HAKIN9

PRACTICAL PROTECTION IT SECURITY MAGAZINE

MALWARE

MALWARE – THE GOOD AND THE BAD

POWER IN THE DATA CENTER: STANDARDS & PRACTICES

Vol.7 No.08
Issue 08/2012(56) ISSN: 1733-7186

PLUS

TOOL TIME: WEBHTTRACK
(IL)LEGAL: THE $35,000,000,000 PROBLEM
EXTRA: ROOTKITS

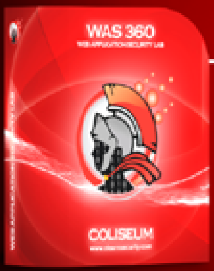
PENETRATION TESTING PROFESSIONAL v. 2

Online Penetration Testing Course

www.elearnsecurity.com

- ✓ 2400+ interactive slides
- ✓ 9 hours video training material
- ✓ 100% hands-on with Hera Labs
- ✓ Extremely in depth and thorough contents
- ✓ Leads to Hands-on ECPPT certification
- ✓ 3 Knowledge domains
- ✓ Web application penetration testing
- ✓ Network penetration testing
- ✓ System security and Exploit Development
- ✓ Lifetime access to course material

Now the most Hands-On course on Penetration Testing :



Coliseum Web Application Security Lab

- ✓ 14 real world vulnerable websites
- ✓ User-exclusive sand-boxed access to labs
- ✓ Multiplatform : PHP, MySQL, MS SQL Server
- ✓ Practice OWASP Top 10
- ✓ Web app analysis, XSS, SQLi, LFI/RFI, CSRF
- ✓ Get inline help if you get stuck

Hera Penetration Testing Virtual Lab

- ✓ VPN access from your own Attack box
- ✓ User-exclusive, non-shared access to labs
- ✓ Guided Exploitation Walkthrough
- ✓ Windows Servers, BSD, Linux, Firewalls, IDS's
- ✓ Different Labs with Different Network topologies
- ✓ On-demand: No Activation, No Expiration

www.elearnsecurity.com

HEY! TEACHER!

LEAVE THEM KIDS ALONE!

THE MOST ADVANCED COURSE ON PENETRATION TESTING

IS SELF-PACED!

WWW.ELEARNSECURITY.COM

Hakin9
team

Editor in Chief: Grzegorz Tabaka
grzegorz.tabaka@hakin9.org

Managing Editor: Marta Jabłońska
marta.jablonska@hakin9.org

Editorial Advisory Board: *Richard Batka, Randy Naramore, Udi Shamir, Bob Johnston, Yury Chemerkin, Remus Ho, Mervyn Heng, Kelly R. Kohl*

Proofreaders: Bob Folden, Nick Malecky

Top Betatesters: Nick Baronian, John Webb, Ivan Burke

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@software.com.pl

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

DTP: Ireneusz Pogroszewski
Art Director: Ireneusz Pogroszewski
ireneusz.pogroszewski@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

Dear Readers,

*I hope you are all well. This is isse is mostly devoted to malware. First article by Richard Batka. At the data center you use 208V single phase high line power [208 V * 24 AMP = 4992 VA or 4.9 KVA] Pushing higher volts is more energy efficient and can actually deliver more power.*

Data centers can be supported with something called three phase power. In this case 208 VOLTS are pushed through 13.8 AMPS x 3 which provides 8611 VA or 8.6 KVA. More in "Power in the data center: standards&practices". Next article By Randy Naramore discusses the good and the bad on malware. Internet threats have played a major role in the research and analysis of malware / viruses, causing companies like Symantec to sponsor research and development of malware fighting tools to better equip consumers to protect themselves from these threats. These consumers are better informed of the dangers of the internet and as a result are ever conscious of the hackers who want to steal their information in order to commit fraud. The malware scene is overly mature, while on the other hand its "releases" usually tend to have extremely short lifecycles, and quickly become part of a family of variations. These and more you will find inside!

We also have for you Social Network Privacy Guide part II by Yury Chemerkin. As an extra article we present you very informative piece of work: " Rootkit Detection Through Open Source Rootkit Detection Software".

It is the last magazine I prepared for you. I want to thank all the authors, betatesters and proofreaders who were helping me to give you the best magazine. I wish you all the best.

So, for the last time:

Marta & Hakin9 Team

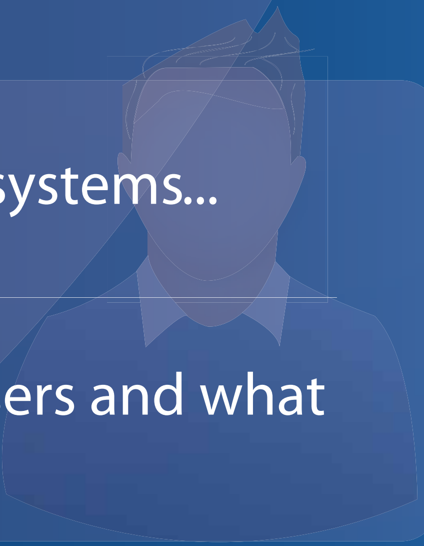
Cheers!

idtheft protect

Be reactive...

- Your systems are being attacked 24 hours a day...
- You understand the threats and are protected against them...

Be proactive...



- My users' behaviour threatens our systems...
- I understand what motivates my users and what threats are coming my way...

ID Theft Protect provides information on threats from a user perspective.

Visit: **http://id-theftprotect.com**

Power In The Data Center: Standards & Practices

First it should be said that all power is basically the same. The power that makes things run at work is the same power that runs things at home.

What you will learn…

- Data center power components
- Basic power calculations
- Efficiency in the data center

What you should know…

- Data center layout and standard power elements

At home you use 120V single phase low line power [120 V * 24 AMP = 280 VA or 2.8KVA]. At the data center you use 208V single phase high line power [208 V * 24 AMP = 4992 VA or 4.9 KVA]. Pushing higher volts is more energy efficient and can actually deliver more power.

What Is 3-Phase Power?

Data centers can be supported with something called three phase power. In this case 208 VOLTS are pushed through 13.8 AMPS x 3 which provides 8611 VA or 8.6 KVA. The use of 3-Phase power is becoming more popular because of its ability to deliver power efficiently.

A WATT(W) measures the real power drawn by the load equipment and is also used as a measurement of the power and heat generated by the equipment. You will hear people sometimes refer to something called a "power factor" which is the ratio of real power to apparent power. Use of a circuit breaker to protect electrical equipment from damage caused by overload or short circuits is common.

Plugs

One thing people don't think about with much frequency are the plugs used in a modern data center. We only have a few standards to worry about-In North America we have the NEMA standard. NEMA stands for National Electrical Manufactures Association. The other standard is IEC – *International Electrotechnical Commission*. NEC (National Electrical Code) is the body that defines the safe installation on of electrical wiring and equipment in North America.

Definitions

- AMPERE (AMP): Measures the amount of electrical current flowing through a circuit during a specific time period.
- VOLT (V): Difference of electric potential between two points on a conducting wire.
- VOLT-AMPS (VA): Equals VOLTAGE x AMPS. This rating is the apparent power, which is the maximum power that a device can draw.
- KILOVOLT-AMPS (KVA): Is the same as above just measured in thousands.
- RPP: Remote Power Panel. A breaker panel like the one you have in your home.
- WHIP: Connects to the RPP on one end and has a series of outlets or connectors on the other end. Servers plug directly into the WHIP or power connectors on a rack PDU.
- BREAKER PANEL: Has 42 positions sometimes called polls. A single phase 120v breaker will utilize 1 poll or 1 position.
- BMS: Building Management System
- PUE: Total Facility Energy / IT Equipment Energy
- PDU: Power Distribution Unit
- CRAC: Computer Room Air Conditioning
- DCO: Data Center Operator

Tip
Spend time reading output from all 3 groups: NEMA, IEC, and NEC.

The NEC code says that a PDU cannot allow a continuous measured load that exceeds more than 80% of a connector or cable rating for a period of more than 3 hours as defined by the NEC. This can also be called a Derated Load (30 AMP Rack PDU = MAX Load 24 AMP).

Power redundancy is critical. Without redundancy when the PDU fails all the equipment in the rack will fail. All equipment should have multiple redundant power supplies and plug into at least two different PDU's.

Tip
Never go above 50% of PDU capacity.

The data center of today typically represents a significant part of a company's asset base. When we say data center we can be talking about a room that houses a few servers to a dedicated facility with over 200,000 square feet of dedicated space.

Colo Facilities

Here in New York City (like most other major cities) we have "CoLo" facilities which are sometimes referred to as hosting hotels. The power component of any data center receives the most attention because it represents one of the largest line items on the Information Technology budget. Use of a CoLo facility can offer you some advantages due to economies of scale.

Effective use of power is important. Phones, servers, monitors, and cell phone towers generate 5% of global greenhouse gas emissions and todays data center is responsible for 15% of that.

Building A Data Center Efficiently

There are some very simple design choices that you focus on when building a data center. All of these recommendations specifically focus on assisting the data center to use power more efficiently.

Measure Pue

Make sure you have the capability to measure the PUE. PUE stands for Power Usage Effectiveness. The formula is simple: PUE = Total Facility Energy / IT Equipment Energy. Basically PUE is the ratio of Total Facility Energy to IT Equipment Energy within your Data Center.

Fact
PUE tells you how effectively you deliver power and cooling to the data center equipment.

Historically speaking- in 2006 the standard PUE of a data center was 2.0. This means that for every 1W of energy consumed 1W of energy overhead was spent by the facility to produce the power and cooling. Work to optimize your data center power consumption in order to lower your PUE.

Fact
A PUE of 1.0 is the ultimate goal.

Fact
In 2011, one data center PUE was recorded as being 1.09.

When it comes to measuring PUE you want to measure as frequently as possible. Every second is preferable. The recorded reading times should be between a quarter to 1 year to allow you to get trending data. Basically, the more frequently you can measure your PUE the more meaningful the results will be.

Tip
Incorporate PUE data into your Building Management System.

Manage Airflow

As I mentioned above first you will want to have the ability to accurately measure your PUE. Once this is achieved this you should move your attention to the management of the Air Flow.

Tip
Separate hot and cold air.
The industry has not created one universal way to do this yet. Conducting a *Computational Fluid Dynamics* (CFD) analysis will help you understand where the air is flowing and where the hotspots are. The design choices you make should be based on the results of the CFD analysis. Don't eliminate the possibility of retrofitting *Computer Room Air Conditioning* (CRAC) systems to achieve greater efficiency.

A company can use cheap available material to build these containment zones between the cold isle and the hot aisle. Some of the more commonly used material is:

- Thin aluminum
- Meat locker curtains

Make sure you seal the gaps between missing hardware in your racks. Do this with face plates that cover space gaps between equipment. Google recently spent $25k in basic material and achieved $65k in savings/year.

Temprature

I remember the first time I went into a data center. It was a hot summer day and I was delivering a new Novell workstation to the office of the DCO (Data Center Operator) whose office was inside the data center. The room was very cold. I remember being amazing by how cold the room felt.

Fact

For a long time the prevailing view was that the data center needed to be at a fixed 72 degrees Fahrenheit or cooler- This is no longer true.

ASHRAE- American Society of Heating, Refrigerating and Air-Conditioning engineers has defined the safe zone to include (up to) 80 degrees Fahrenheit (ASHRAE *www.ashrae.org*). Google runs one corporate data center at 80 degrees Fahrenheit. The center handles 200KW load. Basically- Using less energy = lower carbon footprint.

Natural Cooling

Natural cooling is also referred to free cooling. Without getting into too much detail- Use natural cooling when possible. A practical example would be locating the data center near a reliable water source.

Power Distribution

A typical data center power run looses power at every step. First you need to pull power in from the electrical grid. Then you need to convert the power down to match the voltages that will match the equipment you are hosting in the data center.

This requires multiple conversion stages.

Tip

Reduce the number of conversion stages and you will save power and money.

Learning From The Best

The smart people at Google have figured this out and have also worked hard to help the industry learn about efficient data center design and good power management. One of the areas they identified as a major waste was the UPS: Uninterruptible Power Supply.

Uninterruptible Power Supply Ups

UPS's are typically DC voltage and sit between the input source of power to the facility (AC) and the equipment. This requires a conversion step from AC to DC to charge the batteries. When you lose power and need to utilize those battery's- buy using an inverter to go from DC to AC- the AC then needs to be converted to DC for the equipment in the server rack.

Resources

ASHRAE – American Society of Heating, Refrigerating and Airconditioning Engineers. *www.ashrae.org*

Fact

The modern data center utilizes a design of no less than 3 conversion stages that waste power and money.

Google said why don't we try this- put the UPS onboard the rack/server component which eliminates most of the conversion steps- delivering DC directly to the equipment. #Genius.

Tip

Look to evaluate the efficiency of your AC/DC onboard power supplies. Google says it saves $30/ per year * number of servers with this approach.

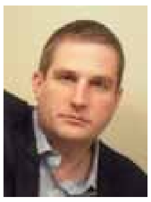
Finance

Industry wide if you follow commonly accepted data center best practices you will achieve cost effectiveness within 12 months.

Summary

In this article we covered the basics of power and power efficiency. The key is to take a holistic approach to power efficiency. Demand a lot from your go to vendors of choice. Set up a structure where vendors can work with each other towards a common goal of power efficiency in the data center. Ask a lot of questions and collect quantifiable numbers at every possible touch point.

RICHARD C. BATKA



Richard C. Batka is a business & technology executive based in New York City.

He has worked for global leaders Microsoft, PriceWaterhouseCoopers, Symantec, Verizon, ThomsonReuters and JPMorgan Chase. A graduate of New York University (honors) he has published multiple books and his articles have appeared in Data Center Magazine, International Linux+ Magazine, Hakin9 Magazine, UNIX/BSD Magazine, and PenTest Magazine. He can be reached via LinkedIn InMail and followed on Twitter at http://twitter.com/richardbatka.

VERACODE
Securing the Software That Runs the World

An ounce of prevention is worth a pound of cure.

We know it's not simple, but we've got the software and solutions to relieve your application security headaches.

Visit us online at www.veracode.com

Malware – The Good and The Bad

In this day and age the internet is as much a way of life as apple pie and baseball but with the endless information that the internet offers there are pitfalls that the internet offers as well. Malware and viruses are a big part of everyday life on the internet so you must take precautions to make sure you are protected from your personal information being stolen as well as the health of your pc or mac.

What you will learn…

- Good and bad about malware

What you should know…

- Basics on malware

The health of your machine is dependent on making sure all of your data is backed up, having updated antivirus software and using good security practices with regards to the websites we visit.

Malware has a huge impact on the global economy as a whole from the antivirus software companies who employ thousands of people to the people who manufacture the firewalls we use to prevent hackers and malware infections from getting into our networks. In various ways malware is a good thing in that we are forced to deal with the bad part of internet life. Knowing that you can be infected while doing common tasks like opening an email, clicking on a link or installing a new program, is a great reminder to be careful with what you do online and offline as well. Having an antivirus program, spyware removers and even the latest secure browsers, will do nothing to protect you if they are not up to date. Take the time to create a weekly maintenance plan that includes downloading the latest Windows updates, get the latest protection definitions for the antivirus program and update any other security software you use. Malicious software has become a critical security threat to all who rely on the Internet for their daily business, whether they are large organizations or home users. While initially a nuisance more than a threat, viruses, worms and the many other variants of malware have developed into a sophisticated set of tools for criminal activity. Computers around the world, some estimate as many as one in five, are infected with malware, often unknown to the owner of the machine. Many of these infected machines are connected through so-called botnets: networks of computers that operate collectively to provide a platform for criminal purposes. These activities include, but are not limited to, the distribution of spam (the bulk of spam now originates from botnets), hosting fake websites designed to trick visitors into revealing confidential information, attacking and bringing down websites, enabling so-called click fraud,‘ among many other forms of often profit-driven criminal uses. There are also reports that indicate terrorist uses of malware and botnets. As security comes at a cost, tolerating some level of insecurity is economically rational. From an economic perspective, the key question is whether the costs and benefits perceived by market players are aligned with social costs and benefits of an activity. In certain situations, the security decisions of a market player regarding malware may be rational for that player, given the costs and benefits it perceives, but its course of action may impose costs on other market players or on society at large. These costs are typically not taken into account by the market player making the initial decision.

The most dangerous group of virus writers is hackers or groups of hackers who intentionally create malicious programs in their own interests. They create such virus and Trojan programs which steal access codes to bank accounts, obtrusively advertise products or services illegally use resources of the infected computer for the purpose of getting money again – to develop spam-business or arrange distributed network attacks further aiming at blackmailing. Malware is the short-cut term for malicious software. It is devised to penetrate your computer without your consent. This term is used to include various forms of aggressive, invasive and bothersome applications. Meanwhile, a computer virus is a program that can copy itself and eventually infects your computer. Though a virus is sometimes referred to malware, spyware and adware, it does not justify such relation for they cannot replicate their selves.

A virus can only infect your computer through the use of the Internet or if carried by removable means like USB drive, floppy disk, CD or DVD. A worm, for example, is self-replicating and spreads to computer networks. It is also used by computer hackers to gain access to your personal information. On the other hand, a Trojan horse is a non-replicating type of malware that pretends to carry out a program beneficial for the user but makes it possible to allow unauthorized access of suspicious sources to your computer. This can be obtained from free software download websites and e-mail attachments.

While a rootkit is a set of programs that are designed to manipulate computer networks, it is also widely used by hackers to constantly gain access to your computer without being detected. Backdoors are just about the same with rootkits as it hides the fact that a hacker has already penetrated your computer. Spyware collects pieces of information about you through the data in your hard drive. It can download its self to your computer and further download other software, which can make your computer run slower. All of these threats to everyday users of the internet have caused corporations, banks, etc. to have to hire specially trained people who have the skills to deal with and counteract the effects of malware so in that way malware is somewhat of a necessary evil.

As time goes on the number of viruses and malware directed at Mac and Linux will undoubtedly increase as the number of people who use these operating systems increase. The numbers of people using windows based machines are more of an

attractive target to the people who write and propagate the malware / viruses. This will in turn force employers to hire people to combat these threats to protect their customers and the businesses they operate which again has a positive effect on the economy. I am by no means an advocate of virus writers or malware authors but merely stating that having threats that come from having the internet forces companies to spend money which does have a positive effect on the global economy.

Internet threats have played a major role in the research and analysis of malware / viruses, causing companies like Symantec to sponsor research and development of malware fighting tools to better equip consumers to protect themselves from these threats. These consumers are better informed of the dangers of the internet and as a result are ever conscious of the hackers who want to steal their information in order to commit fraud. The malware scene is overly mature, while on the other hand its “releases” usually tend to have extremely short lifecycles, and quickly become part of a family of variations. The ones with the longest lifecycles tend to dominate a higher proportion of the Internet’s infected population, and these very same pieces of malware are actually the ones written for gains, be it intellectual or financial ones. They also tend to reach levels of sophistication outpacing the rest, make an impact as well as test the vendor’s understanding and fast response to today’s, even tomorrow’s threats. Mind you, each and every malware is released with a specific purpose, namely its life-cycle is anticipated by the authors themselves, but hijacking botnets or vulnerable infected hosts could extend perhaps, not only its life-cycle, but its ownership as well, and that’s already happening. Malware changes tactics whenever an opportunity appears, so basically, even over a short period of time, all propagation vectors get used. The growing proliferation of malware is raising doubts about the Internet’s future. Current security measures primarily target inbound traffic, but service providers have no incentive to stop attacks and spam at the source. A proposed certification scheme motivates providers to control outgoing traffic, efficiently increasing overall security while preserving the Internet’s open, decentralized structure.

Ironically, what has made the Internet so successful – it’s open and decentralized structure – is also what sustains malicious online activity. Information on newly discovered vulnerabilities propagates quickly, and tools to launch ever more sophisticated attacks are readily accessible. The growing availability of inexpensive personal computers and broadband connectivity, coupled with average users’ poor efforts to secure their operating systems, has further facilitated large-scale intrusions, including the remote hijacking of such systems to launch zombie attacks. Security researchers have made tremendous progress in keeping pace with Internet threats, but there are limits to what technology alone can accomplish. While it is possible to proactively prevent some attacks, most solutions are responses to exploits of unforeseen security flaws, and the current framework encourages neither the dynamic assessment of security risks nor the optimal deployment of prevention and response measures.

In conclusion, the internet is here to stay in one form or another and as it is constantly changing so are the ways that malware as well as the ways in which we protect ourselves from it will continue to change. This change will lead to more research, development and employment in one capacity or another thus having an ever constant positive effect on the global economy.

RANDY NARAMORE

Randy Naramore is a family man whose beautiful wife and 4 kids keep him very busy, he enjoys camping with his family and is an avid outdoorsman. Professionally his interests lie in the Information Security and Forensics fields, his primary functions tend to be directed towards the enterprise web security and forensics field. He is a member of the Birmingham Infragard Chapter and is the Vice President of the International Information Systems Forensics Association chapter in Alabama. Additionally he works as a consultant with “Zero Day Consultants LLC” doing information security and forensics work in the Birmingham, Alabama area. Randy currently holds a number of industry recognized certifications such as C|EH, GWAS, CNE, SFCP, and BCCPA.

The Industry's First Commercial Pentesting Drop Box.

THE Pwn Plug.

FEATURES:

- Covert tunneling
- SSH access over 3G/GSM cell networks
- NAC/802.1X bypass
- and more!

Discover the glory of Universal Plug & Pwn

@**pwnieexpress.com**

PWNIE EXPRESS

t) @pwnieexpress e) info@pwnieexpress.com p) 802.227.2PWN

Evading Technique Employed By Malwares

The battle is heating up, – malwares employ various and strong evading techniques which keeps them undetected by anti virus / hips (host based intrusion prevention programs) softwares. The situation is embarrassing, when new malware is being created in most cases it will not be detected by an anti virus software especially if it was not analyzed before by the anti virus researchers.

What you will learn…

- Techniques malwares employto hide and bypass antivirus softwares

What you should know…

- Basics on Malware

In this article I will explain techniques malwares employ in order to hide and bypass antivirus softwares, debuggers and analysis frameworks.

During my work I have analyzed thousands of samples, I have deigned and write the CAMAL (Coseinc Automated Malware Analysis Lab) *http://www.coseinc.com/en/index.php?rt=subscription* as well I have been asked by several security vendors to write few malware variants which can bypass their scanning engines, I have included in this article few of the techniques I have used to detect debuggers and virtual machine setups during my proof of concept.

Infection Vectors

The first stage will always be the infection stage, there are various way malware can be deployed, from cracks, pirated softwares which are all ready infected to vast scale social engineering attacks which are among the most popular infection vectors:

Emails

The human factor is one of the weakest link in the infection journey, sending an innocent email with – relevant or interesting enough for the victim to open is usually enough. The user will download the malware directly as attachment (this method is less common today due to the fact that mail services prohibit to attach executable files with emails) or more commonly will – provide url which will redirect to *drive-by-download* sites which uses exploits to infect the victim machine usually by using browser exploits.

Exploits

Are programs which take advantages of softwares bugs, vulnerabilities in operating systems and applications. The most “valuable” exploits are the so called 0-days exploits. – 0-days exploits does not have any patch since they were not *disclosed* to the operating system or application vendors hence they have very high chance of infecting the target host.

Social Engineering

Social engineering is one of the most popular method to infect targeted pc.

The method is based on deceiving and manipulation people to execute malicious software or to transfer money.

Among the most popular malwares being propagated via social engineering attacks are fake antivirus softwares usually being referred as RougeWare or RougeAV.

FakeAV is malware which looks like security softwares which fools the user into thinking they have being infected by several threats and in order to remove them they needs to buy further protection like upgrading to the “pro version”.

Vendors Security Mitigation

In order to explain how malwares deploy their evading techniques i will need to explain about mitigation software vendors uses in their products in order to protect users from getting infected.

AntiVirus

Are using two main methods to detect malwares, the first one is called *dictionary* method or better *signature* based, the other method is heuristics which monitors behavioral patterns the malware deploys. – The heuristic can combine more then one signature to increase the detection rate.

Some HIPS (host based intrusion detection system) are using propriety sandboxes which try to emulate the malware instructions sets and to determined if they are malicious.

When the antivirus runs it first verify if the binary is packed, if so it will try to unpack the binary load it to memory and start to disassemble its code.

The anti virus vendors receive thousands of binaries each day from joined feeds, deployed honeypots as well receiving suspicious binaries which has being detected by their heuristics engines as *Trojan.Generic* from their endpoint clients. – By receiving suspected binaries they process the new samples on both static and sandboxed environments tracing the binaries api calls as well comparing the binary stubs against database which all ready holds millions of known (signed) binaries segment they *inspected before*.

The signature consist of hashing and *entire* file scanning which is being compared to a *dictionary* of known viruses all ready recognized by the antivirus engine.

Each antivirus vendor uses its own signature algorithm while most of them uses the same concept.

To better understand the process of producing anti virus signature concept I was using ClamAV antivirus engine to produce a very basic signature.

Luckily ClamAV is free antivirus and can be download from *http://www.clamav.net/lang/en/* – ClamAV is being sponsored by SourceFire and it allows researchers to study signatures and produce new once.

In the following example I have downloaded TDSS detected as W32/FakeAlert.3!Generic – by FPROT.

I have downloaded the sample from *http://www.offensivecomputing.net* which is the best site to obtain malwares for research.

Open your favorite hex editor load the malware and look in the right column, there should be ASCII representation section.

Look for the fields where the API calls can be identified and copy 30 blocks before and 30 blocks after (it’s not a thumb rule, usually you will need to search for specific patterns more deeply; Figure 2).

Make sure to copy the left column (HEX field, blue colored) not the ascii one. (gray colored). Save the output to file.

To write your first signature using wildcards (some regex) we can simply write the following:

```
TDSS.A (Clam)
=41F700439D56F3973F07C8A5*0846DDFDCD6CF8A2E25BD8E1
                    84C53C36690EA0FD54E
```

The signature pattern say that `41F700439D 56F3973F07C8A5` pattern must be located in the beginning of the the pattern while the next pattern `0846DDFDCD6CF8A2E25BD8E184C53C36690EA0FD54E` can be found anywhere in the signature.

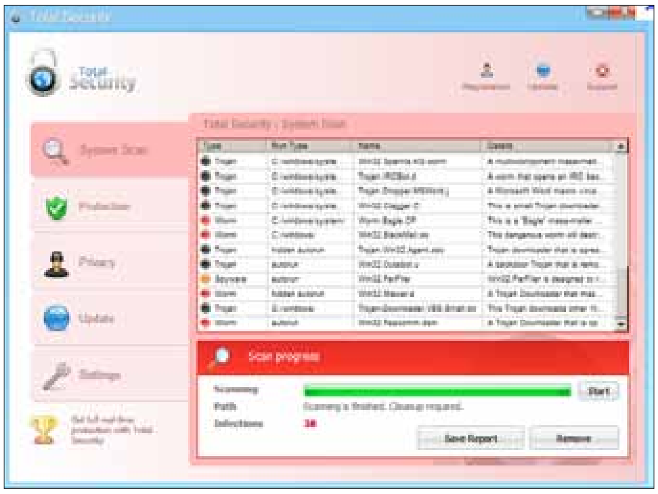


Figure 1. *fakeAV detecting false threats*

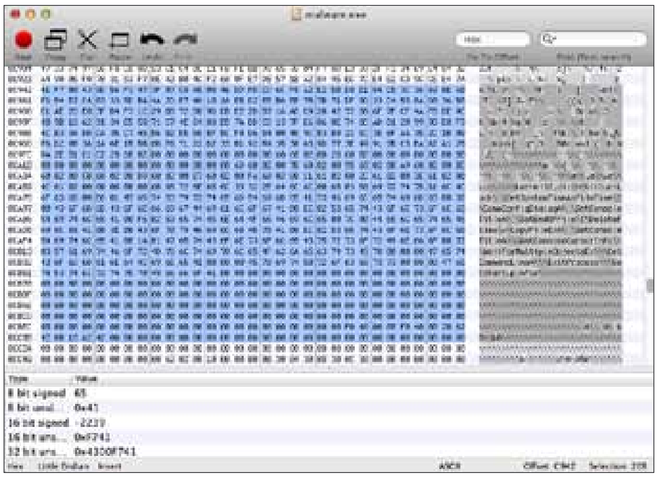


Figure 2. *Fields where the API calls can be identified*

Please note that signatures are usually more complex and combinations between signatures is an option as well.

To find more valuable information regarding writing signatures for ClamAV please refer to: *http://www.clamav.net/doc/latest/signatures.pdf.*

It is essential guide which will clarify the signature process.

NOTE

It is important to understand the nature of Anti Virus signature mechanism and its shortcoming, it is obvious that every vendor uses its own technique to produce signatures for their engines, some Anti Virus vendors offer their engines to other Anti Virus companies so once malware can bypass one engine it will bypass all other vendors which are using the same engine.

Polymorphic Malwares

We should define malwares as an application, it takes time to write efficient application code and since re-code is not an option it simply not efficient and will take to much time. In order to continue evading antivirus signatures engine and continue to propagate malware infections programmers started to use *polymorphic engines* to mutate their code.

The definition for *polymorphic* code is: *continue mutate and keep original algorithm and functionality*.

The idea is to mutate every time the malware its being executed so it will create difficulties for the antivirus engine detecting it.

The most common method which is being used today is writing polymorphic `packers`. the antivirus vendors have problem to sign the polymorphic packers since in order to detect them they will need to *sign* the packers in each mutation phase (some antivirus vendors do sign known packers but they fall short detecting strong polymorphic and unknown one). – It has being demonstrated few times all ready that submitting known malware samples which all ready have signatures like Zeus, SpyEye – etc. to virus total *https://www.virustotal.com/* with unpacked binaries, the detection rate was almost 100%.

When we packed the same samples with our polymorphic packers and submit the same samples to virus total the detection rate was dropped significantly between 3-4% success.

Below is an example for *polymorphic* pseudo code i have presented at Syscan 2010 (*http://www.syscan.org/index.php/archive/view/year/2010/city/sg/pg/speakers*).

The paper can be download from: *http://www.coseinc.com/en/index.php?rt=download&act=publication&file=Camal.pdf.*

Polymorphic Code

```
Start:
GOTO Decryption_Code Encrypted:
     ...lots of encrypted code...
Decryption_Code:
A = Encrypted
Loop: B =*A
      B = B XOR CryptoKey
      *A = B
      A = A + 1
GOTO Loop IF NOT A = Decryption_Code
GOTO Encrypted CryptoKey
      some_random_number
```

Polymorphic and mutation

```
Start:
GOTO Decryption_Code Encrypted: ->> same algorithm
     ... lots of encrypted code...->> same algorithm
Decryption_Code:
     c=2^(c*4)<<--Mutation
     A = Encrypted: ->> same algorithm
Loop: B = *A : ->> same algorithm
      C=C*(A+A)<<--Mutation
      B = B XOR CryptoKey
      *A = B ->> same algorithm
      A = A + 1 <<--Mutation
GOTO Loop IF NOT A = Decryption_Code
      C=C+8 <<--Mutation
GOTO Encrypted CryptoKey
    some_random_number
```

Anti Debugging

Before we go into anti debugging techniques lets first understand what is debugger and how does it works. Debugger is an application which assist in tracing programs, understanding their flow and assists in finding bugs. Debuggers are useful and popular tools during malware analysis process.

For debuggers to work we must have support from the kernel, the reason is that only the kernel have full access to every process which is being created as well to every process registers value (which are essence during the debugging process). The most common practice during debugging is placing breakpoints.

INT3

There are several techniques to place a breakpoint but the most common way is using INT3 instruction known as opcode 0xCC (INT3 is ONLY available on IA32 instruction sets).

Every operating system uses an IDT (interrupt descriptor table) the IDT holds the handlers for every interrupt in the system (except for Unhanded Exception).

When process execute INT (interrupt) specified with the destination operand 3 the CPU halt its execution and generate trap. since the debugger is register as handler to INT3 in the IDT the program execution is being stopped `--break--` and branch out the execution to the debugger.

During the debugging process we can view the registers value, search in the debugged process memory and alter the process flow.

In linux we use the `ptrace()` system call while in windows we use the windows exception mechanism known as the SEH (structure exception handling mechanism) *http://msdn.microsoft.com/en-us/library/windows/desktop/ms679270(v=vs.85).aspx*.

During the debugging process any exception thrown by the debugee (process) will be captured by the debugger.

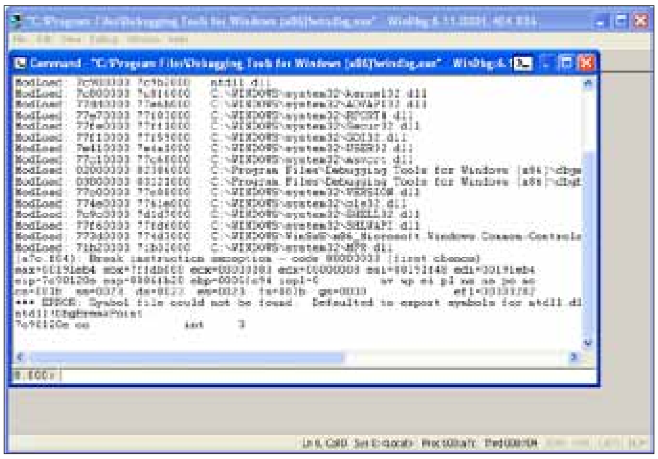
It should be clear now that the malware programmer prefers not to be debugged and certainly not to allow its flow to be broken by the debugger (stop during breakpoint, reveals its registers value).

As stated before malware is an application hence the malware programmer have full access to the OS services (in this article I'm referring to Windows) and can be fully utilize those API and deploy evading technique.

Below an example of – WinDBG placing INT3 (0xCC): see Figure 3.

INT2D

Is usually reserved for Microsoft kernel debugging services.



```
ModLoad: 7c900000 7c9b2000   ntdll.dll
ModLoad: 7c800000 7c8f6000   C:\WINDOWS\system32\kernel32.dll
ModLoad: 77dd0000 77e6b000   C:\WINDOWS\system32\ADVAPI32.dll
ModLoad: 77e70000 77f03000   C:\WINDOWS\system32\RPCRT4.dll
ModLoad: 77fe0000 77ff1000   C:\WINDOWS\system32\Secur32.dll
ModLoad: 77f10000 77f59000   C:\WINDOWS\system32\GDI32.dll
ModLoad: 7e410000 7e4a1000   C:\WINDOWS\system32\USER32.dll
ModLoad: 77c10000 77c68000   C:\WINDOWS\system32\msvcrt.dll
ModLoad: 02000000 [illegible]   C:\Program Files\Debugging Tools for Windows (x86)\dbg
ModLoad: 03000000 [illegible]   C:\Program Files\Debugging Tools for Windows (x86)\dbg
ModLoad: 77c00000 77c08000   C:\WINDOWS\system32\VERSION.dll
ModLoad: 774e0000 7761e000   C:\WINDOWS\system32\ole32.dll
ModLoad: 7c9c0000 7d1d7000   C:\WINDOWS\system32\SHELL32.dll
ModLoad: 77f60000 77fd6000   C:\WINDOWS\system32\SHLWAPI.dll
ModLoad: 773d0000 774d3000   C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls
ModLoad: 71b20000 71b32000   C:\WINDOWS\system32\MPR.dll
(a7c.f04): Break instruction exception - code 80000003 (first chance)
eax=00191eb4 ebx=7ffd[illegible] ecx=00000003 edx=00000008 esi=00191f48 edi=00191eb4
eip=7c90120e esp=0006fb20 ebp=0006fc94 iopl=0         nv up ei pl nz na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000202
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for ntdll.dl
ntdll!DbgBreakPoint:
7c90120e cc              int     3
```

Figure 3. *An example of WinDBG placing INT3 (0xCC)*

When INT2D is being executed it waits for the kernel debugger to handle its exception, in case there isn't kernel debugger attached the exception is passed to the user mode debugging services.

When exception is being executed it points directly to the eip register which in turn *increment by one byte* and breakpoint will be placed. The breakpoint will be served or not served (depend if debugger exist).

If debugger is not attach to the process it will resume normal operation from the address of the exception, if debugger attached to the process it will continue at the *eip* one byte *after* the exception.

NOTE

INT2D has another side effect especially with old versions of *OllyDBG* which will cause it to display fault memory address.

By observing at the address differentiation the malware can determined if its process is being debugged.

Below code is utilizing the INT2D technique.

```
_try{
   _asm{
      int2dh//placing INTERRUPT 2D //
   }
}
_except (EXCEPTION_EXECUTE_HANDLER){
   status = 1;
   MessageBox(NULL,L"clean path",L"clean path",MB_OK);
}

if (status !=1){
   MessageBox(NULL,L"someone attached to my
   process",L"debugger was discovered",MB_OK);
}
```

The ZeroAccess malware utilize the INT2D debugging detection technique.

More information about ZeroAccess can be obtained from: *http://nakedsecurity.sophos.com/2012/06/06/zeroaccess-rootkit-usermode/*.

SetUnhandledExceptionFilter

According to Microsoft MSDN "Enables an application to supersede the top-level exception handler of each thread of a process After calling this function, if an exception occurs in a process that is not being debugged, and the exception makes it to the unhandled exception filter, that filter will call the exception filter function specified by the lpTopLevelExceptionFilter parameter."

In short if the malware is not being *debugged* it will be able to call its specific *handler* (can be anything the malware programmer chooses). in the other hand if the malware is being debugged its own handler will never being called and the malware will break its operation flow or will execute decoy opcodes like loops, far jmps to make the reversing process more challenging.

Below is simple example how to use `SetUnhandledExceptionFilter`

```
#define NOTHING 0
LONG triggerME(LPEXEPTION_POINTERS p)
{
   return EXCEPTION_EXECUTE_HANDLER;
}

int main()
{
    SetUnhandledExceptionFilter ((LPTOP_LEVEL_
                 EXCEPTION_FILTER)&triggerME);
    // here comes the exception
    int x = 0
    int y = 1/x;

 return NOTHING;
}
```

NOTE

Today some debuggers allows the users to pass exceptions to the debugged programs. it doesn't change the fact that lots of malwares are still using the `SetUnhandledExceptionFilter` function.

Memory BreakPoint

Can be achieved by using debugger utilizing the PAGE_GUARD (*http://msdn.microsoft.com/en-us/library/windows/desktop/aa366549(v=vs.85).aspx*).

Feature, the PAGE_GUARD mark the desired memory page (can be buffer allocated using VirtualAlloc, VirtualAllocEx, VirtualProtect, VirtualProtectEx) and when application attempt to access the marked memory page the operating system will raise the `STATUS_GUARD_PAGE_VIOLATION` exception.

Now it should be clear that malware can be utilizing PAGE_GUARD feature to detect when a debugger attach to to process.

Below is simple example how to use VirtualProtect function with PAGE_GUARD:

```
succes  = VirtualProtect (memRegion, 0x10,
PAGE_EXECUTE_READ | PAGE_GUARD, &oldProt);
```

In the above example we use the *VirtualProtect* function to protect our dynamic buffer using PAGE_GUARD, when debugger will try to attach to the malware memory page an exception will be raised and will be catch by the debugger. hence the malware will not be able to receive the exception, this will indicate the existence of debugger and the malware will stop execution.

NTQueryInformationProcess

Function is part of the undocumented functions of windows located inside the *ntdll.dll*.

Sometime malware programmers prefer of using the OS low level API (calling direct to NTDLL) and avoid using the WIn32 API since its closer to the operating system stack and harder to detect especially for instrumentation relaying on Windows 32 API. The "ntdll.dll" export the *native* windows API, it is the user mode interface which operate *without* the Win32 API.

Most of the symbols in the "ndtll.dll" exported with prefix *Nt*.

Malware which is using `NtQeuryInformationProcess` function to detect the presence of debugger consider more advanced.

`NtQeuryInformationProcess` accept 5 parameters, bellow is the function prototype according to MSDN:

```
NTSTATUS WINAPI NtQueryInformationProcess(
  __in       HANDLE ProcessHandle,
  __in      PROCESSINFOCLASS ProcessInformationClass,
  __out      PVOID ProcessInformation,
  __in       ULONG ProcessInformationLength,
  __out_opt PULONG ReturnLength
);
```

We are interested in the first 3 parameters:

The malware needs to verify its not being debugged, the first parameter will be `-1` which will results in getting HANDLE to *it self*.

The second parameter will be the DebugPort value `0x7`, it will return DWORD value in the third parameter `ProcessInfromation`.

If the results is not zero it means that the process is being debugged, the malware then will exit or will generate "noise" like loops, jmps.

NTSetInformationThread

Is another low level native API function. Malware will use this function in order to *detach* debugger which is being attached to it, its more offensive comparing to the other methods.

According to Microsoft `NtSetInformationThread` and `ZwSetInformation` threads are the same function (meaning NetSetInformationThread is a wrapper) *http://msdn.microsoft.com/en-us/library/windows/hardware/ff557675(v=vs.85).aspx.*

Below is the function prototype:

```
NTSTATUS ZwSetInformationThread(
  __in   HANDLE ThreadHandle,
  __in   THREADINFOCLASS ThreadInformationClass,
  __in   PVOID ThreadInformation,
  __in   ULONG ThreadInformationLength
 );
```

We are interested in parameter 1 and 2, the first parameter is the `HANDLE` (the malware it self) the second parameter value will be `0x11` (`ThreadHideFromDebugger`), when we will execute this routine the debugger which was attached to the malware process will detach immediately. Below is code snippet example:

```
//Detach Debugger//
NativeLibrary = LoadLibrary(L"ntdll.dll");
_NtSetInformationThread =
                  GetProcAddress(NativeLibrary,
"NtSetInformationThread");

(_NtSetInformationThread) (GetCurrentThread(), 0x11,
0,0);
MessageBox(NULL,L"Debugger Is Messing With My
                  Process",L"Debugger
Detached",MB_OK);
```

NOTE

`ZwSetInformationThread` is undocumented function, that means that Microsoft doesn't provide any official documentation. For `unofficial` documentation please refer to the following link: *http://undocumented.ntinternals.net/UserMode/Undocumented%20Functions/NT%20Objects/Thread/THREAD_INFORMATION_CLASS.html*.

Sandbox / Virtual Machine Detection

Runtime analysis are very common when analyzing malware behaviour. Most anti virus companies are using runtime analysis techniques in order to trace malware behaviors using Win32 – and low level API instrumentation, as well they are using memory profiling techniques which assist them in generating machine learning and statistical data comparing the run time analysis results and eventually producing new malware signatures.

Runtime analysis is usually being preformed in sandboxed environment both on emulated and virtualized frameworks which analyze thousands of malware samples.

There are numerous malwares which are aware of runtime analysis techniques as well of many sandboxed products and services.

NOTE

In most cases the runtime analysis process takes place inside Virtualized environment.

Virtual Machine Detection

Techniques are being used mainly by Root Kits and advanced packers.

There are various method which assist in detecting the presence of virtual machine. one of the easiest method is to simply search for installation traces during execution.

For example searching and finding one of the following strings prefix on VMWARE framework *VMWARE VIRTUAL IDE HARD DRIVE* or *QEMU HARDDISK* on QEMU hypervisor will results in halting execution.

In the past malwares where searching for the Vmware Tools API which left traces in the windows registry.

Another option is examining the *guest* network interface *MAC* address (first 3 bytes) which will point to the original vendor id. which is the Virtual Machine vendor. (*MAC* address can be changed but people often fail short when it comes to customize their sandbox environment properly).

SIDT Examination (RedPill)

This technique was first introduced by *Joanna Rutkowska*. *IDT* stands for *interrupt descriptor table*, there is exactly one IDT per CPU. the malware needs to detect the IDT address (if there is more then one processor it will need the address of each IDT on every *CPU other wise it might crash or get wrong results*).

In modern X86 architectures there are always multiple CPUs and the application (in this case the malware) have trouble to foresee on which core (CPU) it will be executed.

Luckily Microsoft provides the `SetProcessAffinityMask()` Win32API which provides the option to set the *affinity* mask for the thread of specific process.

The next interesting fact is that query the SIDT to retrieve the IDT address is *not privileged*.

The following code by *Oliver Schneider* demonstrate how to fetch the IDT address:

```
ULONG_PTR GetIdBaseAddress()
{
    #pragma pack(1)
    struct { USHORT Limit; ULONG_PTR BaseAddress;
                 } idtr;
    #pragma pack()
    _asm sidt idtr;
    return iftr.BaseAddress;
}
```

```
int swallow_redpill ()
{
    return (GetIdtBaseAddress() > 0xD0000000);
}
```

NOTE
Joanna Rutkowska was the first to introduce this technique and provided different proof of concept code, the problem in her code is that she didn't consider the multiple *CPU's problem* resulting with her code producing erratic behavior on systems with more then one CPU.

Oliver Schneider code is simple but yet very powerful, – the structure defines the IDTR (Interrupt descriptor – register) and copy (its not privileged hence the malware can perform the query without privileges) by asking the *sidt* to store the address of the *IDTR* into the *idtr* struct.

The last line is returning the `idtr.BaseAddress` (BaseAddress is one of our *idtr* struct elements) to the calling function swallow_redpill.

Bypass Antivirus Using Fud Crypter In A Nuts Shell

As mentioned before, when antivirus scan new binary it first try to determined if it packed. If the binary is packed the antivirus will try to unpack the binary using its built in unpacker and then will use its scanning engine to determined if it infected or not.

The AV engine also examine the imports of the binary and then try to link them against the disassembly code in memory (in case its unpacker manage to unpack the binary).

Understanding the PE data structure will be very good advantage (*http://code.google.com/p/corkami/wiki/PE*) but in short when the loader loads the binary it copy it into memory, the antivirus intercept the binary and try to search for common suspicious words in this memory segments for instance: hacked, owned etc.. (nothing to sophisticated here).

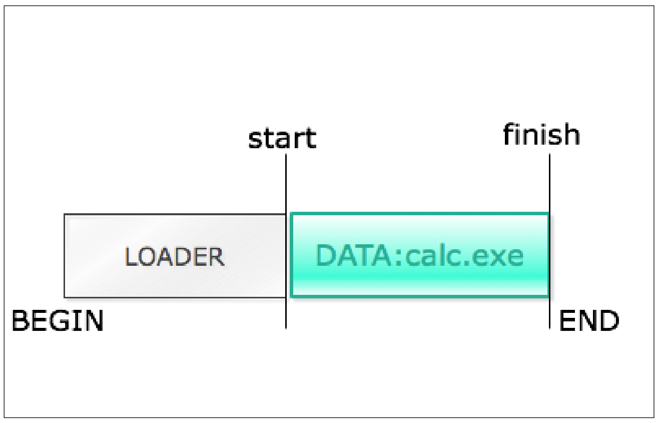


Figure 4. *calc_crypt.exe*

It should be clear that the best method of bypassing the antivirus engine is using a new FUD (fully undetectable) which has not being used before.

I will not cover how to write a crpyter but a good reference can be obtained from: *http://hack2wwworld.blogspot.co.il/2011/04/how-to-make-your-own-100-fud-crypter.html.*

Another resource with source code: *http://genesisdatabase.wordpress.com/2011/02/22/the-official-way-of-writing-a-crypter-in-c-source-code/*.

The malware will need a FUD crpyter and a stub (Decrypter). don't get me wrong but the antivirus vendors sign crypters as well and this is loose loose situation since once the malware uses polymorphic or new loader the antivirus will not detect it.

NOTE
Please bear in mind that most free packers in the internet will be detected by the antivirus scanning engines. there are commercials FUDS which bypass antivirus engines but they are not free (*https://styx-crypt.com/?language=en*).

Let's say that we want to crypt calc.exe, the crypter will need to point to calc.exe and wrap-it (encrypt it).

Usually the crypter will produce another binary for example: *calc_crypt.exe*.

As we see calc.exe will be placed at the end (tail) between two markers (*start, finish*) and this is vital because the stub (Decrypter) size can be changed and as well the payload (calc.exe) (Figure 4).

The payload (calc.exe) will usually be obfuscated using for example some *XOR'ed* because if not it will be detected.

When the loader loads our calc_crypt.exe it will be loaded directly into memory.

Our process will be created in *suspended* mode, the stub will decrypt the payload (*calc.exe*) hopefully in memory and not directly to the hard disk.

And then be executed.

But there is a problem, our *stub* might be signed all ready (like most free crpyters) and will be detected instantly.

To address those issues the malware programmers often does the following:

- Replace the XOR'ed – with strong serpent cipher encryption algorithm (*http://en.wikipedia.org/wiki/Serpent_%28cipher%29*)
- Obfuscate the common used imports (as mentioned before anti virus engines tend to scan the import and link them to the disassembled code in memory), they will need to write certain encoder to hide the strings.

References

- anti reversing guide: *http://www.codeproject.com/Articles/30815/An-Anti-Reverse-Engineering-Guide#StolenBytes*
- MSDN: *http://msdn.microsoft.com/en-US/*
- Eli Bendersky *http://eli.thegreenplace.net/2011/01/23/how-debuggers-work-part-1/*
- Coseinc blupill: *http://www.coseinc.com/en/index.php?rt=download&act=publication&file=Introducing%20Blue%20Pill.ppt.pdf*

- The tail section (calc.exe position) will be changed dynamically.
- Test their crypter against antivirus engines.

Conclusion

Malware is an application which can fully utilizing the operating system resources.

It is no surprise that the infection rate is growing, new threats are being detected long time after they strike and the cyber threats are being taken much more seriously these days.

In the other hand there is a lot to be done, it is true that the antivirus engines detect more then 80% of *known* malwares but stand useless against new malware which employes new evasion techniques. the antivirus vendors must switch tune and redesign their scanning engines which needs to be replaced. their detection concept is old, behavioral approaches combine with smart signatures model needs to be introduced.

In this article I tried to combine few evading techniques malwares are using in order to trick the security researchers and evade the antivirus engines. I'm aware that i didn't cover all techniques – but really hope it will help understating the huge effort reverse engineering face during analysis and in the other hand the difficulties malware programers face when writing new malware variants.

I can be reached in: *udiATcoseinc.com* and will be glad to answer any questions you have.

UDI SHAMIR
Senior Security Researcher
Coseinc Advanced Malware's Labs

Understanding/ Preventing

Drive-by Browser Malware Attacks

Drive-by browser malware comes in many forms by exploiting a vulnerability in your browser or a browser plug-in and opening a file in a hidden frame, then exploiting a security hole in anyone of several applications including any of several components of Adobe or Java to mention a few.

What you will learn…
- Preventing browser malware attacks

What you should know…
- Basics on securing your data

While drive-by malware in general is now attacking virtually if not all platforms the most commonly exploited platform is an Internet browser on Windows. Typically the browser is Internet Explorer, all versions, or Mozilla Firefox.

The challenge of drive-by browser malware attacks is that usually when they occur the user, whether digitally savvy or not, has no idea how it occurred. In fact, it is not uncommon for a totally "not guilty" site being accused. Now, just how is this? The problem stems from the fact that many of us, including myself, have a number of sites which open each time when the browser is opened as they are defined in their 'home page'.

By way of example I have eight differing URLs defined in my home page which means that anytime I open my web browser these eight open and remain open all of the time. Dependent upon your browser settings you may be opening additional sites as when you last closed the browser these URLs were active. Now, if any of these sites display changing advertisements it is likely that at some time you will be hit with drive-by malware. It is simply a matter of time.

Many sites display advertisements but do not constantly change them. These sites are a bit safer yet do not insulate you from drive-bys. It is just that you are less likely to encounter one. On the other hand, those that constantly present varying advertisements without any interaction by the user present the greatest threat. Envision the user having five URLs defined in their homepage, one of which is the corporate and the four others are a media site delivering local news, state news, national news and international news. Virtually all media sites worldwide present changing advertisements and in this example you see four sites providing a continuous variety of advertisements any one of which likely has been compromised. It is only a matter of time before drive-by malware will strike.

A tip to malware examiners who try to determine which site resulted in redirection to the malware site. When it comes to drive-by malware it is often impossible, the reason being that the site responsible was opened in the browser several days prior and the browser nor the system was not shut down. If the source site cannot be identified in close proximity to the date/time of the malware being installed, more than likely it simply cannot be determined.

While it is almost irresistible, it is essential that users do not leave windows open at sites which present changing advertisements. It is possible to block all ads, but that solution is beyond the grasp of typical users. Rather, all need to understand the threat and avoid the practice of remaining on sites which constantly present changing ads. There are tools available which do a credible job of warning users of a questionable site presenting changing ads, but none do a great job.

The best defense is the savvy user… that is those who use a multiple strategy approach. While what I am about to describe might appear to be complicated and challenging, the reality is that each of the components is actually independent of the others and as a result they can be accomplished one at a time. This provides the ability to implement one control at a time and become satisfied that the first control is fine before moving on to the second control, etcetera. Just what are these controls? Let us discuss:

Subscribe to a custom IP address server

There are numerous custom IP address server providers all of which deliver controls not otherwise available. When you use the DNS server of your ISP you are typically left with the DNS servers provided by Internet management without any ability to limit what you may select/be redirected to. On the other hand a custom IP address server commonly referred to as a DNS server will provide you with options often at the basic level for no charge. While there are many I am confident that most will find what they need at opendns.com or dyn.com. Utilizing either and implementing some of their controls in and of itself will reduce vulnerability.

Install a gateway firewall

A gateway firewall is best implemented with a hardware router featuring a number of controls connected between your computer and your ISP modem. Typical routers commonly used today have no intelligence in that all they do is manage network activity between multiple PCs and the Internet. The gateway firewall, or if you prefer intelligent router, includes DHCP services. Many, perhaps most, are using a WiFi router when they connect multiple PCs but most do not include an Internet gateway firewall. Rather its firewall activities deal with unwanted users exploiting your Internet services. Thus, even if you have a WiFi router more than likely you need a hardware based gateway firewall. Again, there are many with some that offer reasonably priced units suitable for home and small businesses. A Google search for "hardware router firewall" will provide a good number of valuable links. Perhaps it would be best if all start with *http://www.firewallguide.com/hardware.htm* which

Note

For Java to be considered current, multiple versions of Java cannot be on the system. Older versions need to be removed, sometimes manually.

Most up-to-date secure versions* as of: July 22, 2012

*This does not mean it is the latest version available; there may be later versions with non-security related fixes

Product	Version	Date	Remarks Hot link
Adobe Acrobat Pro and Standard[1]	10.1.3 9.5.1	04/10/12 04/10/12	Includes MUI *http://www.adobe.com/support/downloads/detail.jsp?ftpID=5324*
Adobe Air	3.3.0.3670	07/11/12	*http://get.adobe.com/air/*
Adobe Flash Player ActiveX	11.3.300.265	07/11/12	*http://www.adobe.com/support/flashplayer/downloads.html*
Adobe Flash Player Plugin	11.3.300.265	07/11/12	*http://www.adobe.com/support/flashplayer/downloads.html*
Adobe Media Player[2]	Obsolete	09/16/10	Adobe Flash Player is its replacement – See *http://www.adobe.com/products/mediaplayer/*
Adobe Photoshop	CS5/CS5.1	08/09/11	Standard and Multiplugin *http://www.adobe.com/support/security/bulletins/apsb11-22.html*
Adobe Reader[1]	10.1.3 9.5.1	04/01/12 04/01/12	Includes MUI *http://www.adobe.com/support/downloads/product.jsp?product=10&platform=Windows*
Adobe Shockwave Player	11.6.5.635	05/08/12	*http://get.adobe.com/shockwave/; http://www.adobe.com/support/security/bulletins/apsb12-02.html*
Apple QuickTime	7.72.80.56	05/15/12	*http://www.apple.com/quicktime/download/*
Java(TM)	1.6.0_33 7.0_05	06/19/12 06/12/12	*http://www.java.com/en/download/manual.jsp*

[1] Adobe Acrobat/Reader, versions 9.5.1 and 10.1.3 are both considered secure. *Earlier versions are not.*
[2] Adobe Media Player is not secure and should be deleted if present. It is no longer supported by Adobe but other sites still offer it!

will provide plenty of information regarding all types of home firewalls.

Employ a reliable desktop antivirus product

There are numerous reliable desktop antivirus products. Many maintain that the Windows Security Essentials is all that you need and to a degree I concur. It is an effective antimalware defensive mechanism. On the other hand, it does not deliver by itself a firewall which is known as Windows Firewall. For many, if not most this combination is adequate especially when the preceding is also in place. On the other hand, Microsoft is not always first to the table and I have found that a number of the commercial services know how to live side by side with the Microsoft tools. I am fortunate in that my ISP provides me with a free commercial service and it gets along with the Microsoft tools. There are too many commercial tools to make an absolute recommendation. The following will provide a comparison that may prove useful. While a bit dated it still is worthy of consideration: *http://www.techrepublic.com/blog/10things/10-free-anti-malware-tools-worth-checking-out/2045*. For what it is worth, my preference is for Avast.

Managing your browser

No matter which browser you use it requires management. The challenge is that there is not any tool available to help you. You must help yourself. Should you choose just what is delivered, more than likely things are happening which you are likely not to agree with. However, there is *http://www.it.northwestern.edu/security/browser-management/* which provides guidance for IE 9, 8 & 7, Firefox, Google Chrome and Safari. A very straight forward guide, it is valuable for all.

This discussion has focused upon the value of management of your systems to ensure that your exposure to malware is kept to a minimum. What it does not address are the details of the components of your systems that are commonly exploited by malware. Failure to keep these components current is likely to expose you to malware exploit even when all other controls are in place. Why? Today there are exploits which are totally web based and when they reach your system by whatever means your last level of defense is that the tools in place are current. Following is a table of those components most commonly exploited and their current version as of 2012-07-22. It is not uncommon for one or more of these to change each month. As a result reference is also included as to where to verify the most current version.

Determining version level of any product can be challenging. When you go to *Control Panel/Programs and Features* (Win7/Vista) or *Add or Remove Programs* (XP) many programs show their version level in their title but often even that version level is inaccurate. To determine the precise version perform the following after you have selected the specific program:

- In Win7/Vista look to the far right you should see Version just beyond the Size field. Note that you may have to adjust field sizes smaller so that you can see the full version level without scrolling to the right.
- In XP simply click on the program name and click on *Click here for support information* and a window will pop-up with useful information including the complete version number

The one application which all should subscribe to is Java(TM) Version 7 Update 5. Very soon further support for Java(TM) Version 6 will be abandoned. Java(TM) Version 8 is expected in the summer of 2013. It is only a matter of time before Java(TM) 6 Version 33, likely the last version, will be compromised if it has not already occurred.

Keeping track of Adobe changes can be difficult. Your greatest concern should be security and Adobe publishes security bulletins. The following URL provides information regarding most Adobe updates, all security bulletins and the opportunity to sign- up for an Email for all future security bulletins – *http://www.adobe.com/support/security/*.

Remaining current to all of the components identified in the attachment is challenging. It is not that uncommon for two or more to change each month. On a monthly basis the latest version of the table will be available at…..

R. E. (BOB) JOHNSTON

U.S. Army Retired Chief Warrant Officer with more than 45 years in information technology and 40 years in information security. Became a Certified Information Systems Security Professional (CISSP) in 1995 and has taught computer security in Asia, Canada, Mexico and the United States as well as digital forensics and other computer security related courses at a local college. Wrote a computer security column for 5 years in the 1980s titled "for the Sake Of Security", penname R. E. (Bob) Johnston, which was published in Computer Decisions. Currently employed as a Senior Security Analyst with a primary focus upon forensic examination of malware infestations. Motto: "When entrusted to process, you are obligated to safeguard".

Is your **MISSION-CRITICAL** security strong enough to stop a **SKILLED ATTACKER?**

Don't guess
Don't believe
Don't hope

KNOW!

acros

An ACROS Penetration Test is conducted exactly like a real attack by a skilled, motivated adversary – only without the damage. We will find the weakest links in your security and use all our knowledge, skills and capabilities to try to achieve exactly what your security measures and policies are there to prevent. If it sounds difficult, we're interested.

Experience the ultimate test of your security.
(After all, the only alternative is to wait for an actual attack.)

ACROS Security – http://www.acrossecurity.com – security@acrossecurity.com

Social Network Privacy Guide – II

This series of articles about security trips how to make social networking is more secure on the top social networks.

What you will learn…

- The most useful ideas and advice how to use a lot of social networks mixing fun and business
- What does the most known social network offer to keep your data in privacy

What you should know…

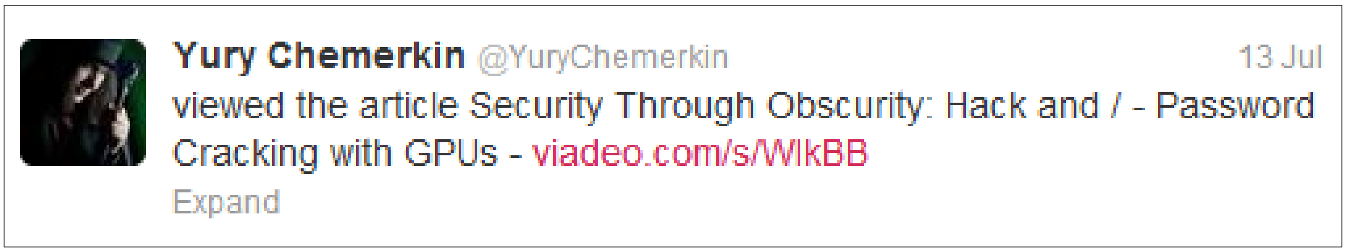
- Basic knowledge how to find and setup security setting on social networks
- Clear understanding of your goal when you start to use a new social network

Social networking services are kind of online service that focuses on building social relations among people shared their information about themselves. This information filled their profiles makes users possible to search and extract necessary information. It means the search will analyse only the actual contents you want (images, video, text, calendar events). Such representation is often based on each user profile as set of social links, interests, public data, and other linked services. Current trend has fast been growing to control mechanism unification for a long time. Each of these social services meets with users desires to less inputting about them. That's why you are allowed to be sign up/in by Facebook button or Twitter button following which you can start to organization your own networks groups by involving others friends via email, social address book or switching your profile into public zone indexed by search engines like Google, Yahoo or Bing. This is so-called individual-centred service whereas online community services are group-centred based on user abilities to share ideas, activities, events, and interests within their individual networks.

Chapter I. Security beyond the whole Picture

Part II. Twitter

Twitter has come in 2006 as a people group telling what they were doing right now as soon as happens like "woke up" or "overslept". Now it is social place to leave messages limited up to 140 characters long as kind of SMS messages. These messages create your so-called timeline that can be followed and these messages can be marked as favorite, retwetted, and replied. Direct messages limited up to 140 characters too.



Yury Chemerkin @YuryChemerkin 13 Jul
viewed the article Security Through Obscurity: Hack and / - Password Cracking with GPUs - viadeo.com/s/WlkBB
Expand

Figure 1. *Normal Tweets*

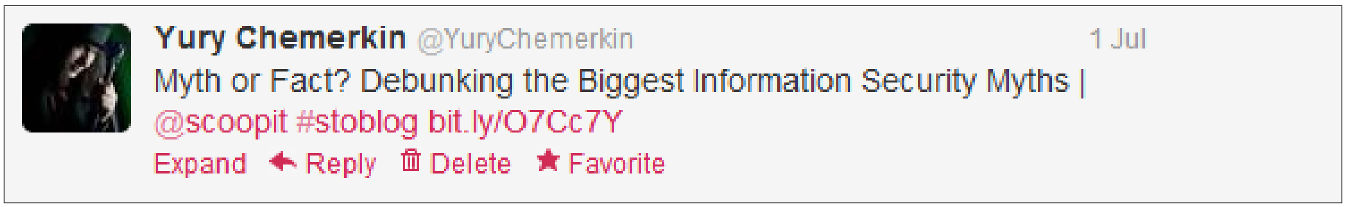

Yury Chemerkin @YuryChemerkin 1 Jul
Myth or Fact? Debunking the Biggest Information Security Myths | @scoopit #stoblog bit.ly/O7Cc7Y

Figure 2. *Mentions*

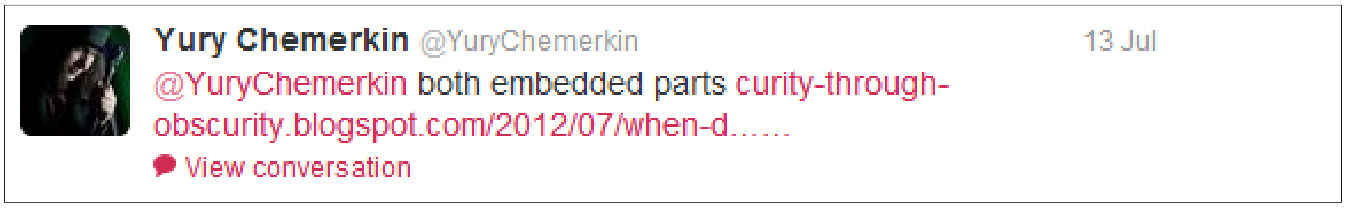

Yury Chemerkin @YuryChemerkin 13 Jul
@YuryChemerkin both embedded parts curity-through-obscurity.blogspot.com/2012/07/when-d......

Figure 3. *Replies*

Some basic terms and definitions

Timeline

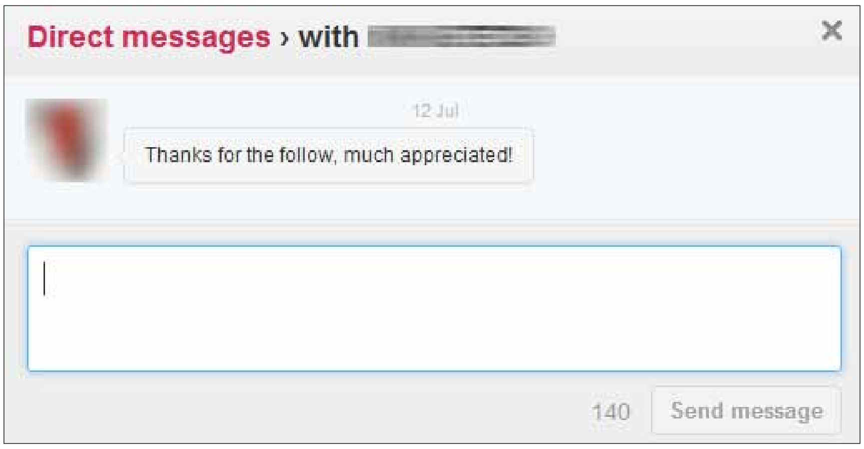
A Twitter timeline collects stream of Tweets listed in real-time order with newest updates are at the top into you will land by view of your homepage.

Types of Tweets

- *Normal Tweets* look like as shown on Figure 1 and are represent a message not more than 140 characters by itself that appear on sender's page and his timeline and on other profile's timeline who are allowed to be seen updates in order to privacy settings. Not that, it has never been appeared on someone profile until it will be retweeted.
- Mentions *look like as shown on Figure 2 and* are represent the same message including another's username preceded by "@" placed I message after one word at least, e.g. "This @yurychemerkin is mention for..." Mentions usually appear on sender's profile among public tweets or someone timeline if this person is following a sender. In addition, mentions may be found in the recipient's Mentions and Interactions tabs, which is accessible only by them. As a normal tweets, it has never been appeared on someone profile until this person wrote the message.
- *Replies*: look like as shown on Figure 3 and are represent a message similar to the mentions except position of "@username". Now it must be placed at the beginning of message, e.g. "@yurychemerkin your blog is cool!" Replies appear on profile page or on timeline, whose owners are following the sender or both of them. Also, replies may be found in the recipient's Mentions and Interactions tabs and never on anyone's profile page, unless they wrote or sent the message. If your Tweets are private, it means no one is allowed to see any of your tweets unless you have given them the right to follow you. Thus, when you send a @reply or mention, only profiles approved by you will be able to see them else you have to unprotect your tweets to make them and your account public.
- *Direct messages* is usually non-public message sent directly to someone who follows you or sends directly to you from someone you follow. Other cases are not allowed. It stores in direct messages folders of sender and recipient as well as on recipient mobile device or email if (s-)he turn on this feature(-s) (Figure 4).

The common privacy rules

Twitter considers the posting another person's private and confidential information as a violation of the Twitter Rules. Such information may be:



12 Jul
Thanks for the follow, much appreciated!

Figure 4. *Direct messages*

- credit card information
- social security or other national identity numbers
- addresses or locations that are considered and treated as private
- non-public, personal phone numbers
- non-public, personal email addresses

Anyway, you may make report about it through the link (*https://support.twitter.com/forms/abusiveuser*).

Twitter solution in protection his users from spam and abuse is by permanent suspension if anyone engaging in the activities specified below:

- Mass account creation is forbidden for Twitter's users and username squatting and account's inactivity for more than 6 are forbidden and will be removed
- Invitation spam technically disabled users to send an invitation repeats.
- Some of declared spam techniques:
  - User has followed and unfollowed a large amount of users in a short amount of time or repeatedly has done it
  - User has less followers than he's following or a large number blocked him
  - User's updates consist mainly of links, especially misleading and malware links
  - User posts too many duplicate data (note, you have to choose a couple of social networks linked with Twitter if these networks are mirrors)
  - User posts unrelated updates to the tag started with #
  - User sends too many duplicated referrals started with @, especially when it looks like a spam
  - User adds too many unrelated users to lists with spam goal
  - User describe false or misleading interests area
  - "Aggressive following" means a misbalance between user's followers and followed list. For example, user cannot follow 10,000 people if only 100 people follow him. When users reach 2000 followed they have to wait until they get more followers in order a misbalance

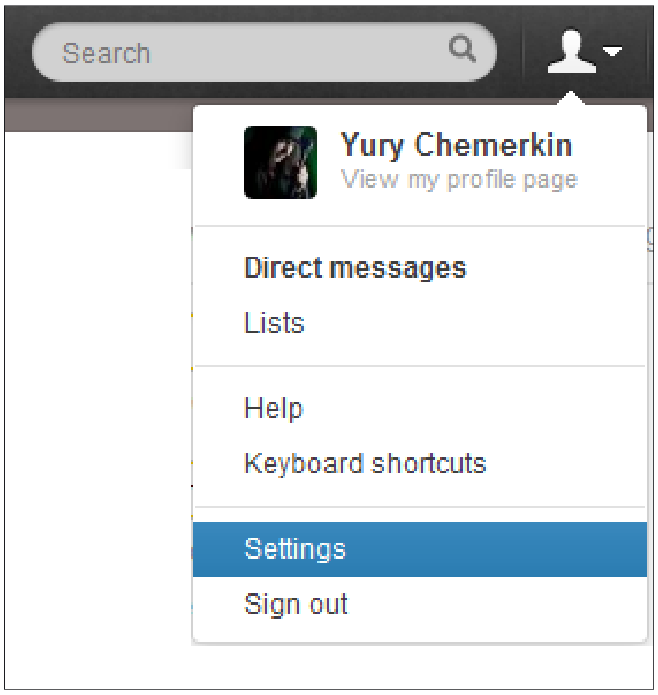


Figure 5. *Dropdown menu*

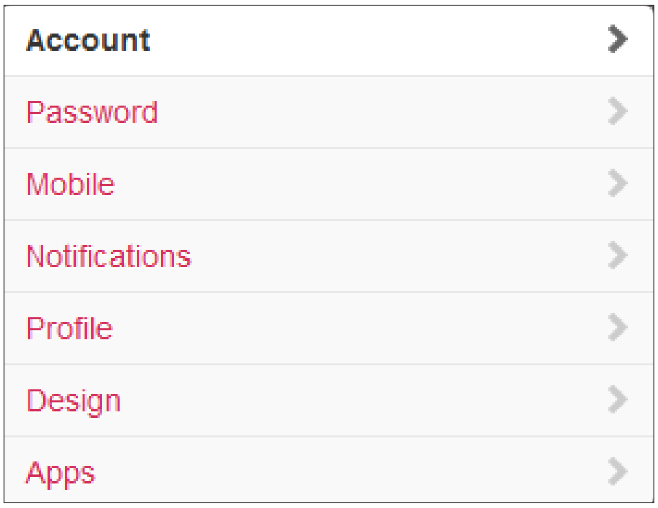
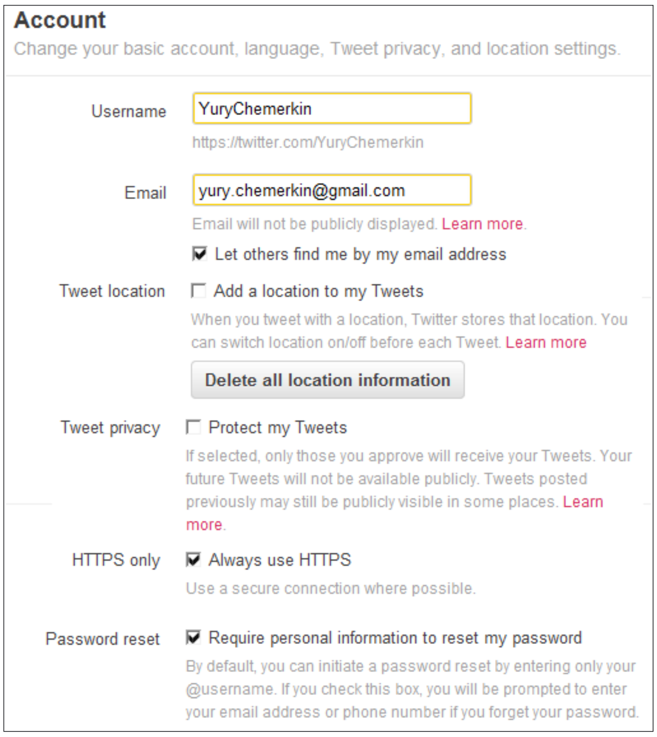


Figure 6. *Account tabs*



Account
Change your basic account, language, Tweet privacy, and location settings.

Username: YuryChemerkin
https://twitter.com/YuryChemerkin

Email: yury.chemerkin@gmail.com
Email will not be publicly displayed. Learn more.
[✓] Let others find me by my email address

Tweet location: [ ] Add a location to my Tweets
When you tweet with a location, Twitter stores that location. You can switch location on/off before each Tweet. Learn more
Delete all location information

Tweet privacy: [ ] Protect my Tweets
If selected, only those you approve will receive your Tweets. Your future Tweets will not be available publicly. Tweets posted previously may still be publicly visible in some places. Learn more.

HTTPS only: [✓] Always use HTTPS
Use a secure connection where possible.

Password reset: [✓] Require personal information to reset my password
By default, you can initiate a password reset by entering only your @username. If you check this box, you will be prompted to enter your email address or phone number if you forget your password.

Figure 7. *Tab "Account" Settings*

  - Some more relate to the technical limits:
  - Updates/Tweets: 1,000 per day that splits into semi-hourly intervals and retweets are counted as updates.
  - Changes to Account Email: four per hour.
  - Following (daily): The technical follow limit is 1,000 per day.
  - Following (account-based): Once an account is following 2,000 other users, additional follow attempts are limited by account-specific ratios.
- Pornography must not everywhere (profile, background, etc.)
- Each new created account to replace previously suspended by Twitter will suspend too.

Fake Twitter Emails

As Twitter does not send emails with attachments and never requests user's Twitter password by email, other emails and messages similar to wrote above tend to be a fake. If you received such email, you should to resend (forward) it to the [spoof@twitter.com] and then delete this email from inbox without downloading any attachments from email. Phishing is another kind to steal sensitive information when intruders send bulk messages in an attempt to revealing such information, like a login and password. It happens often through the website appears legitimate. In case of spoofed emails, such email may contain string header "Twitter/Twitter Support/" while email address differs from real "@twitter.com".

Fake Web Sites

User should always check whether or not link that goes to a fake login page and he is at twitter.com before logging in. You should check links in direct messages when clicking on them, especially be careful when clicking on links that were shortened using an external link shortening service. Even if the link came from a friend, it is possible that their account was compromised and the URL was actually sent out by a spammer. To be sure, you are on Twitter.com before logging in take a look at the URL in the address bar of your browser. Twitter domains will always have the *http://twitter.com/* or *https://twitter.com/* as the base domain. Phishing websites will often look just like Twitter's login page, but will actually be a website that is not Twitter. For example, *http://twitter.example.com* or *https://m.twitter.com.up.com*.

Twitter Setting

Tabs regarded to the Privacy in "Account Settings". To go to the Account Settings click on profile picture on and choose "Settings" as it is shown on Figure 5.

In account section (Figure 6) there are several settings e.g. Email address, GPS location, country and etc. Most of them do not belong to privacy except some you will see on obfuscated Figure 7 below:

- Username
- Email and searching via email
- Tweet location
- Tweet Privacy
- Connection type
- Password reset way

Email address is part of user privacy in way someone may import address book from Gmail, Y!Mail, Windows Live or else to the Twitter and find anyone who uses the same email-address

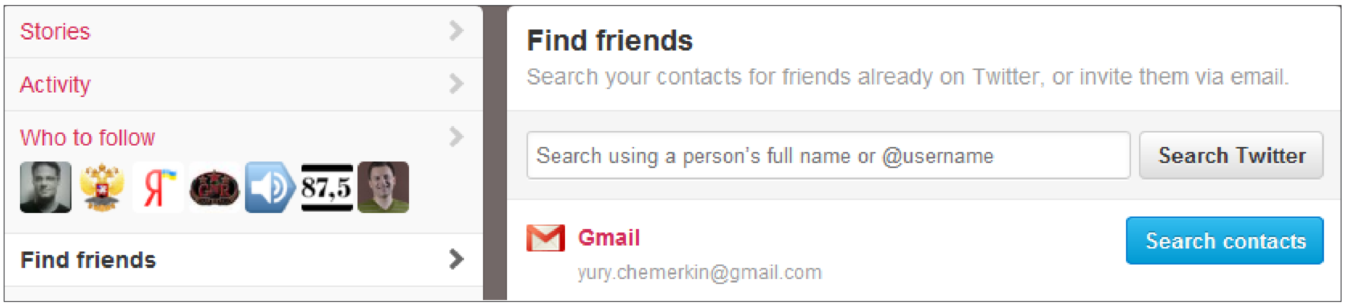
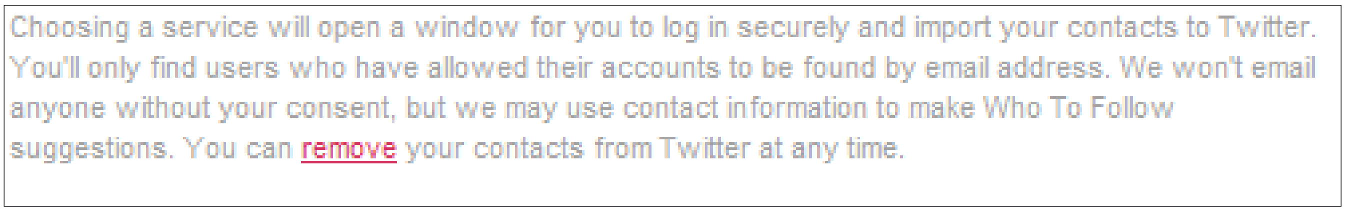

Stories
Activity
Who to follow
87,5
Find friends
Find friends
Search your contacts for friends already on Twitter, or invite them via email.
Search using a person's full name or @username
Search Twitter
Gmail
yury.chemerkin@gmail.com
Search contacts

Figure 8. *Find friends*



Choosing a service will open a window for you to log in securely and import your contacts to Twitter. You'll only find users who have allowed their accounts to be found by email address. We won't email anyone without your consent, but we may use contact information to make Who To Follow suggestions. You can remove your contacts from Twitter at any time.

Figure 9. *Delete "Find friends" data*

stored in his address book. Therefore, check field “Let other find me by my email address” should be unchecked. To find someone on Twitter via email address you need click on “#Discover” and “Find Friends” then where five email services are available (Google, Yahoo, Yandex, Hotmail, and AOL). By the way, you need to enter your email log in credentials when prompted and agree to share your information with Twitter. However, you can remove imported contact info from Twitter at any time: there is a block of text under the email provider list where there is a link to remove your contacts (Figure 8 and Figure 9).

Username may not be a point of privacy itself but when it plays opposite searching via email, you may be interested to hide your username based on your real name. *HTTPS* has become a joke is a classic in most popular social networks (Facebook, Twitter, LinkedIn, G+). This improves your account security and protects you if you are using Twitter over an unsecured Internet connection, like a public Wi-Fi network, where someone may be able to eavesdrop on your site activity. If your settings suddenly don’t use this you need to assign a checked state by several reasons and one of them is that not all fake web-site is already user https like a [*hxxp://www.krishnasings.com/*] was found in June if it looks like Twitter even. If you want to change password you only need to type your current password on the web and retype new password twice. Way to reset

Table 1. *Main Features*

ON	turns ALL your authorized Twitter updates and notifications on.
OFF	turns ALL phone notifications off.
ON [username]	turns on notifications for a specific person on your phone (without “@”). Example: ON Yury-Chemerkin
OFF [username]	turns off notifications for a specific person on your phone (without “@”). Example: OFF Yury-Chemerkin
FOLLOW [username]	allows you to start following a specific user, as well as receive SMS notifications. Example: FOLLOW YuryChemerkin, or f YuryChemerkin
UNFOLLOW [username]	allows you to stop following a specific user. Example: UNFOLLOW YuryChemerkin
LEAVE [username]	stops receiving SMS notifications for a specific user without having to unfollow them. Example: LEAVE YuryChemerkin, or l YuryChemerkin
STOP, QUIT, END, CANCEL or UNSUBSCRIBE	will deactivate your account if you are an SMS-only user.

Table 2. *Additional Features*

@[username] + message	shows your Tweet as a reply directed at another person. Example: @YuryChemerkin newpost
D [username] + message	sends that person a Direct Message that goes to their device, and saves in their web archive.
RT [username]	sends that user's latest Tweet to your followers (also known as a Retweet).
SET LOCATION [place name]	updates the location field in your profile. Example: set location Moscow
SET BIO	edits your Bio information on your Twitter profile. Example: set bio I'm a writer in Hakin9!
SET LANGUAGE [language name]	selects the language you'd like to receive notifications in. Example: set language Russian
SET NAME [name here]	sets the name field on your Twitter profile. Example: set name Yury Chemerkin
SET URL [url here]	sets the URL field on your profile. Example: set url http://re.vu/yury.chemerkin
WHOIS [username]	retrieves the profile information for any public user on Twitter. Example: whois yurychemerkin or w yurychemerkin
GET [username]	retrieves the latest Twitter update posted by that person. Example: get yury-chemerkin or g yurychemerkin
FAV [username]	marks that user's last Tweet as one of your favorites Examples: fav yurychemerkin, or favorite yurychemerkin
STATS [username]	this command returns the given user's number of followers, how many people they're following, and their bio information
SUGGEST	this command returns a listing of Twitter users' accounts we think you might enjoy following

your password means Twitter developers is on right track. Now you may set password recovering via phone that maybe more secure than via email. However, in order to I wrote in article "The backroom message that stolen your deal" (March 2011) [*http://goo.gl/YUzjk*], your SMS messages may include C&C based on SMS for Facebook or Twitter. This *SMS-management* is vulnerable and leads to auto-post anything without asking or notification to the social network additionally charges you and extracts all one-time passwords you need type to approve on public computers. Below I discuss commands and features of SMS exploitation.

Twitter SMS Commands It can help you to perform certain actions, like following a user or marking a friend's update as a favorite (see in *Main Features* and *Additional Features*) (Table 1 and Table 2).

By the way, Facebook has SMS command as well but it is shortly than Twitter. You can text to the Facebook to update your status, message your friends, receive messages, and wall posts from your friends as it happens. All SMS commands send to number "3223" (Table 3).

Adding *location* to the tweets neither for not vulnerable for users without context. For example, user posts about his travel with location tracking during several months and then he makes a post when he is in airport. As you see his last tweet but not includes a location data, because it is enough to rob as a fact nobody is at home. The main idea is its non-systemic actions, tweet and location then intruders need more time and resources to analyze dependence; if not, you will be easy caught! If you want delete location info, you need to know it usually takes around 30 minutes and Twitter grants to delete all location data, but no one grants to delete location data accumulates in Third-Party apps on something RSS news as a Twitter is still allowed to gain data for public profiles via RSS.

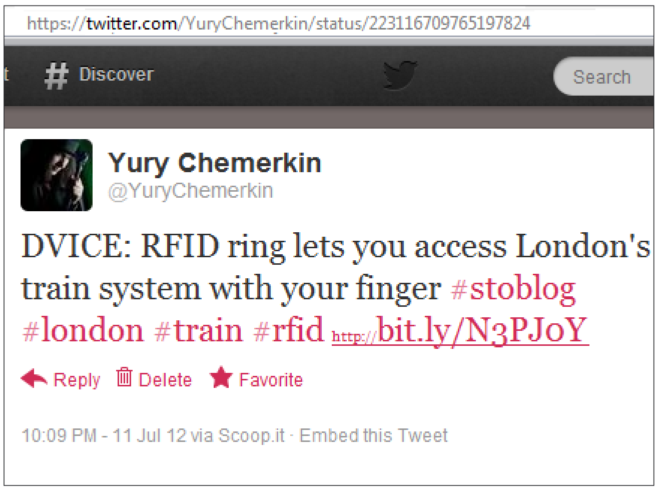
Twitter Privacy turn on/off your account's publicity and means you need approve everyone to see you tweet or never manage it. Public Tweets as a default setting are visible to anyone whether someone has a Twitter account. If you had public twitter profile at once, then these tweets will always be public and searchable despite changing settings. Future tweets that will make after you turn on privacy will be protected. This option may often be interested to control who sees your messages. Moreover, links you made via "t.co" are public, because any links are able to view the content through the world except filtering case. In addition, when privacy is turn on keep in mind that:

- Each follower request have to be approval as I wrote above
- Your Tweets will only be visible to users you've approved
- Only approved users are able to retweet your tweets
- Protected tweets will not indexed by Twitter search, or Google search or any search engine and didn't appear in anywhere
- All @replies users send to the user A will not be seen by him if he wasn't approved before
- You cannot share permanent links to tweets with anyone other than your approved followers. Permanent link is a static URL (except if it was deleted) to the tweet you'll find in browser 's address bar when you click on the tweet and then on his details is often include
    - The exact time the tweet was posted

Did you know?
Your service provider may split SMS messages greater than 160 characters into multiple messages. In this case, the second message will post as a normal Tweet because it does not begin with d+username, as the first message did.

Table 3. *Facebook Commands*

Update status	is at john's party
Add friend by name (or phone number)	add john smith add 1234567890
Subscribe to status	subscribe john smith
Photos help	photos
Unsubscribe	unsubscribe
Help	help
Stop sms	stop
Start sms	on


https://twitter.com/YuryChemerkin/status/223116709765197824

Yury Chemerkin
@YuryChemerkin

DVICE: RFID ring lets you access London's train system with your finger #stoblog #london #train #rfid http://bit.ly/N3PJoY

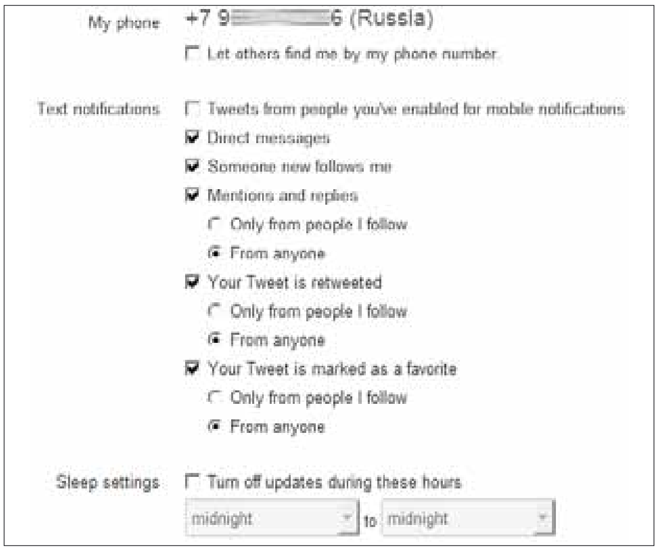
10:09 PM - 11 Jul 12 via Scoop.it · Embed this Tweet

Figure 10. *Permanent links*

- The service/application used to post the Tweet
- Users who retweeted the tweet

Tab “Mobile” in account settings leads you to the phone number management and is represented by text notifications and sleep settings except feature let you find others by your phone number. It is the same graph extraction as email searching, so you need to decide whether this feature should be checked. Check all of text notifications field is a good idea to keep yourself abreast of the news in case your account is hacked. Similar idea is going about email notifications settings is allowed you to control:

- Notification if direct message, reply or mention received
- Notification if retweets, following or mark as favorites happened
- Notification regarding weekly stories


My phone: +7 9[redacted]6 (Russia)

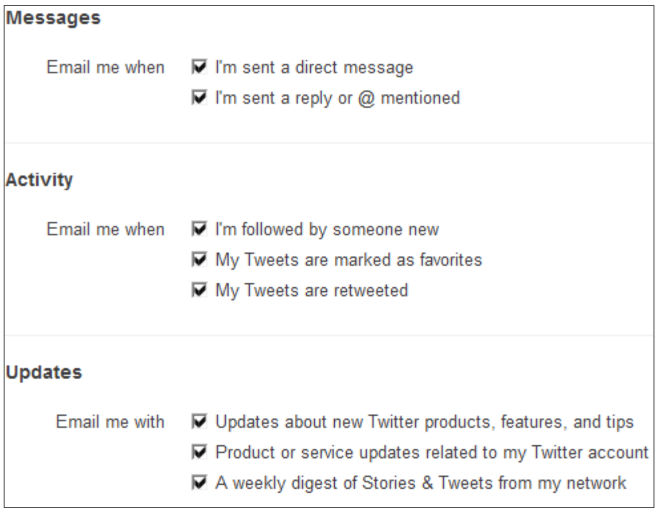
Let others find me by my phone number

Text notifications:
- Tweets from people you've enabled for mobile notifications
- [✓ Direct messages]
- [✓ Someone new follows me]
- [✓ Mentions and replies]
  - Only from people I follow
  - [✓ From anyone]
- [✓ Your Tweet is retweeted]
  - Only from people I follow
  - [✓ From anyone]
- [✓ Your Tweet is marked as a favorite]
  - Only from people I follow
  - [✓ From anyone]

Sleep settings: Turn off updates during these hours

midnight to midnight

Figure 11. *Mobile settings*


Messages

Email me when
- [✓ I'm sent a direct message]
- [✓ I'm sent a reply or @ mentioned]

Activity

Email me when
- [✓ I'm followed by someone new]
- [✓ My Tweets are marked as favorites]
- [✓ My Tweets are retweeted]

Updates

Email me with
- [✓ Updates about new Twitter products, features, and tips]
- [✓ Product or service updates related to my Twitter account]
- [✓ A weekly digest of Stories & Tweets from my network]

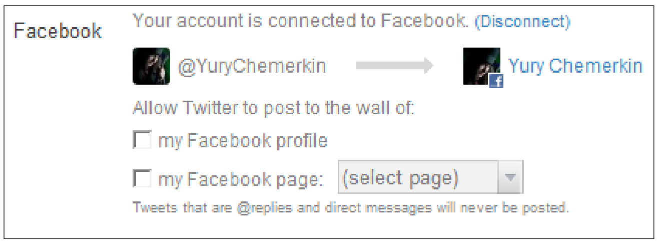
Figure 12. *Email settings*

If you want to add a mobile number perform with actions below:

- Follow [*https://twitter.com/devices*] or
  - Log in to *twitter.com*
  - Click on the person icon and select Settings from the drop-down menu.
  - Click on the Mobile tab.
- Choose your country and enter your mobile number.
- Click Activate phone to start verifying your phone.
- You need to send the word 'GO' via text message to Twitter 8080
- Text the verification code from your phone to that short code.

Tab “Profile” mostly is up to you concerning Your Name, Location, Web site, and biography except linking with Facebook. You should understand if you want separate your tweets and Facebook posts then it’s good idea to create Facebook Page else (if you’re OK with mixing social updates) is good idea to link your Twitter to the Facebook profile only and keep Facebook pages for another content. Your name is a personal or business identifier or real name that displayed in your profile page and used to identify you to friends, especially if your username is mysterious like @XXX.

MakeUseOf wrote an interesting article “Why You Shouldn’t Integrate Facebook, Twitter, & LinkedIn” [*http://www.makeuseof.com/tag/integrate-facebook-twitter-linkedin/*] where compares positive and negative aspects of linked social life stream


Facebook

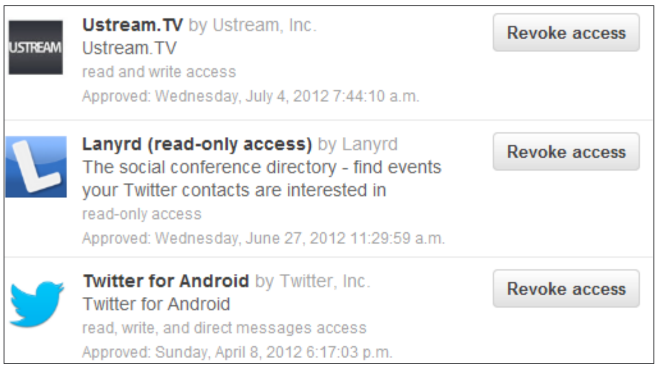
Your account is connected to Facebook. (Disconnect)

@YuryChemerkin → Yury Chemerkin

Allow Twitter to post to the wall of:
- my Facebook profile
- my Facebook page: (select page)

Tweets that are @replies and direct messages will never be posted.

Figure 13. *Facebook linking*


Ustream.TV by Ustream, Inc.
Ustream.TV
read and write access
Approved: Wednesday, July 4, 2012 7:44:10 a.m.
Revoke access

Lanyrd (read-only access) by Lanyrd
The social conference directory - find events your Twitter contacts are interested in
read-only access
Approved: Wednesday, June 27, 2012 11:29:59 a.m.
Revoke access

Twitter for Android by Twitter, Inc.
Twitter for Android
read, write, and direct messages access
Approved: Sunday, April 8, 2012 6:17:03 p.m.
Revoke access

Figure 14. *Applications access settings*

on Facebook and Twitter. Quoting, on Twitter you may want to gain followers by posting a tweet intentionally designed to get a reaction. While random people will love the snark, the less tech-savvy family members may take it seriously, and business contacts may balk at the controversial nature of it. However, on Facebook you may have some personal news that needs sharing with your nearest and dearest. While friends and family will be keen to hear the news, Twitter people will not care and LinkedIn people may resent what they will undoubtedly consider spam and/or utter gibberish.

Tab "Apps" in account settings allow you to manage with third-party access. Here you will see app name, date and time when you granted access this application and access type (read, write, direct messages). You can only revoke access or not to grant access while Facebook gives feature to set access types you want. Best practices say to check from time to time this section and revoke access for application you stopped using.

On another side you're allowed control who can see your Tweets (for Twitter application) on Facebook Application settings by clicking Edit for the Twitter app You still able to change settings to "Only for Me" state when no one tweet will be published on Facebook. Note, some applications may ask Twitter access and never post anything like a Chime.In because you setup your default Chime. In way of posting. However, Viadeo make your accounts cross-posted as soon as you link them.

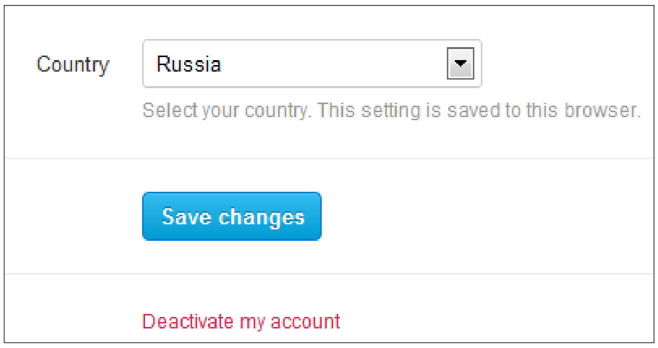
General Activity on Twitter

Delete account

The last feature of tab "account" gives rights to deactivate your profile by agreeing with it and entering password. Your data is going to keep for one month only before it will be deleted. To reactive your account just login in. When a month is over your account is vanished in a few minutes, your data in a few days, and indexed data may keep as long as it can according to the search engine rules, except you write him asking to wipe your data. (Example for Google is at [*http://www.google.com/support/webmasters/bin/answer.py?answer=64033&ctx=sibling*])

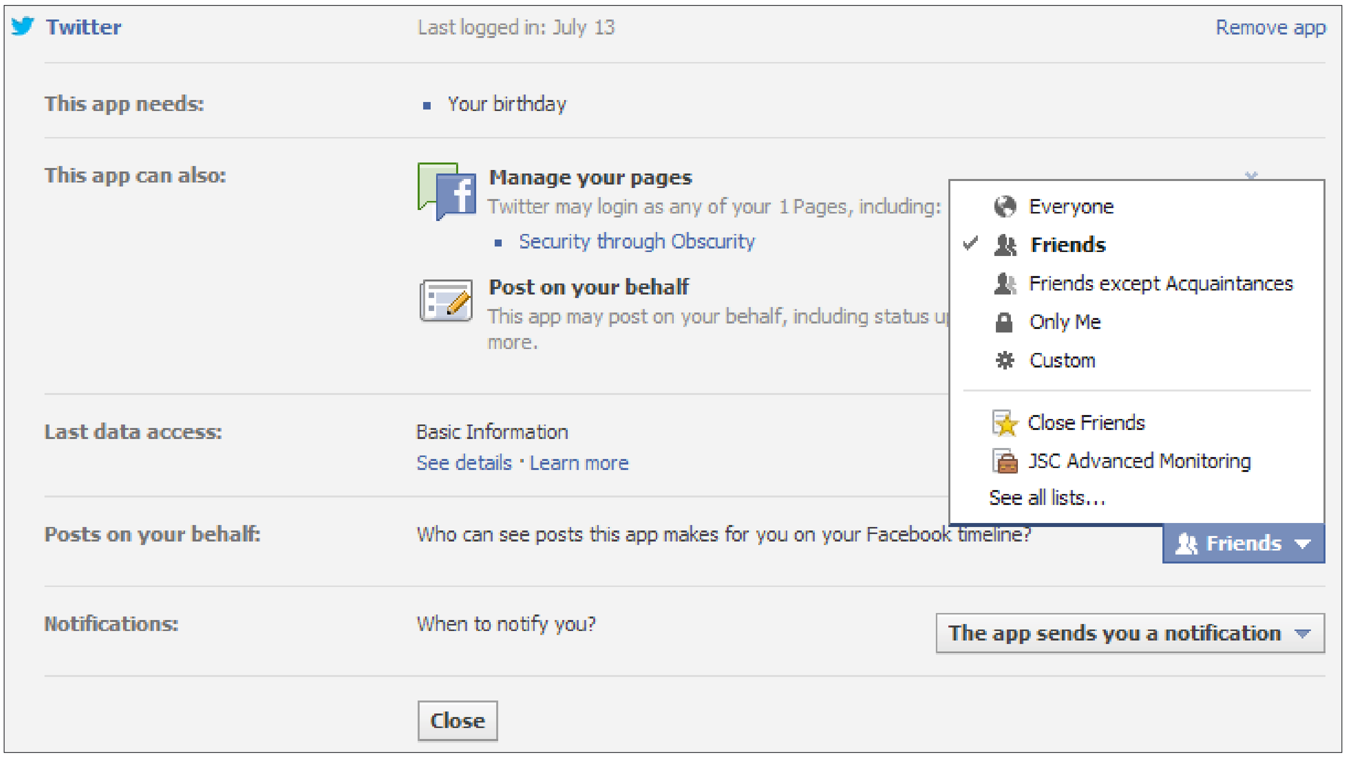
Delete Tweet

How to delete account via long-term deactivation wrote above. You may want to block someone on Twitter also. In this case, you need go to the profile page of the person you want to block and click on


Country: Russia

Select your country. This setting is saved to this browser.

Figure 16. *Delete Account*


Twitter — Last logged in: July 13 — Remove app

This app needs: Your birthday

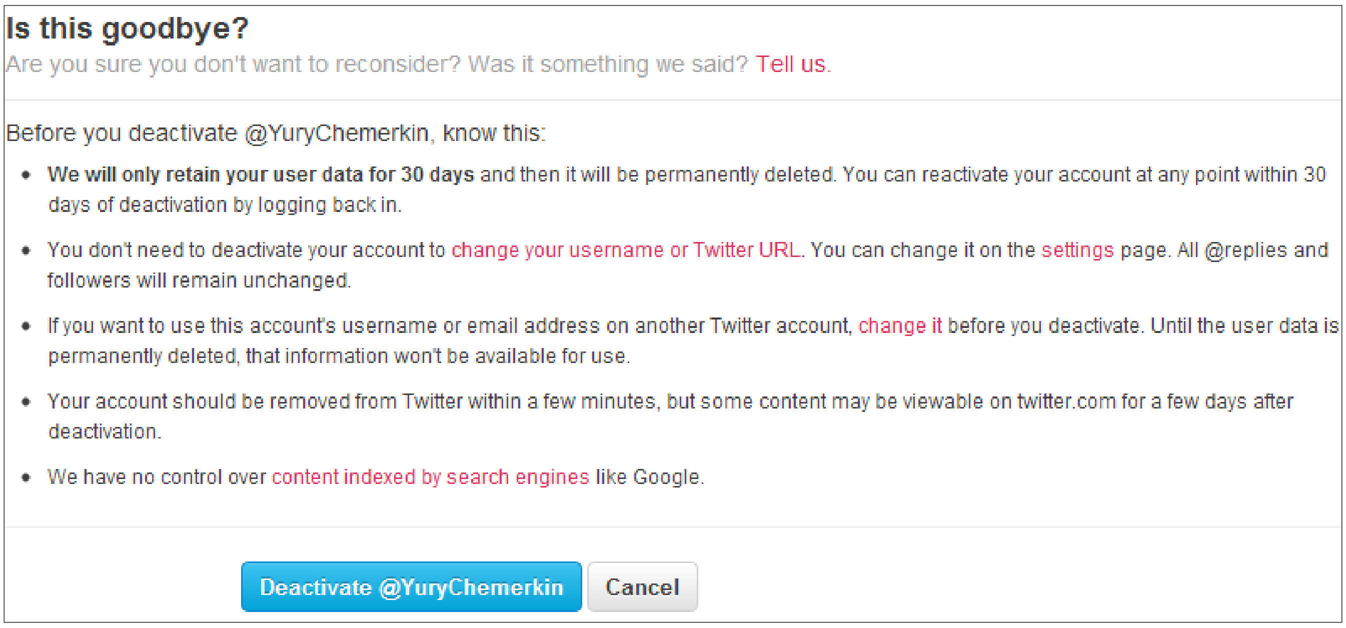
This app can also:
- **Manage your pages** — Twitter may login as any of your 1 Pages, including: Security through Obscurity
- **Post on your behalf** — This app may post on your behalf, including status u... more.

Last data access: Basic Information — See details · Learn more

Posts on your behalf: Who can see posts this app makes for you on your Facebook timeline? — Friends

Notifications: When to notify you? — The app sends you a notification

Figure 15. *Applications access settings*

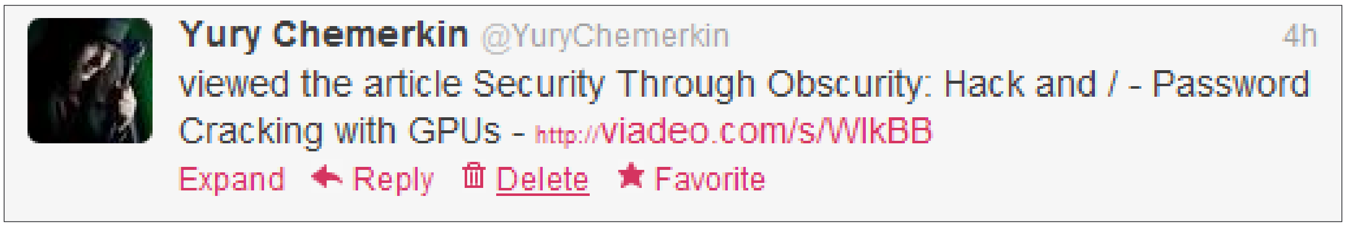

Is this goodbye?

Are you sure you don't want to reconsider? Was it something we said? Tell us.

Before you deactivate @YuryChemerkin, know this:

- **We will only retain your user data for 30 days** and then it will be permanently deleted. You can reactivate your account at any point within 30 days of deactivation by logging back in.
- You don't need to deactivate your account to change your username or Twitter URL. You can change it on the settings page. All @replies and followers will remain unchanged.
- If you want to use this account's username or email address on another Twitter account, change it before you deactivate. Until the user data is permanently deleted, that information won't be available for use.
- Your account should be removed from Twitter within a few minutes, but some content may be viewable on twitter.com for a few days after deactivation.
- We have no control over content indexed by search engines like Google.

Figure 17. *Delete Account*


Yury Chemerkin @YuryChemerkin 4h

viewed the article Security Through Obscurity: Hack and / - Password Cracking with GPUs - http://viadeo.com/s/WlkBB

Figure 18. *Delete Tweet*

his person icon whereupon select Block. Forward blocked accounts cannot add you in their lists or follow you until you unblock them, see your profile picture in their timeline. In addition, you can unfollow or report as spam this person by select "Report @username for spam" in that menu.

To delete tweet.

Locate the Tweet you want to delete and hover mouse over the message and click Delete then.

However, you cannot delete several tweets at a time, only manually one by one.

Grabbing twitter's data from your account

To this day this feature is available only for Facebook users, while Twitter gives users access only to the last few thousand posts made to the site. Twitter has been slower to roll out a similar service, although a number of third-party services and developers have cobbled together ways to let people sift through portions of Twitter's vast collection of messages. On July 24th article in NY Times was published to announce that now Twitter is working on a tool to let their users to have exported all of their tweets that means tweet will be downloaded into a file.

YURY CHEMERKIN

Graduated at Russian State University for the Humanities (http://rggu.com/) in 2010. At present postgraduate at RSUH.

Information Security Researcher since 2009 and currently works as mobile and social infoSecurity researcher in Moscow.

Experienced in Reverse Engineering, Software Programming, Cyber & Mobile Security Researching, Documentation, and Security Writing as regular contributing. Now researching Cloud Security and Social Privacy.

Contacts:

I have a lot of social contacts, that's way to choose the most suitable way for you on Re.Vu http://re.vu/yury.chemerkin

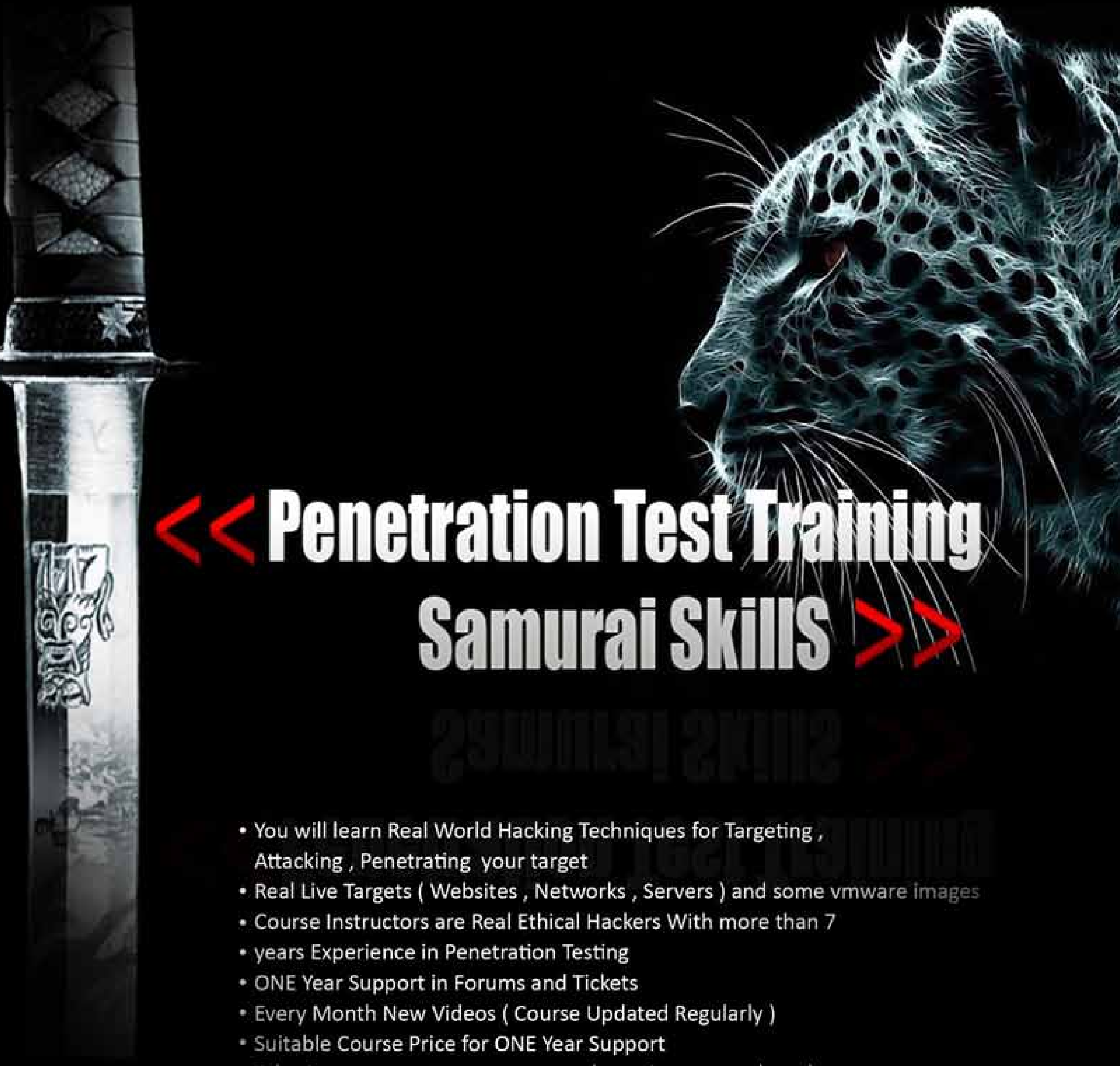
Regular blog: http://security-through-obscurity.blogspot.com

Regular Email: yury.chemerkin@gmail.com

Skype: yury.chemerkin

Other my contacts (blogs, IM, social networks) you will find among http links and social icons before TimeLine section on Re.Vu

CODENAME:
SAMURAI SKILLS COURSE



- You will learn Real World Hacking Techniques for Targeting , Attacking , Penetrating your target
- Real Live Targets (Websites , Networks , Servers) and some vmware images
- Course Instructors are Real Ethical Hackers With more than 7
- years Experience in Penetration Testing
- ONE Year Support in Forums and Tickets
- Every Month New Videos (Course Updated Regularly)
- Suitable Course Price for ONE Year Support
- Take Our course at your own pace (any time , any where)
- Our Course is Totally Different from Other Courses (new Techniques)

WhatsApp InSecurity

The ownership of smartphones and tablets has grown enormously over the past few years. WhatsApp has gained popularity as the cross-platform application to replace traditional messaging services such as Instant Messaging and SMS. How safe is it to use for personal communication?

What you will learn…

- How WhatsApp sends messages
- How to extract content sent via WhatsApp

What you should know…

- Basic packet analysis

Mobility is the way of life nowadays with smartphones and tablets commonly harnessed in developed as well as developing countries. We are seeing huge amounts of data sent across the Internet. In a recent study commissioned by Cisco, it is highlighted that Global mobile data traffic increased by more than 133% to 0.6 Exabytes per month in 2011 from 2010. This number is expected to exponentially grow to 10.8 Exabytes by 2016. This large volume of traffic contains business and personal data such as emails, documents, photos, Instant messages and other web-related content.

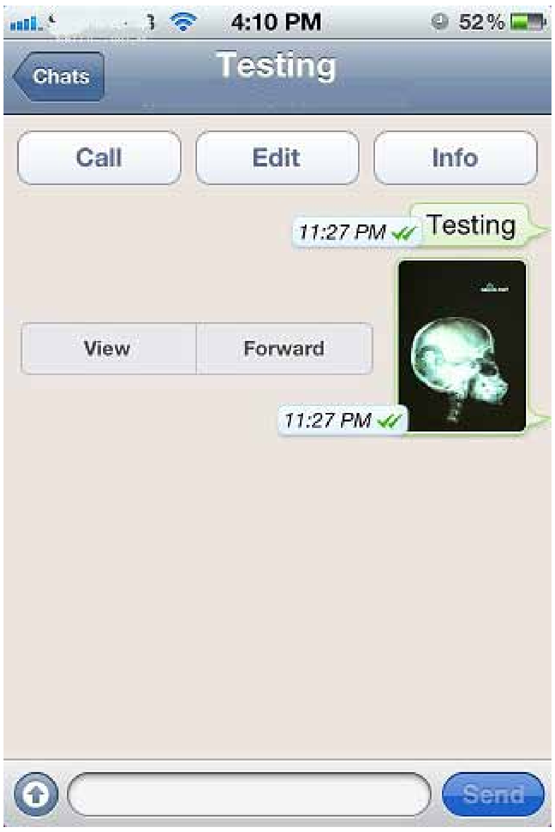


Figure 1. *Sending test message and photo*

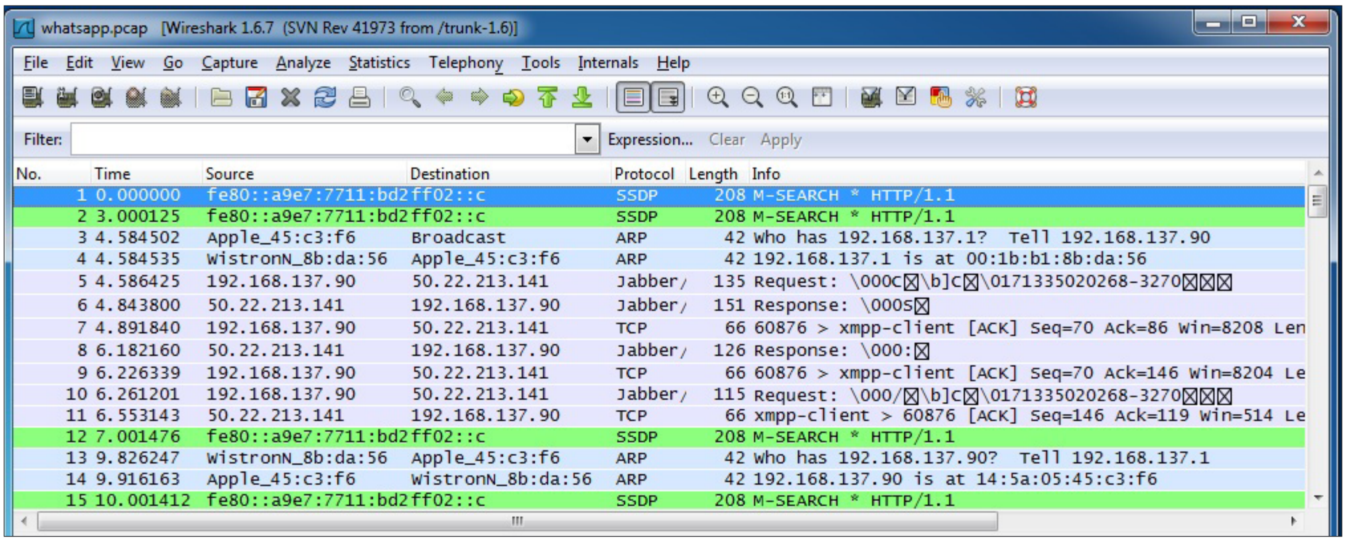
WhatsApp

As people evolve the way they communicate, a bulk of traffic traversing the world comprises text messages instead of voice transmissions.

WhatsApp is a cross-platform mobile messaging application which allows users to exchange text messages over their 3G subscription or Wifi. It is fast gaining popularity and replacing traditional messaging services such as Instant Messaging (eg. MSN Messenger, Yahoo! Messenger) and even SMSs. According to the WhatsApp team, they are delivering more than 1 billion messages a day. The app also supports the sending of photographs, videos and even has the capability of sharing one's location.

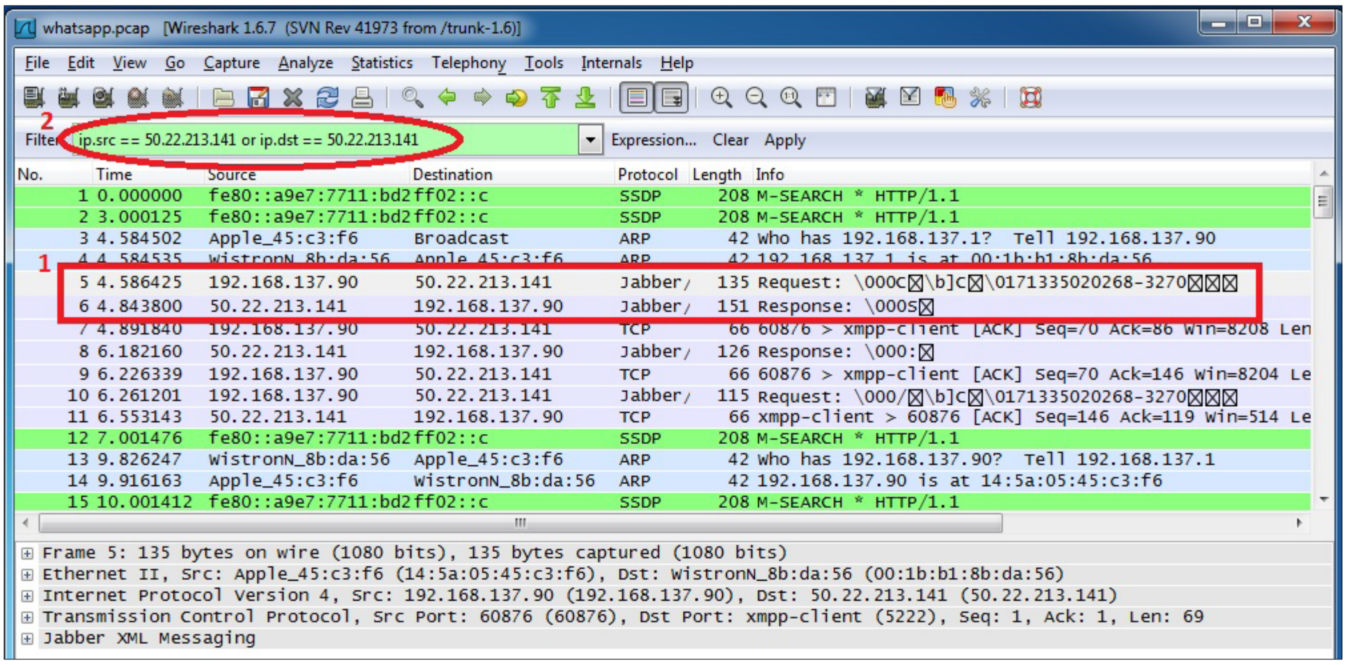
Demonstration

An iPhone was connected to a wireless network to facilitate the demonstration of extract-


whatsapp.pcap [Wireshark 1.6.7 (SVN Rev 41973 from /trunk-1.6)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply

```
No.  Time      Source                Destination        Protocol Length Info
  1 0.000000   fe80::a9e7:7711:bd2 ff02::c            SSDP     208 M-SEARCH * HTTP/1.1
  2 3.000125   fe80::a9e7:7711:bd2 ff02::c            SSDP     208 M-SEARCH * HTTP/1.1
  3 4.584502   Apple_45:c3:f6      Broadcast          ARP       42 Who has 192.168.137.1?  Tell 192.168.137.90
  4 4.584535   WistronN_8b:da:56   Apple_45:c3:f6     ARP       42 192.168.137.1 is at 00:1b:b1:8b:da:56
  5 4.586425   192.168.137.90      50.22.213.141      Jabber/  135 Request: \000C⊠\b]c⊠\0171335020268-3270⊠⊠⊠
  6 4.843800   50.22.213.141       192.168.137.90     Jabber/  151 Response: \000s⊠
  7 4.891840   192.168.137.90      50.22.213.141      TCP       66 60876 > xmpp-client [ACK] Seq=70 Ack=86 Win=8208 Len
  8 6.182160   50.22.213.141       192.168.137.90     Jabber/  126 Response: \000:⊠
  9 6.226339   192.168.137.90      50.22.213.141      TCP       66 60876 > xmpp-client [ACK] Seq=70 Ack=146 Win=8204 Le
 10 6.261201   192.168.137.90      50.22.213.141      Jabber/  115 Request: \000/⊠\b]c⊠\0171335020268-3270⊠⊠⊠
 11 6.553143   50.22.213.141       192.168.137.90     TCP       66 xmpp-client > 60876 [ACK] Seq=146 Ack=119 Win=514 Le
 12 7.001476   fe80::a9e7:7711:bd2 ff02::c            SSDP     208 M-SEARCH * HTTP/1.1
 13 9.826247   WistronN_8b:da:56   Apple_45:c3:f6     ARP       42 Who has 192.168.137.90?  Tell 192.168.137.1
 14 9.916163   Apple_45:c3:f6      WistronN_8b:da:56  ARP       42 192.168.137.90 is at 14:5a:05:45:c3:f6
 15 10.001412  fe80::a9e7:7711:bd2 ff02::c            SSDP     208 M-SEARCH * HTTP/1.1
```

Figure 2. *Running Wireshark*


whatsapp.pcap [Wireshark 1.6.7 (SVN Rev 41973 from /trunk-1.6)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
2
Expression... Clear Apply
1

Filter: ip.src == 50.22.213.141 or ip.dst == 50.22.213.141

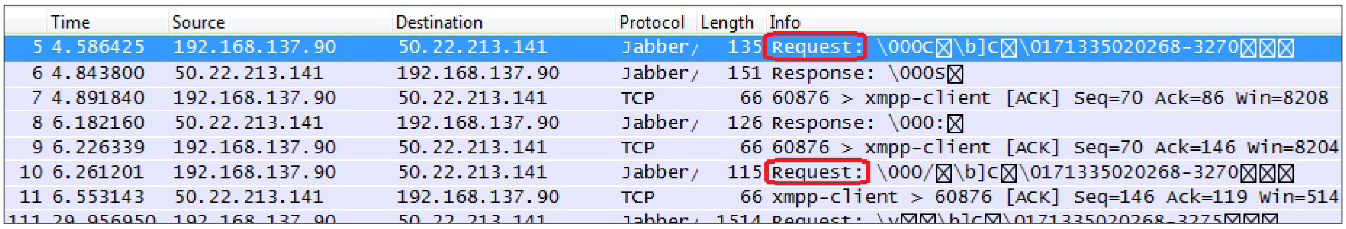
```
No.  Time      Source                Destination        Protocol Length Info
  1 0.000000   fe80::a9e7:7711:bd2 ff02::c            SSDP     208 M-SEARCH * HTTP/1.1
  2 3.000125   fe80::a9e7:7711:bd2 ff02::c            SSDP     208 M-SEARCH * HTTP/1.1
  3 4.584502   Apple_45:c3:f6      Broadcast          ARP       42 Who has 192.168.137.1?  Tell 192.168.137.90
  4 4.584535   WistronN_8b:da:56   Apple_45:c3:f6     ARP       42 192.168.137.1 is at 00:1b:b1:8b:da:56
  5 4.586425   192.168.137.90      50.22.213.141      Jabber/  135 Request: \000C⊠\b]c⊠\0171335020268-3270⊠⊠⊠
  6 4.843800   50.22.213.141       192.168.137.90     Jabber/  151 Response: \000s⊠
  7 4.891840   192.168.137.90      50.22.213.141      TCP       66 60876 > xmpp-client [ACK] Seq=70 Ack=86 Win=8208 Len
  8 6.182160   50.22.213.141       192.168.137.90     Jabber/  126 Response: \000:⊠
  9 6.226339   192.168.137.90      50.22.213.141      TCP       66 60876 > xmpp-client [ACK] Seq=70 Ack=146 Win=8204 Le
 10 6.261201   192.168.137.90      50.22.213.141      Jabber/  115 Request: \000/⊠\b]c⊠\0171335020268-3270⊠⊠⊠
 11 6.553143   50.22.213.141       192.168.137.90     TCP       66 xmpp-client > 60876 [ACK] Seq=146 Ack=119 Win=514 Le
 12 7.001476   fe80::a9e7:7711:bd2 ff02::c            SSDP     208 M-SEARCH * HTTP/1.1
 13 9.826247   WistronN_8b:da:56   Apple_45:c3:f6     ARP       42 Who has 192.168.137.90?  Tell 192.168.137.1
 14 9.916163   Apple_45:c3:f6      WistronN_8b:da:56  ARP       42 192.168.137.90 is at 14:5a:05:45:c3:f6
 15 10.001412  fe80::a9e7:7711:bd2 ff02::c            SSDP     208 M-SEARCH * HTTP/1.1

⊞ Frame 5: 135 bytes on wire (1080 bits), 135 bytes captured (1080 bits)
⊞ Ethernet II, Src: Apple_45:c3:f6 (14:5a:05:45:c3:f6), Dst: WistronN_8b:da:56 (00:1b:b1:8b:da:56)
⊞ Internet Protocol Version 4, Src: 192.168.137.90 (192.168.137.90), Dst: 50.22.213.141 (50.22.213.141)
⊞ Transmission Control Protocol, Src Port: 60876 (60876), Dst Port: xmpp-client (5222), Seq: 1, Ack: 1, Len: 69
⊞ Jabber XML Messaging
```

Figure 3. *Filter IP address*

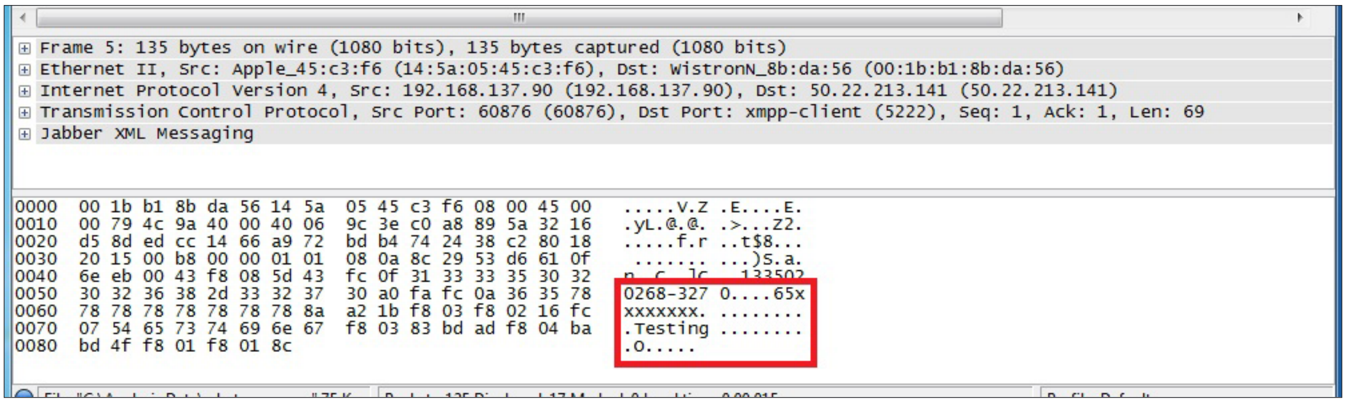
ing text and photos sent via WhatsApp. A test message and a photo were sent with the smartphone connected to the wireless network (Figure 1). Wireshark is engaged to capture all the traffic that passes your wireless interface. It is a network protocol analyser that is capable of capturing network packets and displaying their contents (Figure 2).

WhatsApp uses the Extensible Messaging and Presence Protocol (XMPP) or otherwise commonly known as Jabber. It is an open-standard communications protocol which does not provide encryption of data in transit. Start off the exercise by isolating traffic that is using the "Jabber" protocol. The source of Jabber traffic is likely to be WhatsApp. Reveal only the relevant traffic for analysis by specifying the des-



```
    Time        Source            Destination       Protocol Length Info
  5 4.586425    192.168.137.90    50.22.213.141     Jabber/   135 Request: \000C⊠\b]c⊠\0171335020268-3270⊠⊠⊠
  6 4.843800    50.22.213.141     192.168.137.90    Jabber/   151 Response: \000s⊠
  7 4.891840    192.168.137.90    50.22.213.141     TCP        66 60876 > xmpp-client [ACK] Seq=70 Ack=86 Win=8208
  8 6.182160    50.22.213.141     192.168.137.90    Jabber/   126 Response: \000:⊠
  9 6.226339    192.168.137.90    50.22.213.141     TCP        66 60876 > xmpp-client [ACK] Seq=70 Ack=146 Win=8204
 10 6.261201    192.168.137.90    50.22.213.141     Jabber/   115 Request: \000/⊠\b]c⊠\0171335020268-3270⊠⊠⊠
 11 6.553143    50.22.213.141     192.168.137.90    TCP        66 xmpp-client > 60876 [ACK] Seq=146 Ack=119 Win=514
111 29.956950   192.168.137.90    50.22.213.141     Jabber/  1514 Request: \v⊠⊠\b]c⊠\0171335020268-3275⊠⊠⊠
```

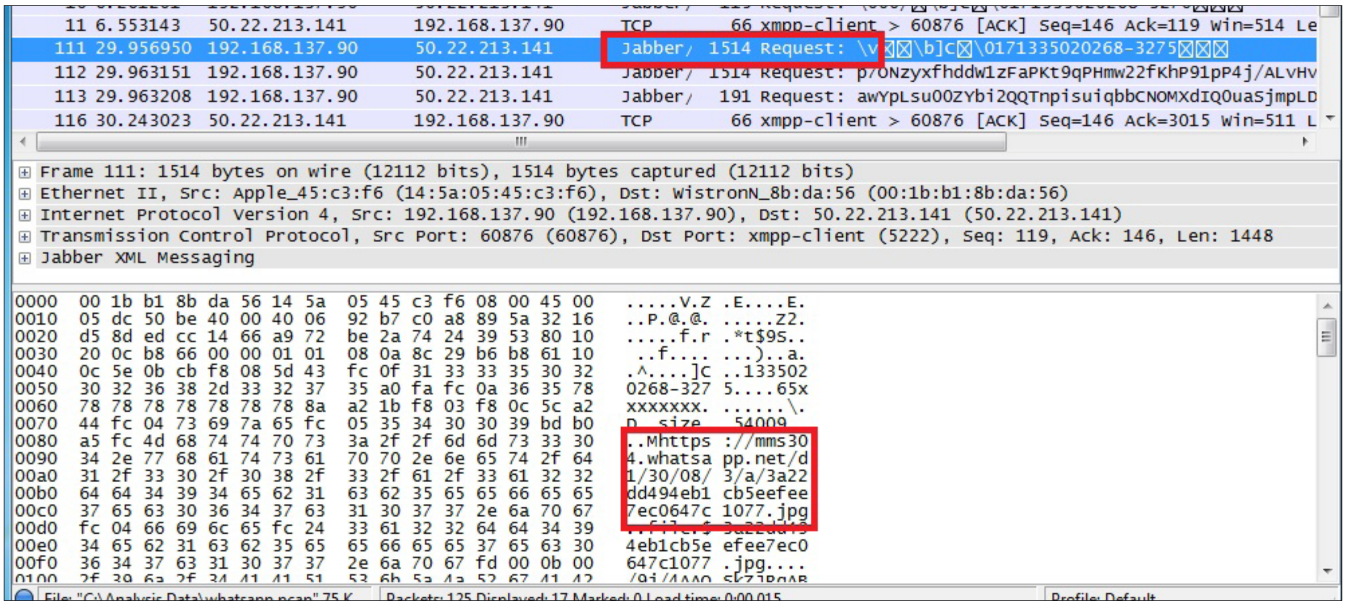
Figure 4. *"Request" Packet*



```
Frame 5: 135 bytes on wire (1080 bits), 135 bytes captured (1080 bits)
Ethernet II, Src: Apple_45:c3:f6 (14:5a:05:45:c3:f6), Dst: WistronN_8b:da:56 (00:1b:b1:8b:da:56)
Internet Protocol Version 4, Src: 192.168.137.90 (192.168.137.90), Dst: 50.22.213.141 (50.22.213.141)
Transmission Control Protocol, Src Port: 60876 (60876), Dst Port: xmpp-client (5222), Seq: 1, Ack: 1, Len: 69
Jabber XML Messaging

0000  00 1b b1 8b da 56 14 5a  05 45 c3 f6 08 00 45 00   .....V.Z .E....E.
0010  00 79 4c 9a 40 00 40 06  9c 3e c0 a8 89 5a 32 16   .yL.@.@. .>...Z2.
0020  d5 8d ed cc 14 66 a9 72  bd b4 74 24 38 c2 80 18   .....f.r ..t$8...
0030  20 15 00 b8 00 00 01 01  08 0a 8c 29 53 d6 61 0f    ....... ...)S.a.
0040  6e eb 00 43 f8 08 5d 43  fc 0f 31 33 33 35 30 32   n..C..]C ..133502
0050  30 32 36 38 2d 33 32 37  30 a0 fa fc 0a 36 35 78   0268-327 0....65x
0060  78 78 78 78 78 78 78 8a  a2 1b f8 03 f8 02 16 fc   xxxxxxx. ........
0070  07 54 65 73 74 69 6e 67  f8 03 83 bd ad f8 04 ba   .Testing ........
0080  bd 4f f8 01 f8 01 8c                               .O.....
```

Figure 5. *Message extracted*



```
 11 6.553143   50.22.213.141   192.168.137.90  TCP       66 xmpp-client > 60876 [ACK] Seq=146 Ack=119 Win=514 Le
111 29.956950  192.168.137.90  50.22.213.141   Jabber/  1514 Request: \v⊠⊠\b]C⊠\0171335020268-3275⊠⊠⊠
112 29.963151  192.168.137.90  50.22.213.141   Jabber/  1514 Request: p/ONzyxfhddW1zFaPKt9qPHmw22fKhP91pP4j/ALvHv
113 29.963208  192.168.137.90  50.22.213.141   Jabber/   191 Request: awYpLsu00ZYbi2QQTnpisuiqbbCNOMXdIQ0uaSjmpLD
116 30.243023  50.22.213.141   192.168.137.90  TCP       66 xmpp-client > 60876 [ACK] Seq=146 Ack=3015 Win=511 L

Frame 111: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
Ethernet II, Src: Apple_45:c3:f6 (14:5a:05:45:c3:f6), Dst: WistronN_8b:da:56 (00:1b:b1:8b:da:56)
Internet Protocol Version 4, Src: 192.168.137.90 (192.168.137.90), Dst: 50.22.213.141 (50.22.213.141)
Transmission Control Protocol, Src Port: 60876 (60876), Dst Port: xmpp-client (5222), Seq: 119, Ack: 146, Len: 1448
Jabber XML Messaging

0000  00 1b b1 8b da 56 14 5a  05 45 c3 f6 08 00 45 00   .....V.Z .E....E.
0010  05 dc 50 be 40 00 40 06  92 b7 c0 a8 89 5a 32 16   ..P.@.@. .....Z2.
0020  d5 8d ed cc 14 66 a9 72  be 2a 74 24 39 53 80 10   .....f.r .*t$9S..
0030  20 0c b8 66 00 00 01 01  08 0a 8c 29 b6 b8 61 10    ..f.... ...)..a.
0040  0c 5e 0b cb f8 08 5d 43  fc 0f 31 33 33 35 30 32   .^....]C ..133502
0050  30 32 36 38 2d 33 32 37  35 a0 fa fc 0a 36 35 78   0268-327 5....65x
0060  78 78 78 78 78 78 78 8a  a2 1b f8 03 f8 0c 5c a2   xxxxxxx. ......\.
0070  44 fc 04 73 69 7a 65 fc  05 35 34 30 30 39 bd b0   D..size. .54009..
0080  a5 fc 4d 68 74 74 70 73  3a 2f 2f 6d 6d 73 33 30   ..Mhttps ://mms30
0090  34 2e 77 68 61 74 73 61  70 70 2e 6e 65 74 2f 64   4.whatsa pp.net/d
00a0  31 2f 33 30 2f 30 38 2f  33 2f 61 2f 33 61 32 32   1/30/08/ 3/a/3a22
00b0  64 64 34 39 34 65 62 31  63 62 35 65 65 66 65 65   dd494eb1 cb5eefee
00c0  37 65 63 30 36 34 37 63  31 30 37 37 2e 6a 70 67   7ec0647c 1077.jpg
00d0  fc 04 66 69 6c 65 fc 24  33 61 32 32 64 64 34 39   ..file.$ 3a22dd49
00e0  34 65 62 31 63 62 35 65  65 66 65 65 37 65 63 30   4eb1cb5e efee7ec0
00f0  36 34 37 63 31 30 37 37  2e 6a 70 67 fd 00 0b 00   647c1077 .jpg....
```

Figure 6. *URL extracted*

tination address in the "Filter" section within Wireshark (Figure 2). Set the filter using the expression:

```
ip.src == <IP address> or ip.dest == <IP address>
```

The filtered traffic shows the communication between the smartphone and WhatsApp server with the "Request" and "Response" packets. Locate the packet that reads "Request:" (Figure 3). This indicates the packet used to send out WhatsApp text messages or data. These packets are the target information you want to funnel out. Upon inspecting the "Request" packet, the test message "Testing" with the recipient cell phone number were clearly shown (Figure 4). This verifies that it is unwise to send private details (eg. your Social Security number, username, password) over WhatsApp. Large "Request" packets are likely to contain photographs or videos. Upon closer scrutiny, a URL starting with "HTTPS" and ending with ".jpg" is observed (Figure 5). Accessing this URL using a web browser uncovers the photo sent by the user and stored on a server hosted by the WhatsApp developers.

Conclusion

Text messaging was traditionally used for short and casual conversations over mobile networks. However, applications (such as WhatsApp) have transformed our behaviour and we are seeing personal and sensitive information being sent over the Internet instead. However, messages sent via WhatsApp are not encrypted and can be easily eavesdropped on. It is wise to be cautious with what data you send via these applications as you never know who is listening in on your conversations.

REMUS HO

Remus Ho works as a Security Consultant in Singapore. He has an intense interest in vulnerability research and malware analysis. Visit his blog (http://labs-werew01f.blogspot.com) to read up on various analysis that he has performed. Special thanks to Mervyn Heng for his assistance in editing this article.

NETCLARITY
PREEMPTIVE, PROACTIVE PROTECTION

NG
INTRUSION DEFENSE
NextGen
Network Access Control

Harden your Network from the Inside Out

Network Access Control

Asset Vulnerability Management

Compliance Auditing and Reporting

Info Security Products Guide 2011 GLOBAL PRODUCT EXCELLENCE AWARDS CUSTOMER TRUST ★★★★★

CRN

SC MAGAZINE BEST BUY ★★★★★

CVE COMPATIBLE cve.mitre.org

RSA TOP THREE INNOVATORS RSACONFERENCE2007

TOLLY Up to Spec CERTIFIED

www.netclarity.net

Available through Partners Worldwide

WebHTTrack

HTTrack Website Copier is an open source tool to download an entire website from the Internet locally onto your desktop for offline browsing.

It is a Windows software that spawned WebHTTrack, its Linux/Unix/BSD release. The tool dumps and mirrors the complete contents of the source website you specify to a local directory by replicating the exact directory structure, files and links.

This is beneficial for a security practitioner who wants to perform offline security testing against a website without impacting the server hosting it.

Install WebHTTrack on Ubuntu 10.04 by entering the following command in your Terminal.

```
sudo apt-get install webhttrack
```

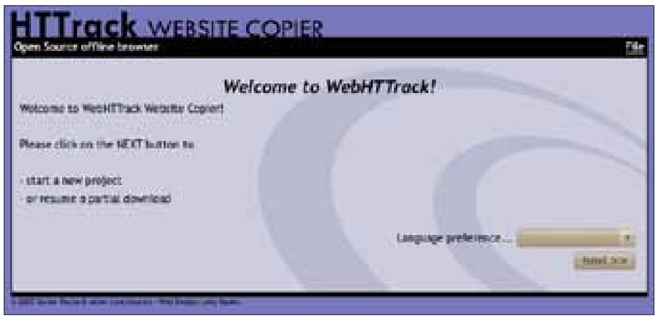
Launch WebHTTrack by clicking on Applications>Internet>WebHTTrack Website Copier. The web interface is now accessible via your default browser. Select your language and click Next.

Give your new project a name, category name and base path before clicking on Next.

Enter details of the URL(s) that you want to mirror locally.

Click Start to initiate the mirroring.

You can monitor the progress of the mirroring. You may opt to skip certain paths or objects and abort the mirror altogether.



Welcome to WebHTTrack!

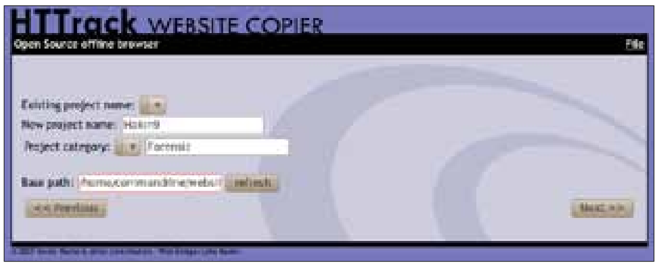
Welcome to WebHTTrack Website Copier!

Please click on the NEXT button to

- start a new project
- or resume a partial download

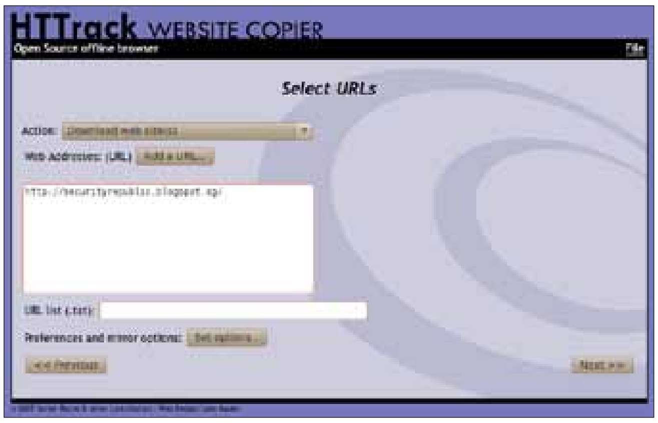
Language preference...

Figure 1. *Web interface*



Existing project name:
New project name: Hakin9
Project category: Forensic
Base path: /home/commandline/websi

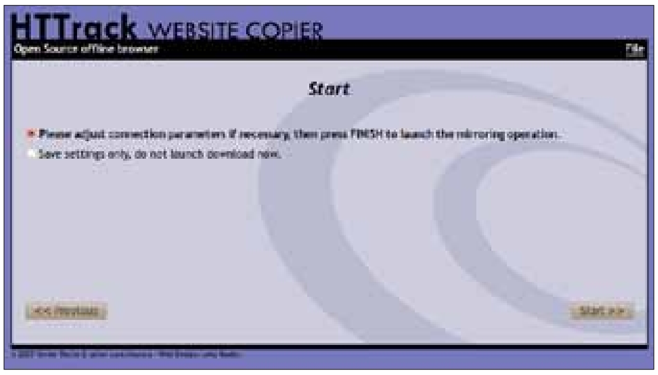
Figure 2. *Project details*



Select URLs

Action: Download web site(s)
Web Addresses: (URL)
http://securityrepublic.blogspot.sg/
URL list (.txt):
Preferences and mirror options:

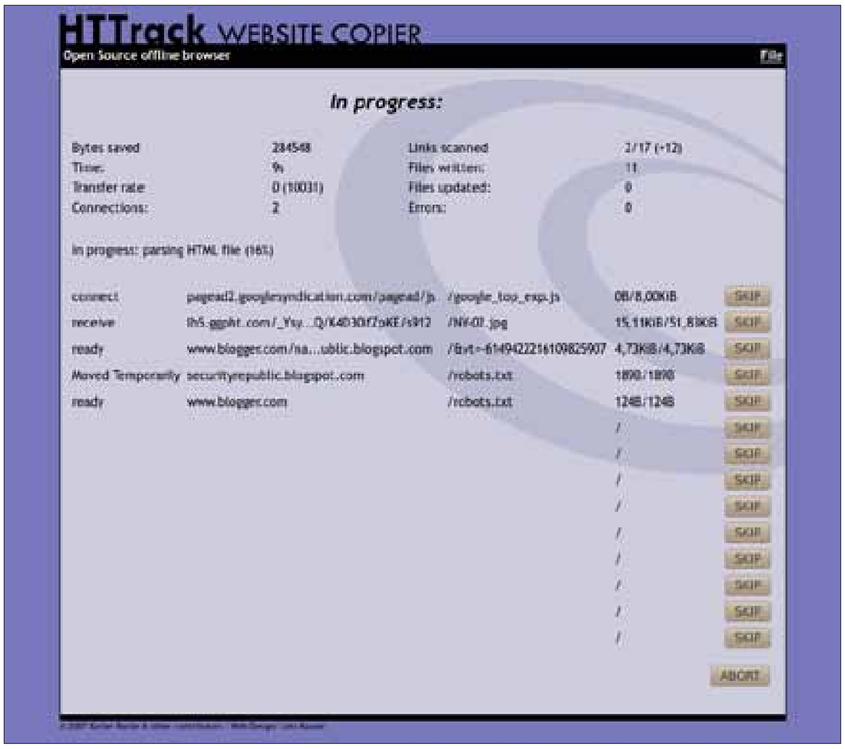
Figure 3. *URLs*



Start

● Please adjust connection parameters if necessary, then press FINISH to launch the mirroring operation.
○ Save settings only, do not launch download now.

Figure 4. *Start mirroring*



HTTrack WEBSITE COPIER
Open Source offline browser

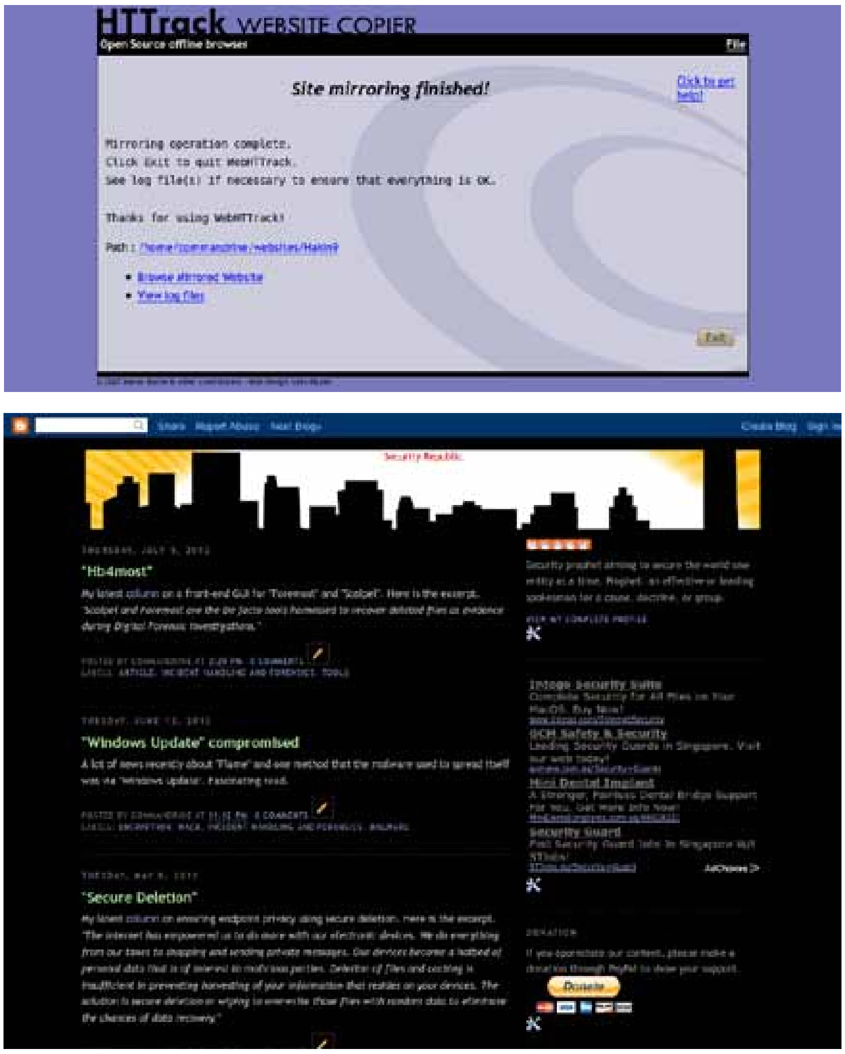
In progress:

```
Bytes saved        284548          Links scanned     2/17 (+12)
Time:              9s              Files written:    11
Transfer rate      0 (10031)       Files updated:    0
Connections:       2               Errors:           0

In progress: parsing HTML file (96%)

connect            pagead2.googlesyndication.com/pagead/js  /google_top_exp.js        0B/8,00KiB         SKIP
receive            lh5.ggpht.com/_Ysy...Q/K4D3Df7oKE/s912   /NV-01.jpg                15,11KiB/51,83KiB  SKIP
ready              www.blogger.com/na...ublic.blogspot.com  /&vt=-6149422216109825907 4,73KiB/4,73KiB   SKIP
Moved Temporarily  securityrepublic.blogspot.com            /robots.txt               189B/189B          SKIP
ready              www.blogger.com                          /robots.txt               124B/124B          SKIP
                                                                                      /                  SKIP
                                                                                      /                  SKIP
                                                                                      /                  SKIP
                                                                                      /                  SKIP
                                                                                      /                  SKIP
                                                                                      /                  SKIP
                                                                                      /                  SKIP
                                                                                      /                  SKIP
                                                                                      /                  SKIP
                                                                                                         ABORT
```

Figure 5. *Progress*



HTTrack WEBSITE COPIER
Open Source offline browser

Site mirroring finished!

Click to get help!

```
Mirroring operation complete.
Click Exit to quit WebHTTrack.
See log file(s) if necessary to ensure that everything is OK.

Thanks for using WebHTTrack!

Path : /home/commandrine/websites/Hakin9
```

- Browse Mirrored Website
- View log files

Exit

Security Republic

"Hb4most"

"Windows Update" compromised

"Secure Deletion"

Figure 6. *Mirror complete*

Once the mirroring is completed, you can directly access the website locally by using the path link at the bottom of the page.

This tool is simple to install and use yet incredibly useful in supporting Application Security testing to find vulnerabilities and also facilitating offline analysis of malicious code as well as malware embedded in websites. It is supported on multiple platforms so try it today.

MERVYN HENG

Mervyn Heng, CISSP, is into Ubuntu, Comic Universe characters, Pop culture and Art outside of Information Security. If you have any comments or queries, please contact him at commandrine@gmail.com.

The $35,000,000,000 Problem

$35,000,000,000. Thirty five billion dollars. It's a big number. It's roughly the same as the net income of all US banks in the first quarter of 2012. It's the same amount the US taxpayer has put in to bailing out General Motors.

What you will learn…

- How code testing supports manual penetration testing
- How insecure code increase the attack surface of organisations
- The relationship between penetration testing, DAST, SAST and the software development lifecycle
- The role of third party software developers in security

What you should know…

- The difference between static and dynamic code testing
- How and why organisations use discovery scanning to manage their risk profile and compliance activities

It' s around the same value as that the Gaddafi family's estimated net assets before their downfall. It's also the same amount that Gartner valued the global security market at in 2011. You can do a lot with that much money – it should solve a lot of problems.

And yet security breaches didn't go away in 2011. Or 2012, for that matter. Worse still, security breaches seem to be getting bigger and bigger. To pick just one example, albeit high-profile, Sony estimated the cost of its May 2011 incident at $171 million. That figure contains all of the investigation, cleanup and insurance needed to resolve the issue, as well as the updates made to the network to prevent further occurrences, but it doesn't include costs associated with lawsuits made against the company, as the outcome is hard to predict.

To zero in on this instance, the same basic attack method, SQL injection, was used repeatedly. SQL injection isn't new. Imperva.com estimates that it has been part of 83% of successful attacks since 2005. This has resulted in 312,437,487 data records lost due to hacking with about 262 million records from various breaches including TJMax, RockYou and Heartland. All of these incidents involved SQL injection attacks. Thirty five billion dollars doesn't seem to have solved this problem, nor all the money that was spent in previous years. Think of all the legal liabilities that leaves open; the technical debt that organizations carry, in terms of unfixed flaws in code, and the losses, documented or otherwise, to consumers.

One has to ask "Why?" The security industry has spent a lot of time, and most of its resources, on securing the network layer and end points. Yet it is applications that traverse both to access the organization's data – the subject of most attacks. Gartner estimates that, of that $35bn, only around $1bn was spent globally on securing applications. Let's put that in perspective – that's about 2% of global security spending. Spread that across every major corporation, government department, and small to medium business that has internet facing applications, and that number starts to look very puny indeed. This starts to look ore than somewhat inadequate. Most organizations will spend more on vending facilities in their offices than they do on application security – but a coffee fix in the morning doesn't do much to stop you getting hacked.

Even worse,the problem is a compound one, across a number of levels. Firstly, it's not the organization's security team that develops code – they merely have to deal with the consequences. Secondly, organizations of all sizeson't just develop applications in house – there's Commercial Off The Shelf (COTS), OEM, open source, outsourced development….the list goes on. Thirdly, between 30 to 70 % of code originates outside the development team, as result of the use of resources such as libraries. These factors makes visibility of security in code development even more difficult. Lastly, and this point should not be underestimated, many developers may not know what good looks like when it comes to security flaws in code. Comp sci courses typically have modules on the superficially "sexy" end of security, like cryptography. But how many include, to any sort of adequate level, teaching on how to prevent SQL injection? Even outside the academic sphere, how often do the specifications for an application actually include a fully articulated set of security requirements for the code?

So we have a picture emerging of a complex web of issues in the application security space. On the one hand, software is vital to business. On the other there is potentially a vast heap of vulnerabilities to address in both new and legacy applications and a finite amount of resources to address this. Where should an organization begin? That is often, quite literally, the first question to address; after all, corporate asset registers are often incomplete or badly kept. There will be corporate web-facing applications developed outside of "normal" channels. Marketing may have run up any number of micro sites to service specific campaigns, only to forget about them. Even when discovery exercises are run, they may well be imperfect or incomplete – for example, if these are based around IP scanning, they'll pick up IP addresses, not the complete universe of possibly thousands of apps running on ports 80 and 443. And let's not forget, the whole exercise is probably being run by the security department – the same department that has a million other responsibilities, from policy to awareness to access management. Worse still, in most organizations, development teams are in a totally different management chain to security, and have worked hard to keep security from "interfering" with the process of delivering applications promptly. The stock of problems is continu-

advertisement

Workbooks.com

Web Based CRM & Business Applications for small and medium sized businesses

Find out how Workbooks CRM can help you

- Increase Sales
- Generate more Leads
- Increase Conversion Rates
- Maximise your Marketing ROI
- Improve Customer Retention

Contact Us to Find Out More

+44(0) 118 3030 100
info@workbooks.com

ally being added to, as new insecure apps are added to older insecure ones.

How, then, do you eat an elephant? The answer, obviously, is: one slice at a time. Let's just remind ourselves of the consequences of not doing this – a business world increasingly dependent on a growing number of insecure applications, with costs of breaches spiraling. Inevitably, these costs would get passed on – back up the to third parties, and to consumers. This is not an attractive prospect. The problem needs to be addressed from the both outside in, and the inside out. Starting from the inside, static code analysis, on code in development, would help the development community identify flaws in code before they hit the production environment. From the outside, carefully structured discovery exercises are needed to enumerate the attack surface of the organization, and hence its current risk profile. Performing dynamic code testing, of applications facing the internet, helps enumerate actual vulnerabilities in production, and therefore acts as a targeting aid for penetration testing. Only by coordinating these elements can a holistic approach be attempted.

What about the third party aspect we identified? A major, coordinated approach would have costs attached. Should the client organization pay for insecure code? Probably not. A supermarket chain wouldn't knowingly accept deliveries of rotten apples week in, week out. Once they know the apples are rotten, they'll either terminate the relationship with their supplier, or put in better contract terms that define the required shelf life of the fruit. Reading it the other way round, there is the opportunity for benefit to the supplier. Well written, secure code has a lower cost of ownership – less need for patching, a lower risk of security breach, and less resulting legal liability. If a third party developer can demonstrate that its products fit this bill, they have competitive advantage. Given the size of the issue, it's clear that their customers, the organizations they develop for, will be demanding this in future. Hence, there is a financial motive for third parties to actively engage in getting this right – and to split the cost of the effort with their enterprise customers. In an ideal world, this would create a virtuous circle, as development teams, both internal and external, compete to produce secure code.

The problem is, we don't live in an ideal world. The suite of measures described all have costs attached. Automation is required to drive down these costs. Most organizations will have trouble finding and allocating enough skilled internal resources to cover the programme management end of all this, let alone the deep technical elements, so outsourcing the exercise makes sense. There will inevitably be pushback from third parties wanting to protect their intellectual property – sharing source code is not a popular activity, hence binary analysis is definitely preferable. Doing nothing is not a real option, and since the problem is ever growing, rapid delivery is essential.

Looking across the market, there really only appears to be one service provider that can tick all these boxes: Veracode. Their SaaS, cloud based services are gaining a growing share of the US markets, with customers ranging from some of the largest names in industry to smaller business. This clearly demonstrates the ability of their service to scale to customer requirements. By comparison, WhiteHat's offerings in the static analysis space are appliance-based, as is HP's Fortify product. Additionally, the latter looks at source code, not binaries, and neither offer the same level of specialist support for remediation as Veracode. In other words, they may be better at finding problems than actually getting them fixed. Moreover, the Veracode business model is the best positioned to tackle the interface between the enterprise and its third parties.

To sum up: big problems require big solutions. It makes little sense for the security industry to hope that the same old approaches will somehow in future address problems they've failed to tackle in the past. More manual penetration testing won't cut it; nor will keeping security out of the software development lifecycle. Failure to take the right track will result in ever increasing costs of failure – an unacceptable outcome, as there's no benefit for anyone involved other than the unethical hacker.

DRAKE
Drake has worked on information security and strategy with government agencies, the military, financial institutions and other blue chip organisations in Europe, the Middle East, and Africa since Boris Yeltsin was President.

ashampoo®

Szukaj nas takze na

www.ashampoo.com

Rootkit Detection

Through Open Source Rootkit Detection Software

Rootkits are computer programs that allow a cyber attacker to covertly take control of a computer and utilize the compromised computer to commit crimes. Additionally, rootkits can also be destructive to a compromised computer by deleting information from the hard drive. The purpose of this research study was to evaluate and analyze a range of open source, free, non fee-based rootkit detection programs.

What you will learn…

- All about rootkits

What you should know…

- Basic knowledge on Information Security

This study evaluated existing research on rootkits, how rootkits operate, and which rootkit detection programs have been previously evaluated. Additionally, this study conducted testing on nine open source rootkit detection programs. This study intended to determine how effective the nine open source rootkit detection programs were at detecting and removing a rootkit. Based on the findings of the testing, three of the nine programs performed well at identifying rootkits, and only one program was successful at removing two of the three rootkits. All of the tested programs failed to identify the FU Rootkit. This failure to identify the FU Rootkit is attributed to the evolution of this rootkit, which allowed it to go undetected. The findings of this study confirm that rootkits are not easily detectable and that they are constantly evolving, which aids cyber criminals in committing their crimes. Moreover, existing rootkit detection programs need to be improved and new methods for detecting rootkits need to be developed inorder to keep pace with the evolution of rootkits.

Rootkits Detection Through Open Source Rootkits Detection Software

As the world became more reliant on computers for storing information electronically, a new valuable target was created. Cybercriminals use rootkit software to facilitate their crimes. Corporations use rootkits to control user actions. Rootkits are applications that provide stealth to hide activities on infected computers or allow access to computers by an unknown attacker. Malicious software (malware) can be destructive by deleting the hard drive or preventing access to an infected computer. Along with rootkits, other types of malware include viruses, botnets, worms, logic bombs and Trojan horses (Hoglund & Butler, 2005).

Hoglund and Butler (2005) stated, “A rootkit is a set of programs and code that allows a permanent and undetectable presence on a computer”. Rootkits, a type of malware, can provide covert access to a system for the purposes of maintaining control and monitoring the compromised computer system remotely all while eluding detection (Blunden, 2009). Rootkits can alter acomputer’s operating system by reporting false information, which allows them to evade detection by security software (Hoglund & Butler, 2005).

A significant issue regarding rootkits is methods of detection. Rootkit developers and rootkit detection methods developers are in constant competition (Davis, Bodmen and LeMaster, 2010). Current detection methods are based on the signature and behavior of rootkits. Therefore, these detection methods can be foiled by merely changing the rootkit’s signature and behavior. Rootkit developers further ensure their success by testing their latest rootkit against multiple current rootkit detection software packages (Hoglund & Butler, 2005). According to Tittel and Korelc (n. d.), "No single tool can correctly identify all rootkits and rootkit like behavior" (para. 3).

The purpose of this research study was to evaluate and analyze open source, free, rootkit detection programs. This study intended to answer the following questions: How effective are open source rootkit detection software programs at detecting rookits? If these programs can detect a rootkit, are they able to remove the rootkit?

As a company's revenues decrease the return on investment (ROI) becomes a significant factor in evaluating monies to be spent. Because computer compromises are generally perceived as rarely occurring, computer security budgets are not viewed as viable investments. Companies will accept the risk of weakened or no computer security, rather than paying to mitigate the potential threat. Because of this, information technology (IT) managers operate computer systems with less than adequate network security. Moreover, less than adequate security can create network vulnerabilities to malicious software (Schneier, 2008).

There are many free rootkit detection programs online that are as good if not better than the expensive programs sold commercially (Griffith, 2010). According to Mediati (2010), "Most of the free products we tested put up identical or nearly identical malware detection scores to the paid varietals put out by the same company" (para. 11). The paid and free detection products found all the same malicious programs used for testing (Mediata, 2010).

There is a deficiency in research making direct comparisons between rootkit detection programs. In the article *Review: Six Rootkit Detection Protect Your System,* detection programs were compared, but only provided limited findings about what was discovered by the scans (Yegulalp, 2007). In the Article *Top Free Tools for Rooting out Rootkit Spies*, the author provides suggestions for choosing a detection program, but the article does not explain the methods for testing to determine which programs the author suggests (Spanbauer, 2008). In the book *Hacking Exposed Malware & Rootkits Security Secrets & Solution,* the authors describe detection applications and their methodologies, but never test the programs to find rootkits (Davis, Bodmer & LeMasters, 2010).

In the article *Best Free Rootkit Scanners/Removers* the programs were reviewed, but only descriptions of the applications' detection methods, features, and ease of use were discussed (Davidson, 2012). This source did not test the detection programs to learn if an actual rootkit would be detected. In the book *Hacking Exposed Malware & Rootkits Security Secrets & Solution,* the authors defined the skill level of the user required to comprehend the scan results for each detection program discussed (Davis, Bodmer & LeMasters, 2010).

In the book *The Rootkit Arsenal*, the author focused on rootkit detection methodologies in great detail. The current generally accepted methods are: signature based, behavioral or heuristic, crossview, integrity based and hardware detection. These methods constitute the theoretical definitions used for finding rootkits. Rootkit detection methodologies define the type of rootkit that each method would be able to detect (Blunden, 2009).

The article *Review: Six Rootkit Detectors Protect Your System,* the author did not disclose or discuss any information about the testing methodologies that were utilized. In the article the author claimed a rootkit was found, which might have merely been a false positive or a misreading of the scan results (Yegulalp, 2007). Furthermore, in *Top free tools for rooting out rootkit spies* the author admitted to never finding a rootkit (Spanbauer, 2008). In addition, *Best Free Rootkit Scanner/Remover*, author did not test the programs, but simply focused instead on the programs features for the review (Davidson, 2012).

This study evaluated existing research on rootkit detection programs, in conjunction with direct testing of detection programs by the author. This project will serve as an informative guide of available open source rootkit detection programs. The findings of this project are intended for information technology (IT) personnel and forensic investigators. Based upon the research and testing results of rootkit detection programs, recommendations will be offered as a guide in the selection of current open source rootkit detection programs. Furthermore, should the results of this study reveal that the open source detection programs fail to identify any rootkits, these inadequacies will be addressed. Additionally, if the open source programs find all of the rootkits a recommendation will be made to identify the best program.

Literature Review

Electronically stored information created a target to tempting to be ignored by cybercriminal. Rootkits allow cybercriminals to covertly access, and control computers or computer networks. A segment of effective security for computers and networks are methods for detecting rootkits. The purpose of the study was to evaluate and analyze open source rootkit detection programs.

There is a significant amount of existing research on rootkits and rootkit detection software. The sources for this project were selected because they directly related to rootkits, their use, and how they can or cannot be detected. The sources were a combination of scholarly papers, news and printed media articles, and published books which provided a complete overview of, and credibility to this project.

Rootkits

Cybercriminals use malware to facilitate their crime. Malware are applications that provide valuable information from an infected computer or allows access to that computer by an unknown attacker. Malware can be destructive by deleting files or preventing access to an infected computer (Hoglund & Butler, 2005).

A rootkit is a collection of tools (e.g., binaries, scripts, configuration files) that intruders utilize to conceal their activity on a computer, which can then allow covert monitoring and control of the system for an extended period. “A well-designed rootkit will make a compromised machine appear as though nothing is wrong, allowing attackers to maintain a logistical outpost right under the nose of the system administrator for as long as they wish” (Blunden, 2009, pp. 10-11). Monitoring can include recording keystrokes and reading email, along with capturing passwords, encryption keys and network traffic. Characteristics that set rootkits apart from all other malware are stealth, and their ability to remain functioning after a computer reboot (Hoglund & Butler, 2005).

The first rootkits were found on SunOS systems in 1994. Two years later, in 1996, the first Linux rootkits appeared on the Internet. One year later in 1997, kernelmode rootkits were being discussed as a possibility. In 1998, Silvio Cesare releases the first non-loadable kernel module patching rootkit code. During the same year, Back Orifice, a full featured backdoor program for Microsoft Windows was released. In 1999, the NT rootkit for Windows was developed by Greg Hoglund, who also wrote the book *Rootkits: Subverting the Windows Kernel*.

The next decade saw an accelerated rate in rootkit development. In 2000, the T0rn Kit and Libproc rootkits were combined with a Trojan horse and released to the Internet. In 2002, computer backdoor software became a standard component of rootkits. Also that same year, Hacker Defender was released and became the most widely used Windows rootkit. Because of the growth in popularity of Windows based systems, rootkit developers switched their focus from developing rootkits in Unix based systems to Windows based systems. In 2004, the FU Rootkit was released and introduced a new technique to conceal running computer processes.

In 2005, the Sony BMG rootkit scandal occurred; this was the first time a rootkit was used for a commercial purpose. In 2006, rootkits became part of every major worm and virus. Moreover, at this time virtual rootkits began being developed. In 2008, rootkits began using the Windows boot, the start up process, to install themselves, by adapting code from eEye bootroot rootkit (Davis, Bodmer, & LeMasters, 2010).

How Rootkits Function

Rootkits are categorized by type and location of where their modifications to the operating system (OS) occur (Sparks,Embleton & Zou, n.d.). The earliest generations of rootkits used the technique of file masquerading. File masquerading replaces a system file with a malicious file. The malicious file poses as the original file, but has additional features that are unknown to the user. The login applications were common targets, which enabled an attacker to capture user names and passwords. First generation rootkits were susceptible to integrity checkers. All files have unique cyclic redundancy check values (CRC) that are stored with a computer and are used to verify current files’ integrity. The changed files would not have the same CRC value as the original file, which would indicate an altered file (Sparks, Embleton & Zou, n.d.).

Second generation rootkits used application program interface (API) hooking.

API hooking is a technique where an application’s normal flow is altered to execute code which was not originally part of the application. Rootkits commonly target the filefind API function to control various OS components. The rootkit hooks the OS, which allows it to run the rootkit’s file hiding function first. Second generation rootkits are immune to integrity checkers. They modify the application within the memory to avoid being detected. Rootkits utilizing memory hooks can be detected with memory based integrity checkers and heuristic methods (Sparks, Embleton & Zou, n.d.).

Third generation rootkits avoided memory detection methods by modifying only data. This method is referred to as *direct kernel object manipulation* (DKOM). These rootkits alter the kernel data structures that the OS relies on to function. By controlling the data, a rootkit can indirectly control the execution path or the function to be executed next. DKOMs are difficult to detect because there is no support for checking underlying architecture for memory monitoring, which is needed to validate access to kernel memory. Detection is also more difficult because the kernel data structures change rapidly, which prevents a means of determining normal activity from malicious activities. Filter drivers are also included in the third generation rootkits. Filter drivers exploit the layered feature of Windows device drivers. Rootkits install between the lower and upper layer device driver transparently and intercept and modify communications (Sparks, Embleton & Zou, n.d.).

Fourth generation rootkits include virtual memory subverting, virtualization, system management mode (SMM) memory, and hardware specific, which infect the basic input-output system (BIOS) or Peripheral Component Interconnection (PCI) expansion cards. Rootkits now include their ability to hide within the BIOS of a computer's motherboard. Unlike, previous rootkits, these rootkits are OS independent. These rootkits do not make any changes to a computer's OS. By being independent, rootkits have increased their stealth capabilities (Sparks, Embleton & Zou, n.d.). After installation into the BIOS, the rootkit continues to infect the.system every time the computer is turned on (Zorz, 2011).

SMM based rootkits have the ability to exist independently of the OS. SMM is Intel's processor management mode. SMM manages low level hardware operations like power, and thermal regulations. SMM has its own memory and execution environment. SMM lacks any security to protect its memory. Virtualization machines based rootkits place the OS into a virtual machine where the rootkit can have full control over the OS (Sparks, Embleton & Zou, n.d.).

According to Hoglund and Butler (2005), "During the early 1990s, many hackers figured out how to find and exploit buffer overflows, the "nuclear bomb" of all exploits. However, the virus-writing community didn't catch on for almost a decade." (p. 26). Buffers are specific amounts of memory that are setup to hold information used by an application. Buffer overflows happen when an application writes more data to the buffer than it can hold. By over writing a buffer, a malicious application can change data stored for another application in the memory. For example, the address of the next instruction to be executed can be changed to the address of a malicious application. Buffer overflows can cause a Windows computer to fail, forcing the computer to reboot and thus complete the installation of the malware (Harris, 2008).

Rootkit Uses

Brenner (2010), states that online malware fraud is estimated to produce in an excess of one hundred billion dollars each year. Malware developers create generic or custom rootkits with a virus, Trojan horse or key logger as payloads. This malware is purchased by cybercriminals who utilize the malicious programs to harvest data, for example credit card information and then sell this information for a profit. The data is purchased by another group of criminals who use the information to commit financial fraud.

Carr (2010) discusses cyber warfare stating that, "Cyber Warfare is the art and science of fighting without fighting, of defeating an opponent without spilling their blood." The author provides information about BIOS-based rootkit attacks. These rootkits elude anti-virus programs and cannot be destroyed by rebooting a computer. Furthermore, they can attack other computers and disable antivirus applications. The author also states that, "After infection occurs, the likelihood of a kernel-level rootkit remaining on the machine is worrisome at best, and catastrophic at worst."

Today's rootkits are more sophisticated and covert than before. Rootkits rarely store anything on hard drives, but instead embed within a computer's memory to increase their stealth. Rootkits take precautions to mask their presence and communications. The author suggests possible precautions in an attempt to avoid BIOS-based rootkits. The first precaution would be to enable the write protection on the motherboard, which would prevent any changes to be made to the BIOs. An additional precaution would be to only allow digitally signed BIOs firmware from the manufacturer to be installed (Carr, 2010).

In the book *Cyber Warfare Techniques, Tactics and Tools for Security Practitioners*, the authors describe how malware is used in cyber warfare. Cyber Attacks compromise target computers or networks by gaining unauthorized access. They then install malware and remote control applications to increase their control of the computer. The installed rootkit hides the presence of the attacker along with all the malware that has been installed (Andress & Winterfeld, 2011).

In July of 2010, a new more advanced malicious application infected particular models of Siemens supervisory control and data acquisition (SCADA) systems. This malware application was called

Stuxnet. Stuxnet targeted Siemens industrial automation and control files to sabotage systems being controlled by SCADAs and for industrial espionage. Stuxnet was composed of a worm, Trojan horse and a rootkit. The rootkit prevented discovery of the worm and Trojan horse activities (Andress & Winterfeld, 2011).

In the article *Sony's DRM Rootkit: The Real Story*, SONY BMG Music Entertainment distributed rootkit technology on each of their music compact disc (CD) to prevent duplication of the music contained on the disc. The rootkit was installed, unbeknownst to the user, on computer systems when the music CD was inserted into the computer. The embedded code modified Windows system files to prevent detection, prevented copying the CD and sent information about the user back to Sony BMG. Further study of the rootkit discovered that attackers could use this program to gain access to infected computers (Schneier, 2005).

In the article *Symantec Caught in Norton Rootkit Flap*, Symantec admitted using rootkit technology to hide a directory from Windows file system. Their thought was to keep customers from deleting Norton System Works files unintentionally. The hidden folder's name was NProtect was used by System Work as a protected recycle bin. The problem arose when other malware utilized this same folder to hide from anti-virus scanners (Naraine, 2006).

In the article *FAQ: Behind the Carrier IQ rootkit*, major cell phone carriers admitted to using programs that gather and track data from smartphones. Carrier IQ software was designed to assist wireless service providers to resolve service problems. The application can be modified to allow the phone's physical location, web usage, and text messages to be sent to the phones carrier. The program uses a keylogger and rootkit for concealment. The keylogger records the buttons on the phone that are pushed. Mobile phone carriers can set triggers for certain user activities and when these activities occur, the carrier is notified (Vijayan, 2011).

How Rootkits Avoid Detection

The evolutions of rootkits began back in the late 1980s when the first log cleaners were created (Hoglund and Butler, 2005). These were a collection of programs designed to remove evidence of an intrusion from a computer. These automatic log cleaner kits searched for files on the compromised computer system which contained records of user logs and the computer's activity. Once the files were located the programs would remove certain entries relating to the attacker's activities or they would delete the file (Davis, Bodmer & LeMasters, 2010).

Rootkit authors are adaptive in response to security measure advancements (Embleton, Sparks & Zou, n.d.). Defensive security measures include anti-virus applications, rootkit detection applications, intrusion-detection systems and firewalls (Hoglund and Butler, 2005). Defensive applications that do not use functions provided by an infected computer's OS have increased possibilities of discovering rootkits (Blunden, 2009).

Defensive security measures, such as anti-virus applications, firewalls, an intrusion detection system (IDS), or an intrusion prevention system (IPS) can all be defeated by rootkits. Windows depends on three subsystems: Win32, POSIX and OS/2. These subsystems are made up of well-documented sets of APIs. APIs are relied on by the taskmanager.exe file, Windows Explorer and the registry editor to interact with the OS, which make them a target for a cyber attack (Hoglund and Butler, 2005).

Rootkits filter what an OS, and security applications view. Rootkits modify the execution paths of the OS files or the data stored about processes, drivers, and network connections. Once the rootkit has been installed, it can block calls made to applications, such as anti-virus programs, by replacing some of the original components. Rootkits can also monitor and intercept parameters, and filter output (Blunden, 2009). Rootkits, to maintain their stealth, misdirect the OS and the user by capturing and changing the information exchanged between the OS and the user (Hoglund and Butler, 2005).

Sources Supporting that Rootkits are Detectable

Hoglund and Butler (2005) suggest some basic rootkit detection methods. First is the monitoring of a computer's memory for the installation of rootkits. Another suggestion is for detection to catch rootkits loading or modifying Windows components, which includes monitoring all functions, registry keys, tables and processes for changes. Specifically, Windows functions need to be monitored to confirm if a dynamic link library (DLL) is loaded into another process's address space.

Two registry keys that need to be monitored for changes are HKEY_LOCAL_MACHINE\System\CurrentControlSet\ Services and HKEY_LOCAL_MACHINE \System\ControlSet001\Services. Additionally, the monitoring of Windows tables for modifications where calls branch outside the table's range. The tables have a starting address and size for modules, which determines table ranges for these modules. Any references to modules outside of the table memory addresses are potential rootkit hijacks. Lastly, checks of all DLLs loaded by a process to search for a function address outside the range of the DLL.

Blunden (2009) states that the import address table (IAT) is the main focus of user-mode rootkit modifications. The table entries should be within the address range of each module and DLLs loaded should remain inside the range of addresses. While going through IATs executables, if any modules exceed the expected ranges then they have been hijacked by a rootkit. The author provides source code for developing methods to detect user-mode rootkits.

The cross view detection can be used to detect kernel object modification. This method depends on the ability to collect the same information from multiple sources. As an example, if Windows Notepad was running the process notepad.exe would be visible in the Windows Task Manager. If Notepad's process was hidden with a rootkit notepad.exe would no longer be visible in the Task Manager. Blunden suggests utilizing netstat.exe, another Windows command-line executable to show network activity, which would show Notepad's active process still visible. This difference would indicate the presence of a rootkit. Cross view detection is a comparison of functions where one depends on Windows components and the other function accesses the object directly.

Davis, Bodmer and LeMaster (2010) describe features added to Windows versions to prevent rootkit installations. Microsoft Windows products have services running in restricted privilege modes. This prevents compromised services from escalating privileges. No-Execute and Address Space Layout Randomization functions were added to prevent buffer overflows. Kernel patch protection prevents modification of kernel objects. Windows requires verified device drivers before allowing the drivers to be loaded into the kernel. Verified device drivers have hash values to ensure that the drivers have not been tamper with and can be verified at run time. User Account Controls are the screens requesting permission to install or execute an application.

The authors describe tainted view or cross-view detection techniques. This methodology compares two snapshots of running files, processes, registry keys, hardware installed and names and number of functions use by each running task. Any differences indicate the possible presence of a rootkit. The view produced by Windows API components is referred to as the tainted-view. The view created by the direct access to the hardware is called the clean or trusted view. Rootkit detectors function by taking snapshots of processes running using Windows APIs followed by snapshots of the processes running by checking the internal threading structure in the kernel, which control program execution and is referred to as the clean view. The detector then compares the snapshots for differences and displays those anomalies for review. The authors discussed software-based rootkit detection programs. Software-based detectors are beneficial when used with other detection applications; where one program will detect something another might not. The authors discuss that many of these detection programs are available for free. They recommend using programs, which are rated highly by magazines, experts or security companies.

Automated Rootkit Detection

Butler and Sparks (2010) state that rootkit detection techniques are categorized into five methods. The methods are: signature based, behavioral or heuristic, crossview, integrity based and hardware detection. Each method has a specific algorithm for operation.

Signature based scanning searches stored files for a specific pattern or fingerprint that identifies a specific rootkit. Anti-virus applications also utilize this method to locate viruses and Trojan Horses on computers. When a pattern match is found the file is flagged as infected and the program allows the user to determine if the offending files is ignored, deleted or quarantined.

The behavioral or heuristic detection method functions by identifying anomalies or occasions when the computer performs outside of its normal operating parameters. Crossview based detection is based on duplication of the same task. Where an application calls common APIs to enumerate the computer's files, processes or registry keys, they are then compared with the same data generated by the application without using the computer's common APIs.

Integrity based detection compares the computer's current state in time to a past state. A snapshot of the computer is taken in time as a baseline which is used to compare against newer snapshots of the computer's state. Any differences between the snapshots would be an indicator of a possible rootkit.

Hardware detection is based on the PCI card. The PCI card is installed into the computer to monitor for rootkit activity. The card has its own central processing unit and uses direct memory access to search for hidden rootkit installations.

Sources Indicating that Rootkits are Undetectable

David, Chan, Carlyle and Campbell (n.d) developed a rootkit, Cloaker, which controlled the central processing unit (CPU), but made no other modifications to the computer. Cloaker avoids detection by never making any modifications to the kernel, which would be detected by existing detection methods. Developing rootkits that infect hardware components have much narrower parameters than designing a rootkit for Windows OS base systems. Cloaker works only on computers that use the one particular CPU model the rookkits was designed for.

Embleton, Sparks and Zou (n.d.) discuss the System Management Mode based rootkit (SMBR). SMM manages low-level hardware operations like power and temperature management and has its own memory and execution space, which makes it a great hiding place for rootkits. They detail their experiences developing their own SMBR.

They began by replacing system handlers in the interrupt descriptor table with a pointer to the malicious hook routine. They then changed the delivery mode of the keyboard interrupt requests from fixed to System Management Interrupts (SMI), in order to route the keystrokes through their SMM handler. The authors focused on using the network card controller to transmit the collected keystrokes. The process of sending packets begins with confirming that the Network Interface Card (NIC) is idle, build a transmit block, create a data packet containing the keystrokes, insert the destination's Internet Protocol (IP) address and then sending the packet.

They discussed the hardware and software limitations of their rootkit and suggested improvements for their rootkit. Hardware rootkits are dependent on specific hardware. Since their rootkit makes no modifications to the operating system, it prevents existing detection methods from detecting it.

Embleton, Sparks and Zou (n.d.) discuss what rootkits are and how they work. They detail how rootkits hide their presence, control interfaces and communications and messages. The authors explain the four generations of rootkits and how they have adapted to overcome the security measures engineered to detect and prevent them. The authors explain how hijacking or hooking is used by rootkits to gain control of a computer and hide from detection.

They also describe advanced rootkits, which are Virtual Memory (VMM) Rootkits, SMM Rootkits, and BIOS and PCI Rootkits. These rootkits all avoid modifications of the OS, which elude detection by current detection methods. The authors discuss how cyber attackers are moving away from OS rootkits to evade existing detection software.

Blunden (2009) states that cyber attackers use rootkits techniques to make it difficult to find evidence of their activities. Attackers use various means to foil disc imaging, some examples include encryption, file system attacks, and file concealment. Commercially available programs are used to protect the rootkit's code along with being used to destroy evidence. Applications that perform file wiping, meta-data shredding, modifying timestamps, and encryption are used to destroy anything that was used, in order to hide their activities or hide pieces and activities of the rootkit.

Existing Comparisons of Rootkit Detection Programs

Davis, Bodmer and LeMaster (2010) conducted review of rootkit detection programs. Their review did not include all of the detection programs selected for this study. The authors did not conduct any actual testing of the reviewed programs. Reviews of the programs were based on descriptions of detection methods used and the programs background. They reviewed the following programs: F-Secures Blacklight, Darkspy, IceSword, GMER, Rootkit Revealer, and Rootkit Unhooker.

Blacklight implements the tainted or cross-view method for discovering rootkits. Blacklight has a simple to use interface and quarantines hidden files by renaming them. The program systematically executes every process on the computer and then checks each running process and then attempts to open the process with the OpenProcess function. If the process is not listed in the EPROCESS list then this process is hidden and needs further study.

Darkspy and Icesword utilize tainted-view detection. These two programs require user interactions to analyze and refresh running processes, and kernel modules views when conditions change. These programs require a high level of experience and knowledge to interpret the results. Icesword allows the user to browse the file system, registry keys and processes to search for differences.

GMER provides good rootkit detection in one package. GMER has limited removal capabilities, but it can delete registry keys and files. It looks for hidden files, processes, services and modified registry keys. The results are color coded to identify possible issues.

RootkitRevealer also utilizes tainted-view detection methods. This program only checks the file system and registry for hidden objects. According to the authors, this program is a user friendly, quick scan rootkit detection solution.

Rootkit Unhooker is a tool for expert users. Unhooker allows the user to view the system service descriptor table (SSDT), shadow SSDT, and low level scans of the file system. Unhooker can find and remove transmission control protocol and the internet protocol (TCP/IP) stack modification by a rootkit. This program has the ability to remove rootkit control over TCP/IP and disable modifications, but leaves the rootkit intact for study by researchers or forensic investigators. An issue with Unhooker is that it will cause the blue screen of death (BSoD), a Windows error screen displayed when a system crashes, when closing the application. Investigators can still use the file created by the BSoD, which contains a copy of all memory at the time of the crash.

Yegulalp (2007) reviewed six rootkit detection programs. The author conducted independent testing of the reviewed programs. The programs reviewed were: Blacklight, Icesword, RKDetector, Rootkit Buster, Rootkit Revealer and Rootkit Unhooker. The author used three rootkits as test samples, which included the FU rootkit, AFX Windows rootkit and Vanquish rootkit.

Blacklight found a hidden process by FU and discovered the other rootkits. Blacklight cleaned the offending files by renaming them and forcing a reboot of the computer. Blacklights interface only allows the user to go forward with no way to go back to check the previous window.

Icesword provides a variety of scans to discover hidden objects. The program seems to have stability issues because during scans the application crashed. The option of *reboot and monitor* causes the computer to restart and watches for attempts to hide processes or registry keys at startup.

RootkitBuster is designed by Trend Micro, and is part of their commercial product line, but has been made available for free. The program is user friendly. The program is a self-contained product, which means that it does not need to be installed. RootkitBuster scans the file system, registry, running processes, drivers and operating system for modifications. The results are presented in a log file and if any found objects are deleted as part of the removal process RootkitBuster forces a reboot. The log file is light on details of the found objects. The application did find the process hidden by all three rootkits. All identified objects were cleaned by deleting them from the computer.

RootkitReavealer will scan the file system and registry entries for hidden objects. The program did show a few false positives in the results, which can be researched and excluded. The results can be exported to a report. RootkitReaveler does not have a mechanism to remove found objects and leaves their removal up to the user. RootkitRevealer did detect AFX Windows Rootkit and the Vanquish rootkit.

Rootkit Unhooker contains six tabs sections: kernel hooks, hidden processes, hidden drivers, hidden files, code hooks and report. The program has a check for virtual machine detector, which use the time response between low level CPU instructions to discover if the operating system is running as a virtual machine. RootkitUnhooker preforms integrity self-test to verify that it has not been modified. The program found all three test rootkits.

The author's results were limited in their reporting. The article only reported whether the rootkits were discovered, and whether the programs removed the rootkits. The article failed to provide more granular information about what was detected.

Romano (2011) discusses rootkits, and makes recommendations for rootkit removal programs. The author recommends the following detection programs: AVG Rootkit Scanner, GMER, Microsoft Standlone System Sweeper Beta, Prevx, RootRepeal, Rootkit Revealer, Sophos Rootkit Scanner and TDSKiller. The author included links to each program's websites for the reader's convenience. The article does not explain how these detection programs were tested or if they were tested. The article does not explain if the author compared these programs against each other. The recommendations appear to rely solely on the authors' experience.

Spanbauer (2008) discusses the detection programs: Blacklight, GMER and Rootkit Buster. The author details which type of rootkit each program searched for, the length of time the scan took, and the ease of use of each program. The article failed to use a sample rootkit for testing. The author states that Blacklight is his favorite, but does not provide any reasons as to why. The article did not compare the detection programs against each other.

Mullins (2011) discusses the following four detection programs: GMER, Icesword, Rootkit Revealer and Tizer Rootkit Razar. The article states that no rootkits were found during testing of these programs. The author recommends using multiple detection programs, because no one program will detect everything. The article details Tizer Rootkit Razor features reasonably well, but only offers a limited review of the other programs.

Rootkits are deployed to monitor or control a computer without the knowledge of the owner of the computer. The literature review presented sources that defined rootkits and their evolution. Additionally, the sources described how rootkits avoid discovery, and also how rootkits modifications to the OS prevent warnings of their presence. Expanding on the information presented, the upcoming Research Methodology section will present results of testing conducted of each of the open source rootkit detection programs.

Research Methodology

The open source rootkit detection programs evaluated in this study were: Blacklight, DarkSpy, GMER, Rootkit Buster, Panda Anti-Rootkit, Rootkit Revealer, Icesword, Sophos Anti-Rootkit and Rootkit Unhooker. The setup of the isolated analysis computer consisted of one Dell Latitude D620 laptop. The software utilized for setting up the testing computer consisted of 32 bit Windows XP Professional – Service Pack 2 and Acronis 2012 True Image Home.

The testing began with the creation of a laptop that was void of malicious code. The process began by booting, or starting, from a known good disc to flash, or reinstall, the laptop's basic input/output system (BIOS). The D620's BIOS was flashed with version A10. Next, the laptop was rebooted with a disc containing the program Darik's Boot and Nuke, which was used to securely wipe the hard drive. These two steps were taken to ensure no malicious code would be present and that the D620 laptop was clean or in a trusted state.

The D620 laptop was then rebooted from a disc containg the Windows XP – Service Pack 2 operating program, which formatted and portioned the laptop's hard drive along with installing the 32 bit Windows XP – Service Pack 2 software. Next, the D620 was imaged with Acronis 2012 True Image Home to a file named Winxp_sp_clean.tib. The image was created so that during the testing process the author could return the laptop to a trusted state if necessary. Moreover, this image was created to ensure that the laptop would be returned to the same clean condition for each program evaluation.

Each of the open source rootkit detection programs were executed to establish a control group. This step guaranteed that no rootkit was present and that any false postitives would be noted. False positives would be if any component of the Windows opeating system were to be listed as a rootkit. These false positives could be excluded from the scan results of the infected laptop.

Rootkits Utilized for Testing

The three rootkits used in this study were: Hacker Defender (HD), FU Rootkit and Vanquish. HD is a user mode or user land rootkit that modifies several Windows APIs. HD hides files, folders,

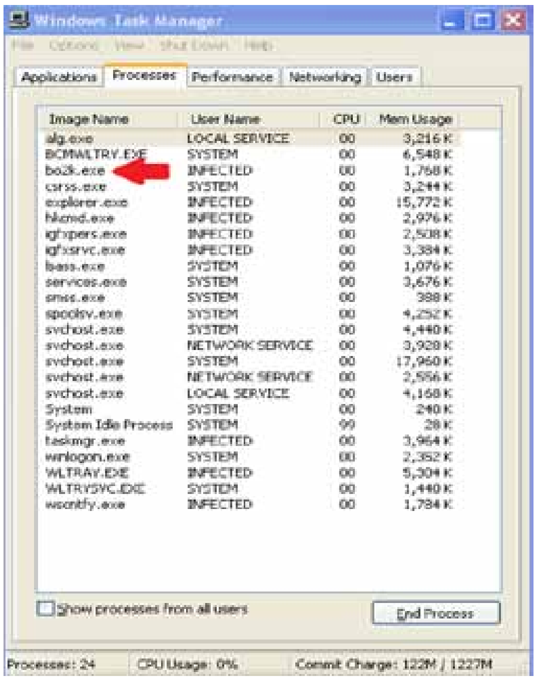

Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users

Image Name	User Name	CPU	Mem Usage
alg.exe	LOCAL SERVICE	00	3,216 K
BCMWLTRY.EXE	SYSTEM	00	6,548 K
bo2k.exe	INFECTED	00	1,768 K
csrss.exe	SYSTEM	00	3,244 K
explorer.exe	INFECTED	00	15,772 K
hkcmd.exe	INFECTED	00	2,976 K
igfxpers.exe	INFECTED	00	2,508 K
igfxsrvc.exe	INFECTED	00	3,384 K
lsass.exe	SYSTEM	00	1,076 K
services.exe	SYSTEM	00	3,676 K
smss.exe	SYSTEM	00	388 K
spoolsv.exe	SYSTEM	00	4,252 K
svchost.exe	SYSTEM	00	4,440 K
svchost.exe	NETWORK SERVICE	00	3,928 K
svchost.exe	SYSTEM	00	17,960 K
svchost.exe	NETWORK SERVICE	00	2,556 K
svchost.exe	LOCAL SERVICE	00	4,168 K
System	SYSTEM	00	240 K
System Idle Process	SYSTEM	99	28 K
taskmgr.exe	INFECTED	00	3,964 K
winlogon.exe	SYSTEM	00	2,352 K
WLTRAY.EXE	INFECTED	00	5,304 K
WLTRYSVC.EXE	SYSTEM	00	1,440 K
wscntfy.exe	INFECTED	00	1,784 K

Processes: 24 CPU Usage: 0% Commit Charge: 122M / 1227M

Figure 1. *A screen capture of the Windows Task Manager showing the Back Orifice process, bo2k.exe, running. The red arrow is pointing to the malicious process*

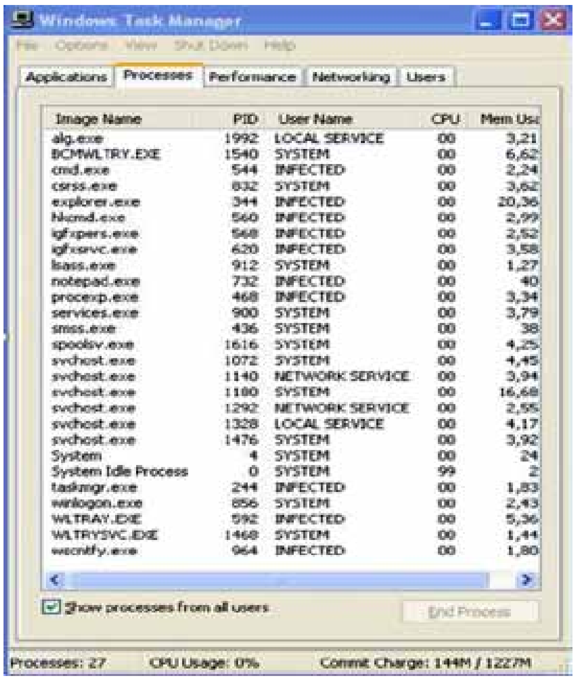

Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users

Image Name	PID	User Name	CPU	Mem Usa
alg.exe	1992	LOCAL SERVICE	00	3,21
BCMWLTRY.EXE	1540	SYSTEM	00	6,62
cmd.exe	544	INFECTED	00	2,24
csrss.exe	832	SYSTEM	00	3,62
explorer.exe	344	INFECTED	00	20,36
hkcmd.exe	560	INFECTED	00	2,99
igfxpers.exe	568	INFECTED	00	2,52
igfxsrvc.exe	620	INFECTED	00	3,58
lsass.exe	912	SYSTEM	00	1,27
notepad.exe	732	INFECTED	00	40
procexp.exe	468	INFECTED	00	3,34
services.exe	900	SYSTEM	00	3,79
smss.exe	436	SYSTEM	00	38
spoolsv.exe	1616	SYSTEM	00	4,25
svchost.exe	1072	SYSTEM	00	4,45
svchost.exe	1140	NETWORK SERVICE	00	3,94
svchost.exe	1180	SYSTEM	00	16,68
svchost.exe	1292	NETWORK SERVICE	00	2,55
svchost.exe	1328	LOCAL SERVICE	00	4,17
svchost.exe	1476	SYSTEM	00	3,92
System	4	SYSTEM	00	24
System Idle Process	0	SYSTEM	99	2
taskmgr.exe	244	INFECTED	00	1,83
winlogon.exe	856	SYSTEM	00	2,43
WLTRAY.EXE	592	INFECTED	00	5,36
WLTRYSVC.EXE	1468	SYSTEM	00	1,44
wscntfy.exe	964	INFECTED	00	1,80

Processes: 27 CPU Usage: 0% Commit Charge: 144M / 1227M

Figure 2. *A screen capture of the Windows Task Manager showing the currently running processes. The Back Orfice (bo2k.exe) process is now hidden from the Task manager*

processes, ports, system drivers and registry keys (Gates, n. d.). FU Rootkit is a kernel mode rootkit, which does not hide itself, but delivers and hides malware applications on the target computer. FU Rootkit is easily discovered because of its lack of stealth (FU, n.d.). Vanquish is a DLL-injection based rootkit. Vanquish hides processes, handles, modules, files, folders, registry keys, services, and logs passwords (Welch and Lee, 2004). Vanquish is a user mode or user land rootkit. User mode or user land is a term used to describe the less secure area of the operating system where programs, DLLs and APIs are executed, which make system calls to interact with the more secure part or core of the operating system.

Rootkit Test Sets

Three compromised Windows XP images were created for the D620 laptop using Acronis 2012 True Image Home. Each compromised image was created by restoring the laptop to a trusted state and then one of the rootkits was installed. An image was then made of the compromised hard drive.

FU Rootkit

FU Rootkit was used to hide the malware program Back Orifice 2000 program, bo2k.exe, on the D620 laptop computer. Back Orifice is an application used for remote administration of Windows based computers (Podrezov, n.d.). The Back Orifice program was initiated by opening the folder BO2K on the desktop, then selecting the bo2k.exe file. Refer to Figure 1 for a screen capture of the Windows Task Manager showing bo2k.exe running.

The next step was to hide the running bo2k.exe process from the Task Manager. To install the FU Rootkit a command prompt window needed to be opened, which was done by clicking on the Start button, then clicking on Run from the menu, and then typing in cmd. This transitioned the computer from operating in Windows mode to Disc Operating System (DOS) mode, and commands were performed by typing text instead of utilizing mouse clicks. Next, the current folder needed to be changed to where the FU Rootkit was stored. The change directory (cd) command was used and the syntax *cd desktop\rootkits\fu_ru\exe* was typed. The command *fu.exe* was typed to initiate the FU Rootkit program. FU Rootkit uses process identifiers to identify the target process to be hidden.

The Process Identifier (PID) for the running Bo2k.exe process was 2020. FU rootkit was utilized to hide the bo2k.exe process, by typing fu –ph 2020 from the command prompt. Refer to Figure 2 for a screen capture of Windows Task Manager showing that the bo2k.exe program was no longer listed as a running process. An image of this drive was created, using Acronis 2012 True Image, with the file name winxp_fu_rootkit_hidden_ process.tib.

Hacker Defender

Hacker defender was used to hide Back Orifice 2000, renamed hxdef_bo2k.exe to correspond to

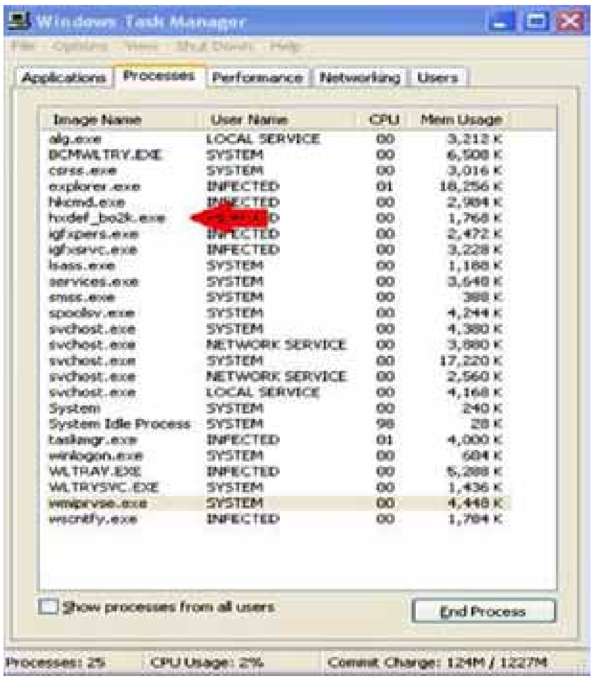


Image Name	User Name	CPU	Mem Usage
alg.exe	LOCAL SERVICE	00	3,212 K
BCMWLTRY.EXE	SYSTEM	00	6,508 K
csrss.exe	SYSTEM	00	3,016 K
explorer.exe	INFECTED	01	18,256 K
hkcmd.exe	INFECTED	00	2,984 K
hxdef_bo2k.exe	[illegible]	00	1,768 K
igfxpers.exe	INFECTED	00	2,472 K
igfxsrvc.exe	INFECTED	00	3,228 K
lsass.exe	SYSTEM	00	1,188 K
services.exe	SYSTEM	00	3,648 K
smss.exe	SYSTEM	00	388 K
spoolsv.exe	SYSTEM	00	4,244 K
svchost.exe	SYSTEM	00	4,380 K
svchost.exe	NETWORK SERVICE	00	3,880 K
svchost.exe	SYSTEM	00	17,220 K
svchost.exe	NETWORK SERVICE	00	2,560 K
svchost.exe	LOCAL SERVICE	00	4,168 K
System	SYSTEM	00	240 K
System Idle Process	SYSTEM	98	28 K
taskmgr.exe	INFECTED	01	4,000 K
winlogon.exe	SYSTEM	00	604 K
WLTRAY.EXE	INFECTED	00	5,288 K
WLTRYSVC.EXE	SYSTEM	00	1,436 K
wmiprvse.exe	SYSTEM	00	4,448 K
wscntfy.exe	INFECTED	00	1,784 K

Figure 3. *A screen capture of the Windows Task Manager showing the renamed Back Orifice process, hxdef_bo2k.exe, running. The red arrow points to the renamed Back Orfice process with hxdef_ left inserted*

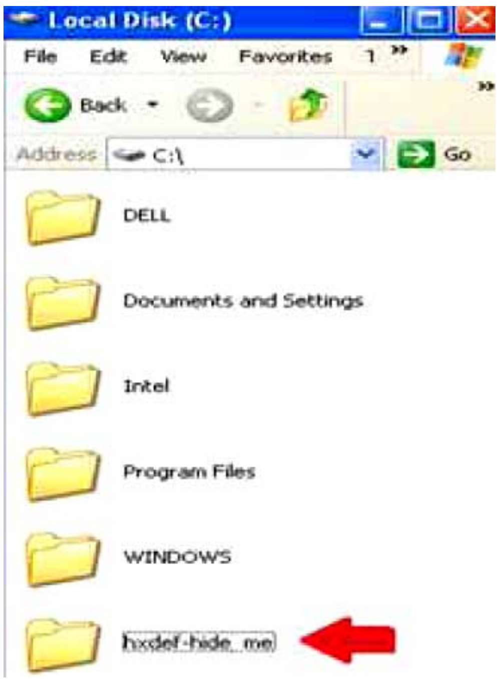


Figure 4. *A display window of the folders on the C drive. The red arrow points to the folder with hxdef, which will be hidden by Hacker Defender*

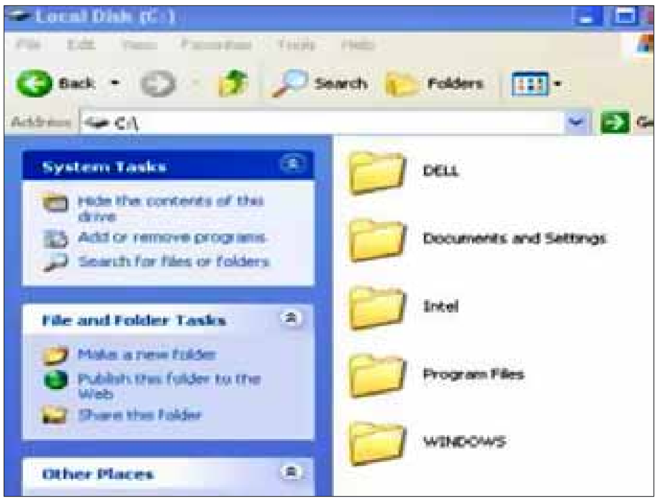
this test set, and a folder named hxdef-hide_me, which contained all of the Hacker Defender files, on the C drive of the compromised computer. This was accomplished by running Hacker Defender in default mode by selecting the file hxdef100.exe. Refer to Figure 3 for a screen capture of the Windows Task Manager with the process hxdef_bo2k running. Refer to Figure 4 for a screen capture of the hxdef-hide_me folder on the C drive.

The next step was to hide the running process hxdef100.exe from the Task Manager, and the hxdef-hide_me folder on the C drive. This was accomplished by running Hacker Defender in default mode by selecting the file hxdef100.exe. Refer to Figure 5 for a screen capture of the Windows Task Manager with hxdef100.exe process no longer listed as a running process. Refer to Figure 6 for screen capture of the folder listing for the C drive, which no longer displayed the folder hxdef-hide_me. Next, an image of the compromised computer was created using Acronis 2012 True Image Home, which was named winxp_hacker _defender.tib.

Vanquish

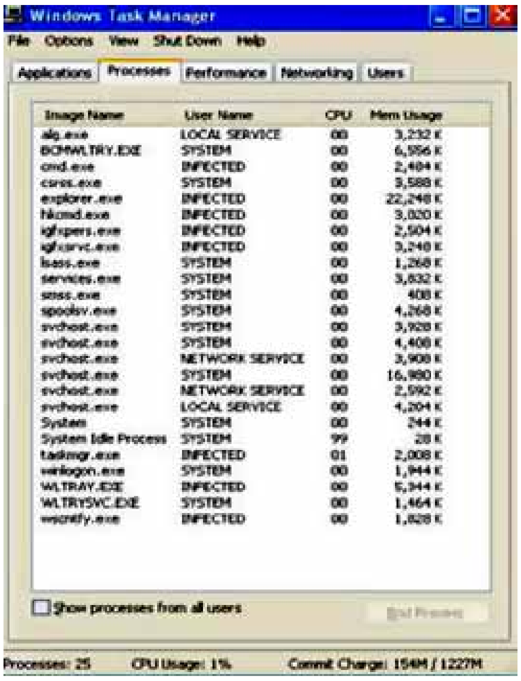
Vanquish was used to hide Back Orifice 2000, renamed vanquish_bo2k.exe to correspond with this test set, and a folder named vanquish, which contained all of the Vanquish files, on the C drive and desktop of the compromised computer. Refer to Figure 7 for a screen capture of the Windows Task Manager with the process vanquish_bo2k.exe running. Refer to Figure 8 for a screen capture of the folder listings, which showed the vanquish folder, before Vanquish was initiated.

The next step was to hide the running process vanquish_bo2k.exe from the Windows Task Manager, and the vanquish folder on the C drive. This was accomplished by running Vanquish in default mode by selecting the file vanquish.cmd and then chosing yes to modifying the registries to add the Vanquish's dll file. Refer to Figure 9 for a screen capture of the Windows Task Manager with the vanquish_bo2k.exe process no longer listed as a running process. Refer to Figure 10 for a screen capture of the folder listing of the C drive, which



```
DELL
Documents and Settings
Intel
Program Files
WINDOWS
```

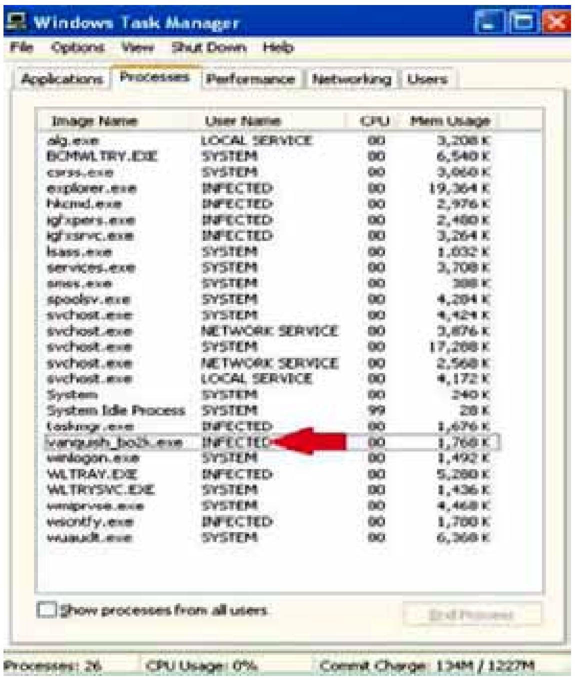
Figure 6. *A screen capture of the folder list of C drive showing Hacker Defender rootkit folder is hidden*


Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users
Show processes from all users
End Process

```
Image Name            User Name          CPU   Mem Usage
alg.exe               LOCAL SERVICE      00      3,232 K
BCMWLTRY.EXE          SYSTEM             00      6,556 K
cmd.exe               INFECTED           00      2,404 K
csrss.exe             SYSTEM             00      3,588 K
explorer.exe          INFECTED           00     22,248 K
hkcmd.exe             INFECTED           00      3,020 K
igfxpers.exe          INFECTED           00      2,504 K
igfxsrvc.exe          INFECTED           00      3,248 K
lsass.exe             SYSTEM             00      1,268 K
services.exe          SYSTEM             00      3,832 K
smss.exe              SYSTEM             00        408 K
spoolsv.exe           SYSTEM             00      4,268 K
svchost.exe           SYSTEM             00      3,928 K
svchost.exe           SYSTEM             00      4,408 K
svchost.exe           NETWORK SERVICE    00      3,908 K
svchost.exe           SYSTEM             00     16,980 K
svchost.exe           NETWORK SERVICE    00      2,592 K
svchost.exe           LOCAL SERVICE      00      4,204 K
System                SYSTEM             00        244 K
System Idle Process   SYSTEM             99         28 K
taskmgr.exe           INFECTED           01      2,008 K
winlogon.exe          SYSTEM             00      1,944 K
WLTRAY.EXE            INFECTED           00      5,344 K
WLTRYSVC.EXE          SYSTEM             00      1,464 K
wscntfy.exe           INFECTED           00      1,828 K

Processes: 25     CPU Usage: 1%     Commit Charge: 154M / 1227M
```

Figure 5. *A screen capture of the Windows Task Manager showing the currently running processes after Hacker Defender was running. The Back Orfice process, hxdef_bo2k. exe, is now hidden.*


Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users
Show processes from all users
End Process

```
Image Name            User Name          CPU   Mem Usage
alg.exe               LOCAL SERVICE      00      3,208 K
BCMWLTRY.EXE          SYSTEM             00      6,540 K
csrss.exe             SYSTEM             00      3,060 K
explorer.exe          INFECTED           00     19,364 K
hkcmd.exe             INFECTED           00      2,976 K
igfxpers.exe          INFECTED           00      2,480 K
igfxsrvc.exe          INFECTED           00      3,264 K
lsass.exe             SYSTEM             00      1,032 K
services.exe          SYSTEM             00      3,708 K
smss.exe              SYSTEM             00        388 K
spoolsv.exe           SYSTEM             00      4,204 K
svchost.exe           SYSTEM             00      4,424 K
svchost.exe           NETWORK SERVICE    00      3,876 K
svchost.exe           SYSTEM             00     17,208 K
svchost.exe           NETWORK SERVICE    00      2,568 K
svchost.exe           LOCAL SERVICE      00      4,172 K
System                SYSTEM             00        240 K
System Idle Process   SYSTEM             99         28 K
taskmgr.exe           INFECTED           00      1,676 K
vanquish_bo2k.exe     INFECTED           00      1,768 K
winlogon.exe          SYSTEM             00      1,492 K
WLTRAY.EXE            INFECTED           00      5,280 K
WLTRYSVC.EXE          SYSTEM             00      1,436 K
wmiprvse.exe          SYSTEM             00      4,468 K
wscntfy.exe           INFECTED           00      1,700 K
wuauclt.exe           SYSTEM             00      6,368 K

Processes: 26     CPU Usage: 0%     Commit Charge: 134M / 1227M
```

Figure 7. *A screen capture of the Windows Task Manager showing the renamed Back Orifice process, vanquish_bo2k. exe, running. The red arrow is pointing to the malicious process*

no longer showed the folder vanquish. Next, an image of the compromised computer was created using Acronis 2012 True Image Home, which was named winxp_vanquish.tib.

Test Results

The rootkit detection program to be tested was installed on the imaged compromised hard drive. A scan was executed to determine if the installed rootkit would be detected, identified, and removed or if the detection program offered suggestions on how to remove the rootkit. This process was repeated each time a rootkit detection programs was tested. Before each test the D620 laptop was restored to a clean state to prevent interference from the previous detection program that was tested. The detection program *Darkspy* could not perform in this testing. The program does not have the capability to operate on computers with multiple processors. The computer utilized for this study, as do most computers today, have multiple processors.

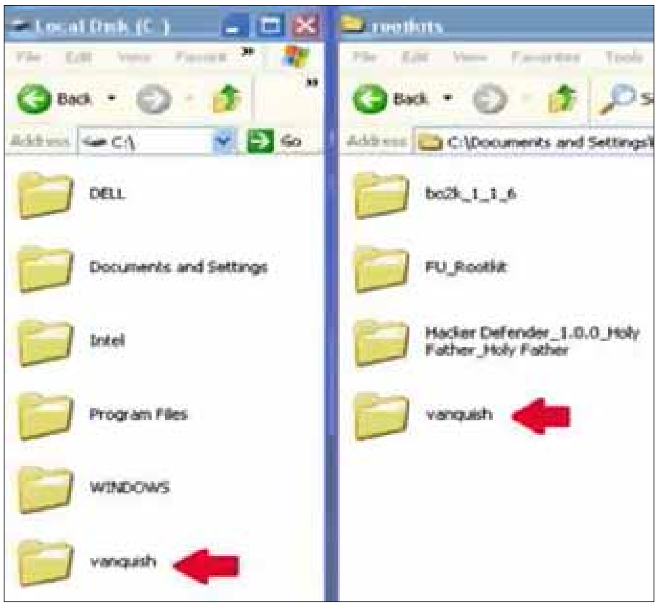
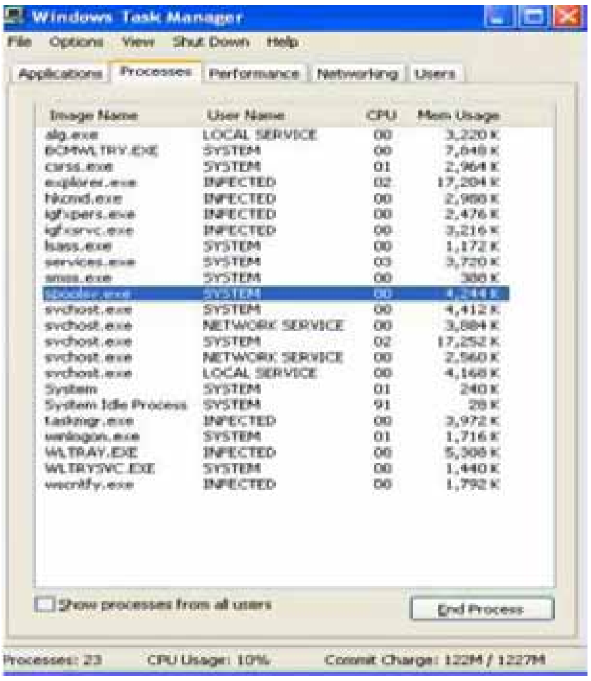


Figure 8. *A screen capture of the folder list of C drive and on the desktop showing folders named Vanquish*


Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users
Show processes from all users
End Process

```
Image Name            User Name         CPU   Mem Usage
alg.exe               LOCAL SERVICE     00     3,220 K
BCMWLTRY.EXE          SYSTEM            00     7,648 K
csrss.exe             SYSTEM            01     2,964 K
explorer.exe          INFECTED          02    17,204 K
hkcmd.exe             INFECTED          00     2,980 K
igfxpers.exe          INFECTED          00     2,476 K
igfxsrvc.exe          INFECTED          00     3,216 K
lsass.exe             SYSTEM            00     1,172 K
services.exe          SYSTEM            03     3,720 K
smss.exe              SYSTEM            00       388 K
spoolsv.exe           SYSTEM            00     4,244 K
svchost.exe           SYSTEM            00     4,412 K
svchost.exe           NETWORK SERVICE   00     3,084 K
svchost.exe           SYSTEM            02    17,252 K
svchost.exe           NETWORK SERVICE   00     2,560 K
svchost.exe           LOCAL SERVICE     00     4,168 K
System                SYSTEM            01       240 K
System Idle Process   SYSTEM            91        28 K
taskmgr.exe           INFECTED          00     3,972 K
winlogon.exe          SYSTEM            01     1,716 K
WLTRAY.EXE            INFECTED          00     5,308 K
WLTRYSVC.EXE          SYSTEM            00     1,440 K
wscntfy.exe           INFECTED          00     1,792 K

Processes: 23     CPU Usage: 10%     Commit Charge: 122M / 1227M
```

Figure 9. *A screen capture of the Windows Task Manager showing the currently running processes after Vanquish was run. The Back Orfice process, vanquish_bo2k.exe, is now hidden*

FU Rootkit

Blacklight, copyright 2005–2007, was executed by selecting the fsbl.exe file. The program's license terms were displayed, and accepting them initiated the Blacklight program. Selecting the *scan* button initiated the scan of the computer. The finished scan report indicated that Blacklight had found no hidden items.

GMER, version 1.0.15.15641, was executed by selecting the gmer.exe file. This caused threferee quick scan to automatically begin. Selecting the *Scan* button initiated the full scan of the computer. The finished scan report indicated that GMER had found no system modifications.

Rootkit Buster, version 5.0.0.1050, was executed by selecting the rootkitbuster_v5_1050.exe file. The program's license terms were displayed, which were accepted. Selecting the *Next* button initiated Rootkit Buster, and then selecting the *Scan* button initiated the full scan of the computer. The finished scan report indicated that Rootkit Buster had found no threats.

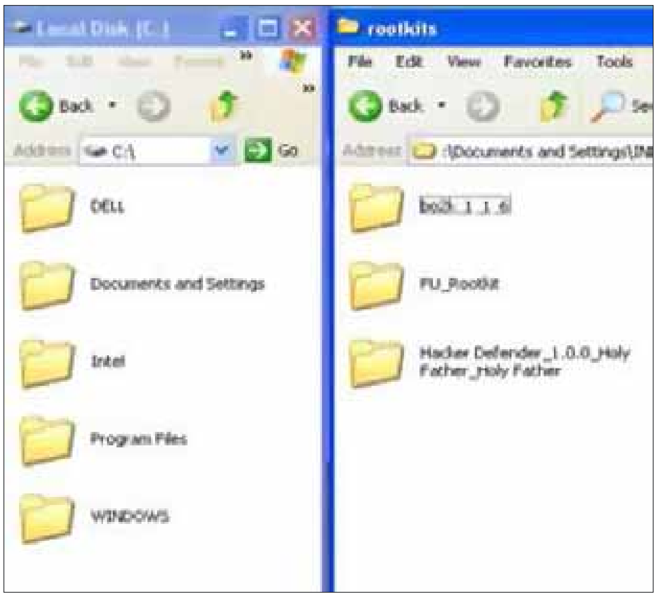


Figure 10. *A screen capture of the folder list of C drive and the computer's desktop showing Vanquish folders are hidden by Vanquish*

The Panda Anti-Rootkit, version 1.08.00, was executed by selecting the pavark.exe file. The program's license terms were displayed, and selecting the *Accept* button opened the application. Selecting the *Start Scan* button began the scan of the computer. The finished scan report indicated that Panda Anti-Rootkit had not detected any rootkits.

Rootkit Revealer, version 1.7, was executed by selecting the rkdetector2.exe file. The program's license terms were displayed; selecting the *Agree* button opened the program.

Selecting *File* in the toolbar and then selecting *Scan* from the drop down menu started the scan of the computer. The finished scan report showed that two discrepancies had been found. Both discrepancies involved registry keys, which were not related to the installed rootkit. Rootkit Revealer had no option to fix or quarantine the discrepancies.

Icesword, version 1.2, was executed by selecting the icesword.exe file. Selecting the *Scan* button loaded information about the computer. Icesword has no automatic scan capabilities, and is setup for more experienced users. To view the current running processes requires selecting the gear icon labeled process under functions on the left side of the programs window. Icesword displayed the results in red. Bo2k.exe was not listed with the other running processes.

Sophos Anti-Rootkit, version 1.5.20, was executed by selecting the sar_15_sfx.exe file. The license terms were displayed; selecting the *Accept* button opened the Sophos Anti-Rootkit program. Next, the installation window opened and selecting the *Install* button began the program installation. After completing the installation, selecting the *Yes* button to initiated the program. Next, select the *Start* Scan button began the scan of the computer. The finished scan report indicated no hidden items found by the scan.

Rootkit Unhooker, version 3.7.300.505, was executed by selecting the rku3.7.300.505.exe file. Next, the installation window opened and selecting the *Install* button began the installation of the program. Then, selecting the *Close* button completed the installation.

Next, selecting the Windows *Start* button, selecting the *All Programs* menu option, then selecting Rootkit Unhooker folder, and finally selecting the program Rootkit Unhooker from the menu, initiated the program. Once the program was running, selecting the *Processes* tab to view the current running processes, Bo2k.exe was not listed.

Hacker Defender

Blacklight, copyright 2005–2007, was executed by selecting the fsbl.exe file. The program's license terms were displayed, and accepting them initiated the Blacklight program. Selecting the *scan* button initiated the scan of the computer. The finished scan report indicated that Blacklight had found the hidden process hxdefbo2k.exe, and eight hidden files that were Hacker Defender files. Blacklight did not find the hidden folder hxdef-hide_me on the C drive. Next, selecting the *Show All Processes* button showed that Hacker Defender (hxdef100.exe) was running. Moreover, Blacklight failed to identify hxdef_bo2k, the Windows remote administration program. Selecting the *Next* button at the bottom right portion of the program window cleans the hidden items that the program located. Blacklight failed to quarantine or stop Hacker Defender from running.

GMER, version 1.0.15.15641, was executed by selecting the gmer.exe file, which caused a quick scan of the computer to automatically begin. The first scan found one hidden process and two hidden services. Next, selecting the *Yes* button GMER conducted a full scan of the computer. The full scan found Hacker Defender's hidden items, including one process, one library, two services, one folder, and seven files. GMER has no automatic removal of the found threats, nor does it have the capabilities for the user to remove the found threats manually.

Rootkit Buster, version 5.0.0.1050, was executed by selecting the rootkitbuster_v5_1050.exe file. The program's license terms were displayed, which were accepted. Selecting the *Next* button initiated Rootkit Buster, and then selecting the *Scan* button initiated the full scan of the computer. The finished scan report indicated that Rootkit Buster had found eighteen Hacker Defender threats, eleven files, five registry entries, and two services. Rootkit Buster did find the hxdef100.exe file, but missed hxdef_bo2k.exe. Next, selecting the *Fix Now* button at the bottom of the program window, Rootkit Buster returned all Hacker Defender's hidden folders to being viewable, deleted the Hacker Defender files and stopped the process.

The Panda Anti-Rootkit, version 1.08.00, was executed by selecting the pavark.exe file. The program's license terms were displayed, and selecting the *Accept* button opened the application. Selecting the *Start Scan* button began the scan of the computer. The finished scan report indicated that The Panda Anti-Rootkit had detected Hacker Defender four times and one unknown rootkit. The unknown rootkit was a Hacker Defender file. The

Advanced Report revealed Hacker Defender's five hidden files, one process and two registry keys. Anti-Rootkit missed the hidden process hxdef_bo2k.exe. Next, selecting the *Remove Rootkits* button, The Panda Anti-Rootkit moved the files to quarantine and stopped Hacker Defender from running following a computer reboot, which occurred as a result of selecting the *Restart Now* button.

Rootkit Revealer, version 1.7, was executed by selecting the rkdetector2.exe file. The program's license terms were displayed; selecting the *Agree* button opened the program. Selecting *File* in the toolbar and then selecting *Scan* from the drop down menu started the scan of the computer. The finished scan report indicated two discrepancies. These two discrepancies were registry keys, neither of which was related to the installed rootkit. Rootkit Revealer had no option to fix the discrepancies or suggestions for addressing these discrepancies.

Icesword, version 1.2, was executed by selecting the icesword.exe file. Selecting the *Scan* button loaded information about the computer. Icesword has no automatic scan capabilities, and is setup for more experienced users. To view the current running processes requires selecting the gear icon labeled process under functions on the left side of the programs window. Icesword displayed issues in red. The file hxdefBo2k.exe was not listed with the other running processes, but under win32 service Hacker Defender was flagged in red. The author selected the *Files* button and navigated to the location of the hidden folders and files. Icesword did find the hidden folder hxdef-hide_me and all hidden files, but none of the hidden objects was flagged in red or in a way that would draw attention.

Sophos Anti-Rootkit, version 1.5.20, was executed by selecting the sar_15_sfx.exe file. The license terms were displayed; selecting the *Accept* button opened the Sophos Anti-Rootkit program. Next, the installation window opened and selecting the *Install* button began the program installation. After completing the installation, selecting the *Yes* button to initiated the program. Next, selecting the *Start* Scan button began the scan of the computer. The finished scan report located twenty-four hidden items. Hacker Defender was found as a hidden process, along with twelve hidden registry keys and eleven hidden files. Under *Issues to be removed*, selecting each check box to the left of the issue and then selecting the *Cleanup checked items* button, removed or quarantined the selected item. The program deleted all of the Hacker Defender files. SophAnti-Rootkit did not find the hidden folder hxdef-hide_me or the hidden process for Remote Windows Administration (hxdefbo2k.exe).

Rootkit Unhooker, version 3.7.300.505, was executed by selecting the rku3.7.300.505.exe file. Next, the installation window opened and selecting the *Install* button began the installation of the program. Then, selecting the *Close* button completed the installation. To execute the application, we left clicked on the Start button, all programs, Rootkit Unhooker and the application Rootkit Unhooker from the menu options. Selecting the *Processes* revealed that Rootkit Unhooker had listed the Hacker Defender as a process hidden from Windows, which should raise suspicions requiring further research of that process. The file hxdefBo2k.exe was not listed. The *files* tab failed to report hidden files on the system. These findings indicate that this application would be useful for more experienced users.

Vanquish

Blacklight, copyright 2005–2007, was executed by selecting the fsbl.exe file. The program's license terms were displayed, and accepting them initiated the Blacklight program. Selecting the *Scan* button initiated the scan of the computer. The finished scan report indicated that Blacklight had found eight hidden items. The eight items found were the Vanquish files and the renamed Back Orifice 2000 file. Blacklight missed both hidden folders named Vanquish on the C drive and on the desktop. Next, selecting the *Show All Processes* button did not does show the Vanquish (vanquish.exe) or the Windows Remote Administration application (vanquish_bo2k.exe) currently running. Selecting the *Next* button, at the right bottom of the program window, initiated cleaning the hidden items. Blacklight failed to quarantine or stop Vanquish or Back Orifice from running.

GMER,version 1.0.15.15641, was executed by selecting the gmer.exe, which caused the quick scan to automatically began. The first scan found no modifications. Next, a full scan of the system was started by selecting the *Scan* button. The full scan of the computer found Vanquish hidden items, including four processes, four libraries, one folder and five files. In addition, GMER has tabs for the user to review processes, modules, services, files, registry, rootkit and malware, autostart and CMD, which can all be scanned individually. GMER does not have the capability automatically remove found threats, which leaves the removal of the discovered rootkit to the user's discretion.

Rootkit Buster, version 5.0.0.1050, was executed by selecting the rootkitbuster_v5_1050.exe file. The program's license terms were displayed, which were accepted. Selecting the *Next* button initiated Rootkit Buster, and then selecting the *Scan* button initiated the full scan of the computer. The finished scan report indicated that Rootkit Buster had found 14 threats. It had found Vanquish's 13 hidden files and 1 registry entry. Rootkit Buster not did find the vanquish.exe or vanquish_bo2k.exe process running. Next, selecting the *Fix Now* button at the bottom of the program window, brought the hidden folders back into view, deleted the Vanquish files and stopped the vanquish.exe process after a reboot, which occurred after selecting the *Restart Computer* button.

The Panda Anti-Rootkit, version 1.08.00, was executed by selecting the pavark.exe file. The program's license terms were displayed, and selecting the *Accept* button opened the application. Selecting the *Start Scan* button began the scan of the computer. The finished scan report indicated that Panda Anti-Rootkit had not detected any rootkits.

Rootkit Revealer, version 1.7, was executed by selecting the rkdetector2.exe file. The program's license terms were displayed; selecting the *Agree* button opened the program. Selecting *File* in the toolbar and then selecting *Scan* from the drop down menu started the scan of the computer. The finished scan report indicated that two discrepancies had been found. These discrepancies were registry keys. Both discrepancies were related to Windows XP opeating system and not to Vanquish. The discrepancies were listed because the registry key names contain an asterisk. Rootkit Revealer had no option to fix the discrepancies or suggestions for how to fix the discrepancies.

Icesword, version 1.2, was executed by selecting the icesword.exe file. Selecting the *Scan* button loaded information about the computer. Icesword has no automatic scan capabilities, and is setup for more experienced users. To view the current running processes requires selecting the gear icon labeled process under functions on the left side of the programs window. Icesword displayed issues in red. Icesword did not flag any of Vanquish's files in red, or any of the hidden objects. Nothing was flagged in red to draw the user's attention.

Sophos Anti-Rootkit, version 1.5.20, was executed by selecting the sar_15_sfx.exe file. The license terms were displayed; selecting the *Accept* button opened the Sophos Anti-Rootkit program. Next, the installation window opened and selecting the *Install* button began the program installation. After completing the installation, selecting the *Yes* button to initiated the program. Next, selecting the *Start* Scan button began the scan of the computer. The finished scan report indicated that no hidden items had been found. Anti-Rootkit did not find the hidden Vanquish folder or the hidden process for Remote Windows Administration (vanquish_bo2k.exe). Sophos Anti-Rootkit does not identify this type of rootkit.

Rootkit Unhooker, version 3.7.300.505, was executed by selecting the rku3.7.300.505.exe file. Next, the installation window opened and selecting the *Install* button began the installation of the program. Then, selecting the *Close* button completed the installation. To execute the application the *Start* button was selected, all programs, Rootkit Unhooker and the application Rootkit Unhooker from the menu options. Selecting the *Processes* tab confirmed that Rootkit Unhooker had not found any hidden processes. The Remote Windows Administration application Vanquish_bo2k.exe also was not listed. The *Files* tab failed to report hidden files.

This concludes the Methodology portion of this project. In this section open source rootkit detection programs were tested. The evaluations consisted of a search and the removal of the rootkits, which was verified by execution of a scan by the same detection program. In the following Discussion of the Findings section, the author will discuss in detail the results of the procedures used to test the open source rootkit detection programs.

Discussion of the Findings

This purpose of this study was to evaluate the effectiveness of open source rootkit detection programs. The programs evaluated were Blacklight, DarkSpy, GMER, Rootkit Buster, Panda Anti-Rootkit, Rootkit Revealer, Icesword, Sophos Anti-Rootkit, and Rootkit Unhooker. Additionally, the detection programs were analyzed to determine if they could remove any discovered rootkits. The following are the detailed test results.

Rootkit Detection Program Testing Results

All of detection programs were run on a clean laptop computer the author utilized a disc wiping program before beginning the testing, to ensure no malware was present. The computer was running 32 bit Windows XP – Service Pack 2 OS, installed after using the disc wiping program. All of the rootkit detection programs failed to locate any of the files related to the FU Rootkit. The following are the testing results for each of the rootkit detection programs.

Blacklight

Hacker Defender rootkit

The scan of the 70 gigabyte hard drive, with 3 gigabytes used by the installed programs, took 40 seconds to complete. The program found eight hidden items related to the Hacker Defender rootkit. Seven of the files were Hacker Defender files and one was the Back Orifice program. The program is able to show all active processes. The hidden Hacker Defender processes were displayed, but these same processes were not listed in Windows Task Manager. Blacklight claimed to have renamed the hidden items, but after a reboot of the computer another scan was performed, and the eight hidden items were found again. Blacklight was unsuccessful at removing the Hacker Defender files.

Vanquish rootkit

The scan of the computer's hard drive also took 40 seconds to complete. The application found nine hidden objects, eight were Vanquish rootkit files and one was the Back Orifice file. Blacklight's show all processes failed to find the process hidden by Vanquish. Selecting the *Next* button caused the nine hidden objects to be displayed. Blacklight claimed to have renamed the hidden items, but after a reboot of the computer another scan was performed, and the nine hidden items were found again. Blacklight was unsuccessful at removing Vanquish files.

DarkSpy

DarkSpy is longer useable on computers with multiple processors. The application immediately displays an error message window with the message about not supporting multiple central processing units environment. All current computers have multiple CPUs, which makes Darkspy no longer a viable program.

GMER

Hacker Defender rootkit

The scan of the 70 gigabyte hard drive, with three 3 gigabytes used by the installed programs, took 4 minutes and 32 seconds to complete. GMER's scan of the computer found six hidden items that all begin with hxdef, which were Hacker Defender objects. GMER highlighted the hidden objects in red to alert the user. GMER can export the results to file for later analysis by selecting the *Save* button. GMER has no removal option for found objects.

Vanquish rootkit

The scan of the computer's hard drive also took 4 minutes and 32 seconds to complete. GMER's scan of the computer found 20 hidden objects related to the Vanquish rootkit. GMER caused a window to display, warning of rootkit activity. There was a folder GMER identifed as a file which was named vanquish. GMER has no option for removing the found hidden objects.

Rootkit Buster

Hacker Defender rootkit

The scan of the 70 gigabyte hard drive, with 3 gigabytes used by the installed programs, took 43 seconds to complete. Rootkit Buster's scan of the computer found 18 threats related to the Hacker Defender rootkit. Rootkit Buster has three post scan options: fix now, full results and scan again. The *fix now* option runs the rootkit removal process, the *full results* option produces a report that can be saved for future analysis, and the *scan again* option starts the scan process. Selecting the box at the top of the column of boxes in front of the threats inserts a check in the box for all of the threats found. Next, selecting the fix now button initiates removing the threats. Next, selecting *reboot computer* button rebooted the computer, followed by another scan, and no threats were found. Rootkit Buster successfully removed the Hacker Defender files.

Vanquish rootkit

The scan of the computer also took 43 seconds to complete. The scan found fourteen threats related to the Vanquish rootkit. Rootkit Buster has three post scan options: fix now, full results and scan again. The *fix now* option runs the rootkit removal process, the *full results* optioin produces a report that can be saved for future analysis, and the *scan again* option starts the scan process. Selecting the box at the top of the column of boxes in front of the threats inserts a check in the box for all of the threats found. Next, selecting the fix now button initiates removing the threats. Next, selecting *reboot computer* button rebooted the computer, followed by another scan, and no threats were found. Rootkit Buster successfully removed the Hacker Defender files.

Panda Anti-Rootkit

Hacker Defender rootkit

The scan of the 70 gigabyte hard drive, with three 3 gigabytes used by the installed programs, took 30 seconds to complete. Panda Anti-Rootkit found five files related to the Hacker Defender rootkit, which were all of the files associated with the rootkit. The program displayed the five discovered files. The program has three options: remove rootkits, do not remove rootkits, and advanced report. The remove rootkits option runs the rootkit removal process, the do not remove rootkits option only causes the program to shut down, and the advanced report option produces a report that can be saved for future analysis.

To remove the found items, the *remove rootkits* option was selected. Panda Anti-Rootkit removed the five detected files and two registry keys, which only took a few seconds. Next, the program caused the laptop to automatically rebooted, Panda Anti-Rootkit started up again automatically, and then displayed the five files and two registry keys that were removed. Another scan of the hard drive found no threats.

Panda Anti-Rootkit was successful at removing files related to the Hacker Defender rootkit. Panda Anti-Rootkit also requested that the discovered files be sent to the program developer for product improvement.

Vanquish rootkit
Panda Anit-Rootkit failed to locate any files related to the Vanquish rootkit.

Rootkit Revealer

Hacker Defender rootkit
The scan of the 70 gigabyte hard drive, with three 3 gigabytes used by the installed programs, took 45 seconds to complete. Rootkit Revealer found 26 discrepancies related to the Hacker Defender rootkit. Rootkit Revealer has no options to remove discrepancies. The program has a save option in the file drop down menu, which can be used to write the results to a text file for further analysis.

Vanquish rootkit
Rootkit Revealer failed to locate any discrepancies related to the Vanquish rootkit.

Icesword

Hacker Defender rootkit
IceSword does not have an automatic scan capability like the other applications. To find the registry keys the user would have to manually search the registries to locate any modified keys. The processes are scanned by selecting the process button. The scan of the processes only took a few seconds. Icesword located the Hxdef100.exe files, which was highlighted in red to alert the user to the threat. Next, the win32 service icon was selected and the Hacker Defender rootkit was highlighted in red, and as a running process. IceSword has no removal option for found objects.

Vanquish rootkit
Icesword failed to locate any of the files or processes related to the Vanquish rootkit.

Sophos Anti-Rootkit

Hacker Defender rootkit
The scan of the 70 gigabyte hard drive, with three 3 gigabytes used by the installed programs, took 3 minutes and 3 seconds to complete. Sophos Anti-Rootkit found 24 hidden items related to the Hacker Defender rootkit. Sophos Anti-Rootkit has no option to remove hidden items.

Vanquish rootkit
Sophos Anti-Rootkit failed to locate any items related to the Vanquish rootkit.

Rootkit Unhooker

Hacker Defender rootkit
Rootkit Unhooker does not have an automatic scan capability like the other applications. Rootkit Unhooker scans for rootkits by selecting the process tab. The program discovered one process, related to the Hacker Defender rootkit, which was hidden from Windows API. The hidden process was hxdef100.exe. None of the other tabs had any hidden objects listed. This application requires a lot of skill to use it effectively. Rootkit Unhooker has no automatic removal option for found objects. Rootkit Unhooker has a drop down menu where the user can unhook, kill process or wipe a file.

Vanquish rootkit
Rootkit Unhooker failed to identify any hidden objects related to the Vanquish rootkit.

Rootkit Detection Program Analysis

After completing the evaluation of the detection programs, there were three detection programs that performed better than the rest. Rootkit Buster program performed the best of the detection programs tested for this study. The two other rootkit detection programs that performed well were Blacklight and GMER. These programs found two of the three rootkits used during this evaluation. None of the detection programs were able to find the FU rootkit. In a previous study, both Rootkit Buster and Blacklight were able to locate the FU rootkit (Yegulalp, 2007). Refer to table 1 for a summary of the testing results.

Limitations with this Research Study

The three rootkits used for this research were not the most recent versions. The latest versions were unavailable because rootkit developers are unwilling to provide their rootkits for analysis. Rootkits developers strive to remain anonymous to produce rootkits that are undetectable. Rootkits continue to evolve to avoid detection. Newer versions have increased their stealth on a computer system. The FU rootkit went undiscovered by all programs used in this study. There is a possibility, had the detection programs been more recent versions, the FU Rootkit would have been identified. Furthermore, the detection programs may have performed better had they been allowed to access the Internet and receive updates.

All of the detection programs were free for personal use. The free versions may have had some of the program's features disabled until they are purchased. These programs appeared to depend on Windows APIs, which were subverted by two of the rootkits, used to test these detection applications. Six of the detection applications were executed without having to install them, while two of the applications performed installation on to the laptop. All detection applications were either run or installed after the rootkit was already installed to simulate a response to an anomaly.

Recommendations

To date, there is no single program that can detect all known rootkits. Multiple programs are needed to detect the various rootkits, and rootkits developers continue to change how rootkits perform in order to avoid being discovered by detection programs. Rootkits are transitioning from hiding within the OS and memory of a computer to hardware components of computers. Rootkits can hide within a computer's BIOS, CPU, and add-on components that utilize PCI cards.

Rootkit modifications to Windows OS are designed to hide and execute programs. Rootkits provide the means for attackers to use a computer that they can gain access to for their own purposes. An attacker's use of a compromised computer is almost limitless. The attacker's could use the compromised computer to hide their identity and location while commiting cyber crimes. Compromised computers increase the difficulty of tracing the identity and illegal computer activity back to the cyber attacker.

The literature reviewed discussed rootkit detection programs, and the results of the testing of those detection programs. The sources highlighted some of the same detection programs that were used in this study, however the FU rootkit files that were discovered in those studies, were undetected in this study. The difference in the findings could be attributed to a newer version of the FU rootkit that was used in this study. Furthermore, this provides additional evidence that rootkits do continue to evolve to better avoid detection.

Based on the author's research, Rootkit Buster, by Trend Mirco, performed the best, of the open source rootkit detection programs tested. Rootkit Buster found two of the three test rootkits and was able to completely remove the discovered rootkits. Rootkit Buster also found the hidden Windows remote administration program and revealed it.

This findings of this study verified that multiple rootkit detection programs are needed to be completely effective. Rootkit Buster would need to be utilized in conjunction with another program in order to detect the FU rootkit. Rootkit Buster employs cross-view detection to detect rootkits. To be most effective at detecting rootkits, a second program employing behavioral or heuristic detection, which none of the other programs tested utilize, is needed. Open source heuristic rootkit detections programs were not readily available at the time of this study. This combination of rootkit detection methods would significantly increase the probability of rootkit detection.

Recommendations for Future Research

Future research could be conducted on rootkit detection programs. A greater number of open source rootkit detection programs could be tested in a comparative format. Specifically, se-

Table 1. *Summary of the Rootkit Detection Programs Scan Results*

	Blacklight	GMER	Rootkit Buster	Panda Anti-Rootkit	Rootkit Revealer	IceSword	Sophos Anti-Rootkit	Rootkit Unhooker
FU	No Hidden Items	No Hidden Items	No Hidden Items	No Hidden Items	No Hidden Items	No Hidden Items	No Hidden Items	No Hidden Items
Hacker Defender	8 Hidden Items	4 Hidden Items	18 Hidden Items	5 Hidden Items	27 Hidden Items	1 Hidden Item	24 Hidden Items	1 Hidden Items
Vanquish	9 Hidden Items	20 Hidden Items	14 Hidden Items	No Hidden Items	No Hidden Items	No Hidden Items	No Hidden Items	No Hidden Items

lecting rootkit detection programs from each of the five different detection methods, signature based, heuristic or behavior based, cross view based, integrity based, and hardware based, to evaluate and analyze each of the programs' effectiveness. This study's methodologies could be utilitzed to evaluate and analyze the detection programs.

Future research could be conducted on detecting rootkits that hide in hardware. Such a study could focus on rootkits that hide in the computer's BIOS, video card, or network card. This research would need to focus on one manufacturer's hardware and on devising a method for discovering the presence of a rootkit. In this kind of study, possible detection methods may include using other components in the computer to verify if something is being manipulated, which would indicate the presence of a rootkit.

Future research could be conducted into identifying the connection between botnets and rootkits. This kind of study could establish how rootkits are used to hide and control the computer infected with the botnet program. Possible detection methods might include network packet captures analysis, or the discovery of other indication of the infection.

Furthermore, additional research could include using antivirus, host intruder protection systems, and host intruder detection systems used for rootkit detection. The research could evaluate a few of each application to determine if rootkit activity could be detection. Also if the program does detect a threat how the threat was handled could also be analyzed.

Lastly, future research could use recent rootkits to evaluate the performance of the open source rootkit detection programs. This testing could be repeated with newer rootkits or even different rootkits to see if the new threats could be detected. The testing could also include if the rootkit detection programs could remove the discovered threats.

Conclusion

The purpose of this research study was to evaluate and analyze open source, free, rootkit detection programs. The detection programs tested in this study were Blacklight, DarkSpy, GMER, Rootkit Buster, Panda Anti-Rootkit, Rootkit Revealer, Icesword, Sophos Anti-Rootkit, and Rootkit Unhooker. This study intended to answer the following questions: How effective are open source rootkit detection software programs at detecting rookits? If these programs can detect a rootkit, are they able to remove the rootkit?

Rootkits are used to hide Windows processes, libraries, files, folders and services from the user, from the OS, and from anti-virus programs. These hidden Windows components serve as back doors for attackers to gain access to the computer, to monitor user activities or to attack another computer. Rootkits allow attackers to hide their activities from the user. Attackers protect themselves by executing hacking programs from the compromised computer, allowing them to commit various cyber crimes once they gain access to that computer.

There were nine open source rootkit detection programs evaluated for this study. Each of these programs was tested against three rootkits. The outcome of the testing process verified that open source rootkit detection programs can discover rootkits. Open source rootkit detection programs are viable alternatives to purchased commercial applications. However, the findings of the study also supported those author's opinions, within existing literature regarding rootkit detection, that there is no single rootkit detection program that will identify all rootkits.

The findings showed that rootkits may not always be detectable by detection programs. Rootkit developers design rootkits to be undetectable, and are continually evolving in their methods to hide rootkits. The version of the FU Rootkit utilized in this study was undetectable by all of the detection programs, which is further proof of the need for continued detection development. As rootkits methodologies change to avoid detection, so can to the methodologies for programs that detect them.

KELLY R. KOHL

Fusic

Fusion of Society, IT and Culture

Founded in 2003 in Fukuoka, Japan. Fusic provides several IT related services all around Japan. Among the services we provide are: web development, contract-based software development (such CMS and CRM), etc. We also developed our own web-based presentation service "Zenpre", and e-Commerce platform "Ureru-net-kokoku-tsukuru", and serve consumer through ASP. Currently, we also play a leading role in the mobile applications development in platforms such iPhone and Android.

浜崎陽一郎

Yoichiro Hamasaki
Vice-President
Co-Founder

納富貞嘉

Sadayoshi Noutomi
President
Founder

Fusic Co.,Ltd http://fusic.co.jp/ info@fusic.co.jp

Fukuoka Head Office
Shin-nihon build.9F, 2-4-22 Daimyo Chuo-ku, Fukuoka-shi, 810-0041, JAPAN
+81-92-737-2616 +81-92-737-2617

Fukuoka Laboratory
East Fukuoka General Office 4F, 1-17-1 Hakata Station East,Hakata-ku, Fukuoka-shi, 812-0013, JAPAN

Tokyo Branch
Okura build. 3F, 1-4-10, Shibadaimon, Minato-ku, Tokyo, 105-0012, JAPAN
+81-3-6450-1633 +81-3-6450-1634

[IS IT IN YOUR DNA?]

[GEEKED AT BIRTH]

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 Games and Simulation
Strategic Technology Development
Technology Forensics
Technology Product Design
Technology Studies
Virtual Modeling and Design
Web and Social Media Technologies

University of Advancing Technology
UAT
Learn. Experience. Innovate.

You can talk the talk.
Can you walk the walk?

www.uat.edu > 877.UAT.GEEK

PLEASE SEE **WWW.UAT.EDU/FASTFACTS** FOR THE LATEST INFORMATION ABOUT DEGREE PROGRAM PERFORMANCE, PLACEMENT AND COSTS.