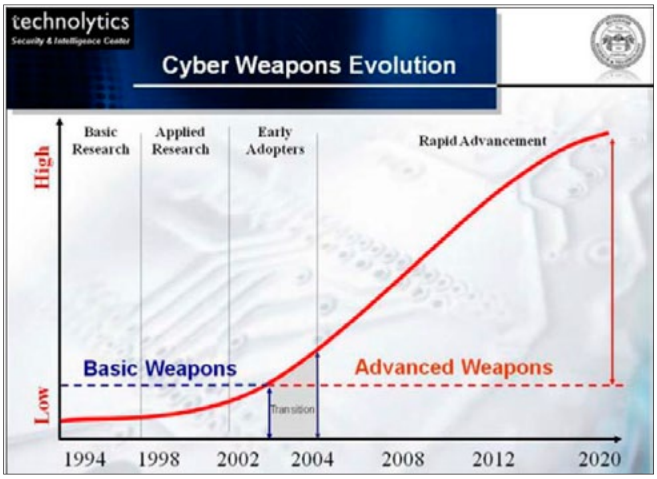


Figure 1. *Evolution of Cyberweapons (Technolytics, 2012)*

Part 1 Final Thoughts about Cyberwarfare Operations

In the words of Deb Radcliff, in an article published in SC Magazine in September 2012, “we are already in a cyberwar” (Radcliff, D., 2012). But as I was performing my research, it occurred to me that a country like the U.S., might in the future unleash such a devastating cyberattack that it could cripple the enemy’s ability to communicate surrender. I think that the moral implications of such circumstances need to be justly considered as a matter of the laws of war, because if a country continues to attack an enemy that has indicated that they are defeated and want to surrender, this shifts the moral ground from which the U.S. may have it was conducting its cyberwarfare operations. This is one other unintended consequence of cyberwarfare and one that needs to be carefully considered.

Part 2 – U.S. Policy Appraisal Related to Cyberwarfare and Cyberdeterrence

This section will examine current U.S. Policy related to cyberwarfare and cyberdeterrence.

Current U.S. Policy Covering Cyberwarfare Threats

The current written policy related to cyberwarfare threats can be found in President Obama’s Defense Strategic Guidance 2012, a 16-page policy documented that was published on January 3, 2012. The excerpt related specifically to cyberwarfare and cyber threats is shown below:

> *“To enable economic growth and commerce, America, working in conjunction with allies and partners around the world, will seek to protect freedom of access throughout the global commons – those areas beyond national jurisdiction that constitute the vital connective tissue of the international system. Global security and prosperity are increasingly dependent on the free flow of goods shipped by air or sea. State and non-state actors pose potential threats to access in the global commons, whether through opposition to existing norms or other anti-access approaches. Both state and non-state actors possess the capability and intent to conduct cyber espionage and, potentially, cyber attacks on the United States, with possible severe effects on both our military operations and our homeland. Growth in the number of*

> *space-faring nations is also leading to an increasingly congested and contested space environment, threatening safety and security. The United States will continue to lead global efforts with capable allies and partners to assure access to and use of the global commons, both by strengthening international norms of responsible behavior and by maintaining relevant and interoperable military capabilities (Obama, 2012)."*

The first explicit Obama Administration policy acknowledging the realities of cyber threats were published in a 30-page document titled International Strategy for Cyberspace in May 2011.

> *"Today, as nations and peoples harness the networks that are all around us, we have a choice. We can either work together to realize their potential for greater prosperity and security, or we can succumb to narrow interests and undue fears that limit progress. Cybersecurity is not an end unto itself; it is instead an obligation that our governments and societies must take on willingly, to ensure that innovation continues to flourish, drive markets, and improve lives. While offline challenges of crime and aggression have made their way to the digital world, we will confront them consistent with the principles we hold dear: free speech and association, privacy, and the free flow of information.*
>
> *"The digital world is no longer a lawless frontier, nor the province of a small elite. It is a place where the norms of responsible, just, and peaceful conduct among states and peoples have begun to take hold. It is one of the finest examples of a community self-organizing, as civil society, academia, the private sector, and governments work together democratically to ensure its effective management. Most important of all, this space continues to grow, develop, and promote prosperity, security, and openness as it has since its invention. This is what sets the Internet apart in the international environment, and why it is so important to protect.*
>
> *"In this spirit, I offer the United States' International Strategy for Cyberspace. This is not the first time my Administration has address the policy challenges surrounding these technologies, but it is the first time that our Nation has laid out an approach that unifies our engagement with international partners on the full range of cyber issues. And so this strategy outlines not only a vision for the future of cyberspace, but an agenda for realizing it. It provides the context for our partners at home and abroad to understand our priorities, and how we can come together to preserve the character of cyberspace and reduce the threats we face (Obama, 2011)."*

Though the Obama Administration reviewed and approved President Bush's CNCI policy in May 2009, Obama, who is regarded as the most technology-savvy president that has ever occupied the White House, went much further to acknowledge the importance of cyberspace to the American economy and the American military, and the importance of defending the U.S. from adversaries that could threaten us via cyberspace. Obama's policy also acknowledges the reality that future wars will be fought on the realm of cyberspace, and has thus funded the preparation of the U.S. armed forces to prepare for conflict in cyberspace (Gerwitz, 2011).

What is the effectiveness of current policy when it concerns this particular threat issue?

The Obama Administration's policies have been effective in raising the awareness of the U.S. population as to the importance of protecting assets that are connected in cyberspace. These policies have also been effective in providing for the preparation of the U.S. military to deal with conflict in cyberspace.

However, the present policy has not been effective as a deterrence to cyber threats presented by potential national enemies and non-state actors. As recently as September 23, 2012 – September 30, 2012, cyber attacks in the form of distributed denial of service (DDOS) attacks from the Middle East against several major U.S. banks based have publicly demonstrated the ire of the attackers and also the vulnerabilities of banks with a customer presence in cyberspace (Strohm and Engleman, 2012).

Short-Term and Long-term Ramifications of Current Policy

In the short-term, the Obama Administration's policies regarding cyberspace have done much to raise the awareness of cyberspace as an area that requires protection for the public good and prosperity of the American people. These policies have also served to show our allies and our potential enemies that the U.S. has the intention of defending cyberspace and all our interests that are connected to it. In the long-term, these policies will probably evolve to reveal in a general, unclassified way, stronger defenses, stronger deterrent capabilities and probably offensive cyberweapons.

On the legislative front, as recently as September 23, 2012, Chairman of the Senate Homeland Security Committee, Senator Joseph Lieberman (D., Connecticut), realizing that Congress would fail to pass cybersecurity legislation to designed to help protect the United States and its people, sent an urgent letter to President Obama to ask for the creation of a new Presidential Executive Order that would address several current cybersecurity issues, that includes how and when and where law enforcement can become involved in cybersecurity issues (Kerr, 2012). Though many digital privacy rights advocates, including the Electronic Frontier Foundation, the Electronic Privacy Information Center, and the American Civil Liberties Union have strenuously fought recent cybersecurity legislation, it is expected by many cybersecurity experts that if President Obama is reelected in November 2012, an Executive Order drafted and signed by the Obama Administration provide the tools that the federal government wants. Even if President Obama is not reelected in November 2012, it is expected that some expedient action on the part of the new president would probably take place even before Congress could successfully agree upon and pass such legislation.

Allies and Adversaries Connected to this Specific Policy?

It is entirely likely that there are classified versions of the International Strategy for Cyberspace policy that address the nature of how U.S. policies regarding the defense of cyberspace will affect our allies and our adversaries. But since it has been publicly revealed that the Obama Administration has conducted offensive cyberwarfare operations against Iran between June 2009 and June 2010, it is also likely that both our allies and our enemies have a clearer understanding of U.S. capabilities as well as the intent to use cyberweapons when it deems it is in its best interests to do so.

Part 2 Conclusion

The good news is that President Obama and his Administration apparently have an acute awareness of the importance of the cyberspace to the American economy and the American military. The bad news is that because we are already in some form of cyberwarfare that appears to be rapidly escalating, it remains to be seen what effects these cyberattacks and the expected forthcoming Executive Orders that address cybersecurity will have on the American people and our way of life. Nevertheless, it will be necessary to act prudently, carefully balancing our freedoms with our need for security, and also considering the importance of enabling and protecting the prosperity of the now electronically connected, free enterprise economy that makes the U.S. the envy of and the model for the rest of the world.

Part 3 – Strategic Comparative Analysis in Cyberwarfare and Cyberdeterrence

This section will present a strategic comparative analysis of the present state of cyberwarfare and cyberdeterrence issues as that relate to other countries that could be considered adversaries, now or in the not too distant future.

What Other Countries / Regions of the World Are Concerned with This Same Threat Issue?

The countries that are primarily concerned with cyberwarfare and cyberdeterrence threat issues are the same countries that already have the greatest cyberwarfare capabilities and also the most to lose in the event of a full-scale cyberwarfare attack.

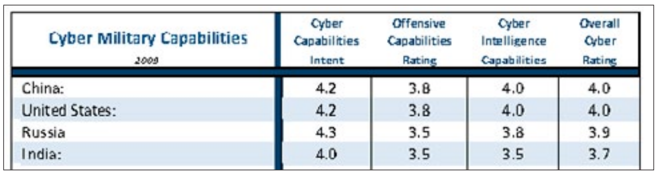
The diagram below from a 2009 study shows the comparative cyberwar capabilities of the 66 largest countries in the world.

Countries Regions of the World That Do Not Place a High Priority on This Threat Issue

Countries that are more focused on the survival and welfare of their citizens, coupled with the fact that they are largely consumers of Internet and computer capabilities versus being able to afford to channel resources into the development of cyberweapons or the resources required to develop a credible cyberdeterrence strategy. It is also ironic that the U.K. with its stature and status does not rank higher on the list shown in Table 1.

Some of the Current Policies Being Employed by These Other States / Regions in Regards to the Threat

China, Russia, and India, each of which are in the top four of the countries listed in Table 1, have well-defined cyberwarfare policies and strategies.



Cyber Military Capabilities 2009	Cyber Capabilities Intent	Offensive Capabilities Rating	Cyber Intelligence Capabilities	Overall Cyber Rating
China:	4.2	3.8	4.0	4.0
United States:	4.2	3.8	4.0	4.0
Russia	4.3	3.5	3.8	3.9
India:	4.0	3.5	3.5	3.7

Figure 2. *Country Cyber Capabilities Ratings (Technolytics, 2012)*

Ironically, the U.S., which occupies the number 2 position in that same table, does not yet have well-defined cyberwarfare policies and strategies. For comparison, Table 2 below shows a summary of the policies and strategies of China, Russia and India.

Successes and Failures of the Various Alternative Policies around the Globe

Despite some of the negative press from the Stuxnet virus, this collaborative effort by the U.S. and Israel has been looked at with both fascination and as an event that has quickly and successfully heralded in a new age of warfare, the age of cyberwarfare. However, many still feel that in the absence of publically defined policies and strategies by the Obama Administration, it invites a secretive and even random appearance of and the continued use of cyberweapons (Sanger, 2012).

Areas of Joint Communication / Operation / Cooperation that Exist or Should Exist Across Countries Dealing with This Threat Issue

Apparently, the U.S. has already created one or more rather sophisticated cyberweapons with the help of Israeli cyberweapon experts. At least one of these cyberweapons, the Stuxnet Worm, was effectively used to impede the development of Iran's nuclear material refinement program from 2009 to 2010 (Langer, 2010).

It is likely however, that through the auspices of the United Nations, or perhaps some G20 accord, there may be some general consensus on the importance of defining the appropriate uses cyberweapons. There also needs to be some agreement on types of response to cyberattacks, and effective methods of cyberdeterrence.

China and Its Role in Cyberwarfare Capabilities

China is probably doing a better job than the realm of cyberwarfare for three reasons: 1) the government has invested considerable resources into their cyberwarfare capabilities; 2) the number of personnel devoted to cyberwarfare efforts is reportedly in the tens of thousands; and 3) the Chinese government is able to easily operate under a cloak of secrecy and conduct operations without fear of cyberwarfare activities being leaked to Chinese press agencies (Hagestad, 2012).

Part 3 Conclusion

This paper has presented a brief strategic comparative analysis of countries with cyberwarfare capability.

Table 1. *Summary of Cyberwarfare Policies and Strategies of China, Russia, and India*

Country	Policy	Strategy
China	China supports cyberwarfare capabilities, especially providing such capabilities in the People's Liberation Army.	The Chinese will wage unrestricted warfare and these are the principles: Omni-directionality Synchrony Limited objectives Unlimited measures Asymmetry Minimal consumption Multi-dimensional coordination djustment, control of the entire process (Hagestad, 2012).
Russia	Russia supports cyberwarfare capabilities, especially providing such capabilities in the Russian Army. The nature of cyberwarfare and information warfare requires that the development of a response to these challenges must be organized on an interdisciplinary basis and include researchers from different branches – political analysts, sociologists, psychologists, military specialists, and media representatives (Fayutkin, 2012).	The ability to achieve cyber superiority is essential to victory in cyberspace. (Fayutkin, 2012).
India	India supports cyberwarfare capabilities, especially providing such capabilities in the Indian Army. "It is essential for efficient and effective conduct of war including cyber-war. The war book therefore needs to specify as how to maintain no-contact cyber war and when the government decide to go for full-contact or partial-contact war then how cyber war will be integrated to meet overall war objectives (Saini, 2012)."	Strategies are still under development, but will follow the guidance of policies related to the conduct of war. (Saini, 2012)

Part 4 – Conflict Resolution in Cyberwarfare and Cyberdeterrence

This section will present the ideas of conflict analysis and resolution as they relate to cyberwarfare.

Current Academic Research on This Threat Problem

Since 2007, as the existence of well-orchestrated cyberwar attacks such as the DDoS attacks on Estonia (2007), Georgia (2008), and Kyrgyzstan (2009), as well as the Stuxnet (2010), Duqu (2011), and Flame (2012) have all become known to the world through security researchers, their victims, and the media. As a result, it has become apparent most who are watching this area that cyberspace has now become the new realm onto which the field of international conflict has been extended, and that cyberwarfare is now no longer a theoretical issue that could one day threaten those participants and systems that rely upon connections to the Internet and Internet-connected networks. Unfortunately however, the present findings and research on cyberwarfare related events shows that the U.S. is playing catch-up and doing so badly (Turanski and Husick, 2012).

Intellectual Positions and Theoretical Explanations That Have Been Staked Out on This Threat Problem

As recently as the 2008 – 2009 timeframe, John Boyd's conflict model known as Observe – Orient – Decide – Act (OODA) began to be applied to analyze the ideas of "cybernetic warfare" and "net-centric warfare." The model itself has been analyzed for its ability to simply demonstrate the nature of the complexity of conflict, complete with factors of ambiguity, unpredictability, and so the model has also been used to define the nature of life itself. Yet, the model is also impacted by the chaotic nature of life and reality. The further shows the similarity between actual cyberwarfare events and this model. Other characteristics of the OODA loop model are its continuous nature and the feedback loops that provide data on which to base some form (or forms) of decision and action. The OODA Loop model is shown in the diagram below:

However, one key distinction between Boyd's OODA model and cybernetic warfare is Boyd's "focus on the conditions of emergence transformation of systems through information rather than merely the manner in which information is processed by a fixed organizational schema." Boyd would argue that Claude Shannon and others tend to overemphasize the view of information related to structure as opposed to information as a process (Bousquet, 2009).

Joint Publication (JP) 5-0, Joint Operation Planning

As recently as December 2006, the Joint Chiefs of Staff provided an inside look into how the U.S. National War Plan was created and maintained. In the document titled, Joint Publication (JP) 5-0, Joint Operation Planning. While this publically available, 264-page, document is unclassified, it does provide an extraordinary look into the strategic military thinking, principles, and guidance of the Joint Chiefs of Staff and the National Command Authorities as they create policies and strategies that enforce the national strategic objectives of the United States. This document that was created during the Bush administration is also significant because it is one of the first official publically known such documents that included cyberspace as part of the operational realm of conflict, along with air, sea, land, and space for conducting military operations (U.S. DoD, JCS, 2006). The high-level diagram be-

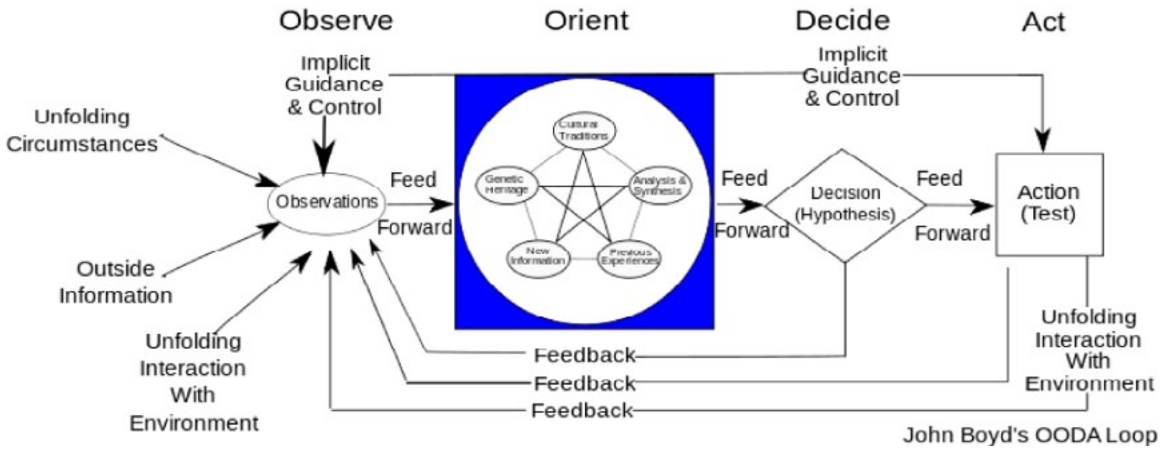


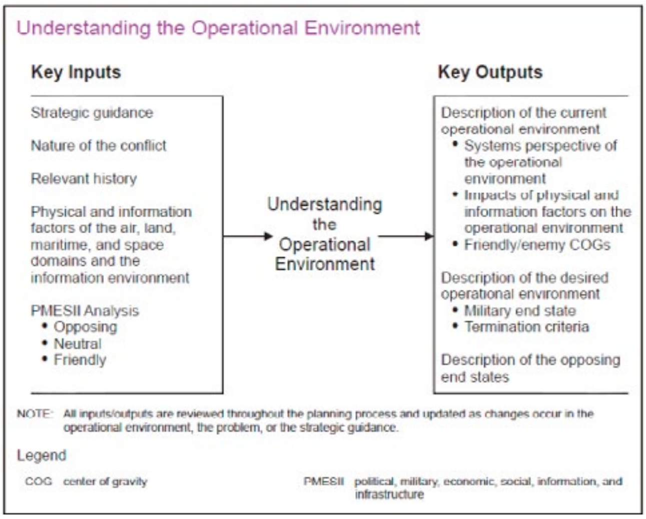
Figure 3. *Boyd's OODA Loop Model (Bousquet, 2009)*

low shows simply the concept of the inputs and the outputs that lead to understanding the operational environment of conflict, and it compares somewhat to the OODA figure shown earlier: Figure 4.

To further illustrate the intent of the Joint Chiefs of Staff to the diagram below to visually explain the interconnected nature of the realms related to the operational environment of conflict and the nature of the systems analysis required for decision making (Figure 5).

The JCS also described the environment of conflict as a place where simultaneity of operations would and this environment would include the information environment and cyberspace:

> *"Simultaneity refers to the simultaneous application of military and nonmilitary power against the enemy's key capabilities and sources of strength. Simultaneity in joint force operations contributes directly to an enemy's collapse by placing more demands on enemy forces and functions than can be handled. This does not mean that all elements of the joint force are employed with equal priority or that even all elements of the joint force will be employed. It refers specifically to the concept of attacking appropriate enemy forces and functions throughout the OA (across the physical domains and the information environment [which includes cyberspace]) in such a manner as to cause failure of their moral and physical cohesion (U.S. DoD, JCS, 2006)."*



Understanding the Operational Environment

Key Inputs

Strategic guidance

Nature of the conflict

Relevant history

Physical and information factors of the air, land, maritime, and space domains and the information environment

PMESII Analysis
- Opposing
- Neutral
- Friendly

Understanding the Operational Environment

Key Outputs

Description of the current operational environment
- Systems perspective of the operational environment
- Impacts of physical and information factors on the operational environment
- Friendly/enemy COGs

Description of the desired operational environment
- Military end state
- Termination criteria

Description of the opposing end states

NOTE: All inputs/outputs are reviewed throughout the planning process and updated as changes occur in the operational environment, the problem, or the strategic guidance.

Legend

COG center of gravity

PMESII political, military, economic, social, information, and infrastructure

Figure 4. *Understanding the Operational Environment (U.S. DoD, JCS, 2006)*

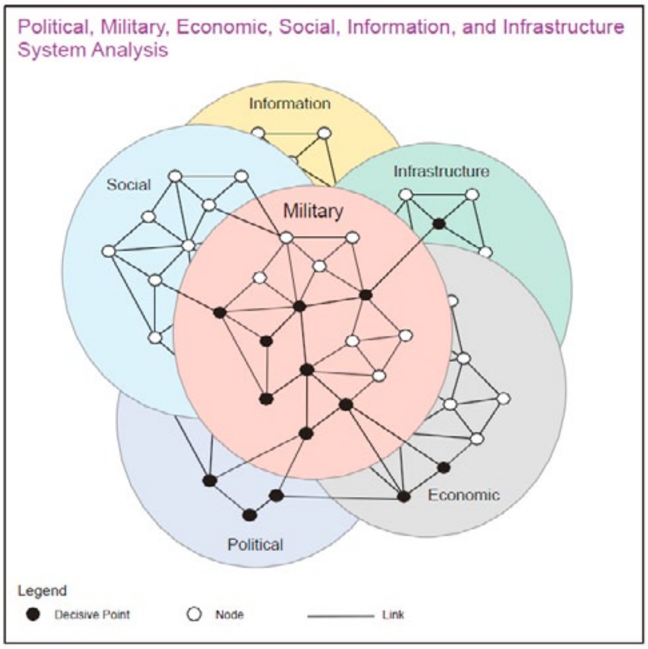


Figure 5. *Understanding the Interconnected Nature of the Realms Related to the Operational Environment of Conflict and the Nature of the Systems Analysis Required for Decision Making (U.S. DoD, JCS, 2006)*

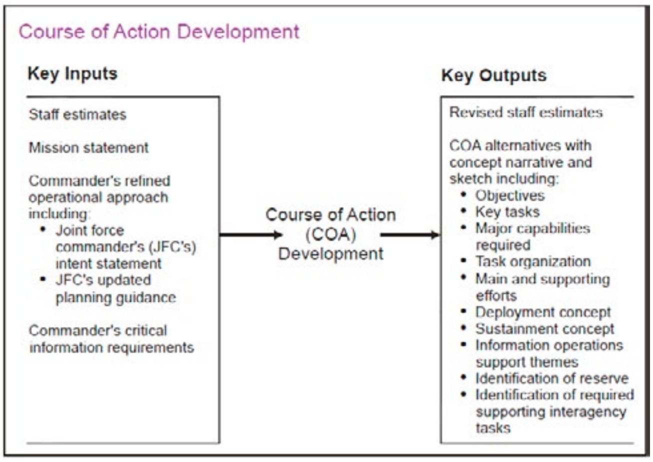
Therefore, the JCS also created a Course of Action framework for determining the best courses of action in a conflict environment, and here again, cyberspace is included in that realm of options in which a course of action could and would be developed (U.S. DoD, JCS, 2006) (Figure 6).

Options in Conflict

Based on the current state of where the U.S. stands with the lack of coherent and cohesive incorporated into its National CONOPSPLAN, and the potential for unintended consequences where the unilateral use of cyberweapons can and will occur, I see three possible options for the U.S., and each of these options has advantages and disadvantages.

Part 4 Conclusion

This section has presented a brief look at the U.S. Military's recognition of cyberspace as an extension of the operational environment of conflict and



Course of Action Development

Key Inputs

Staff estimates

Mission statement

Commander's refined operational approach including:
- Joint force commander's (JFC's) intent statement
- JFC's updated planning guidance

Commander's critical information requirements

Course of Action (COA) Development

Key Outputs

Revised staff estimates

COA alternatives with concept narrative and sketch including:
- Objectives
- Key tasks
- Major capabilities required
- Task organization
- Main and supporting efforts
- Deployment concept
- Sustainment concept
- Information operations support themes
- Identification of reserve
- Identification of required supporting interagency tasks

Figure 6. *Course of Action Development (U.S. DoD, JCS, 2006)*

a comparison of the options that exist for resolving the issues that threaten America's ability to create the coherent and cohesive policies and strategies that will define its ability to effectively conduct cyberwarfare and cyberdeterrence in the future.

Part 5 – Policy Generation Related to Cyberwarfare and Cyberdeterrence

This section will present the ideas for the creation of national policy or enhancement of existing national policy related to cyberwarfare and cyberdeterrence issues.

Current U.S. Policy Covering Cyberwarfare Threats

As started earlier in the Part 2 – Policy Analysis, the current written policy related to cyberwarfare threats can be found in President Obama's Defense Strategic Guidance 2012, a 16-page policy documented that was published on January 3, 2012. It has already been noted that this policy has not been effective in deterring cyberattacks and other acts of cyberwar.

Challenges Related to Cyberwar and Cyberdeterrence Policy and Strategy Creation

The creation of policies and strategies related to cyberwar and cyberdeterrence are complicated by six major issues:

- The lack of international definition and agreement on what constitutes an act of cyberwar (Markoff and Kramer, 2009).
- The lack of the ability to clearly attribute the source of an attack (Turzanski and Husick, 2012).
- The ability for non-state actors to conduct potent cyberattacks (Turzanski and Husick, 2012).
- The inability to clearly define what the exact nature of critical infrastructure targets (Turzanski and Husick, 2012).
- The massive proliferation and reliance on of ubiquitous, highly insecure, vulnerable systems based on SCADA technologies during the 1980s and 1990s (Turzanski and Husick, 2012).
- The continually changing landscape of information technology including the vulnerabilities and threats related to systems that are obsolete, yet remain in operational use for several years past their intended useful life.

A Single Integrated Operational Plan for War

During the 1950s and 1960s, when it became evident that nuclear weapons could play a major role in strategic warfare, the United States, utilized a think-tank of individuals, both military and civilian, to craft the strategic war-fighting plans of the U.S. that would deal with very real possibility that tactical and possibly strategic nuclear weapons may be required during a major wartime scenario. The first such war plan was called the Single Integrated Operational Plan (SIOP). The process of its creation involved the use of intelligence data about potential enemies, a threat assessment process, and then a process whereby the identified likely targets would be prioritized and matched with weapons. The process of matching weapons to targets also included intricate sequence timings, and the various event triggers that would result in the execution of such

Table 2. *Comparing Options for Incorporating Cyberwar and Cyberdeterrence Policies and Strategies into the U.S. National CONOPS Plan*

Option	Description	Advantage	Disadvantage
1	Create policies that mandate the inclusion of cyberwarfare and cyberdeterrence into the U.S. National CONOPS Plan	Prevents unintended consequences of unilateral use or unplanned use of cyberweapons	Takes time, politics, skills, knowledge, and money
2	Limited creation and application of policies that mandate the inclusion of cyberwarfare and cyberdeterrence into the U.S. National CONOPS Plan	Prevents some possible unintended consequences of unilateral use or unplanned use of cyberweapons	Still requires some time, political wrangling, skills, knowledge, and money
3	Do nothing whatsoever related to cyberweapons and U.S. National CONOPS Plan. Just continue to the present trend to continue to conduct cyberwarfare operations on an ad hoc basis in secrecy, and allow the situation with current cyberwarfare threats to continue (Sanger, 2012).	Saves time, political wrangling, and money	Unintended consequences of unilateral use or unplanned use of cyberweapons

attacks. In the 1980s, the SIOP evolved into something called the OPSPLAN and later, it was renamed the CONOPS Plan, but it has always been kept up to date and tested at least semiannually so that all involved would know their roles if the nation command authorities deemed it necessary to execute this intricate war plan (Freedman, 2003).

Note that as far back as the 1970s, there were 24 defined levels of conflict between the U.S. and a potential adversary, ranging from a war of words, all the way to strategic nuclear war. No matter what the name of it was, the national war plan has always been a key tool of the national command authorities for understanding what military responses would be required in the event of these various levels of conflict.

Recommendations for the U.S. Cyberwarfare Policy and Strategy

It is not unreasonable to assume that the path towards a coherent and cohesive U.S. policy and set of strategies regarding the use of cyberweapons will follow a path that is similar to the strategic war plan maturity path from Hiroshima to the SIOP. Today, in the absence of any clear policy on the use of cyberweapons, Crosston advocates the agreement on a policy of "Mutually Assured Debilitation" in which everyone with cyberweapons would come to a general understanding that the use of these weapons would result in the expectation that massive destruction would be unleashed on every participant's assets (Crosston, 2011). This makes perfect sense considering that the "Mutually Assured Destruction" nuclear deterrence policy was effective and worked well during the Cold War from the 1950s through 1990s.

Yet, today, I believe that once a coherent and cohesive U.S. policy on cyberwarfare and cyberweapons is defined by the National Command Authorities, there should be an eight-step process that could result in the development and rapid maturation of a strong national strategy U.S. Cyberwarfare:

- Define the doctrines and principles related to cyberwarfare and the needs under which cyberwarfare would be conducted.
- Create the policies that embody these doctrines and principles.
- Conduct the intelligence gathering to accurately understand the landscape of the cyber battlefield.
- Perform the analysis to create the strategy
- Create the strategic plan and tactics
- Conduct regular war games, at least twice yearly to test the strategic plan and tactics
- Analyze and document the results of the cyberwarfare war games.
- Refine the strategies and tactics for cyberwarfare and cyberdeterrence based on the results of analyzing the outcomes of the cyberwarfare war games

Note that it is also essential to continually assess the capabilities of Information Technology so that tools that our cyberwarfare fighters are using are

Table 3. *A 10-step Remedy toward the Creation of National Policy (Kramer, et al, 2009)*

Idea	Explanation
Unify Policy Direction	Effective policies will not be created by a single person or entity, but they require centralized leadership to unify their direction and intent.
Specialize Policy Direction	Recognizing that one size does not fit all, specialized policies need to be created for varies infrastructures and industries to ensure maximum protection.
Strengthen and Unify Regulation	Regulations must be strengthened to be more effective, or new, more effective regulations must be created.
Define State and Local Roles	A workable Federal policy must have the involvement of state and local authorities to be effective
Define International Interfaces	This is required because cyberspace is connected internationally and because there is still lack of international agreement on many aspects of cyberwar.
Mandate Effective Systems Engineering for Infrastructure-related Software	Ensure that there is a realization and commitment for the need to have higher minimum standards for the quality of software that is related to infrastructure.
Don't Take No for an Answer	Ensure that stakeholders and those responsible participants realize the resolute, unwavering commitment toward a workable policy solution
Establish and Implement Clear Priorities	This will ensure the best allocation of financial and management resources.
Inform the Public Clearly and Accurately	The public needs to understand the efforts being made to protect the U.S.
Conduct a Continuing Program of Research	Keep the policy updated and relevant to changing technologies.

References

- Bousquet, A. (2009). The Scientific Way of Warfare: Order and Chaos on the Battlefields of Modernity. New York, NY: Columbia University Press.
- Bush, G. W. (2008). Comprehensive National Cybersecurity Initiative (CNCI). Published by the White House January 2008. Retrieved from *http://www.whitehouse.gov/cybersecurity/comprehensive-national-cybersecurity-initiative* on January 5, 2012.
- Carr, J. (2012). Inside Cyber Warfare, second edition. Sebastopol, CA: O'Reilly.
- Clarke, R. A. and Knake, R. K. (2010). Cyberwar: the Next Threat to National Security and What to Do About It. New York, NY: HarperCollins Publishers.
- Crosston, M. (2011). World Gone Cyber MAD: How "Mutually Assured Debilitation" Is the Best Hope for Cyber Deterrence. An article published in the Strategic Studies Quarterly, Spring 2011. Retrieved from *http://www.au.af.mil/au/ssq/2011/spring/crosston.pdf* on October 10, 2012.
- Czosseck, C. and Geers, K. (2009). The Virtual battlefield: Perspectives on Cyber Warfare. Washington, DC: IOS Press.
- Edwards, M. and Stauffer, T. (2008). Control System Security Assessments. A technical paper presented at the 2008 Automation Summit – A Users Conference, in Chicago. Retrieved from *http://www.infracritical.com/papers/nstb-2481.pdf* on December 20, 2011.
- Fayutkin, D. (2012). The American and Russian Approaches to Cyber Challenges. Defence Force Officer, Israel. Retrieved from *http://omicsgroup.org/journals/2167-0374/2167-0374-2-110.pdf* on September 30, 2012.
- Freedman, L. (2003). The Evolution of Nuclear Strategy. New York, NY: Palgrave Macmillan.
- Gerwitz, D. (2011). The Obama Cyberdoctrine: tweet softly, but carry a big stick. An article published at Zdnet.com on May 17, 2011. Retrieved from *http://www.zdnet.com/blog/government/the-obama-cyberdoctrine-tweet-softly-but-carry-a-big-stick/10400* on September 25, 2012.
- Gjelten, T. (2010). Are ‚Stuxnet' Worm Attacks Cyberwarfare? An article published at NPR.org on October 1, 2011. Retrieved from *http://www.npr.org/2011/09/26/140789306/security-expert-u-s-leading-force-behind-stuxnet* on December 20, 2011.
- Gjelten, T. (2010). Stuxnet Computer Worm Has Vast Repercussions. An article published at NPR.org on October 1, 2011. Retrieved from *http://www.npr.org/templates/story/story.php?storyId=130260413* on December 20, 2011.
- Gjelten, T. (2011). Security Expert: U.S. ‚Leading Force' Behind Stuxnet. An article published at NPR.org on September 26, 2011. Retrieved from *http://www.npr.org/2011/09/26/140789306/security-expert-u-s-leading-force--behind-stuxnet* on December 20, 2011.
- Gjelten, T. (2011). Stuxnet Raises ‚Blowback' Risk In Cyberwar. An article published at NPR.org on December 11, 2011. Retrieved from *http://www.npr.org/2011/11/02/141908180/stuxnet-raises-blowback-risk-in-cyberwar* on December 20, 2011.
- Hagestad, W. T. (2012). 21st Century Chinese Cyberwarfare. Cambridgeshire, U.K.: IT Governance.
- Hyacinthe, B. P. (2009). Cyber Warriors at War: U.S. National Security Secrets & Fears Revealed. Bloomington, IN: Xlibris Corporation.
- Jaquith, A. (2007). Security Metrics. Boston, MA: Addison Wesley.
- Kaplan, F. (1983), The Wizards of Armageddon: The Untold Story of a Small Group of Men Who Have Devised the Plans and Shaped the Policies on How to Use the Bomb. Stanford, CA: Stanford University Press.
- Kerr, D. (2012). Senator urges Obama to issue ‚cybersecurity' executive order. An article published at Cnet.com on September 24, 2012. Retrieved from *http://news.cnet.com/8301-1009_3-57519484-83/senator-urges-obama-to-issue-cybersecurity-executive-order/* on September 26, 2012.
- Kramer, F. D. (ed.), et al. (2009). Cyberpower and National Security. Washington, DC: National Defense University.
- Langer, R. (2010). A Detailed Analysis of the Stuxnet Worm. Retrieved from *http://www.langner.com/en/blog/page/6/* on December 20, 2011.
- Libicki, M.C. (2009). Cyberdeterrence and Cyberwar. Santa Monica, CA: Rand Corporation.
- Markoff, J. and Kramer, A. E. (2009). U.S. and Russia Differ on a Treaty for Cyberspace. An article published in the New York Times on June 28, 2009. Retrieved from *http://www.nytimes.com/2009/06/28/world/28cyber.html?pagewanted=all* on June 28, 2009.
- Mayday, M. (2012). Iran Attacks US Banks in Cyber War: Attacks target three major banks, using Muslim outrage as cover. An article published on September 22, 2012 at Poltix.Topix.com. Retrieved from *http://politix.topix.com/homepage/2214-iran-attacks-us-banks-in-cyber-war* on September 22, 2012.
- McBrie, J. M. (2007). THE BUSH DOCTRINE: SHIFTING POSITION AND CLOSING THE STANCE. A scholarly paper published by the USAWC STRATEGY RESEARCH PROJECT. Retrieved from *http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA423774* on September 30, 2012.
- Obama, B. H. (2012). Defense Strategic Guidance 2012 – Sustaining Global Leadership: Priorities for 21st Century Defense. Published January 3, 2012. Retrieved from *http://www.defense.gov/news/Defense_Strategic_Guidance.pdf* on January 5, 2012.
- Obama, B.H. (2011). INTERNATIONAL STRATEGY for Cyberspace. Published by the White House on May 16, 2011. Retrieved from *http://www.whitehouse.gov/sites/default/files/rss_viewer/international_strategy_for_cyberspace.pdf* on May 16, 2011.
- Payne, K. B. (2001). The Fallacies of Cold War Deterrence and a New Direction. Lexington, KY: The University of Kentucky Press.
- Pry, P. V. (1999). War Scare: Russia and America on the Nuclear Brink. Westport, CT: Praeger Publications.
- Radcliff, D. (2012). Cyber cold war: Espionage and warfare. An article published in SC Magazine, September 4, 2012. Retrieved from *http://www.scmagazine.com/cyber-cold--war-espionage-and-warfare/article/254627/* on September 7, 2012.
- Saini, M. (2012). Preparing for Cyberwar – A National Perspective. An article published on July 26, 2012 at the Vivikanda International Foundation. Retrieved from *http://www.vifindia.org/article/2012/july/26/preparing-for-cyber-war-a-national-perspective* on October 14, 2012.
- Sanger, D. E. (2012). Confront and Coneal: Obama's Secret Wars and Surprising Use of America Power. New York, NY: Crown Publishers.
- Schmidt, H. S. (2006). Patrolling Cyberspace: Lessons Learned from Lifetime in Data Security. N. Potomac, MD: Larstan Publishing, Inc.

state of the art and that they are effective and perform well as they are integrated into the cyberwar war fighting environment.

Recommendations for the U.S. Cyberdeterrence Policy and Strategy

A strongly worded, explicit U.S. national policy regarding cyber deterrence would serve to further strengthen the U.S. in cyberspace as well as protect critical infrastructure and our allies. According to a 1997 paper that was prepared by the U.S. Army for the Clinton administration, Toward Deterrence in the Cyber Dimension these would be recommended elements of such a policy:

- Continue to design, create, possess, and use offensive cyber warfare capabilities when necessary
- Develop a defensive system for surveillance, assessment, and warning of a cyber attack. (I think such capability presently exists now)
- A declaration that any act of deliberate information warfare resulting in the loss of life or significant destruction of property will be met with a devastating response (U.S. Army, 1997).
- I would also include Crosston's idea of Mutually Assured Debilitation (Crosston, 2011).

Final Thoughts on the Creation of a National Policy on Cyberwar and Cyberdeterrence

According to Kramer, the table below contains the 10-step remedy for creating a policy that would protect the U.S. in cyberspace.

Part 5 Conclusion

This section has presented a brief look at the importance of creating a set of publicly available, coherent and cohesive national policies and strategies that will facilitate U.S. capabilities to effectively conduct cyberwarfare and cyberdeterrence operations now and in the future. At the present moment, the lack of such policies effectively represents a window of risk and uncertainty during a time when cyber threats and cyber attacks are growing at an exponential rate. That has the elements of a real potential for a cyber disaster if this weak policy situation is not resolved as soon as possible. Here, I presented a set of processes and a framework by which the U.S. can quickly address the national challenges of effectively creating the urgently needed national policies and integrated strategies for conducting cyberwarfare and cyberdeterrence operations now and in the future.

Conclusion

This paper has presented a brief look at the importance of creating a clear set of publicly available, coherent and cohesive national policy. It then advocated the incorporation of strategies that will address U.S. intentions and capabilities to effectively conduct cyberwarfare and cyberdeterrence operations now and in the future, into the U.S. CONOPS Plan.

WILLIAM F. SLATER, III
DET 630 – Cyberwarfare and Cyberdeterrence
Bellevue University

MATTHEW CROSSTON

References

- Schmitt, E. and Shanker, T. (2011). U.S. Debated Cyberwarfare in Attack Plan on Libya. An article published in the New York Times on October 17, 2011. Retrieved from *http://www.nytimes.com/2011/10/18/world/africa/cyber-warfare-against-libya-was-debated-by-us.html* on October 17, 2011.
- Stiennon, R. (2010). Surviving Cyber War. Lanham, MA: Government Institutes.
- Strohm, C. and Engleman, E. (2012). Cyber Attacks on U.S. Banks Expose Vulnerabilities. An article published at BusinessWeek.com on September 28, 2012. Retrieved from *http://www.businessweek.com/news/2012-09-27/cyber-attacks-on-u-dot-s-dot-banks-expose-computer-vulnerability* on September 30, 2012.
- Technolytics. (2012). Cyber Commander's eHandbook: The Weaponry and Strategies of Digital Conflict, third edition. Purchased and downloaded on September 26, 2012.
- Turzanski, E. and Husick, L. (2012). "Why Cyber Pearl Harbor Won't Be Like Pearl Harbor At All..." A webinar presentation held by the Foreign Policy Research Institute (FPRI) on October 24, 2012. Retrieved from *http://www.fpri.org/multimedia/2012/20121024.webinar.cyberwar.html* on October 25, 2012.
- U.S. Army. (1997). Toward Deterrence in the Cyber Dimension: A Report to the President's Commission on Critical Infrastructure Protection. Retrieved from *http://www.carlisle.army.mil/DIME/documents/173_PCCIPDeterrenceCyberDimension_97.pdf* on November 3, 2012.
- U.S. Department of Defense, JCS. (2006). Joint Publication (JP) 5-0, Joint Operation Planning, updated on December 26, 2012. Retrieved from *http://www.dtic.mil/doctrine/new_pubs/jp5_0.pdf* on October 25, 2012.
- Waters, G. (2008). Australia and Cyber-Warfare. Canberra, Australia: ANU E Press.

U.S. Government Says No Way To Huawei

The U.S. government asserts that the Chinese telecommunications behemoth, Huawei, poses a threat to its national security. Huawei insists that the claims are uncorroborated, and that they are being hindered by unsubstantiated, non-specific concerns, and profess they pose absolutely no threat to U.S. national security.

Through examination of how the U.S. has handled its involvement with Huawei, this paper seeks to answer the question: "Is there tangible evidence backing the stance the U.S. has taken against Huawei, or are decisions being made based on skepticism, speculations, and long-standing bias?"

Introduction

The U.S. government asserts that the Chinese telecommunications giant, Huawei, poses a threat to its national security because of their relationship with the Chinese government and the *People's Liberation Army* (PLA). Huawei insists that the claims are uncorroborated, and that they are being hindered by unsubstantiated, non-specific concerns, and profess they pose absolutely no threat to U.S. national security.

Through examination of how the U.S. has handled its involvement with Huawei, this paper seeks to answer the question: "Is there tangible evidence backing the prohibitive stance the U.S. government has taken against Huawei, or have decisions been guided by skepticism, speculations, a long-standing bias against China, or for protectionist reasons?"

This paper will also examine how Huawei is faring in other parts of the world, take a closer look at some of the United States' closest allies and scrutinize how they are dealing with Huawei. Additionally, this paper will examine alternative solutions, outsider views on the topic, and new result-oriented paths that could potentially lead to new resolutions in regards to the United States stance on Huawei.

Evolution of the Case, Nature of the Threat

U.S. political and intelligence forces continue to have major concerns about the possible threat that Huawei poses to national security, and blocked the company from competing for many projects within the U.S. There exists an abundance of reports filled with speculations, assumptions, and concerns that Huawei may have secret ties with the Chinese government and a special relationship with the PLA, which could lead to threats to national security and espionage through backdoors or modifications implanted in Huawei equipment. The U.S. government has an obligation to its citizens to act accordingly to protect its national security, but, at the same time, has to weigh the potential ramifications of shutting Huawei out without concrete evidence rather than unsubstantiated fears.

The theme of the *House Permanent Select Committee on Intelligence* (HPSCI) investigation hearing held on the 13th of September, 2012, echoed continued concerns the U.S. government has in regards to the supply chain of information technology equipment coming from China. The HPSCI continued its investigations of the alleged security threat Huawei and fellow Chinese company ZTE pose to the United States. Representatives from both Huawei and ZTE appeared before the committee for an open hearing on national security threats.

Summarizing the opening statements of Chairman Mike Rogers, some of the long-standing concerns the U.S. has with Huawei are (Rogers, 2012):

- Huawei reaps the benefits of billions of dollars in Chinese government financing. There exist concerns over believed ties to the Chinese government.

- Have heard reports about backdoors or unexplained beaconing from the equipment sold by both companies. Sources overseas say there is a reason to question whether the companies are tied to the Chinese government or whether their equipment is as it appears.
- Concerns over the ability to modify or steal information from government and corporate entities provides China access to expensive and time consuming R&D that assists China's place in the world.
- Huawei and ZTE provide opportunities for Chinese Intelligence agencies to insert malicious hardware or software implants into critical telecommunications components and systems. Under Chinese law, Huawei and ZTE would likely be required to cooperate with any request by the Chinese government to use their systems or access for malicious purposes.

Earlier this year the committee ramped up the pressure on Huawei to disclose details about its ties to the Chinese Government. The 11-page letter, which reads like a laundry list of accusations, focused on questions about everything from the alleged funding the company receives from the Chinese government – to inquires on how board members received their posts. The letter was released to the media from Mr. Rogers to Huawei founder and Chairman Ren Zhengfei (Cheng, Greene, 2012).

Mr. Zhengfei's past is just one of many concerns that the U.S. political and intelligence community has with Huawei. In a Northrop Grumman Corp report (Adams, Bakos, Krekel, 2012) dealing with Chinese capabilities for computer network operations and cyber espionage, prepared for the U.S.-China Economic and Security Review Commission, it was reported that Mr. Zhengfei is the former director of the GSD Information Engineering Academy, the PLA's primary center for telecommunications research, and there are concerns over Huawei's perceived relationship with the Chinese government and the PLA.

The Northrop Grumman Security report goes on to talk about the collaboration of U.S. and Chinese information security firms, pointing out that the Symantec, Inc, and Huawei's joint venture was the only major partnering between a western information security firm and a Chinese high technology company, the partnership was dissolved in 2011 after 4 years of operations. The report spells out concerns of these types of mergers, stating that the risk of loss of intellectual property, and long term competitiveness for U.S. companies. The report states that intellectual property theft is a concern for virtually all U.S. businesses operating in China.

Northrop Grumman is not alone in supplying U.S. decision makers reports on the potential threats Huawei may pose; Lockheed Martin, the U.S.-China Economic and Security Review Commission, the *Department of Homeland Security* (DHS), the U.S. Department of Commerce, the Pentagon, the RAND Corporation, and the *Committee on Foreign Investment in the United States* (CFIUS) have all have released publicly accessible reports, each reporting similar suspicions on the potential threat Huawei presents to U.S. national security.

The U.S. is putting up quite the wall for Huawei to climb, but in reviewing the above listed documents, it would be something of a stretch to say that there is any tangible proof to any of the claims. The concerns raised in these findings should absolutely be considered and investigated further, but it's challenging to wholly trust the findings when there seems to be an absence of solid, tangible evidence to back the reported concerns.

An example where the HPSCI may be stretching, or taking liberties to solidify their position against Huawei: In the hearing on the 13th of September, Michele Bachmann, a majority member of the HPSCI, repeatedly pressed Charles Ding, Corporate Senior Vice President of Huawei on allegations that Huawei had violated intellectual property rights. The situation she was referring to surrounded a patent infringement lawsuit Cisco Systems filed against Huawei in 2003, and was settled 20 months later. The U.S. court dismissed Cisco's claim "with prejudice" following the end of a third party's review process (Leyden, 2004).

The hearing ended on a sour note, committee chairman Mike Rogers closed by saying (Greene, 2012): "I can say that I'm a little disappointed today, I was hoping for a little more transparency... Other inconsistencies worry me greatly."

What the U.S. is Currently Doing to Address the Huawei Threat

On the 8th of October, 2012, *House Permanent Select Committee on Intelligence* (HPSCI) members Chairman Rogers and Ranking Member Ruppersberger held a news conference to announce the release of their report: "Investigative Report on the U.S. National Security Issues Posed by Chinese Telecommunications Companies Huawei and ZTE." The report came after an 11-month investigation by the committee, and the results do not bode well for Huawei (Wolf, 2012a). The report recommends that U.S. companies who are considering, or are currently doing business with Huawei or ZTE to find another vendor, and recommends that U.S. government systems exclude equipment or component parts from these companies (Rogers & Ruppersberger, 2012). The report illustrates the concerns that HPSCI has with Huawei and ZTE regarding their suspected connections with the Chinese government. The HPSCI fears that Huawei networking equipment could be purposefully modified to be utilized by the Chinese government or the PLA for malicious cyber activities via secret backdoors, which could be used for spying, intellectual property theft, or command and control of U.S. networks, and have thus blocked Huawei equipment from being used in critical infrastructure, and is the driving argument behind why the committee has advised the commercial industry to cease all dealings with Huawei.

The HPSCI report is the latest step in ongoing efforts by the U.S. to block Huawei from expanding into the U.S. marketplace. The panel argues that there are long-term security risks supposedly linked with Huawei's equipment and services, but does not provide any hard evidence to back up its concerns. The onus was placed on Huawei to prove its innocence in the face of charges which appear to be heavily based upon rumors, allegations, and speculations instead of hard facts, and in the eyes of the committee, they failed to dispel those concerns, and thus find themselves all but blacklisted in the U.S.

Huawei is the world's second-largest telecom company, operating in more than 150 countries, with more than two-thirds of its annual revenue of $32.4 billion being earned outside of China (Wolf, 2012b). With that kind of global presence and revenue being generated outside China's boarders, there is no question that Huawei has to be concerned with its global image, and the possible damages the damning HPSCI report may have on their growth and stability in the international market, particularly in nations who may wish to score points with the United States by standing in agreement with them on their decision to scrutinize or ban Huawei equipment in their countries. Fortunately for Huawei, not every nation is so reluctant to do business.

Huawei Counters: Unsubstantiated, Non-Specific Concerns

It would be economic suicide on the international stage for Huawei if they were to engage in the sort of activities the U.S. is concerned about. Mr. Charles Ding, Senior Vice President of Huawei, attempted to disarm U.S. concerns in his testimony before the HPCSI with the following arguments (Ding, 2012): “Huawei is an independent, employee-owned company that operates in more than 140 countries. Neither the Chinese Government nor the PLA is involved in business decisions. Much equipment used in U.S. networks is developed and manufactured in China. Improper behavior would blemish our reputation, and strike a fatal blow to the company's business operations. It would be immensely foolish to risk involvement in national security or economic espionage. Huawei has been in the U.S. for 10 years and has 1,700 employees here. In 2011 alone, Huawei procured $6.6 billion of goods and services from U.S. companies, and has invested $500 million in the U.S. over the last 5 years.” He requested HPSCI to provide any proof that Huawei has engaged in national security or economic espionage.

A Lot of Talking, but Where's the Proof?

Huawei strongly denies being under Chinese government control or having ties to the PLA, but Michael Juneau-Katsuya, a former senior intelligence officer who served as Asia-Pacific Bureau Chief for the Canadian Security Intelligence Service, says (Rohrlich, 2012): "It cannot exist any other way... The Chinese are masters at hiding their true intentions and they have been practicing it since Sun Tzu wrote *The Art of War* 2,000 years ago, what we saw was a masquerade, a smokescreen to make us believe these companies are not linked to the Chinese government.”

A report prepared for the US-China Economic and Security Review Commission by Bryan Krekel (2009) of Northrop Grumman, states that Huawei is a supplier of specialized telecommunications equipment, training and related technology to the PLA, and has received direct funding for Research and Development on Command, Control, Communications Computers, Intelligence, Surveillance and Reconnaissance (C4ISR) systems capabilities. He argued that Huawei originated as a state research institute and continues to receive preferential funding and support from the PLA. The report continues by stating that Huawei also provides certification training and related engineering training to PLA personnel assigned to communications and *Information Warfare* (IW) related positions – he points to provincial level Communist Party military newspapers as his source.

The most blatant finger-pointing comes from F. Michael Maloof (2012), a former senior security policy analyst in the Office of the Secretary of Defense, according to him the Chinese government has "pervasive access" to some 80% of the world's communications, giving it the ability to undertake remote industrial espionage and even electronic sabotage of critical infrastructures in the U.S. and other countries. He claims the Chinese government and PLA are acquiring their access through Huawei and ZTE. When considering Maloof's report, it should be noted that he was stripped of his security clearance in 2001.

Without Facts it's Rhetoric, Get to the Tech

Leave it to German hackers to present something tangible. Security researchers Felix Linder, the head of security firm Recurity Labs, and colleague Gregor Kopf, a security consultant, gave a presentation at this year's Defcon hacker's conference, which exposed vulnerabilities in Huawei routers. The vulnerabilities which include a session hijack, a heap overflow, and a stack overflow, were found in the firmware of Huawei's AR18 and AR29 series routers. These exploits could be used to take control of the devices over the Internet. Linder described the security of the devices as: "the worst ever," and said they are bound to contain additional vulnerabilities. In layman's terms, Linder says that an attacker could get access to the system, log in as administrator, change the admin password, and reconfigure the systems, which would allow for interception of all the traffic running through the routers. (Constantin, 2012. Miles, 2012).

In an interview with CNET after the Defcon talk, when asked about reports that Huawei routers have back doors per the Chinese government's request, Linder said: "They don't need to. You just need to have Huawei people running your network or help run your network... If you have so many vulnerabilities, they are the best form of attack vectors (Miles, 2012)."

Dan Kaminsky, Security expert and chief scientist at DKH, said: "If I were to teach someone from scratch how to write binary exploits, these routers would be what I'd demonstrate on. What Linder has shown is that the 15 years of secure coding practice that we've learned about – the things to do or not to do – have not been absorbed by the engineers at Huawei (Constantin, 2012)."

Though Linder and Kopf only tested two types of routers, in a resulting technical support publication by Huawei (Huawei, 2012a), they acknowledged that Recurity Labs had uncovered security vulnerabilities that affected AR18/28/46/19/29/49 series access routers and S20/30/35/39/51/56/78/85 series switches.

How Long Has the U.S. Been Attempting to Block Huawei's Growth in the U.S.?

This latest investigation into Huawei is just the latest link in an ongoing attempt to block Huawei from planting a foothold in the U.S. Huawei has made multiple attempts to grow their business in the U.S., but have faced major resistance, all on the grounds of national security. Some examples:

- In 2011, the Commerce Department blocks Huawei's application to build a wireless network for America's first responders (Lake, 2011).
- In 2011, CFUIS blocked a proposed investment in 3Leaf (Flicker, & Parsons, 2011).
- In 2010, CFUIS blocked a proposed investment in 2Wire and Motorola (Flicker & Parsons, 2011).
- In 2010, a group of Republican lawmakers raised concerns about Huawei's bid to supply mobile telecommunications equipment to Sprint Nextel Corp (Carew & Wohl, 2011).
- In 2008, CFUIS blocked a proposed sale of 3Com to Huawei (Lake, 2011).

What are the Ramifications of the U.S. Stance on Huawei?

On the eve of the HPSCI hearing, a paper by Dan Steinbock was published on Huawei's U.S. website. The paper alleges they are being blocked by the U.S. on false suspicions, and unsubstantiated "allegations based on allegations." The paper opens with a quote about the McCarthy-era Communist witch-hunting of the 1950s. It continues by stating the roadblock is not the American marketplace, but the U.S. government – the question is why? The paper says that if there is any substance to unstated allegations in Washington, these should be specified, in the absence of clearly stated and specific evidence, the case against Huawei does not represent U.S. values (Jowitt, 2012).

"Huawei employs 140,000 people worldwide, less than 1.3% of its personnel are in the U.S. In light of business potential this translates to missed opportunities... As long as barriers continue to deter Chinese foreign direct investment in the US the unequivocal message is that America is open for business, but not for Chinese business... Indeed, US trade and investment policy is at risk for being perceived as unipolar. In the long-run, such threats may return to haunt US corporations and *their* efforts to grow and expand in foreign markets, which are today more vital than ever before to American corporations (Steinbock, 2012)."

Huawei's Recent Success Around the World

Not every nation is as concerned as the U.S. in regards to Huawei's alleged security threats, suspected PLA ties, or assumed Chinese government control. Listed are just a few recent examples of success that Huawei is having in other nations:

- Huawei won its largest managed services contract in Europe through a five-year agreement with Sunrise in Switzerland (Middleton, 2012).
- Ethiopia's government is preparing to sign a two-year deal with Huawei and ZTE for a $1.3 billion government telecommunications contract (Davison, 2012).
- Huawei was awarded a multi-million dollar contract to supply fiber equipment for the Ultra-Fast Broadband (UFB) network in Christchuch, New Zealand (Beach, 2012).
- Brazil has signed a deal with Huawei to develop mobile Internet in 450 MHz band (Prescott, 2012a).
- Chile's Department of Telecommunications and Huawei have signed a wireless technology cooperation agreement (Prescott, 2012b).

Australia Following the Lead of the U.S. and Being Called Discriminatory

Australia's recent history with Huawei may serve to illustrate an example of one nation kowtowing to another in order to maintain its positive relationship. In 2010, RailCorp had no security concerns with Huawei when it awarded the $225 million contract to supply its mobile GSM system for railways; Australian intelligence agencies didn't raise any security concerns with Transport for NSW when Huawei was appointed (Colley, 2012). However, by 2011, Huawei was banned from working on the $37.4 billion Australian National Broadband Network (NBN) project based on similarly vague security concerns in line with those concerns the U.S. has with Huawei. A key difference between Australia and the U.S. is where the U.S. has warned all companies to discontinue current or future dealings with Huawei, Australia has publicly encouraged Huawei to grow their commercial success in Australia (Ramli, 2012).

In a submission to the committee, Huawei stated it was concerned that new laws could discriminate against companies from a particular country. "We believe the principle of non-discrimination should be clearly set out in any legislative reform," Huawei said, adding that companies should have a chance to address specific security concerns. In April, China's Ministry of Commerce expressed its anxiety about the Australian government's decision to ban Huawei from the project, calling the decision unfair (Grubel, 2012). The same opinion of the U.S. is sure to be echoed in China if the U.S. is unable to produce specific, concrete security concerns to back their position against Huawei.

Canada's Interaction with Huawei

Though Huawei has been met with stiff opposition in the U.S. and Australia, they have made major inroads in Canada. Both the Canadian and regional Ontario governments have shown an eagerness to work with Huawei, even officially praising the Chinese company's partnerships in Canadian telecom projects with Bell, Telus, SaskTel, and Wind mobile. Earlier this year, Prime Minister Stephen Harper visited China and said he was honored to have witnessed the signing of large contracts for Huawei to provide Bell and Telus with the latest LTE high-speed wireless networks throughout Canada (Weston, 2012).

That being said, as is the case in Australia where things were going without a hitch and then suddenly altered course, the day after the release of the HPSCI report blasted Huawei; Canada strongly indicated that it would exclude Huawei from helping build a secure Canadian government communications network because of possible security risks. This decision required Canada to invoke a national security exception to let it discriminate without violating international trade obligations (Palmer, 2012).

United Kingdom's Interaction with Huawei

One of the more puzzling factors in this case is the U.K. – arguably one of the closest U.S. allies, is acting in direct opposition to the stance the U.S. has taken in regards to Huawei. Last September, when HPSCI was prepping to warn American telecommunications networks about the dangers of dealing with Huawei, Prime Minister David Cameron of Britain, was posing for photos with Ren Zhengfei, the chief executive of Huawei. During that meeting Mr. Cameron announced that Mr. Zhengfei had agreed to expand the company's operations with an investment of $2 billion.

One Reason Britain may be warm to the idea of having Huawei expand there is the boost Huawei's presence will have on the UK economy. Huawei currently has 800 employees in Britain, and intends to create 700 more positions in the next five years. They also have a research center in Ipswich, and plan to increase technical centers throughout the country (Pfanner, 2012).

Mr. Cameron's government says it has no plans to change its relationship with Huawei in the wake of the U.S. committee's recommendations, but rather it utilizes a "trust-but-verify" approach to the partnership. Huawei set up a Cyber Security Evaluation Center two years ago in Banbury, England, where its engineers work alongside officials of Government Communications Headquarters, a British spy agency, to vet equipment that Huawei is producing for use in Britain (Pfanner, 2012).

Given the habitually close cooperation between U.S. and U.K. on matters of national security, seeing them take diametrically opposed approaches to Huawei is surprising.

More on the Evaluation Approach employed by the U.K.

Huawei has enjoyed successful growth in the United Kingdom, even when some of the United Kingdom's closest allies, such as the United States and Australia have met the company much more scrutiny, and in the case of the United States have declared the company a threat to national security and have simply decided the risk was too great to utilize their equipment. The United Kingdom curtailed its concerns with Huawei by installing checks and balances. The first major step to ensuring that Huawei equipment was safe for use was by way of implementing a Cyber Security Evaluation Centre in 2010 in Banbury, Oxfordshire (Huawei, 2012b). The evaluation centre is run with United Kingdom intelligence agency Government Communications Headquarters (GCHQ), which is the centre for Her Majesty's Government Signal Intelligence (SIGINT) activities (Chorley, 2012). The centre works to test networking equipment and software that will be sold in the United Kingdom. The testing is welcomed by the company, "Believe no one and check everything." Huawei's global cyber-security officer John Suffolk says (McClenaghan, 2012).

If one centre for keeping a close eye on Huawei in the United Kingdom wasn't enough, in 2011 the UK Internal Audit Centre of Excellence was established; further supporting Huawei's UK operations from both an internal control and governance perspective and to provide the impetus for the development of a leading edge global internal audit service in Huawei. It also aims to encourage collaboration between the company and the relevant internal audit professional institutions in both the UK and China (Comms Business, 2011).

Huawei is now looking to make a similar push in Australia to alleviate concerns there. John Lord, chairman of Huawei Australia says: "Huawei has done a very poor job of communicating about ourselves and we must take full responsibility for that." He followed up by stating that the company needed to be more open and would give the Australian authorities complete and unrestricted access to its software source code and equipment (Burt, 2012).

Lord says that Huawei is proposing the creation of a national cyber-security evaluation center, where telecom equipment from all vendors could be tested for security risks and vulnerabilities. The center would allow for greater transparency into communications technologies being used in Australia, and would be paid for by various telecom equipment vendors and operated by "security-cleared Australian nationals." He argues that no single country, agency, vendor, or telco (sic) has all the answers to solving cyber security issues (BBC, 2012).

The Heritage Foundation's Take on the Situation

Dr. Bucci and Dr. Scissors of The Heritage Foundation (2012) gave their thoughts on fixing the problem: "The Intelligence Committee's main finding is correct: Huawei and ZTE should not be considered reliable partners for work involving sensitive systems. It is equally true that the particular nature of telecommunications means that the Committee's finding should not be extended to other industries where there is no equivalent to cyber attack. There was a rush of Chinese investment into the U.S. in the first half of 2012, benefiting America considerably and bringing no security threat... other public and private-sector decisions are considerably more difficult. When national security is not involved, the private sector should take the lead on how best to respond to cyber and intellectual property threats. These will vary by sector and by the company and should not be evaluated at the level of national policy."

What Theories Exist Which Attempt to Explain the Huawei Threat?

There exists a plethora of theories which attempt to explain why the U.S. is so adamantly against using Huawei equipment, ranging from the realm of rational to that of conspiracy theories. The predominant theories/arguments are:

- Huawei really is a threat. According to the report by Rogers and Ruppersberger of the *US*

House Permanent Select Committee on Intelligence (HPSCI), even though the unclassified documents don't provide any clear facts to back the committee's decision to consider Huawei a threat to national security, they claim that there exist classified documents which do demonstrate the real and present threat (Rogers & Ruppersberger, 2012).

- Reuters reports that two people familiar with the Huawei Probe, who wished to remain anonymous, said that the 18-month review actually found no clear evidence of security vulnerabilities being intentionally placed in Huawei equipment. "We knew certain parts of government really wanted evidence of active spying... we would have found it if it were there," said one of the people. Additionally, Chris Johnson, a former CIA analyst on China, said that he had been told that the White House review had come up empty on past malicious acts (Menn, 2012).
- The HPSCI decision to block Huawei may have more to do with politics than with security concerns (Bremmer, 2012).
- State capitalism and the challenge it poses have expanded enough that the government is officially worried about them; free-market capitalism and state capitalism are increasingly at odds (Bremmer, 2012).
- According to John Lord, U.S. stance on Huawei is "protectionism, not security." Lord says: "The fiery rhetoric of the U.S. Committee's report may make good headline-fodder in an election year, but it should really be seen as a missed opportunity. It missed the opportunity to address the real issues at stake, to increase awareness of the common threats we face, and to develop methods of countering these threats in a realistic way. When all telecoms equipment is produced by an interdependent global supply chain, simply blacklisting a single vendor or country will not make critical infrastructure more secure (Sharwood, 2012)."
- Cisco is so entrenched with the U.S. government that open competition had to be controlled. "Cisco is likely to be one of the biggest beneficiaries of the investigation, and Huawei has suggested in the past that Cisco is often behind the efforts to besmirch Huawei's reputation. (Greene & Tibken, 2012).

How should current policy be changed, adapted, reformed, or replaced to deal with the Huawei threat?

The following sections will present alternatives to the United States government's policy on utilizing other nation's hardware or software for critical infrastructure. The recent case between the HPSCI and Huawei which ended in the U.S. denying Huawei the ability to bid on critical infrastructure projects in the U.S. has raised concerns that the U.S. may be acting in a prejudicial manner towards outside nations, or acting in a protectionist fashion.

According to the Department of Homeland Security (2008): "Protecting and ensuring the continuity of the critical infrastructure of the United States are essential to the nation's security, public health and safety, economic vitality, and way of life… Critical infrastructure are the assets, systems, and networks, whether physical or virtual, so vital to the United States that their incapacitation or destruction would have a debilitating effect on security, national economic security, public health or safety, or any combination thereof."

Below are my recommendations to U.S. policy makers on how to better handle and deal with threats they feel are present in companies like Huawei in the future:

- The U.S. government should be working hand-in-hand with third parties from the public sector to address emerging threats. A major effort should be made to reach out to individuals outside of the government arena for their technical expertise on security threats. A push for partnership with security industry leaders, hackers, and individuals who have highly-sought after technical skills, and a true passion and understanding of security, should be made to form a better way forward in identifying and uncovering threats and vulnerabilities. Better information sharing between the private and public sectors strengthens the whole.
- The U.S. government should expand works with global partners towards a global approach to dealing with cybersecurity research and development, and approaches on dealing with emerging threats and vulnerabilities. President Obama has hinted at this direction in the National Strategy for Global Supply Chain Security (2012): "The Federal Government cannot achieve this alone. Partnerships with state, local, and tribal governments, the private sector, and the international community are critical to realizing our shared goal of building a new framework to strengthen and protect this vital system."
- A drastic shift would be for the U.S. government to consider deploying a secure, closed-network for critical infrastructure. A network

which has no connectivity to the existing Internet would greatly mitigate the threat to critical infrastructures from other nations, non-state hackers, and hackers present within the United States itself.

- The U.S. government should base its decisions to not utilize products from companies such as Huawei on facts rather than ambiguous security concerns. Rather than accusing Huawei of being aligned with the People's Liberation Army, or China's government – with little proof of either, stick to tangible arguments for not utilizing Huawei's equipment:
  - Huawei's line of products are full of security holes and are programmed with substandard coding. They simply do not meet the stringent standards expected of equipment utilized in critical infrastructures. Until coding substantially improves, the risk is too great to utilize Huawei's equipment on U.S. critical infrastructure.
  - A majority of technicians working on critical infrastructure are trained and skilled in working on Cisco or Juniper equipment, the monetary cost involved with retraining these employees to manage and operate new, foreign equipment, would have be factored into the total cost.
- Institute a U.S. only, Commercial off-the-shelf (COTS) or Government off-the-shelf (GOTS) hardware/software policy for vital components of critical infrastructure and only allow desired or required equipment or software from other nations as an exception to policy. In doing so, outside nations would only be able to bid on, or compete for the contracts on select elements of critical infrastructure projects. The burden would then be put on the U.S. government to clearly demonstrate why they are choosing to utilize products from outside the U.S. (or off the GOTS list) if similar equipment or software existed or could be developed within a reasonable amount of time and at a reasonable cost from within the United States. Deciding to utilize equipment or software developed or manufactured in foreign nations would be done so with the knowledge that the door would have to be open for other nations to freely bid on those particular elements. For example, if Japan were to develop a new network intrusion detection solution that the U.S. government was interested in utilizing on critical infrastructure systems, by opting to utilize said gear, the floor would be open for China, the UK, and any other nation to pitch their own solution in fair competition.

Though this measure may seem somewhat extreme, until greater efforts are made to work with other nations to find clear solutions in regards to cybersecurity threats, it would negate any room for the arguments presented by foreign organizations or nations that the U.S. government is being selective, or acting in a protectionist manner, or generally being unfair to organizations because of their affiliations, real or imagined, secret or otherwise, with foreign security agencies or governments. From a risk management perspective, The U.S. government should simply declare that its nation's critical infrastructure is just that, critical, therefore they must do everything in their power to ensure that we are working to mitigate risks and alleviate threats.

What areas of current policy are more in need of dramatic reform?

The U.S. Government should not be advising private organizations or swaying free markets by way of instructing U.S. Companies not to buy Huawei, or any other IT gear coming from China, without first clearly demonstrating, with unequivocal proof, that there is a real and present danger associated with deploying and utilizing said equipment. They could warn that there's proof that the products do not match the quality of other vendors, but should not issue warnings without true examples.

According to NIST: Piloting Supply Chain Risk Management Practices for Federal Information Systems (Bartol, Moorth, & Swanson, 2010), the following steps should be followed in regards to supply chain concerns:

- Proper oversight of suppliers. This includes actively managing suppliers through contracts/ Service-Level Agreements (SLAs).
- Audit the development process. Use trusted third-party auditing mechanisms in the life cycle for assessing the exit criteria for each life cycle step (e.g., vetting the requirements analysis, architecture design).
- Perform quality assurance and quality control, e.g., of security features.

There currently exists no way for the U.S. to adequately perform these measure in regards to Huawei, either these steps need to be developed, or Huawei (or any foreign supplier for that matter) should immediately be disqualified from competing on critical infrastructure projects; the UK found a workaround by standing up security centers in the UK, where its security agencies

References

- Adams, P., Bakos, G., Krekel, B. (2012). Occupying the information high ground: Chinese capabilities for computer network operations and cyber espionage. Retrieved 27 September, 2012, from: *http://www.uscc.gov/RFP/2012/USCC%20Report_Chinese_CapabilitiesforComputer_NetworkOperationsandCyberEspionage.pdf*
- BBC (2012). Huawei offers access to source code and equipment. Retrieved 24 October, 2012, from: *http://www.bbc.co.uk/news/business-20053511*
- Barkakati, N., Dacey, R., Rhodes, K., Willemssen, J. (2004). GAO: Technology assessment: Cybersecurity for critical infrastructure protection. Retrieved 10 November, 2012, from: *http://www.gao.gov/new.items/d04321.pdf*
- Bartol, N., Moorthy, R., Swanson, M. (2010). NIST: Piloting supply chain risk management for federal information systems. Retrieved 06 November, 2012, from: *http://csrc.nist.gov/publications/drafts/nistir-7622/draft-nistir-7622.pdf*
- Beach, J. (2012). Huawei wins Kiwi fibre contract. Retrieved 07 October, 2012, from: *http://www.telecoms.com/39595/huawei-wins-kiwi-fibre-contract/*
- Bremmer, I. (2012). America's way or Huawei. Retrieved 26 October, 2012, from: *http://blogs.reuters.com/ian-bremmer/2012/10/26/americas-way-or-huawei/*
- Bucci, S., Scissors, D. (2012). China cyber threat: Huawei and American policy toward Chinese companies. Retrieved 23 October, 2012, from: *http://www.heritage.org/research/reports/2012/10/china-cyber-threat-huawei-and-american-policy-toward-chinese-companies*
- Burt, J. (2012). Huawei proposes security center in Australia. Retrieved 26 October, 2012, from: *http://www.eweek.com/networking/huawei-proposes-security-center-in-australia/*
- Capaccio, T. (2012). China's U.S. debt holdings aren't threat, Pentagon says. Retrieved 08 September, 2012, from: *http://www.bloomberg.com/news/2012-09-11/china-s-u-s-debt-holdings-aren-t-threat-pentagon-says.html*
- Carew, S., Whol, J. (2011). Huawei backs away from 3Leaf acquisition. Retrieved 29 September, 2012, from: *http://www.reuters.com/article/2011/02/19/us-huawei-3leaf-idUSTRE71I38920110219*
- Cheng, R., Greene, J. (2012). Inside Huawei, the Chinese tech giant that's rattling nerves in DC. Retrieved 25 September, 2012, from: *http://news.cnet.com/8301-1035_3-57484472-94/inside-huawei-the-chinese-tech-giant-thats-rattling-nerves-in-dc/*
- Chorley, M. (2012). British security at risk from deals with Chinese telecoms giant Huawei, ex-MoD chief warns. Retrieved 22 October, 2012, from: *http://www.dailymail.co.uk/news/article-2221484/British-security-risk-deals-Chinese-telecoms-giant-Huawei-ex-MoD-chief-warns.html*
- Colley, A. (2012). RailCorp had no problem with Huawei. Retrieved 12 October, 2012, from: *http://www.theaustralian.com.au/australian-it/railcorp-had-no-problem-with-huawei/story-e6frgakx-1226494077572*
- Comms Business (2011). Huawei UK established new Internal Audit Centre of Excellence. Retrieved 24 October, 2012, from: *http://www.commsbusiness.co.uk/RSS_News_Articles.cfm?NewsID=15167*
- Constantin, L. (2012). Hackers reveal critical vulnerabilities in Huawei routers at Defcon. Retrieved 28 September, 2012, from: *http://www.computerworld.com/s/article/9229785/Hackers_reveal_critical_vulnerabilities_in_Huawei_routers_at_Defcon?taxonomyId=12*
- Davison, W. (2012). ZTE, Huawei to be awarded Ethiopian telecommunications contracts. Retrieved 11 October, 2012, from: *http://www.bloomberg.com/news/2012-10-11/zte-huawei-to-be-awarded-ethiopian-telecommunications-contracts.html*
- Department of Homeland Security (2008). Critical infrastructure sectors. Retrieved 05 November, 2012, from: *http://www.dhs.gov/critical-infrastructure-sectors*
- Ding, C. (2012). Written statement for Charles Ding: Permanent Select Committee on Intelligence, U.S. House of Representatives. Retrieved 21 September, 2012, from: *http://intelligence.house.gov/sites/intelligence.house.gov/files/documents/091312HuaweiTestimony.pdf*
- Eckert, S. (2006). Protecting critical infrastructure: The role of the private sector. Retrived 10 Novenber, 2012, from: *http://www.ridgway.pitt.edu/LinkClick.aspx?fileticket=Bezaq7AdjxA%3D&tabid=233*
- Flicker, S. M., Parsons, D. M. (2011). Huawei – CFIUS Redux: Now it gets interesting. Retrieved 29 September, 2012, from: *http://www.paulhastings.com/assets/publications/1868.pdf*
- Greene, J. (2012). Lawmakers frustrated by Huawei, ZTE during hearings. Retrieved 25 September, 2012, from: *http://news.cnet.com/8301-1035_3-57512430-94/lawmakers-frustrated-by-huawei-zte-during-hearings/*
- Greene, J., Tibken, S. (2012). Lawmakers to U.S. companies: Don't buy Huawei, ZTE. Retrieved 26 October, 2012, from: *http://news.cnet.com/8301-1035_3-57527782-94/lawmakers-to-u.s-companies-dont-buy-huawei-zte/*
- Grubel, J. (2012). China's Huawei urges Aurstralia not to discriminate on telco security. Retrieved 26 September, 2012, from: *http://uk.reuters.com/article/2012/09/14/us-australia-huawei-security-idUKBRE88D05820120914*
- Huawei (2012b). About Huawei in the UK. Retrieved 22 October, 2012, from: *http://www.huawei.com/uk/about-huawei/huawei-uk/index.htm*
- Huawei (2012a). Statement on Recurity Lab revealing security vulnerabilities in Huawei AR series routers. Retrieved 28 September, 2012, from: *http://support.huawei.com/support/pages/news/NewsInfoAction.do?actionFlag=view&doc_id=IN0000052595&colID=ROOTENWEB%7CCO0000000170*
- Jowitt, T. (2012). Huawei complains at US ‚roadblock' on trade. Retrieved 29 September, 2012, from: *http://www.techweekeurope.co.uk/news/huawei-slams-us-trade-roadblock-92602*
- Krekel, B. (2009). Capability of the People's Republic of China to conduct cyber warfare and computer network exploitation. Retrieved 25 September, 2012, from: *http://www.uscc.gov/researchpapers/2009/NorthropGrumman_PRC_Cyber_Paper_FINAL_Approved%20Report_16Oct2009.pdf*
- Lake, E. (2011). China bid blocked over spy worry. Retrieved 28 September, 2012, from: *http://www.thedailybeast.com/articles/2011/10/11/u-s-blocks-china-telecoms-bid-to-build-wireless-network-over-spying-concerns.html*
- Leyden, J. (2004). Cisco drops Huawei lawsuit. Retrieved 28 September, 2012, from: *http://www.theregister.co.uk/2004/07/29/cisco_huawei_case_ends/*
- Maloof, F. M. (2012). China: ‚Pervasive access' to 80% of telecoms. Retrieved 27 September, 2012, from: *http://www.wnd.com/2012/07/chinese-have-pervasive-access-to-80-of-worlds-telecoms/*

work alongside Huawei developers, and are able to address issues on the spot.

There exists a lot of speculation, skepticism and concern from the U.S. over the supposed ties between Huawei, the Chinese government and the PLA. The concerns themselves may be valid, but without the backing of tangible evidence, the U.S. is skirting defamation of Huawei. Instead of speculating on the unknown, perhaps the HPSCI should take a page from the German hackers and focus on real, tangible concerns such as Huawei equipment's firmware having such slapdash coding that secret Chinese backdoors aren't required.

If the U.S. has true security concerns involving Huawei, they should be made transparent to both the public and to Huawei. Presenting unsubstantiated allegations as the reason not to allow a global telecommunications leader to do business in the U.S. is not in aligned with its free market system, and may lead to the U.S. being perceived as being "discriminatory" like its Australian allies.

The U.S. is quite happy to do business with every other Chinese company; facts being facts, the majority of U.S. material goods market is unquestionably dominated by Chinese goods, the two economies are symbiotically dependent upon one

References

- McClenaghan, M. (2012). Chinese telecoms giants are taking over the world. Should we be scared? Retrieved 22 October, 2012, from: *http://www.thebureauinvestigates.com/2012/08/06/chinese-telecoms-giants-are-taking-over-the-world-should-we-be-scared/*
- Menn, J. (2012). White House-ordered review found no evidence of Huawei spying: sources. Retrieved 18 October, 2012, from: *http://www.reuters.com/article/2012/10/18/us-huawei-spying-idUSBRE89G1Q920121018*
- Middleton, J. (2012). Huawei wins outsourcing deal in Switzerland. Retrieved 07 October, 2012, from: *http://www.telecoms.com/45288/huawei-wins-outsourcing-deal-in-switzerland/*
- Mills, E. (2012). Expert: Huawei routers are riddled with vulnerabilities. Retrieved 29 September, 2012. from: *http://news.cnet.com/8301-1009_3-57482813-83/expert-huawei-routers-are-riddled-with-vulnerabilities/*
- Obama, B. (2012). National strategy for global supply chain security. Retrieved 07 November, 2012, from: *http://www.whitehouse.gov/sites/default/files/national_strategy_for_global_supply_chain_security.pdf*
- Palmer, R. (2012). Update 1 – Huawei faces exclusion from planned Canada government network. Retrieved 09 October, 2012, from: *usa-china-huawei-canada-idUKL1E8L9J6020121009*
- Pfanner, E. (2012). Chinese telecom firm finds warmer welcome in Europe. Retrieved 10 October, 2012, from: *http://www.nytimes.com/2012/10/12/business/global/huawei-chinese-telecom-company-finds-warmer-welcome-in-europe.html?pagewanted=all&_r=0*
- Prescott, R. (2012a). Brazil signs deal with Huawei to develop mobile Internet in 450 MHz band. Retrieved 07 October, 2012, from: *http://www.rcrwireless.com/americas/20120712/internet/brazil-govt-signs-agreement-huawei-develop-mobile-internet-450-mhz/*
- Prescott, R. (2012b). Chile, China sign wireless technology cooperation agreement. Retrieved 07 October, 2012, from: *http://www.rcrwireless.com/americas/20120627/industry-2/chile-china-sign-wireless-technology-cooperation-agreement/*
- Ramli, D. (2012). Huawei suggests NBN ban is political. Retrieved 10 October, 2012, from: *http://www.afr.com/p/technology/huawei_suggests_nbn_ban_is_political_7petLqXkAQ82ZfA5i3hgCI*
- Rogers, M. (2012). Opening statements for Chairman Rogers: Hearing on threat posed by Chinese telecommunications companies. Retrieved 20 September, 2012, from: *http://intelligence.house.gov/sites/intelligence.house.gov/files/documents/09132012RogersOpening.pdf*
- Rogers, M., Ruppersberger, D. (2012). Investigative report on the U.S. national security issues posed by Chinese telecommunications companies Huawei and ZTE. Retrieved 25 October, 2012, from: *http://intelligence.house.gov/sites/intelligence.house.gov/files/documents/Huawei-ZTE%20Investigative%20Report%20(FINAL).pdf*
- Rohrlich, J. (2012). Chinese military tied to telecoms, says former intelligence agent. Retrieved 24 September, 2012, from: *http://www.minyanville.com/sectors/technology/articles/Huawei-Chinese-government-House-Intelligence-Committee/9/18/2012/id/44118?page=1*
- Sharwood, S. (2012). Huawei says US stance is ,protectionism.’ Retrieved 24 October, 2012, from: *http://www.theregister.co.uk/2012/10/24/huawei_accuses_usa_of_protectionism/*
- Steinbock, D. (2012). The case for Huawei in America. Retrieved 30 September, 2012, from: *http://www.chinausfocus.com/foreign-policy/unipolar-trade-in-a-multipolar-world-the-case-for-huawei-in-america/*
- U.S. Treasury (2012). Major foreign holders of treasury securities. Retrieved 09 October, 2012, from: *http://www.treasury.gov/resource-center/data-chart-center/tic/Documents/mfh.txt*
- Weston, G. (2012). Chinese firm’s Canadian contracts raise security fears. Retrieved 07 October, 2012, from: *http://www.cbc.ca/news/politics/story/2012/05/15/pol-weston-huawei-china-telecom-security-canada.html*
- Wolf, J. (2012a). U.S. lawmakers seek to block China Huawei, ZTE U.S. inroads. Retrieved 08 October, 2012, from: *http://intelligence.house.gov/press-release/chairman-rogers-and-ranking-member-ruppersberger-warn-american-companies-doing*
- Wolf, J. (2012b). U.S. panel to probe new wave of complaints against Huawei, ZTE. Retrieved 10 October, 2012, from: *http://articles.chicagotribune.com/2012-10-10/news/sns-rt-us-usa-china-huawei-ztebre8960nh-20121007_1_huawei-and-zte-chinese-telecom-equipment-makers-representatives-intelligence-committee*

another. Nearly every phone, computer, or networking component, to include the U.S. government's preferred networking suppliers, Cisco and Juniper, are either wholly or partially manufactured, assembled, or touched in some fashion by Chinese hands.

If the decisions made by U.S. lawmakers are honestly based on security concerns, on the back of the argument that Huawei may be required to cooperate with requests of the Chinese government to modify equipment for malicious purposes, wouldn't the same argument apply to every company in China?

In absence of hard facts, it's appropriate to question if there exist alternative reasons for the U.S. to ban Huawei. Could decisions have more to do with American manufacturers Cisco and Juniper having a hard time competing with low-cost Chinese rivals? Are we witnessing a form of economic protectionism, instead of earnest national security concern? Could this aid in explaining why the U.K. is willing to work with Huawei even when its close ally is issuing strong warnings against Huawei?

If China were really interested in damaging the U.S., it wouldn't require telecommunication backdoors or espionage, they could simply cash in $1.1 trillion of U.S debt they hold (U.S. Treasury, 2012), but according to the Pentagon, China's debt holdings aren't a threat (Capaccio, 2012).

Regardless of how U.S. lawmakers feel about Huawei, there exists no tangible 'smoking gun' to connect Huawei to the Chinese government or the PLA in such a way to factually deem them a threat to national security. Instead of simply saying no to the company, lawmakers should have considered, much like our closest allies have, a workable solution to the problem, such as the approach the United Kingdom has taken with its security centre solution.

Huawei seems to be more than happy to provide full disclosure with the United Kingdom, and work with Britain's intelligence agencies to vet their hardware/software to alleviate concerns. Where there's a will there's a way, the United States could opt to have a similar center for analyzing security vulnerabilities in Huawei's equipment, or perhaps could opt for a "Chinese owned, American made" solution.

With the available information demonstrating no clear, present threat, this author can only conclude the United States government is not acting in line with our free market, capitalist principles, but rather it is picking winners and losers in regards to network providers.

Conclusion: How would these alternative policy prescriptions help solve or improve the flaws in current policy?

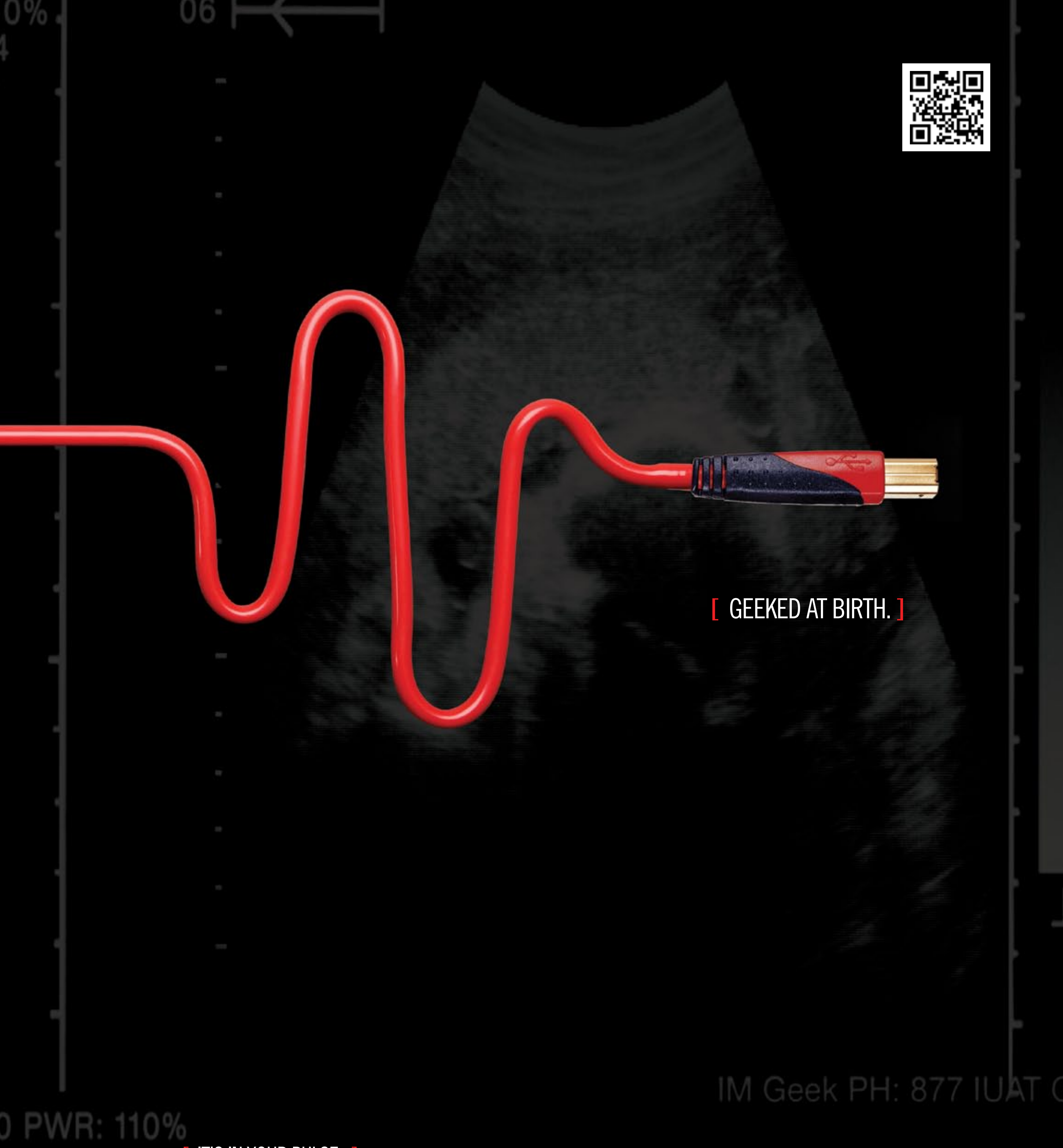
By working to bridge public and private sectors, including third-party individuals and working with global partners to come up with new solutions, the U.S. government would have a much broader pool of talent to work from, which would likely lead to solutions which they themselves may not be able to devise themselves may be presented. "With approximately eighty-five percent of U.S. key infrastructures privately owned or operated, the private sector is an increasingly important actor in the new security issues associated with homeland security. (Eckert, 2006)." Having that much skin in the game, there's no question that the private sector should have a voice, and make contributions in the way forward.

For the interim, until practical, measurable, and efficient measures and controls can be erected to ensure the security of critical infrastructures, the government should consider utilizing U.S. only GOTS/COTS equipment, or working to develop a closed-off network which would add an additional layer of security to critical systems.

This alternative to current policy direction would lend to ensuring that U.S. critical infrastructure was as secure as it could be, and would lessen the international view of the U.S. acting in a prejudicial manner towards outside nations, or acting in a protectionist fashion.

Security of critical infrastructure systems should be on the top of policy maker's lists, but approaches taken to address threats should be done in such a way as to not offend trading partners or the rest of the world.

TERRANCE J. STACHOWSKI, CISSP, L|PT



[IT'S IN YOUR PULSE.]

LEARN:

- Advancing Computer Science
- Artificial Life Programming
- Digital Media
- Digital Video
- Enterprise Software Development
- Game Art and Animation
- Game Design
- Game Programming
- Human-Computer Interaction
- Network Engineering
- Network Security
- Open Source Technologies
- Robotics and Embedded Systems
- Serious Game and Simulation
- Strategic Technology Development
- Technology Forensics
- Technology Product Design
- Technology Studies
- Virtual Modeling and Design
- Web and Social Media Technologies

www.uat.edu > 877.UAT.GEEK

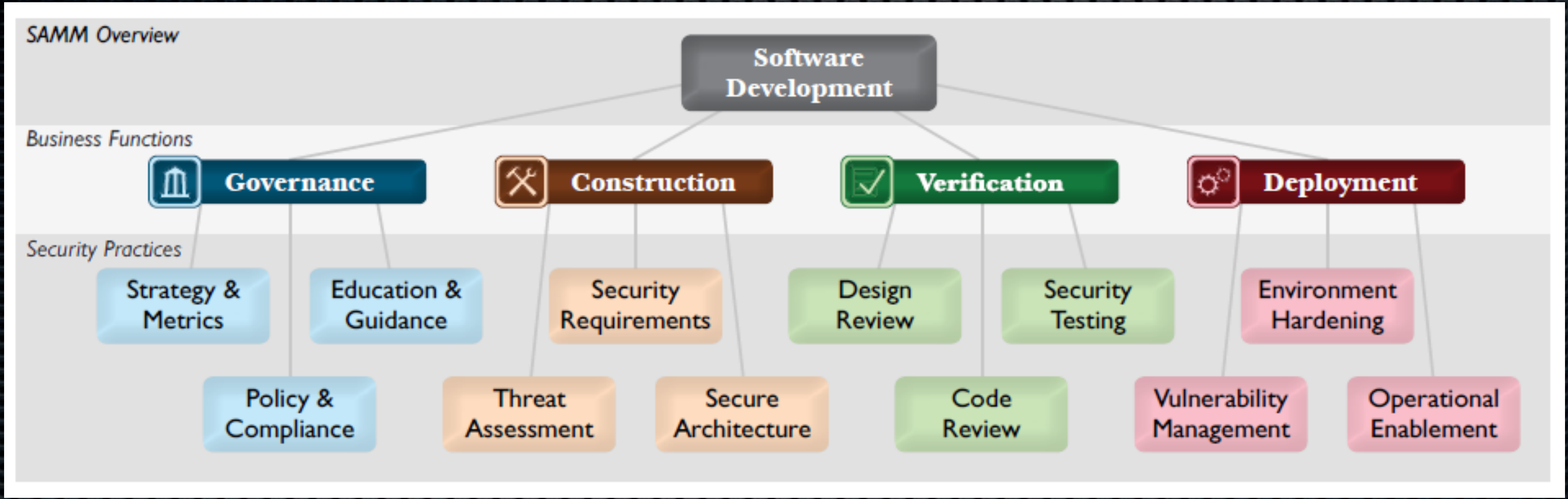
University of Advancing Technology
UAT
Learn. Experience. Innovate.

You can talk the talk.
Can you walk the walk?

PLEASE SEE **WWW.UAT.EDU/FASTFACTS** FOR THE LATEST INFORMATION ABOUT DEGREE PROGRAM PERFORMANCE, PLACEMENT AND COSTS.

OWASP Foundation

"We help protect critical infrastructure one byte at a time"



SAMM Overview

Software Development

Business Functions

Governance

Construction

Verification

Deployment

Security Practices

Strategy & Metrics

Policy & Compliance

Education & Guidance

Threat Assessment

Security Requirements

Secure Architecture

Design Review

Code Review

Security Testing

Vulnerability Management

Environment Hardening

Operational Enablement

- **140+** Checklists, tools & guidance
- **150** Local chapters
- **20,000** builders, breakers and defenders
- **Citations:** ***NSA, DHS, PCI, NIST, FFIEC, CSA, CIS, DISA, ENISA*** and more..

Learn More: http://www.owasp.org