

# HAKING

# Mobile Security

Vol.2 No.2  
Issue 02/2012(3) ISSN: 1733-7186

## DATA HANDLING ON IOS DEVICES

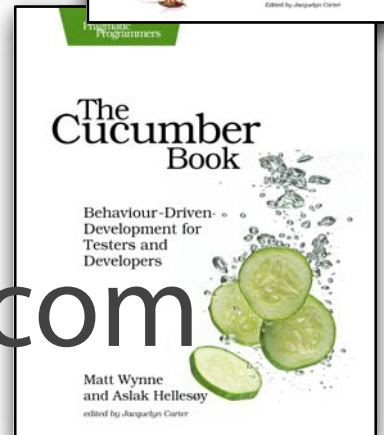
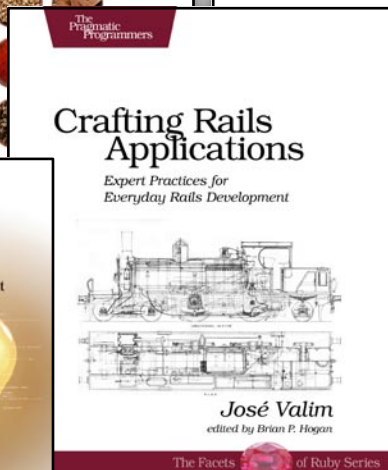
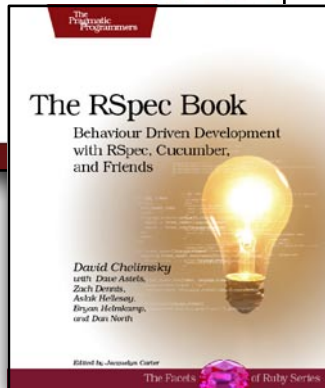
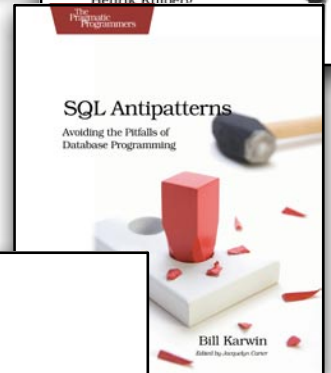
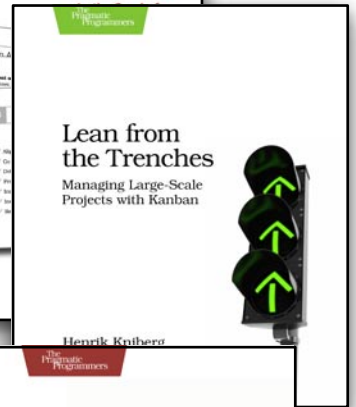
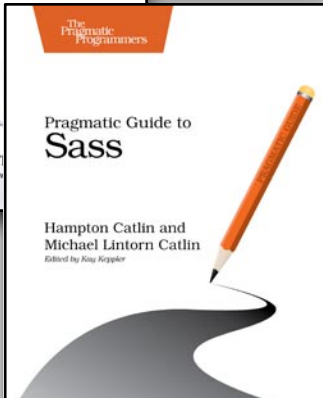
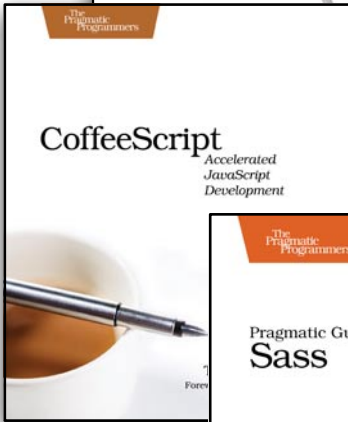
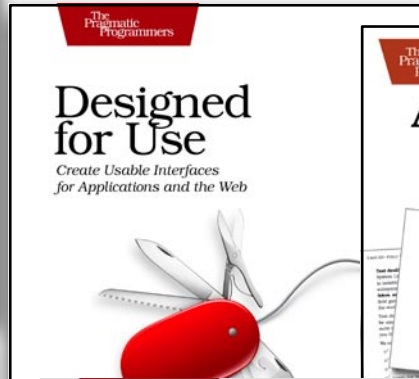
**A BITS' LIFE**

**WHEN DEVELOPERS API SIMPLIFY  
USER-MODE ROOTKITS DEVELOPING**

PLUS

**ANDROID FORENSICS  
SOCIAL ENGINEERING**

Keeping You At the  
Top of Your Game.  
Paper • eBooks • Dropbox



Pragmatic  
Bookshelf

www.pragprog.com

# CODENAME:

# SAMURAI SKILLS COURSE



## << Penetration Test Training Samurai Skills >>

- 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 )

## Mobile Security

team

**Managing:** Angelika Gucwa  
[angelika.gucwa@hakin9.org](mailto:angelika.gucwa@hakin9.org)

**Editor in Chief:** Grzegorz Tabaka  
[grzegorz.tabaka@hakin9.org](mailto:grzegorz.tabaka@hakin9.org)

**Senior Consultant/Publisher:** Paweł Marciniak

**Marketing Director:** Angelika Gucwa  
[angelika.gucwa@hakin9.org](mailto:angelika.gucwa@hakin9.org)

**Production Director:** Andrzej Kuca  
[andrzej.kuca@hakin9.org](mailto:andrzej.kuca@hakin9.org)

**DTP:** Ireneusz Pogroszewski  
**Art Director:** Ireneusz Pogroszewski  
[ireneusz.pogroszewski@hakin9.org](mailto:ireneusz.pogroszewski@hakin9.org)


**Proofreaders:** Michael Munt, Nick Baronian, Dan Dieterle, Bob Folden, Kelly Kohl, Aby Rao, Jeffrey Smith

**Top Betatesters:** Keith Applegarth, Nick Baronian, Venay Bhana, Ayo Tayo – Bologun, Jose Bosio, Rick Chandler, Amit Chugh, Sieng Chye, Dan Dieterle, Jürgen Eckel, Patrik Gange, Alexander Groisman, Eric Hansen, Shane Hartman, José Herrera, Tyler Hudak, Marek Janáč, Michal Jáchim, Kelly Kohl, Matteo Massaro, Michael Munt, David Prokop, Jonathan Ringle, Rissone Ruggero, Antonio Saporita, Tim Singletary, Daniel Sligar, Jeffrey Smith, Tim Thorniley, Tom Updegrave, Robert Wood, Bert White, David von Vistaux, Bert White, Rebecca Wynn

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

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

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

All rights to trade marks presented in the magazine are reserved by the companies which own them.  
To create graphs and diagrams we used [smartdraw.com](http://smartdraw.com) program by  SmartDraw

Mathematical formulas created by Design Science MathType™

### DISCLAIMER!

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

### Dear Readers,

We are giving in your hands the third issue devoted to Mobile Security. We hope you've enjoyed the previous ones.

In this issue you will find lots of interesting information about security. After reading this issue, your phones will remain safe and hackers will no longer threaten them. The authors, who have contributed to this issue, should be proud of themselves. We are sure you will read this issue from cover to cover.

In the third issue of Hakin9 Mobile Security you can read an article written by Dominic Chell. In his article Dominic helps in understanding of the common pitfalls for developers when implementing transport and data storage mechanisms in iOS apps. From the article, the readers will learn not only how to evaluate iOS apps for common issues but also how to securely resolve them.

The next articles written by Yury Chemerkin concerns cybercrime. Our smartphones have been equipped with operating systems that compare in complexity with those on desktop computers. The past several years of mobile malware are no longer only in the theory. There has been malware, loss, theft, data communication interception, exploitation, etc. If you want be safe – it is essential that you read his series of articles.

In the last issue we wrote about a very “hot” topic – Social Engineering. In this issue Ruben Thijssen written another very interesting article. It is never enough of that kind of knowledge! Did you know that Android had more than 36% share of the market by the end of the first quarter of 2011? In Manish Chasta' article you can read about Android Forensics. Android is becoming commonly used platform. As Android is capturing market, it is becoming favourite target platform for hackers. You should know it!

Tomasz Cedro and Marcin Armand Kuzia describe in very interesting way a binary system in a Bits' Life. Machines make our lives easier and do the majority of boring and time consuming tasks for us. However, they are not yet as intelligent and autonomous as their creators, so their applications are still limited. On the other hand, the digital world gives us an opportunity to develop our skills and create better world. This article is an essential reading.

Enjoy the reading!

Angelika Gucwa  
and Hakin9 Team

## MOBILE SECURITY

### 06 Data Handling on iOS Devices

by *Dominic Chell*

With over half a million apps in the App Store, Apple's trademark slogan "There's an app for that" is bordering on reality. We use these apps for online banking, social networking and e-mail without really knowing if they're communicating and storing our personal data securely. With Apple controlling over 52% of the mobile market [1], iOS apps are becoming more closely scrutinised in a world where the security of our personal data is paramount. In the last year, MDSec's consultants have performed an increasing number of security assessments of iOS applications and their supporting architecture where data security is paramount, specifically the retail/business banking sector.

### 16 When developers API simplify user-mode rootkits developing – part 1

by *Yury Chemerkin*

This is a series of articles about shell extensions that enhance high-level features of any operation system. However, such possibilities not only enrich platform but simplify developing trojans, exploits that leads to the new security holes. Mostly this kind of extensions are known as user-mode rootkits.

### 22 Android Forensics

by *Manish Chasta*

Smartphones are changing the IT and Communication landscape vastly. A Smartphone can do almost every good thing a computer can do. Today most of the corporate

employee access and manage their official e-mails through the e-mail client installed on their Smartphone. Right from booking movie tickets to making fund transfers, all e-commerce and online banking transactions can be done using a Smartphone. With high speed of 3G, Smartphones are getting more popular specially among working professionals and students.

### 30 Social Engineering

by *Ruben Thijssen*

Ever walked into a shop planning on buying a product, but after talking to the salesman leaving with a more expensive product that has more options than you needed? Ever listened to a politician and thought during his or her speech: 'that's a pretty good idea', but the next day you realize what all the downsides are to his or her statement? And ever kept buying that one person drinks because you thought it would get you somewhere, but suddenly the person in question left and you didn't even get a phone number?

## ABITS' LIFE

### 34 A Bits' Life

by *Tomasz Bolesław Cedro & Marcin Armand Kuzia*

Have you ever wondered what makes all these devices around you alive? I might have to give you a bit of bad news – this is not a black magic of any kind, neither any supernatural powers, not even the Jedi Force; it is just a simple set of interesting ideas, well described with a language called science and technology.

# Data Handling on iOS Devices

With over half a million apps in the App Store, Apple's trademark slogan "There's an app for that" is bordering on reality. We use these apps for online banking, social networking and e-mail without really knowing if they're communicating and storing our personal data securely.

With Apple controlling over 52% of the mobile market (Mobile/Tablet Top Operating System Share Trend – NetMarketShare <http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=9&qpcustomb=1>), iOS apps are becoming more closely scrutinised in a world where the security of our personal data is paramount. In the last year, MDsec's consultants have performed an increasing number of security assessments of iOS applications and their supporting architecture where data security is paramount, specifically the retail/business banking sector.

## Introduction

Transport and data storage security are two of the greatest risks facing mobile applications; it is imperative that these are implemented securely to prevent data theft when using untrusted networks or in the event that a device is lost or stolen.

This article will provide the reader with an understanding of the common pitfalls for developers when implementing transport and data storage mechanisms in iOS apps. The reader will learn not only how to evaluate iOS apps for common issues but also how to securely resolve them. For more information on iOS data handling issues and other vulnerabilities affecting iOS apps, the reader is encouraged to subscribe to the MDsec research page and blog (MDsec Research and Blog <http://www.mdsec.co.uk/research>, <http://blog.mdsec.co.uk>).

## Data Storage

The protection of data stored on a mobile device is perhaps one of the most important issues that an

application developer has to deal with. It is imperative that developers protect sensitive data that is stored client-side in a secure manner. Developers wishing to encrypt sensitive content on the device should employ the Data Protection API. Unfortunately, it is common practice to find even apps from large multinationals storing their sensitive data in clear text. A good example of this was highlighted in 2010 where vulnerabilities in the Citigroup online banking application caused it to be pulled from the AppStore, as reported by The Register (Citigroup iPhone Data Storage Issues [http://www.theregister.co.uk/2010/07/27/citi\\_iphone\\_app\\_weakness/](http://www.theregister.co.uk/2010/07/27/citi_iphone_app_weakness/)):

*In a letter, the US banking giant said the Citi Mobile app saved user information in a hidden file that could be used by attackers to gain unauthorized access to online accounts. Personal information stored in the file could include account numbers, bill payments and security access codes...*

While this article will only focus on app data storage and how applications can use the Data Protection API, an in depth presentation on iPhone encryption has been performed by Jean-Baptiste Bedrune and Jean Sigwal of SogetiESEC (iPhone data protection in depth <http://esec-lab.sogeti.com/dotclear/public/publications/11-hitbamsterdam-iphonedataprotection.pdf>).

Client-side data can be stored in a number of forms, including but not limited to:

- custom created files
- databases
- system logs
- cookie stores

Table 1. Application directory structure

Directory	Description
Application.app	Stores the static content of the application and compiled app. This content is signed and checked at runtime.
Documents	A persistent store for application data; this data will be synched and backed up to iTunes.
Library	This folder contains support data used by the app such as configurations, preferences, cache data and cookies.
tmp	This folder is used to store temporary files.

Listing 1. Kik Attachments

```
mbp:Documents $ file fileAttachments/057a8fc9-0daf-4750-b356-5b28755f4ec4
fileAttachments/057a8fc9-0daf-4750-b356-5b28755f4ec4: JPEG image data, JFIF standard 1.01mbp:Documents $
```

- plists
- data caches

All of these may contain sensitive data that should be protected if the handset were lost or stolen. This data will generally be stored within the application's sandboxed container.

Applications are stored on the file-system as the *mobile* user under the `/var/mobile/Applications` directory where a unique GUID is used as a sub directory container to store the app data. The application directory structure is as follows: Table 1.

An attacker looking to extract application data is likely to find it within this directory structure in one form or another.

Let's take a look at a real world app, taken from the AppStore. Kik Messenger is a social networking application with a 4+ star rating from 6405 ratings on the

AppStore and well over 1 million users. The application allows users to send free instant messages via the devices data connection. In order to do this, the user must sign-up for a free Kik account.

Within the Kik application directory is the preferences plist, `Library/Preferences/com.kik.chat.plist` that is use by the app to store configuration information, including the users username, password and e-mail address, as shown below (obscured for reader): Figure 1.

The plist detailed above is not protected by the Data Protection API and therefore resides unencrypted on the file-system while the device is enabled, regardless of lock state. This is a classic example of misuse of data storage as sensitive information such as credentials should be stored in the keychain rather than on the file-system as a plist.

In addition to the above, Kik stores other information on the device, including the SMS chat history and contact information. This data is stored in a sqlite database in `Documents/kik.sqlite` which again is not encrypted: Figure 2.

A common dilemma and one faced by the Kik application is that it is a real-time app that receives messages while backgrounded and regardless of lock state. If the app was to apply `NSFileProtectionComplete`, it would not be able to access the SQLite store when the phone is locked. Partial mitigation might be achieved by encrypting the data until the first phone unlock by setting the `NSFileProtectionCompleteUntilFirstUserAuthentication` constant. Subsequent reboots would cause the data to be encrypted, however this is only available from iOS 5.

The Kik app also allows users to send attachments such as photos within IMs, these are stored unencrypted in the `Documents/fileAttachments` directory. For example, the following shows a photo sent via an IM attachment: Listing 1.

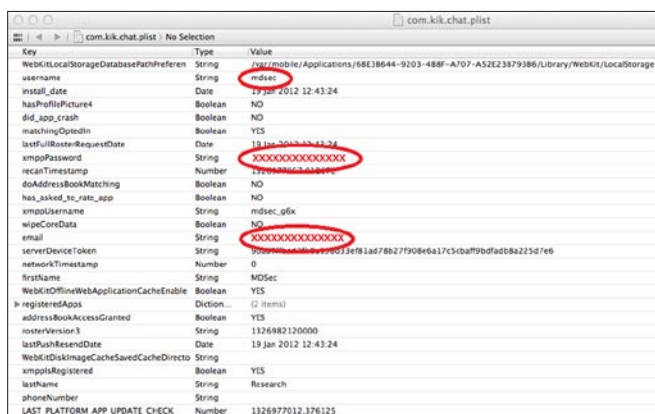


Figure 1. KikNSDefaults Content

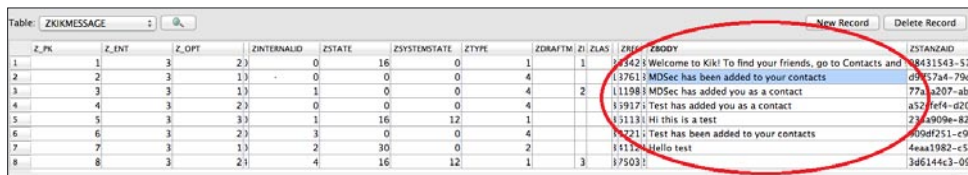


Figure 2. Kik SQLite Database

**Table 2.** Data Protection Levels

Level	Description
No Protection	The file is not encrypted on the file-system.
Complete Protection	The file is encrypted on the file-system and inaccessible when the device is locked.
Complete Unless Open	The file is encrypted on the file-system and inaccessible while closed. When a device is unlocked an app can maintain an open handle to the file even after it is subsequently locked, however during this time the file will not be encrypted.
Complete Until First User Authentication	The file is encrypted on the file-system and inaccessible until the device is unlocked for the first time. This helps offer some protection against attacks that require a device reboot.

**Table 3.** Data Protection Attributes

NSData	NSFileManager
NSDataWritingFileProtectionNone	NSFileProtectionNone
NSDataWritingFileProtectionComplete	NSFileProtectionComplete
NSDataWritingFileProtectionCompleteUnlessOpen	NSFileProtectionCompleteUnlessOpen
NSDataWritingFileProtectionCompleteUntilFirstUserAuthentication	NSFileProtectionCompleteUntilFirstUserAuthentication

It is worth considering that iOS itself does not apply data protection to photos stored on the device; however it is a risk that the app could potentially avoid.

The Data Protection API allows four levels of file-system protection which are configurable by passing an extended attribute to the `NSData` or `NSFileManager` classes. The possible levels of protection are: Table 2.

In order to apply one of the above levels of protection, one of the following extended attributes must be passed to the relevant class: Table 3.

For example, consider an application that needs to save some data to the filesystem, but does not require access to the file while the device is locked, such as an

app that allows you to download documents and then later view them. As the app does not require access to the files when the device is locked, it can take advantage of the complete protection by setting the `NSDataWritingFileProtectionComplete` or `NSFileProtectionComplete` attributes: Listing 2.

In this scenario, the document will only be accessible while the device is unlocked. The OS provides a 10 second window between locking the device and this file being unavailable. The following shows an attempt to access the file while the device is locked: Listing 3.

Developers wishing to apply the relevant protection levels to data stored on the device can achieve this in

**Listing 2.** Example of Implementing `NSDataWritingFileProtectionComplete`

```
-(BOOL) getFile
{
    NSString *fileURL = @"http://www.mdsec.co.uk/training/wahh-live.pdf";
    NSURL *url = [NSURL URLWithString:fileURL];
    NSData *urlData = [NSData dataWithContentsOfURL:url];
    if ( urlData )
    {
        NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
        NSString *documentsDirectory = [paths objectAtIndex:0];

        NSError *error = nil;
        [urlData writeToFile:filePath options:NSDataWritingFileProtectionComplete error:&error];

        return YES;
    }
    return NO;
}
```



# You've been using us for years.



World-Leading Security Consulting.



Email [sales@mdsec.co.uk](mailto:sales@mdsec.co.uk)

Tweet [@MDSecLabs](https://twitter.com/MDSecLabs)

[www.mdsec.co.uk](http://www.mdsec.co.uk)

**Table 4.** Keychain attributes

Attribute	Description
kSecAttrAccessibleAlways	The keychain item is always accessible.
kSecAttrAccessibleWhenUnlocked	The keychain item is only accessible when the device is unlocked.
kSecAttrAccessibleAfterFirstUnlock	They keychain item is only accessible after the first unlock from boot. This helps offer some protection against attacks that require a device reboot.
kSecAttrAccessibleAlwaysThisDeviceOnly	The keychain item is always accessible but cannot be migrated to other devices.
kSecAttrAccessibleWhenUnlockedThisDeviceOnly	The keychain item is only accessible when the device is unlocked and cannot be migrated to other devices.
kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly	The keychain item is accessible after the first unlock from boot and cannot be migrated to other devices.

a similar manner to the above by passing the relevant attribute that best fits the developer's requirement for file access.

In conclusion, iOS leaves data protection very much in the developer's hands, providing granular controls to configure the level of protection that can be applied to data written to the filesystem. Unfortunately, it is common to find that developers do not take advantage of this protection and leave sensitive data at risk of compromise.

## iOS Keychain

The iOS keychain is an encrypted container used for storing sensitive data such as credentials whilst restricting apps to accessing only their own keychain items unless they are a member of a keychain access group.

Similarly to files on the filesystem, a protection level can be applied using the Data Protection API. The following table describes the available protection levels for keychain items: Tabela 4.

### Listing 3. Attempting to Access the Document When Locked

```
test-iPhone:/var/mobile/Applications/7F5ED565-781E-47FD-8787-4C76CD7A4DD5 root# ls -al Documents/ total 372
drwxr-xr-x  2 mobile mobile   102 Jan 20 15:24 ./
drwxr-xr-x  6 mobile mobile   204 Jan 20 15:23 ../
-rw-r--r--  1 mobile mobile 379851 Jan 20 15:24 wahn-live.pdf
test-iPhone:/var/mobile/Applications/7F5ED565-781E-47FD-8787-4C76CD7A4DD5 root# strings Documents/wahn-live.pdf
strings: can't open file: Documents/wahn-live.pdf (Operation not permitted)
test-iPhone:/var/mobile/Applications/7F5ED565-781E-47FD-8787-4C76CD7A4DD5 root#
```

### Listing 4. Sample Provisioning Profile

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>application-identifier</key>
<string>my.company.VulnerableiPhoneApp</string>
<key>get-task-allow</key>

<key>keychain-access-group</key>
<array>
<string>my.company.VulnerableiPhoneApp</string>
</array>
</dict>
</plist>
```

**Listing 5. Example of Adding a KeyChain Item**

```

- (NSMutableDictionary *)getkeychainDict:(NSString *)service {
return [NSMutableDictionary dictionaryWithObjectsAndKeys:
        (id)kSecClassGenericPassword, (id)kSecClass, service, (id)kSecAttrService, service,
        (id)kSecAttrAccount, (id)kSecAttrAccessibleWhenUnlocked, (id)kSecAttrAccessible, nil];
}

- (BOOL) saveLicense:(NSString*)licenseKey {
static NSString *serviceName = @"my.company.VulnerableiPhoneApp";
NSMutableDictionary *myDict = [self getkeychainDict:serviceName];
SecItemDelete((CFDictionaryRef)myDict);
NSData *licenseData = [licenseKey dataUsingEncoding:NSUTF8StringEncoding];
[myDict setObject:[NSKeyedArchiver archivedDataWithRootObject:licenseData] forKey:(id)kSecValueData];

OSStatus status = SecItemAdd((CFDictionaryRef)myDict, NULL);

if (status == errSecSuccess) return YES;

return NO;
}

```

**Listing 6. SQLite KeyChain Groups**

```

test-iPhone:/var/Keychains root# sqlite3 keychain-2.db "select agrp from genp"
ichat
com.apple.apsd
apple
my.company.VulnerableiPhoneApp
test-iPhone:/var/Keychains root#

```

**Listing 7. NSURLConnection Example**

```

@implementation insecuressl

int main(int argc, const char* argv[])
{
NSString *myURL=@"https://localhost/test";
NSURLRequest *theRequest = [NSURLRequest requestWithURL:[NSURL URLWithString:myURL]];
NSURLResponse *resp = nil;
NSError *err = nil;
NSData *response = [NSURLConnection sendSynchronousRequest:

NSString * theString = [[NSString alloc] initWithData:response encoding:NSUTF8StringEncoding];
[resp release];
[err release];

return 0;
}
@end

```

```

Handshake Type: Client Hello (1)
Length: 135
Version: TLS 1.0 (0x0301)
▷ Random
  Session ID Length: 0
  Cipher Suites Length: 58
▷ Cipher Suites (29 suites)
  Compression Methods Length: 1
▷ Compression Methods (1 method)
  Extensions Length: 36
▷ Extension: server_name
▷ Extension: elliptic_curves
▷ Extension: ec_point_formats
    
```

Figure 3. iOS 4.3 SSL Client Hello

Keychain items can be added using the `SecItemAdd` or updated using the `SecItemUpdate` methods, which accept one of the above attributes to define the protection level to apply. By default all keychain items are created with a protection level of `kSecAttrAccessibleAlways` which will allow access at any time and allows migration to other devices.

Applications' access to keychain items is limited by the entitlements they are granted. The keychain uses application identifiers stored in the `keychain-access-group` entitlement of the provisioning profile for the app; a sample provisioning profile that allows keychain access only to the app's keychain is shown Listing 4.

As previously noted, an app can add an item to the keychain using the `SecItemAdd` method; consider the following example app that wishes to store a license key in the keychain and only requires access to the item when the device is unlocked: Listing 5.

Firstly, the app creates a dictionary of key-value pairs which are the configuration attributes for the keychain. In

```

▽ Cipher Suites (29 suites)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_RC4_128_SHA (0xc007)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA (0xc008)
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
  Cipher Suite: TLS_ECDHE_RSA_WITH_RC4_128_SHA (0xc011)
  Cipher Suite: TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (0xc012)
  Cipher Suite: TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA (0xc004)
  Cipher Suite: TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA (0xc005)
  Cipher Suite: TLS_ECDH_ECDSA_WITH_RC4_128_SHA (0xc002)
  Cipher Suite: TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA (0xc003)
  Cipher Suite: TLS_ECDH_RSA_WITH_AES_128_CBC_SHA (0xc00e)
  Cipher Suite: TLS_ECDH_RSA_WITH_AES_256_CBC_SHA (0xc00f)
  Cipher Suite: TLS_ECDH_RSA_WITH_RC4_128_SHA (0xc00c)
  Cipher Suite: TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA (0xc00d)
  Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
  Cipher Suite: TLS_RSA_WITH_RC4_128_SHA (0x0005)
  Cipher Suite: TLS_RSA_WITH_RC4_128_MD5 (0x0004)
  Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
  Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
  Cipher Suite: TLS_RSA_WITH_DES_CBC_SHA (0x0009)
  Cipher Suite: TLS_RSA_EXPORT_WITH_RC4_40_MD5 (0x0003)
  Cipher Suite: TLS_RSA_EXPORT_WITH_DES40_CBC_SHA (0x0008)
  Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
  Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
  Cipher Suite: TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x0016)
  Cipher Suite: TLS_DHE_RSA_WITH_DES_CBC_SHA (0x0015)
  Cipher Suite: TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA (0x0014)
  Compression Methods Length: 1
    
```

Figure 4. iOS 4.3 SDK Cipher Suites

this instance the app sets the `kSecAttrAccessibleWhenUnlocked` attribute to allow access to the keychain item whenever the device is unlocked. The app then sets the `kSecValueData` attribute to the value of the data that it wishes to store in the keychain, in this instance the license key data, and adds the item to the keychain using the `SecItemAdd` method. Under the hood, the keychain is simply a SQLite database and can be queried like any other database. For example, to find out the list of the keychain groups the following query can be executed: Listing 6.

On a jailbroken phone, it is possible to dump all the keychain items for any application under the same caveats previously detailed with the Data Protection API. This is achieved by creating an app that is assigned to all the relevant keychain-access-groups and querying the keychain service to retrieve the protected items (Keychain Dumper <https://github.com/ptoomey3/Keychain-Dumper>).

## Transport Security

Most iOS applications will perform some network communication and due to the nature of mobile devices this communication may often occur over an untrusted or insecure network such as hotel or café WiFi, mobile hotspots or GSM. Consequently, it is imperative that this communication is performed in a secure manner.

iOS apps will commonly interact with online web applications; these operations are often performed using the `NSURLConnection` class. This class takes an `NSURLRequest` object and performs a HTTP(S) request with it. The API uses a default set of SSL ciphers to perform secure connections; unfortunately the API is not granular enough to allow the developer to select which ciphers from the suite to negotiate with. There are some differences between the transports that are negotiated for different versions of the SDK, for example version 5.0 uses TLS 1.2 and offers 37 suites by default while 4.3 uses TLS 1.0 and negotiates 29 suites. However, in the case of the 5.0 SDK, none of the cipher suites offered are considered *weak*.

```

▽ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
  Content Type: Handshake (22)
  Version: TLS 1.2 (0x0303)
  Length: 181
  ▽ Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 177
    Version: TLS 1.2 (0x0303)
    ▷ Random
      Session ID Length: 0
      Cipher Suites Length: 74
    ▷ Cipher Suites (37 suites)
      Compression Methods Length: 1
    ▷ Compression Methods (1 method)
      Extensions Length: 62
    ▷ Extension: server_name
    ▷ Extension: elliptic_curves
    ▷ Extension: ec_point_formats
    ▷ Extension: signature_algorithms
    
```

Figure 5. iOS 5.0 SSL Client Hello

Consider the following example which will perform a simple HTTPS connection to the localhost: Listing 7.

Compiling the application with both the 5.0 and 4.3 SDKs and then running it while monitoring the communication produces different results.

For version 4.3 of the SDK, the application negotiates a TLS1.0 session with one of 29 cipher suites, as shown in Figures 3 and 4.

Using version 5.0 of the SDK, the application negotiates a TLS1.2 session with one of 37 cipher suites, as shown in Figures 5 and 6.

In the above 4.3 SDK negotiation, the following cipher suites can be considered weak:

- TLS\_RSA\_WITH\_DES\_CBC\_SHA
- TLS\_RSA\_EXPORT\_WITH\_RC4\_MD5
- TLS\_RSA\_EXPORT\_WITH\_DES40\_CBC\_SHA
- TLS\_DHE\_RSA\_WITH\_DES\_CBC\_SHA
- TLS\_DHE\_RSA\_EXPORT\_WITH\_DES40\_CBC\_SHA

In order to prevent Man-in-the-Middle attacks, it is essential for iOS applications to prohibit self-signed certificates. The default behaviour for the `NSURLSessionRequest`

```

▼ Cipher Suites (37 suites)
Cipher Suite: Unknown (0x00ff)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xc024)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xc023)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_RC4_128_SHA (0xc007)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA (0xc008)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
Cipher Suite: TLS_ECDHE_RSA_WITH_RC4_128_SHA (0xc011)
Cipher Suite: TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (0xc012)
Cipher Suite: TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384 (0xc026)
Cipher Suite: TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256 (0xc025)
Cipher Suite: TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384 (0xc02a)
Cipher Suite: TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256 (0xc029)
Cipher Suite: TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA (0xc004)
Cipher Suite: TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA (0xc005)
Cipher Suite: TLS_ECDH_ECDSA_WITH_RC4_128_SHA (0xc002)
Cipher Suite: TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA (0xc003)
Cipher Suite: TLS_ECDH_RSA_WITH_AES_128_CBC_SHA (0xc00e)
Cipher Suite: TLS_ECDH_RSA_WITH_AES_256_CBC_SHA (0xc00f)
Cipher Suite: TLS_ECDH_RSA_WITH_RC4_128_SHA (0xc00c)
Cipher Suite: TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA (0xc00d)
Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA256 (0x003d)
Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
Cipher Suite: TLS_RSA_WITH_RC4_128_SHA (0x0005)
Cipher Suite: TLS_RSA_WITH_RC4_128_MD5 (0x0004)
Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (0x0067)
Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (0x006b)
Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
Cipher Suite: TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x0016)
Compression Methods Length: 1

```

Figure 6. iOS 5.0 SDK Cipher Suites

#### Listing 8. `allowsAnyHTTSPCertificateForHost` Example

```

#import "loadURL.h"

@interface NSURLRequest (DummyInterface)
+ (BOOL)allowsAnyHTTSPCertificateForHost:(NSString*)host;
+ (void)setAllowsAnyHTTSPCertificate:(BOOL)allow forHost:(NSString*)host;
@end

@implementation loadURL
-(void) run
{
    NSURL *myURL = [NSURL URLWithString:@"https://localhost/test"];
    NSMutableURLRequest *theRequest = [NSMutableURLRequest requestWithURL:myURL cachePolicy:NSURLRequestReloadIgnoringCacheData timeoutInterval:60.0];
    [NSURLRequest setAllowsAnyHTTSPCertificate:YES forHost:[myURL host]];
    [[NSURLConnection alloc] initWithRequest:theRequest delegate:self];
}

```

#### Listing 9. `didReceiveAuthenticationChallenge` Example

```

-(void) connection:(NSURLConnection *)connection didReceiveAuthenticationChallenge:(NSURLAuthenticationChallenge *)challenge
{
    if ([challenge.protectionSpace.authenticationMethod isEqualToString:NSURLAuthenticationMethodServerTrust])
    {
        [challenge.sender useCredential:[NSURLCredential credentialForTrust:challenge.protectionSpace.serverTrust] forAuthenticationChallenge:challenge];
        [challenge.sender continueWithoutCredentialForAuthenticationChallenge:challenge];
    }
    return;
}

```

## Listing 10. Disabling SSL Validation using CFNetwork

```
- (void)onSocket:(AsyncSocket *)sock didConnectToHost:(NSString *)host port:(UInt16)port {
    NSMutableDictionary *settings = [[NSMutableDictionary alloc] initWithCapacity: 3];
    [settings setObject:[NSNumber numberWithInt:YES]
    forKey:(NSString *)kCFStreamSSLAllowsExpiredCertificates];
    [settings setObject:[NSNumber numberWithInt:YES]
    forKey:(NSString *)kCFStreamSSLAllowsAnyRoot];
    [settings setObject:[NSNumber numberWithInt:NO]
    forKey:(NSString *)kCFStreamSSLValidatesCertificateChain];
```

## References

- Mobile/Tablet Top Operating System Share Trend – NetMarketShare – <http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=9&qpcustomb=1>
- MDSec Research and Blog – <http://www.mdsec.co.uk/research>; <http://blog.mdsec.co.uk>
- Citigroup iPhone Data Storage Issues – [http://www.theregister.co.uk/2010/07/27/citi\\_iphone\\_app\\_weakness/](http://www.theregister.co.uk/2010/07/27/citi_iphone_app_weakness/)
- iPhone data protection in depth – <http://esec-lab.sogeti.com/dotclear/public/publications/11-hitbamsterdam-iphonedataprotection.pdf>
- Keychain Dumper – <https://github.com/ptoomey3/Keychain-Dumper>
- OWASP Mobile Security Project – [https://www.owasp.org/index.php/OWASP\\_Mobile\\_Security\\_Project](https://www.owasp.org/index.php/OWASP_Mobile_Security_Project)

class is to reject self-signed certificates and raise an `NSURLErrorDomain` exception. However, it is not uncommon to see developers override this behaviour to accept any certificate, presumably to allow self-signed certificates deployed in pre-production environments. The certificate validation can be disabled for the requested domain using the `allowsAnyHTTSPCertificateForHost` method, similar to that in the following example: Listing 8.

The `allowsAnyHTTSPCertificateForHost` method is a private method and using it in production code may result in the application being rejected from the App Store. An alternate approach for bypassing SSL verification that is not uncommon is using the `continueWithoutCredentialForAuthenticationChallenge` selector, implemented within the `NSURLConnection` delegate method `didReceiveAuthenticationChallenge`, as shown Listing 9.

The CFNetwork framework provides an alternate API for implementing SSL, indeed the framework allows greater control and customisation of the SSL session for the developer. Similarly to `NSURLRequest`, it is not uncommon to see developers weaken the SSL configuration. CFNetwork however provides more granular controls, allowing the application to accept expired certificates or roots, allow any root or even perform no validation on the certificate chain.

Consider the following `onSocket` delegate method, taken from a real-world application: Listing 10.

Unfortunately, when using the CFNetwork framework, there is no clear method of modifying the cipher suite and again, the SDK default set of ciphers is used.

In conclusion, it is imperative for mobile applications to implement transport methods in a secure manner

and in the default mode and using the latest SDK, this is likely to be the case when developing an iOS application. However, the APIs do allow the transport security to be weakened and it is not uncommon to see this implemented by developers.

## Conclusion

Apple provides developers with a configurable framework to implement both transport and data storage mechanisms in a secure manner. Unfortunately, MDSec regularly observe that developers do not always take advantage of these features, which can lead to our personal data being left at risk. As the mobile app market continues to grow, there lies a responsibility with app developers to ensure that our data is handled securely. The community as a whole should seek to produce standards and guidelines for mobile developers to adopt. OWASP mobile security (OWASP Mobile Security Project [https://www.owasp.org/index.php/OWASP\\_Mobile\\_Security\\_Project](https://www.owasp.org/index.php/OWASP_Mobile_Security_Project)) is one such project that although still in its infancy, seeks to raise awareness about mobile app issues. Hopefully, 2012 will see an increased focus on mobile security issues and raise the bar for app developers across all platforms.

## DOMINIC CHELL

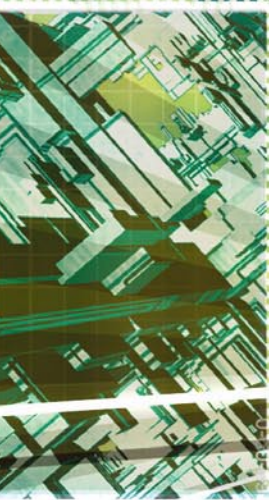
*Dominic is a director of MDSec, a UK based security consultancy specialising in a range of technical security assessment services including Mobile security. MDSec's published work includes popular titles such as the Web Application Hacker's Handbook. As a researcher, Dominic has been publicly acknowledged by numerous vendors, including Apple, for vulnerability disclosure.*

# The Industry's First Commercial Pentesting Drop Box.

# THE Pwn Plug.



Air Freshener?



Printer PSU?  
...nope



## FEATURES:

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



**PWNIE EXPRESS**

**@pwnieexpress.com**

Discover the glory of  
Universal Plug & Pwn

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

# When Developers

## API Simplify User-mode Rootkits Developing

This is a series of articles about shell extensions that enhance high-level features of any operation system. However, such possibilities not only enrich platform but simplify developing trojans, exploits that leads to the new security holes. Mostly this kind of extensions are known as user-mode rootkits.

Cybercrime is becoming a growing threat to society. The thefts of information, website crashing or manipulating online payment traffic are also increasing. Many organizations offer various services in the battle against digital crime, such as network or data monitors and extractions tools. It is interesting mainly to authorities and financial institutions, but they are accessible to every organization.

Smartphones have been equipped with operating systems that compare in complexity with those on desktop computers. This trend makes them vulnerable to lot of the same threats as desktop OS. The past several years of mobile malware are no longer only in the theory. There has been malware, loss, theft, data communication interception, exploitation, etc. This has been because of the decreasing cost of mobile devices which allowed them to enrich the software interfaces that allows users to interact better with the cyber and the physical worlds. For example, mobile devices are often pre-installed with a number of applications, including clients for location-based services and general-purpose web browsers used for chatting and social networking.

The increasing number of mobile-related exploits tends to impact security breaches which have put the industry at a critical point. That's the greatest risk to all mobile OS moving to involve the rapid development, and distribution throughout so-called app markets While most of application markets provide an ideal transportation mechanism for the delivery of malicious software to the heart of industry's networks. According to Juniper Networks the following mobile malware take place:

- Mobile device and OS market share and mobile malware infection rates are linked

- Mobile malware uses the same techniques as PC malware to infect mobile devices.
- The greatest mobile malware risk comes from rapid proliferation of applications from app stores.
- RIM BlackBerry, Google Android, and Apple iOS operating systems suffer predominantly from spyware applications

It's totally right. If you want to distribute malware take the most popular application or game, most popular device what you are going to infect and place malware application into „warez“-storage. For example, take any game for Android/iOS, that is not free (like angry birds) and place a supposedly cracked version on the non-official android market. The application does not even need to be the game. To prove this idea let's go back to November 2010 when security researchers Jon Oberheide and Zach Lanier unveiled an Android exploit at an Intel security conference in Oregon. They showed that the Android security model include a security flaw that allows an application to invisibly download additional exe-applications, known as APK files, without requiring the user's permissions. Their proof-of-concept malware did not contain any actual malicious code; it simply portrayed itself as bonus levels for *Angry Birds* that, once installed, would open up more levels for the player. In reality, nothing related to *Angry Birds* was ever included in the application. However, Oberheide and Lanier proved that users could be tricked into downloading this application, and that the application could download and install additional applications without prompting the user to approve the additional installs, or to verify and agreement required for the background applications to



be installed. Unfortunately, BlackBerry suffers from the same problem.

Mobile environments make every attractive targets to attackers. Important personal and financial information can easily be compromised because phone usage is a part of day-to-day user activities. For example, mobile devices are being used for chatting, email, storing personal data even financial data, pictures, videos, GPS tracks and audio notes.

A rootkit is a stealth type of malicious software – designed to keep itself, other files, or network connections hidden from detection. A rootkit typically intercepts common API calls to modify or filter information from the operating system to keep itself hidden. For example, it can intercept requests to file explorer and cause it to keep certain files hidden from display, even reporting false file counts and sizes to the user. There are legitimate uses for rootkits by law enforcement, parents or employers wishing to retain remote command and control and/or the ability to monitor activity on their employee's or children's computer systems.

Another example, rootkits are commonly used to conceal keyloggers, which steal sensitive user data, such as passwords and credit card numbers, by silently logging keystrokes. Rootkits are design to maintain access to targeted computers. Rootkits can also disable the firewall/antivirus tools by replacing files, changing settings or modifying what the antivirus application can see. None of these activities are directly visible to the user because the rootkit conceals its presence.

There are several kinds of rootkits. They are bootkits, firmware user-mode kernel and hypervisor. A further discussion needs to compare kernel mode with user-mode rootkits.

Kernel mode rootkits involve system hooking or modification in kernel space which is the ideal place because it is at the lowest level, highest level of security and thus, is the most reliable and robust method of system hooking.

As a system call's execution path leaves user mode and enters kernel mode, it must pass through a gate. This gate must be able to recognize the purpose of the incoming system call and initiate the execution of code inside the kernel space and then return results back to the incoming user mode system call. It's some kind of a proxy between user mode and kernel mode. One of the rootkit techniques is to simply modify the data structures in kernel memory. For example, kernel memory must keep a list of all running processes and a rootkit can simply remove themselves and other malicious processes they wish to hide from this list. Rootkits do this by inserting or patching the process that list running processes and then filter what processes are reported as running.

User mode rootkits involve system hooking in the user or application space. Whenever an application makes a system call, the execution of that system call follows a predetermined path and after that rootkit can intercept it. One of the most common user mode techniques is the one in memory modification of system libraries.

That's why applications run in their own memory space and the rootkit needs to patch the memory space of every running application to provide self-control. Moreover, the rootkits have to monitor for new applications to patch those programs' memory space too (need to explain the security rings if they are referenced).

User-mode rootkits run in Ring 3, along with other applications as user, rather than low-level system processes. They have a number of possible installation vectors to intercept and modify the standard behaviour of *application programming interfaces* (APIs). Some inject a dynamically-linked library like a DLL file on Windows into processes, and are thereby able to execute inside any target process to spoof it. Injection mechanisms include:

- Use of vendor-supplied application extensions. For example, Windows Explorer as well as any mobile platform like BlackBerry has public interfaces that allow third parties to extend its functionality.
- Interception of messages.
- Exploitation of security vulnerabilities.
- Function hooking or patching of commonly used APIs, for example, to mask a running process or file that resides on a file system.

Windows-based rootkits are modifying paths and system structures these methods are used to mask network activity, registry keys, and processes Rootkits modify all the things which could alert a user to the fact that a malicious program is active in the system. Implementation in user mode rootkits is relatively easy. Most often, a method based on hooking API functions is used to modify the path to executables.

Many of rootkits techniques are well documented and use *normal* applications. Example: a desktop firewall program may use similar hooking things to watch and alert the user to any outgoing network connections while a rootkit will use it to hide their backdoor activities. The legitimizing effect of commercial rootkit software is leading away from user-mode and toward kernel-mode techniques at first glance. However, user-mode rootkits still have the ability to bypass security applications Non-official market places are now widely available on the Internet therefore malware writers don't even ponder over how to spread it. It's very easy to integrate several technologies into one malware attack.

User-mode rootkits exist for \*NIX, Windows and are known for mobile devices such as Android or BlackBerry.

## Why not BlackBerry?

BlackBerry smartphone applications include inherent virus protection and spyware protection that is designed to contain and prevent the spread of viruses and spyware to other applications. Security is known as the cornerstone of the BlackBerry system that allows users to confidently access sensitive information.

Previous attacks on BlackBerry included: several exploits, password thefts, screen information, chats messages and other data. All of these described attacks are possible to put into practice on application (user-mode) level. While root mode provides powerful ability to feel like God, application level has the ability for wide spreading, easy distribution, misleading and finally easy developing. The most popular solution in security field is to operate as always under attack. Well-established products will provide the end user with some protection. Meanwhile vendors start to develop security measures as hackers continue to develop new rootkit/exploits. That's why an application level is one of most interesting to research and actually it will always be relevant and useful to take it under investigation. Forensic investigation can be conducted on cloud or mobile devices, but you're still able to extract most data. (Computer and servers) However, it's really a problem when accessing the network card on high level at first glance, but don't forget about browser plugin, IM plugin and etc. Therefore this type of level is based on opened API for developers clearly leads to exploitation.

Coming back to my previous articles I'm going to refresh some ideas of these attacks. First article discussed password protection (*Is Data Secure on the Password Protected Blackberry Device?* Hakin9,

February 2011) mainly password masking as point of protection. It's obvious that in most cases masking passwords doesn't even increase security, but it does cost your business due to login failures. Visualizing the system status has always been among the most basic usability principles. Showing asterisks while users enter complex codes definitely fails to comply. Other background was keystroke emulation as a kind of underlying principle of direct interaction with screen. BlackBerry has an API managed with applications that eventually ask user to choose restriction via showing API-requests. Why does a user agree with it? Some application such as phone manager have no way to communicate with high level hardware to catch incoming calls and auto pick up. It seems there is no failure because user installs applications and allows their policies. However, the issue covers all screens that can be managed by keystroke emulation or navigation emulation, or any textbox. One exploit shown in the article which was by „noising” the inputting text field. The major concept is in using the most complex password in range of 14-18 symbols with case-sensitive. That's right, you're obliged to use the most complex password and you never see the noise-symbol until unmasking happen, which usually leads to device wiping (Figure 1 and 2).

Next bad idea discussed in the same article based on *global* permission of screen-capturing as previous keystroke emulation. You can choose whether you want to use such application, for example, to capture a lot of screenshots to make a video tutorial are only between *yes* and *no*. Thus, in my other article (*Why is password protection a fallacy point of view?* Hakin9 Extra, June 2011) I showed that you can sniff password data. Two issues for further discussions:

- Keylogger
- Datalogger

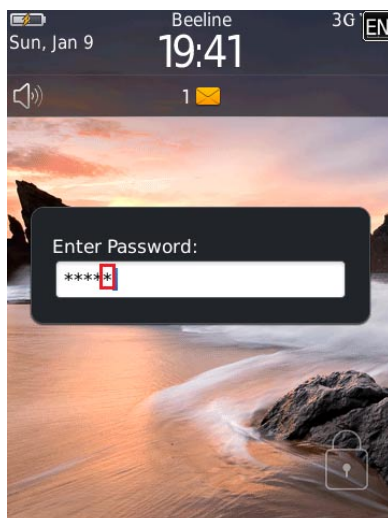


Figure 1. Noising password field

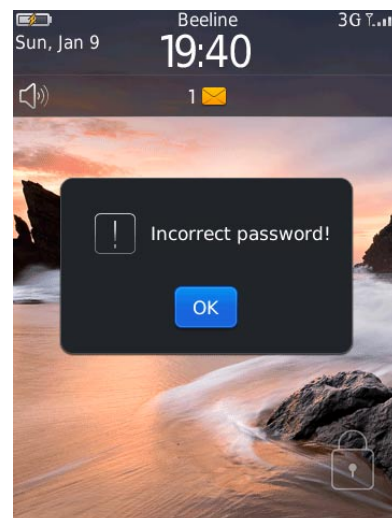


Figure 2. Result of noising password field

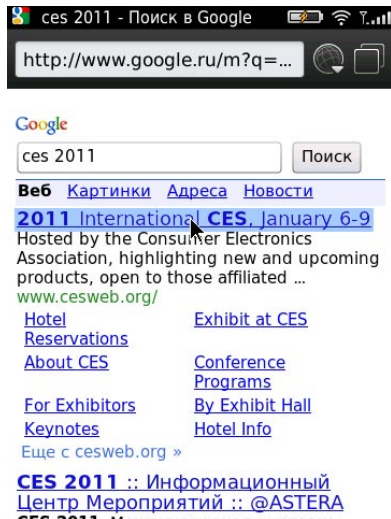


Figure 3. Screen-capture of browser #1

Many thanks to Apple's patent (if I'm not mistaken) to the asterisk masking lagging. More details: when you touch screen to type a character a big-scaled preview appears. When you do the same while typing password into masked text box you can see that every character is going to be masked by asterisk or black circle in about~1-2 second afterward. It's quite true to all mobile devices. But if you use a hardware keyboard you will never see it. Reasonably, password preview is only used when the keyboard is a sure type or multitap keyboard. The bold keyboard is a full keyboard so it won't duplicate that behavior. Such preview is screenshot-able. Average statistical data shows 300-400 msec is good to steal each button press like letters, numeric, switching between letter and numeric or symbols or pressing some kind of *shift/caps lock*. If you want to be totally sure set a 100 msec timer to get everything (Figure 3-7).

Despite the *big bug* in good managing password by your brain and previous exploitation each user has a problem with password managers. BlackBerry gives an

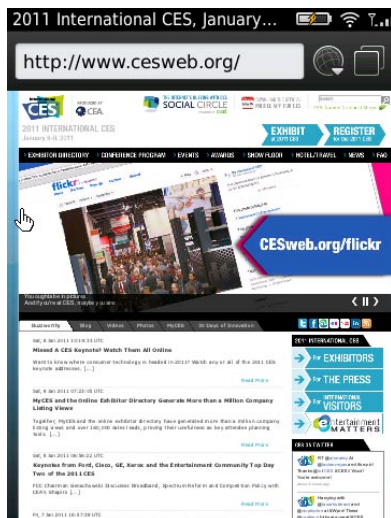


Figure 4. Screen-capture of browser #1

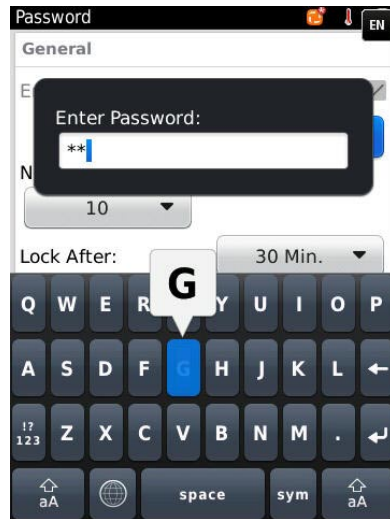


Figure 5. Screen-capture of password's creation window

opportunity to choose pre-installed Password Keeper or BlackBerry Wallet. Both can be screen-captured! A good question is why can't we take control over all windows application (even 3rd party apps)? Some kind of answer: Clipboard manager restricts data extracting while window of Password Keeper/BlackBerry Wallet is active until you minimize it or close it. Is it really needed in protection by password masking (Figure 8)?

Next failures with password discussed in that article highlight attacker capabilities to steal password while you're syncing your device with a PC (discussed Windows-based PC only) from password by catching sync-event and start to screen capturing for example. There's protect the PC side too. There are four attacks there. Unfortunately, we can't get a screen-capture.

First of them dealt with previous version of BlackBerry Device Manager (4-5 version) developed by C++. It divides into two builds (Windows Seven and Windows XP). Second of them was referring to BlackBerry Desktop Manager (6 version) developed mainly by .NET. Third of them expanded the previous in case

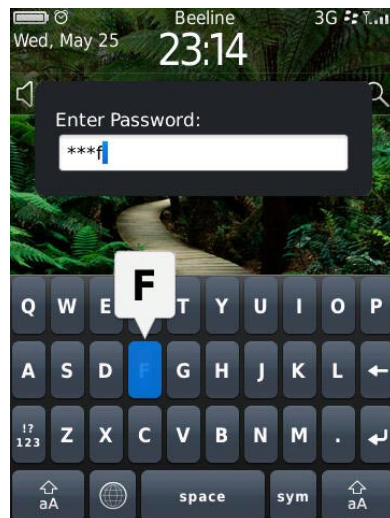


Figure 6. Screen-capture of device-unlocking

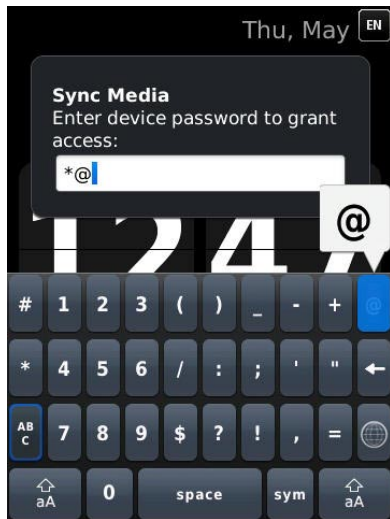


Figure 7. Screen-capture of device-unlocking while PC-syncing

of getting password that types before backup starts. Fourth of them dealt with silent connecting if you've got a device password to wipe for example.

Recalling knowledge about system messages and system object answers to us that edit box is a simple field for typing characters ~32k in length with a `passwordchar` property. It has default #0 value or NULL or \0. Other masking character could be a black circle or asterisk or anything else. 0x25CF is unicode character of black circle. Every system object like modal window or textbox responds to API subroutine such as `SendMessage` or `PostMessage`. Both subroutines send the specified message to a window or windows. But if you need to post a message in the message queue associated with a thread you should use the `PostMessage` function. Parameters' syntax is the same. First parameter is (Type: `HWND`) a handle to the window whose window procedure will receive the message. Second parameter is (Type: `UINT`) a message to be sent. Other two parameters (Type: `WPARAM`, Type: `LPARAM`) represent an additional message-specific

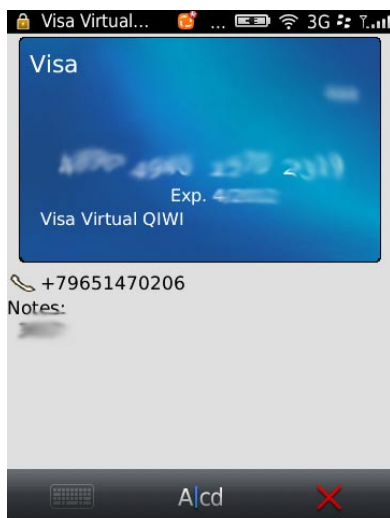


Figure 8. Screen of BlackBerry Wallet

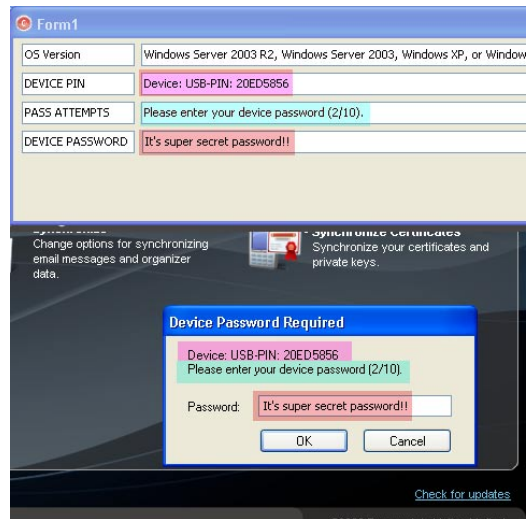


Figure 9. Attacking BlackBerry Device Manager

information. It's easy to guess that we need in `WM_GETTEXT` (0x000D) message because it copies the text that corresponds to a window into a buffer provided by the caller. However, if the editbox is masked you can't copy text, because you get a NULL-pointer. Well then do unmask, copy and mask again.

In 2003 a MS Windows `PostMessage` API Unmasked Password Weakness was found. Declared affects for MS Windows 2000, XP and could effectively allow unmasked passwords to be copied into a user's clipboard or other buffer. `EM_SETPASSWORDCHAR` (Type `UINT`, Message) messages sets the password mask character in password edit box controls. `PostMessage` may be abused in combination with `EM_SETPASSWORDCHAR` messages to cause an unmasked password to be placed into a buffer which could potentially be accessed through other means by an unauthorized process. The unmasked password can be copied while this is occurring. If we try to use this code in Vista or Windows 7 we get nothing, because it's more correct to set system hook in owner address space via

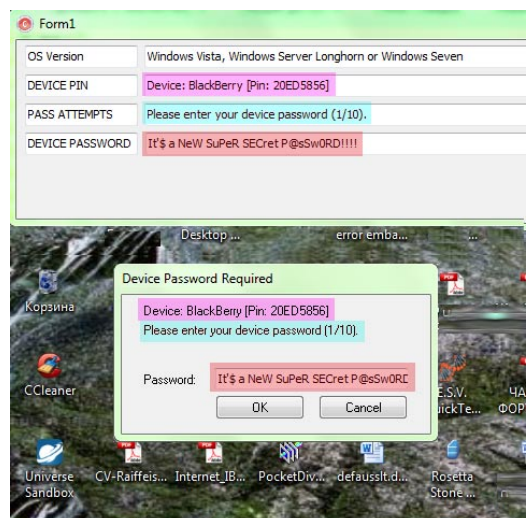


Figure 10. Attacking BlackBerry Desktop Software

### Listing 1. FaceBook Additional Info

```
FaceBook Additional Info
Friendly name: Facebook
Description: Facebook?@ for BlackBerry?@ smartphones makes it even easier to connect and share while you're on
the go...
Version: 2.0.0.37
Vendor: Research In Motion Limited
Copyright: (null)
```

### Listing 2. FaceBook Additional Info

```
PhoneArguments phoneArgs = new PhoneArguments(PhoneArguments.ARG_CALL, premium_number);
Invoke.invokeApplication(Invoke.APP_TYPE_PHONE, phoneArgs);
```

loading a DLL-Cather. But at this rate you should know OS version, right? Roughly, we need a so called Major Version to distinct XP and Seven. Most of this repeats previous parts when you deal with BlackBerry Desktop Manager. You can filter by application name or etc to gain access to type the password. Three HwndWrapper as classname text were presented in my article to catch backup-password as some kind of extended filters (Figure 9 and 10).

Most of these attacks filled forensics article (*To Get Round To The Heart Of Fortress*, Hakin9 Extra, August 2011). This article discovered problem with antiforensic methods especially when we talk about live forensics. As example, we can extract device information, hardware Id, PIN, OS Version, some more applications like a facebook (Listing 1).

Using live methods we are able to extract Address Book, Calendar Events, Call History, Browser history and bookmarks, Memos and Tasks, Screen-shots, Camera-shots, Videocamera-shots, Clipboard, Location tracking (cell, wifi, gps, bluetooth), SMS/MMS/Emails, Pictures, Videos, Voice notes, and other file, IMs, etc. BlackBerry EXIF-Picture information tells you about filename, details of camera e.g. RIM BlackBerry Torch as name, DateTime, Resolution or GPS. Also, you can find saved IM history (even BBM) on internal storage or SD-storage. All files filled IM history has a simple CSV-format: Date/Time, ID Sender, ID Receiver, and Data.

The security conference InfoSecurity Russia 2011 mentioned several API to protect from forensics analyzing in point of emails and PIN. It reconstructs PIN or email message with any property of flag, such received, rejected, read, delivered and etc. Idea consists in simulating a lot of messages to water down the purport of the proof thread.

To imagine how many type of information your device owns just look above. And it's only personal data. There many trojans that steal money by calling to Antarctica or

Dominican Republic. If every call costs at least \$3 per minute, then one compromised device can lead to \$10k per month. Don't forget that developing is easy and needs to be compact. Look a Listing 2.

Whereas `premium_number` is a string and `PhoneArguments` may used by default.

However, BlackBerry Enterprise Solution has several powerful rules to manage with emails, phone numbers or sms/mms. Administrator can add rules like `+7x` to disallow incoming or outgoing calls (or both types) to Russian Federation. The same with emails despite of spam filtering.

The idea was to highlight the failure of group policy such as I have discussed it. Some of them are obvious if you're an Android user. When users try to download any application or game a permission request asking to allow it to install BlackBerry is quite contrary but if you allow sms or email permission to application and you activate any possible action in relation to message like deleting, creating, reading, intercepting and etc. while Amazon (AWS) The Cloud has feature to control in custom policy mode any possible API declared as a developer API.

### YURY CHEMERKIN

*Graduated from 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 a mobile and social infosecurity researcher in Moscow.*

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

*E-mail:*

*[yury.chemerkin@gmail.com](mailto:yury.chemerkin@gmail.com) ([yury.chemerkin@facebook.com](mailto:yury.chemerkin@facebook.com))*

*Facebook: [www.facebook.com/yury.chemerkin](http://www.facebook.com/yury.chemerkin)*

*LinkedIn: [www.linkedin.com/in/yurychemerkin](http://www.linkedin.com/in/yurychemerkin)*

# Android Forensics

Smartphones are changing the IT and Communication landscape vastly. A Smartphone can do almost every good thing a computer can do. Today most of the corporate employee access and manage their official e-mails through the e-mail client installed on their Smartphone.

**R**ight from booking movie tickets to making fund transfers, all e-commerce and online banking transactions can be done using a Smartphone. With high speed of 3G, Smartphones are getting more popular specially among working professionals and students.

As Smartphone market is growing, it is also catching bad guy's attentions. For bad guys or hackers, it is easy to target mobile users as they are less aware and bother less about the risks associated with the mobile and mobile applications.

There are number of Mobile Operating Systems present in the market. Among these Mobile OS, Android, iOS and RIM are more popular than others. Android is the most widely used Mobile OS present in the market. According to Gartner report, Android had more than 36% share of the market by end of the first quarter of 2011.

It is quite obvious that the widely used platform is likely to be targeted more, as in the case of Microsoft Windows Operating System. A hacker wants to target mass and for doing that he has to target the most commonly used platform. Android is one such commonly used platform. As Android is capturing market, it is becoming favorite target platform of hackers.

It is always a challenge for forensic examiners to discover the evidences from the Android devices. Android has a different and newer file system, directory structure, runtime environment, kernel and libraries which make Android more complex to forensic examiner. We will discuss detailed forensics steps to examine Android device in later part of this article.

## How Android can be used in Cyber Crime

Android can be used in cyber crime in two ways:

- Android device is targeted by the attacker.
- Android device is used as a means to carry out cyber crime.

Let us consider some of the real world cases. What if an Android device is discovered from a crime location?? What all evidences can be discovered from the device?? Where exactly to look for the evidences??

These are some challenges faced while doing forensic analysis on Android device. First we will see what all bad things can be done with the Smartphone (or how a Smartphone can be used in various criminal activities).

## Cyber Crimes through Smartphones

**Software Theft:** Software theft is now a common attack. If codebase of your software is stolen and sold to your rival, he can make a great loss to your company. Your rivals are ready to invest huge money to obtain source code of your key software.

A Smartphone can carry large volume of sensitive data. It can be used in carrying codebase of any key software of any company. There are security guards and other mechanism in place to check the employees and visitors, if they are carrying any business critical information in any form. But still they hardly check for Mobile phones.

In one classic case of Software theft, an unhappy employ of a company used to carry all source code of the key software of the company in her smart phone. She first copied the code in her phone's external storage and then deleted the same data from the phone. When her phone was observed at security check, nothing was found in her phone. When she reached home, she used

a tool to recover the deleted data. This way she took all the data out from her company and latterly she sold the source code to the rival of her employer.

**Terrorist Activities:** Terrorists also use Smartphones to exchange and store the information. They use Smartphones to communicate with the other member of the terrorist organization. They also use GPS to find locations. They can store various data in the Smartphone like maps or photos of target locations, encrypted and stagno files, instructions etc. They can use the phone to click photos of target locations.

**Pornography/Child Pornography:** Pornography is fully banned in a number of countries. And child pornography is considered a big offence across the world. Smartphone can be used to store, view, capture and exchange such kind of materials.

**Sexual Harassment Cases:** Smartphone can play big role in sexual harassment kind of cases. If a Smartphone is discovered from accused, a forensic examiner can get treasures of information from the device.

**Financial Crimes:** Every other bank is developing banking and other non-financial application to facilitate their mobile customers. These applications can be used for malicious activities by hackers. A Smartphone recovered in financial fraud cases can give many evidences about the case.

**Murder Cases:** Even in murder or other criminal cases, a Smartphone can provide evidence useful in solving the case. Right from call records and SMSes to facebook records or GPS data can be recovered from the phone.

Let us think about, what all evidences can be recovered from a Smartphone?? Where to look in the Smartphone?? We will discuss in coming section that what all evidences we can discover from a Smartphone:

### Interesting locations for Forensics Investigation

- Phone Browser Memory
- Application storage
- External Card
- SQLite database files
- SMS
- GPS data
- Call records
- Contact list
- Social networking application (Facebook, Twitter, Orkut) records
- Messenger (Yahoo, MSN) records
- Email client data
- System storage
- Data stored in external card

How investigation of Android device is different than other Smartphones?? Does forensic investigators

really needs to learn something special to analyze Android devices?? Can evidences be discovered from the device?? Are they admissible in the court of law??

Next section of the article will answer all the above questions in further detail.

### Forensic Process of Android Device

Forensic process of Android phone will comprise of following steps:

**Seizing Android device:** If an Android device or any Smartphone is discovered from any crime location, first thing a forensic investigator should do is to click the photos of the crime scene including the photo of the device. If phone is ON, take photo of display as well.

If you find mobile to be ON then keep charging the mobile so that the battery does not drain. In case, we don't charge the phone and the phone goes OFF, we may lose the important data especially regarding current or recent applications. If phone is OFF at the time it was recovered, keep it OFF. Seize all other available accessories i.e. memory card, data cable etc.

As soon as we recover anything, start labeling it. It is required to maintain and present a *chain of custody* at the court of law. A label should have the following minimum information on it:

- What is the evidence?
- How did you obtain it?
- When was it collected?
- Who all have handled it?
- Why did that person handle it?
- Where has it travelled and where was it ultimately stored?
- What is Case ID?

*Chain of Custody* is a chronological documentation of individuals who had physical possession of the evidence. Maintaining the chain of custody is vital and it guarantee

The image shows a 'Chain of Custody Form' with the following structure:

- EVIDENCE** (Large bold header)
- Agency \_\_\_\_\_
- Collected By \_\_\_\_\_
- Item # \_\_\_\_\_ Case # \_\_\_\_\_
- Date \_\_\_\_\_ Time \_\_\_\_\_
- Description \_\_\_\_\_
- Location \_\_\_\_\_
- Remarks \_\_\_\_\_
- CHAIN OF CUSTODY** (Section header)
- Received from \_\_\_\_\_
- By \_\_\_\_\_
- Date \_\_\_\_\_ Time \_\_\_\_\_
- Received from \_\_\_\_\_
- By \_\_\_\_\_
- Date \_\_\_\_\_ Time \_\_\_\_\_

Figure 1. Chain of Custody Form

the integrity of the evidence, right from collection to the final test result. Chain of custody is something *must to have* document in any criminal trial. If proper Chain of custody has not been maintained, court may not consider that evidence in making final verdict.

## Creating 1:1 Image

Creating image is the most important task in any forensic analysis. It is the thumb rule in forensic investigation that you cannot work on primary evidences if you want them to take in the court of law. For that we need to create bit-by-bit image of the target device.

What is bit-by-bit image and how it is different from the copy-paste the content of entire disk??

If we copy and paste the content of a disk, this will only copy visible, hidden and system files. Whatever is deleted or not accessible by the OS would not be copied by *copy* command. So, for a thorough analysis, it is required to create a 1:1 image of the disk. Bit-by-bit image is as good as the original image. Thorough analysis is not the only reason we need to take 1:1 image, it is also required by the court of law. If you have not taken 1:1 image, your evidences are not admissible in the court of law.

How we will take image of an Android device?? How I can verify that my image is exact bit-by-bit copy of original disk or device?? How can I establish the authenticity of the image??

There are two locations to be taken image of in case of Android device. One is the device and other is the external card. We will see in following section that how to create a bit-by-bit image of the Android device. But before that we will see how to verify the image.

Before starting the imaging of original disk, calculate the hash value of it and note that down. Now after taking image, calculate the hash value again for the image. If hash values are same for both the image and disk, we can be sure that we have taken exact image of the original disk. Now we can work on image and evidences discovered from the image can be taken to the court along with the hash values calculated. The hash value establishes the authenticity of the image that it has not been tampered.

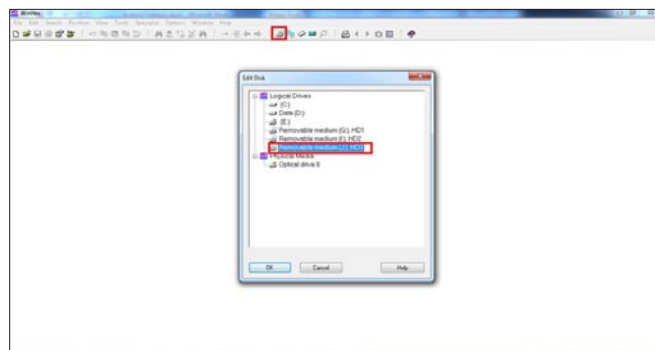


Figure 2. Winhex Opening SD Card

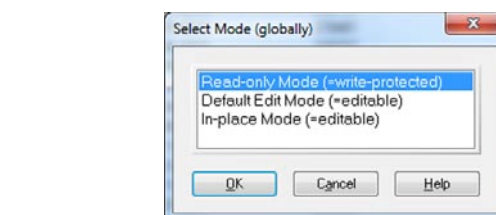


Figure 3. Winhex Making device Write Protected

One more thing we should take care of before creating image is to make the target device in *write protected* mode. Whenever you connect any device to your computer, there are chances that some data can be written on the devices by any software, application or OS. In that case your evidence (device) is no more genuine. Just to avoid this kind of situation, make the disk or device write protected. To do that, use write protected cables present in the market. In this article, we will make device write protected by software to explain the technique.

*Creating image of external memory card:* We will start with simpler part of imaging, which is creating image of the memory card. In most of the cases, file system of the memory card is FAT32 and it is easy to image. There are lots of free and commercial tools available in the market which can help us in creating image of the memory card. We will use free version of Winhex to do that. Winhex is a powerful forensic tool. It is available in both freeware and commercial versions.

## Note

Only commercial tools should be used to discover the evidences in case you want to take evidences to the court.

Here are the detailed steps to take the 1:1 image of the memory card:

First remove the SD card and connect the card to the computer with any card reader. Now we will make the device *write protected* through Winhex. Follow below step to do that: Open the disk in Winhex: Figure 2.

Go to *Options* then *Edit mode* and select first option *write protected* mode: Figure 3.

Now calculate the hash value for SD card.

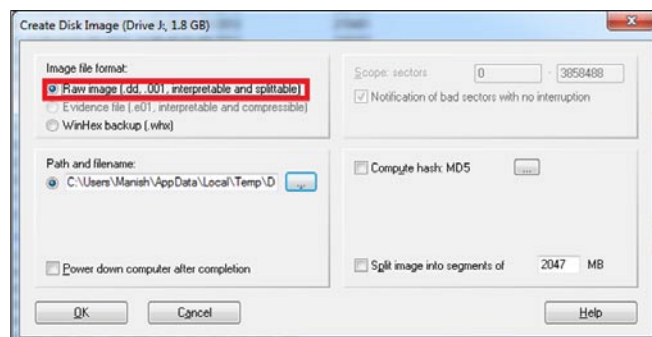


Figure 4. Winhex Creating Image



To calculate hash, go to *Tools* then *Compute hash* and choose any Hashing algorithm. We have to compare this hash value with the hash value computed earlier for the image. Now we create the image of the disk. Go to *File* menu and click on *Create Disk Image* option for creating an image. Choose *Raw image* option (.dd) to create image, as *dd* image is interpreted by almost all commercial and open source forensic tools (Figure 4).

Image of memory card is created. We will use this image for analysis in later part of the article.

### Creating Image of Android device

This is a tricky part. Android does not provide any direct way to access or view its internal directories or system files and directories. But internal or system locations may have most critical data stored. Almost all applications write some application data and temporary data in these directories only. `/data/data` is the most interesting location for the forensic investigator which is not accessible to the user. Only application or root users have access to these locations.

How we can access Android internal directory structure?? How to create the image of the Android internal directory structure??

For this we need to obtain ROOT permission on the Android OS. In Android terminology, we need to ROOT

the device to get the superuser permission. There are various techniques available in the market that can help you in rooting your Android phone. Among them, Odin3 software is one such popular tool. All you need to do is to check the *build number* of your phone. You can check

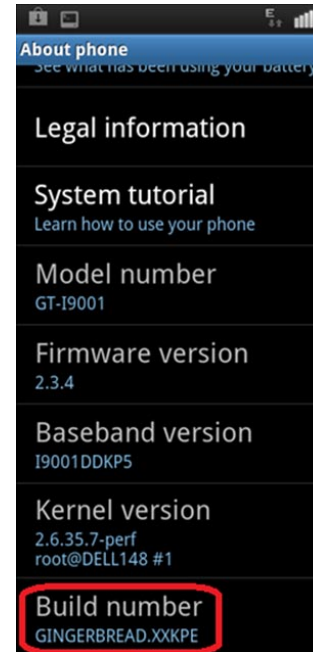


Figure 5. Kernal Build Number

a d v e r t i s e m e n t

# You've been using us for years.



## World-Leading Security Consulting.

```

$ export PATH=/data/local/bin:$PATH
$ su
# cd data/data/com.facebook.katana
# ls
lib
databases
files
app_composer_temp
cache
# cd databases
# ls
fb.db
webview.db
webviewCache.db
uploadmanager.db
# dd if=fb.db of=/mnt/sdcard/external_sd
/mnt/sdcard/external_sd: cannot open for write: Is a directory
# dd if=fb.db of=/mnt/sdcard/external_sd/fb.db
7742+0 records in
7742+0 records out
3963904 bytes transferred in 0.161 secs (24620521 bytes/sec)
    
```

Figure 6. DD Command

it by visiting the following location in any Android phone: Settings-> About Phone-> Build number. Now Google for the rooted kernel for this build number and pass all the files to Odin3 software. This way you can ROOT your phone. There number of good tutorial available in the market on Android Rooting. As per my knowledge ROOTING is legal and it does not void any warranty. Still check local laws before rooting your phone. I have never come across such situations; still it is a general belief that rooting may harm your system or you may lose your entire data stored on the phone.

## Note

In the rooting process, something will be written on target device and as I mentioned earlier, we can't write anything on the phone, if we want to take that into court of law as evidence. The method and technique explained in this example may not be accepted by the court. In this case, one can take approval in advance from the court. That is again subject to local laws. So now we have root access on the phone, what next??

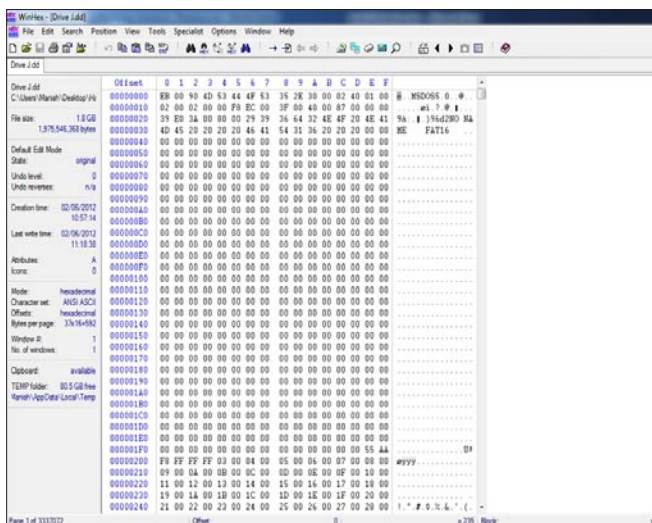


Figure 7. Winhex Reading Image

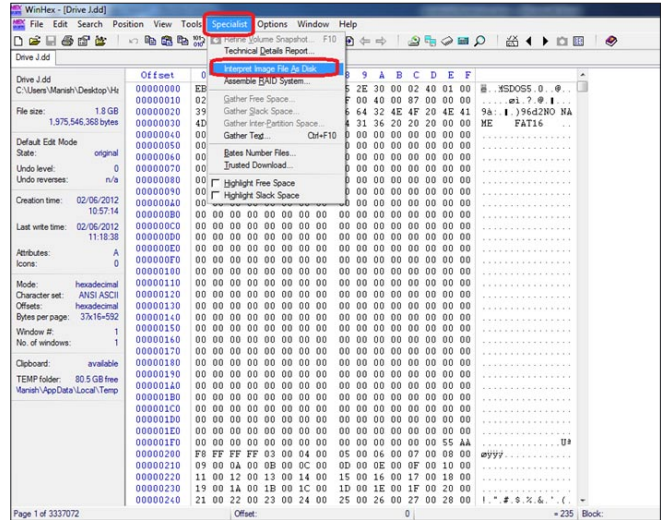


Figure 8. Winhex Interpreting Image as Disk

As it is known to all that Android uses Linux kernel 2.6. By downloading *Terminal Emulator* application from the Android Market, we can run almost all Linux commands. So, to create image of device, we will be using *dd* command. DD stands for Data Description, it does low-level copying of data in Linux. The dd command will help us in creating bit-by-bit image of Android device.

To take backup, insert a fresh SD card in device and copy the target data there. Typical syntax of DD command:

```
dd if=/dev/fd0 of=tmp.image
```

Where if is input file and of is output file. Again, output of the dd command is understood by all commercial and open source forensic tools including WinHex, EnCase, Helix, Forensic Toolkit etc.

To take the backup of the Android system folders, go to `/proc/mnt` file and open the mnt file. You will have similar to following structure:

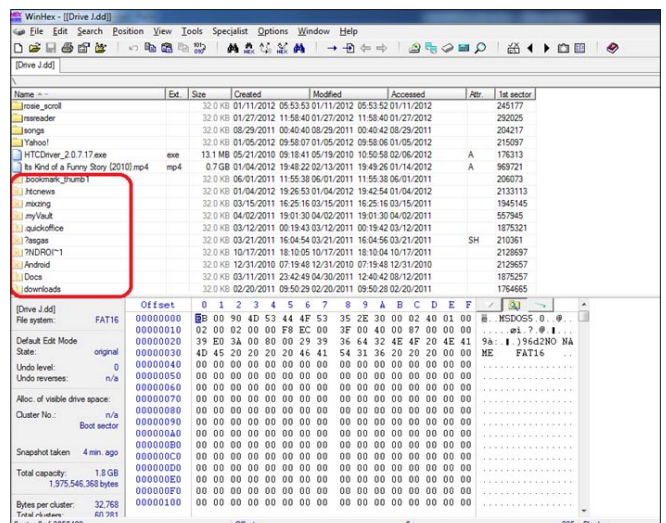


Figure 9. Winhex Hidden Files and Folders

```

dev: size erasesize name
mtd0: 000a0000 00020000 „misc“
mtd1: 00480000 00020000 „recovery“
mtd2: 00300000 00020000 „boot“
mtd3: 0fa00000 00020000 „system“
mtd4: 02800000 00020000 „cache“
mtd5: 093a0000 00020000 „userdata“

```

Copy one by one location through DD command.

To understand the concept, we will be copying some directories with dd command as shown Figure 6.

### Recovering Data

Now we are done with the imaging part. The image created in above steps can be accepted by any forensic tool. We will be using free version of Winhex to recover and analyze the data as well.

In most of cases, criminal deletes suspicious data or even format the entire disk. Suppose in any pornography related case, we hardly find anything in the device, because all data has been intestinally deleted. So, before starting analysis part, it is recommended to recover all deleted or destroyed data first.

To recover deleted data, open the image file in Winhex. Go to File menu then Open option, select the image file and click ok. Figure 7 show the opened image file.

As we can see from the above screenshot, all data is represented in hex form. To make the data understandable, we need to interpret the image as disk. To do that to *Specialist* menu and click on Interpret *Image File As Disk* as shown Figure 8.

Folders highlighted in the Figure 9 are the deleted folders.

To recover deleted files or folder, right click on target folder and click on *Recover/Copy* and select location to save the file.

There are number of tools available to recover deleted or destroyed data. All well known forensic tools

like FTK or EnCase have inbuilt feature of identifying and restoring deleted data.

### Analyzing the Data

Analyzing Android data is a bit different; one should know the important locations to be checked out. More manual intelligence is required in this step of forensic analysis of Android device. For example, in case of money laundering related cases; email, browser data and banking application related data must be looked at to discover any clue. Same is true in the case of sexual harassment case; emails, social networking data, SMS will be interesting locations to search for evidences.

For example below file was recovered from Skype application. I have used same dd command to recover this file. The format for this file was .DAT. I have opened this file in a text editor (notepad in this case). You can see email addresses, Skype ids (one is mine ☺), chat records; everything in plain text. Same way you can get useful information from other applications like Facebook, Yahoo Messenger, Twitter etc. All application related data can be found at the following location: /data/data/com.application/; (Figure 10).

### Analyzing SQLite database files

SQLite database files are most interesting files for forensic investigators. One will get most critical information here, even username and passwords in some cases.

SQLite is a lightweight database (RDBMS) and used by almost all Smartphone OS like Android, iOS and Blackberry. SQLite files can be found at the following location:

/data/data/com.application\_name/databases

For example we want to see all SQLite files created and maintained by Facebook. Then we need to look at following location for db files:

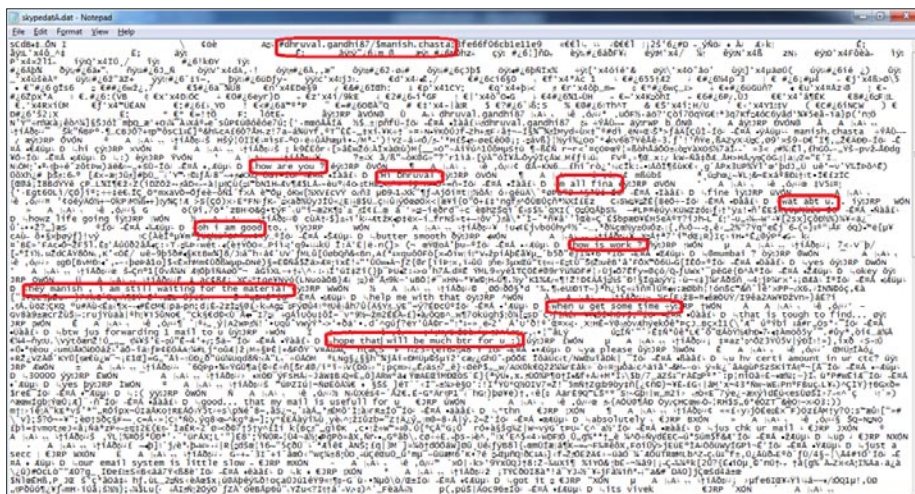


Figure 10. Skype File

/data/data/com.facebook.katana/  
databases

All SQLite files stored with .db format. I have copied a few sample .db files (from Facebook, email client etc) using dd command to explain analysis of SQLite database files.

To understand the concept, I will be using free version of Epilog tool. Epilog is a powerful tool for all kind of SQLite files.

Open a db file in Epilog tool, in this example we are opening

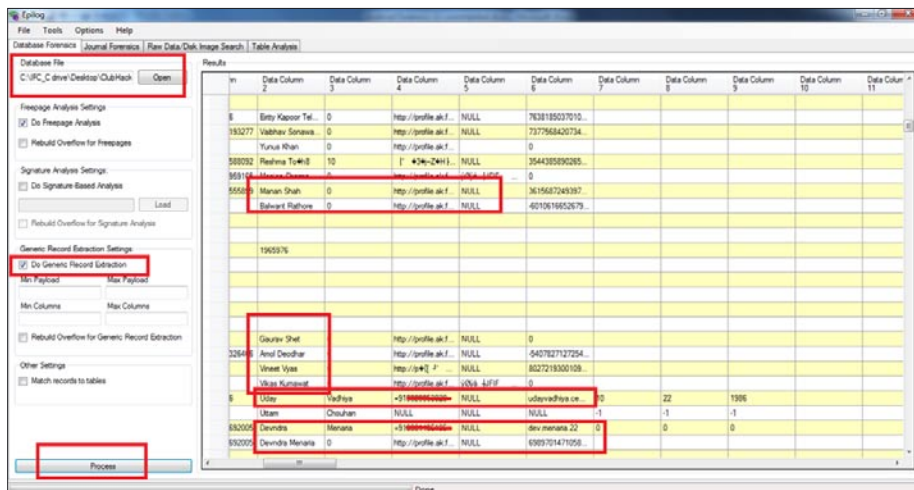


Figure 11. Epilog Tool fb.db file analysis

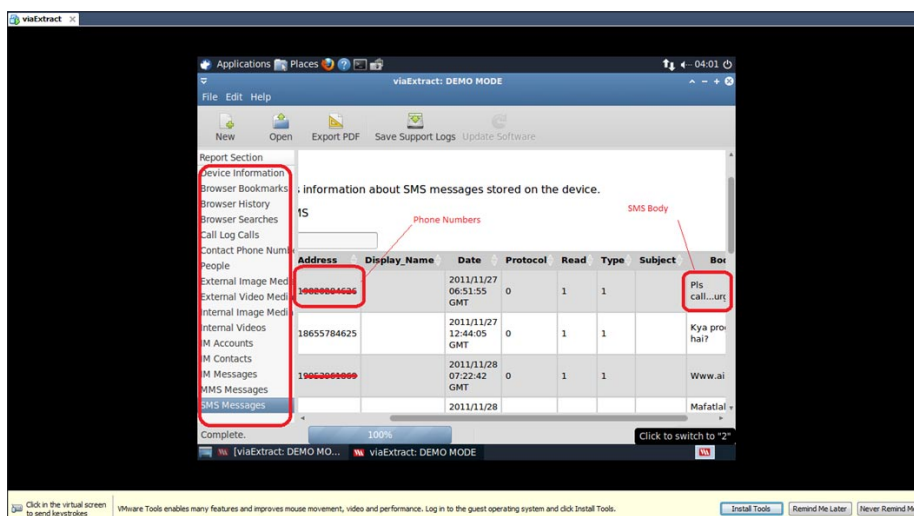


Figure 12. viaExtract Report

fb.db (Facebook db file). Check *Do Generic Record Extraction* checkbox and click on process (Figure 11).

You can observe in above screenshot, fb.db file contain some really useful information. In our case, we can see full names, email ids, phone numbers of the friends added in Facebook friendlist of the suspect. By opening the correct db file, we can even find all the chat logs, personal messages and other details. In some cases, you may even find username and password stored in a SQLite files.

## viaExtract Tool

There are a number of good forensic tools available in the market, out of them I found viaExtract tool to be very useful and easy to use for Android forensic. This tool is specially meant for Android forensic by viaForensic.

In this tool, you just need to connect the phone to the machine where viaExtract is installed. Phone should be in USB *debugging mode*. To make phone in USB *Debugging mode*, go to Settings-> Applications-> Development and select USB Debugging mode. Now you just need to click Next and tool will recover and analyze the device. As an output, you get final report

with all the useful information like Contacts, SMSes, IM records etc.

Figure 12 shows the HTML report from viaExtract tool, we can see all SMS details here.

## Note

Even viaExtract will write something on the device.

## Reporting Evidences

Reporting has to be done on case to case basis. There are different ways of reporting evidences in corporate cases and criminal cases. Reported evidences should be clear, give direct or indirect reference to the possible scenarios of crime.

In a criminal case, where we want to present evidences in the court of law, it is also required to map the findings with respective laws. In addition to evidences, it is also required to present *Chain of Custody*. Again reporting depends on country to country, as the Cyber Laws varies with geography.

## Conclusion

To summarize, analyzing Android for forensic purpose employs totally different techniques than the traditional forensics. It involves

heavy manual intelligence and interference. Maintaining integrity of primary evidences is also a challenge. There are tools available in the market for Android Forensics but still there are gaps to be filled and a lot to be done in this direction. After learning about forensic process, it will be a fun learning Anti-Forensic on Android device ☺ .

## MANISH CHASTA

*Manish Chasta is a CISSP, CHFI and Certified Digital Evidence Analyst, working with IndusFace Consulting (Mumbai) as Principal Consultant. He is having more than 5.5 years of experience in Information and Application security. He is currently managing team of security engineers and doing a vast research in Mobile Application Security. He is also handling prime customer accounts for the company. He has authored numerous security articles for ClubHack and Palisade. He has audited 200+ mobile and web-applications in the areas of Internet Banking, Core Banking (Flexcube), Finance, Healthcare, CRM, telecom and eCommerce. He has delivered trainings in the field of Digital Forensics, Application Security and Ethical Hacking to multiple clients.*

*Email id: chasta.manish@gmail.com*



**Classroom & Distance Learning Programs on  
Information Security & Ethical Hacking**

**Web Application Security**

**Network Security**

**Exploit Development**

**Reverse Engineering & Malware Analysis**

[www.innobuzz.in](http://www.innobuzz.in)  
[mag@innobuzz.in](mailto:mag@innobuzz.in)



[@innobuzzks](https://twitter.com/innobuzzks)



[/innobuzz](https://www.youtube.com/innobuzz)



[/innobuzz](https://www.facebook.com/innobuzz)



# Social Engineering

Ever walked into a shop planning on buying a product, but after talking to the salesman leaving with a more expensive product that has more options than you needed?

**E**ver listened to a politician and thought during his or her speech: *that's a pretty good idea*, but the next day you realize what all the downsides are to his or her statement?

And ever kept buying that one person drinks because you thought it would get you somewhere, but suddenly the person in question left and you didn't even get a phone number?

This happens all the time and it happens to everyone. Luckily the examples above are harmless situations (except for your wallet and self-esteem of course), mostly because the other person didn't have any malicious intentions.

*Malicious Intention: Performing actions to obtain confidential information.*

People that talk you into believing their story or leading you towards certain behavior are people who understand (knowingly or unknowingly) how humans interact. They understand how people make decisions. Especially when emotion plays a roll (money, political views and attraction are the causes for a certain emotion or a set of emotions). These day's we like to call this: *Social Engineering* (SE).

SE is a term that is most commonly used in the IT-Security sector. It seems that SE is a term that is something from the last few years. Often this term is directly associated with people that pretend to be someone else to manipulate and/or misuse people for personal benefit. I think this term is miss-interpreted in many situations. SE is a skill that we see all through history and the association between SE and malicious behavior is incorrect.

SE is much more than just tricking people into doing stuff for you or handing over confidential information.

In this article we will look further into SE. I'll explain what SE is and how it can be useful by everybody (not just for the bad guys).

Although SE is a massive topic, most of the information won't be useful for you when protecting yourself against these attacks – you don't need to be an expert on human behavior to identify SE attacks.

This article I will describe why the human brain is so vulnerable to lies and deceit. The article will describe a couple of very effective SE scenario's – which are driven by a malicious intention – that are used on a daily basis. This follow up will describe a few basic rules you can use for detecting malicious Social Engineers. Finally some references and material in case you are interested in SE.

## What is Social Engineering?

There are several descriptions for what Social Engineering and a Social Engineer is. A very good – and in my opinion complete description – is given in a recently released book (*Social Engineering: The Art of Human Hacking* by Christopher Handnag; 2011; ISBN: 978-0-470-63953-5) on SE by Christopher Handnag (lead developer of [www.social-engineer.org](http://www.social-engineer.org)).

In his book he writes:

*'SE is the art or better yet, science, of skillfully maneuvering human beings to take action in some aspect of their lives that may or may not be in the "target's" best interest'*

The above description shows us that SE is much more than manipulating people. One of the key aspects you need to know before you are able to manipulate human behavior is to understand how people communicate. A

good Social Engineer is a person who understands human interaction and behavior and knows how to maneuver people into taking actions. Suddenly the term SE is much more than *pretending to be someone else to manipulate and/or misuse people for personal benefit*'

Some good examples of social engineers without any malicious intentions are

- Politicians (getting their message across with the intention of gaining more support)
- People in marketing (selling their product to the largest amount of people)
- Sales men and woman (maneuvering you into buying the right product)
- Children (getting what they want)
- Doctors (bringing a message across to advice on the current situation in the most understanding and tactful way)
- Lawyers (Obtaining the information they need to defend their client)

All these people know how to maneuver people in certain ways to get what they want.

In all of these cases the intention isn't *malicious* but simply their job or normal human behavior. But what if malicious intentions are part of the interaction? Why do people fall for simple lies? What makes humans so vulnerable against SE attacks?

### Vulnerabilities in the human brain

Now you understand what Social Engineeris we can continue to understanding why people are vulnerable to deception.

The human mind is a very interesting system that works in mysterious ways. Let me give you an example. If I say: Don't think about a pink elephant. What just popped in your head? I bet it is a pink elephant. But I just told you not to think about it! This is a simple example (completely useless though..) for a malicious attacker (unless you know someone who will give you all their money or deepest secrets when they think of a pink elephant). But it shows that the human brain processes information even if a different instruction is given. Of course the human brain isn't able to allow direct command injection without being in very unique situation. The human brain is way too complex for that. So how does it work?

As I described before, the human brain processes a lot of information. Most of the time you are unaware of how much information is being processed. Good examples of this process is often used by magicians, illusionists, mentalists or other forms of trickery (e.g. Derren Brown) tricks your brain into believing something happened.

The inability to process all the information that is going on around you is very normal for one simple reason: there is simply too much information.

A lot of techniques are used to manipulate people. I'll describe the four most commonly used attacks by Social Engineers.

### Technical Attacks

Technical attacks are very common. This technique is mostly used for spreading malicious software. Typical to this attack method is that there is no direct interpersonal contact with the victim (e.g. email, pop-up, website). Most of the time this method is used on large scale with a small to medium impact (also known as the shotgun approach). Most of the time the goal is simply to get the user to run a specific piece of software giving the attacker access to the system or obtaining sensitive (personal) data. We see examples of this type of attack almost daily. Both in a mass and targeted form.

### Ego Attacks

This is a way of manipulating someone by appealing to their ego. This is an attack that takes skill. Appealing to someone's ego too much and people get suspicious.

Too less and you're missing your goal. This attack can be performed in two ways, either the ego is lowered or lifted by the attacker. Lowering someone's ego/self-esteem can be done by giving someone the feeling they will have to prove themselves. Of course your goal here is to obtain classified information while your victim has a sudden need to prove themselves to you. Also playing on the obligated feeling we humans naturally have to help somebody can be very useful.

Most of the time the victim realizes too late that classified information has been given because in most ego attacks emotion played a big role. Triggering a victim into having a specified emotion can be a very powerful situation for a Social Engineer because most of the time emotions are overrule logic. There are numerous examples where classified information. To increase the effectiveness of this attack, try to find out what weakens the person emotionally. Examples: alcohol, pretty woman, money, etc..

### Sympathy Attacks

A sympathy attack plays even more on the victims need to help people. Not a lot of people like to be responsible for your problems. So people will try to keep you out of trouble. Urgency is a very important part of this attack. You don't want to give the target the opportunity to *check things out*. Making the target feel sorry for you can also be a very powerful angle for this attack.

Example: I have seen people from the support desk giving *administrator* passwords to users that were in need of help. The person in question was very good

(unknowingly probably) in triggering and sympathy situation and the support desk employee was very trustworthy.

## Intimidation Attacks

Finally *intimidation* is a good attack in combination with the right attitude. Projecting an influential personality (e.g. authority figure, law enforcement) can create an immediate cooperative response from your victim. Threats (e.g. job sanctions, criminal charges) when resistance is given, are very useful, because this again creates the feeling of haste and eliminates the need of *checking things out*.

Example: Letting your target believe you are a high ranking police officer with an urgent need for critical information regarding an ongoing investigation generates an almost immediate helpful response without being asked any questions.

Remember: people working for a law enforcement agency always have to be able to identify themselves. You are always allowed to request this identification. Also, if someone is from a law enforcement agency, this does not entitle them to know everything.

Now that you are aware of the basic attacks Social engineers use it is important to know how to defend yourself against malicious Social Engineers. But before that you'll first have to understand a little bit about lying.

## Why are we so Bad in Detecting Lies and Deceit?

You'll have to understand: why do we lie? Even better, why are we so unskilled in detecting lies? These days and all through history a lot of research has been done regarding lies. Out of these studies comes forth that everybody lies. People lie to make themselves look better or to mask their opinion to prevent hurting someone's feelings. On top of that, we also lie to ourselves. We all want to be happy. To do this we have to ignore all the suffering around us (e.g. war, hunger, murder and theft). So naturally the human brain lies, and rejects the truth in specific occasions to ensure happiness. This mechanism is what we call a 'truth bias' which results in the inability or in other words, unskilled in spotting and detecting lies without proper training (even with proper training detecting lies is not an exact science).

## Preventing (malicious) SE attacks

The most important parts about preventing SE attacks is simply to pay attention and stay calm. As you read before, SE attacks often go hand in hand with putting you on the spot where you are tricked into triggering an emotion you dislike or very much enjoy.

People will always lie and try to trick you. Failing in detecting this is not *being stupid* but being human.

The most important part of failing in detecting SE attacks is making your company aware of the fact that a successful SE attack has been performed, what data has been obtained, what the risks are this SE attack created, what to do about it and how to prevent it in the future (this is the exact same process in case of a technical security incident). The main thing you can do about preventing SE attacks is education!

But of course, I will leave you with some basic rules you can use in your day to day job:

## Who are you talking to?

Knowing who you are communicating with is something you cannot check easily. Systems should be in use to make this easier. Almost every company has an intranet page with a list (most of the time even with photos) of current employees. This list can be very useful if the person on the telephone is indeed an employee. You can for example say that you need to check up on that and that you will call the person in question back in a couple of minutes (everybody can wait a couple of minutes). People who will not wait should raise a flag of interest. Also setting up control questions can be very useful. Masking these questions in the conversation makes it even better. Maybe you can SE the Social Engineer.

## How valuable is the information in question?

Knowing how valuable the information is your working with is half the job. If you are aware of the fact that you are talking about information regarding critical systems bells and red lights should be ringing and flashing. Situations where critical information is at stake should make you ask more questions than usual towards the person you are communicating with. People with good intentions will always understand why you ask these questions. If they ask why the paranoia or suspicion just explain that the questions are there for security reasons. A legitimate person won't and can't argue with that. The value of the information and the intensity of checking the credibility of the party that requests the information goes hand in hand.

As you can see the two rules we are using are basically the same rules we use in Internet technology. We want to make sure that we are talking with the person who is who they say they are (Public-Private key encryption and digital signing). If we are exchanging more valuable information more of the technologies in question are being used at the same time.

## Want to Learn More?

Social-Engineering is a big topic with a lot of ground to cover. *Neuro-linguistic programming* (NLP), micro expressions and hypnosis. I can keep going with topics that could be useful. But this is not a book. So if you are



interested, here is a list with material you should take a look at:

- We have to start with the book Social-Engineering, hacking the human mind. This is a great book for people who know nothing about SE or people who already read a lot about this topic. It offers a wide range of topics and go very far into detail.
- *Ghost in the wires* is the latest book by Kevin Mitnick. This book gives you a great insight in on one of the world most famous social engineers. You get a great insight in what kind of consequences a SE attack can have on you company.
- Some other topics that are worth looking into are:
  - Body language (Allan & Barbara have some great books about this) ii. Smalltalk, no not the programming language. Actually making conversation with strangers is a great skill for a social engineer. Your quality is in the amount of time you spend while building report. If you can do this quick you already have a great skill you can use. It doesn't matter what your profession is.
  - Learn to love communication. There is nothing better than having fun while practicing your skills on random people.
  - Look at children. Children are great social engineers. They are masters in manipulating their parents and in lying. Try to learn from them because the signals adults show are still the same, just more subtle.

### Conclusion

As you hopefully remember I said in the beginning that SE is much more than working with malicious intent. The main part of this article was focused on malicious intention.

The main reason for this is because I want you to be aware of the harm people can do simply by giving you a call. I hope you understand that SE is a skill that you can use in your day to day life. Getting people to perform an action, if it is working on a project,

making someone feel good or simply having a nice conversation simply being aware of the conversation and what is happening is incredibly important. This makes people feel comfortable and relax around you. So good communication skills will help you (especially if you don't have any malicious intention) in work and private life.

---

### RUBEN THIJSSSEN

*My name is Ruben Thijssen and I am a Software Developer for Amrap.org in Sydney Australia. At Amrap.org I'm responsible for the development of web applications that support the promotion of Australian music. Next to my job I have a deep interest in human behavior and IT-Security, both the technical and the organization side. My twitter account (@rubenthijssen) and website <http://www.damnsecure.org>.*

# Join

## hakin9 team!



If you would like to help our team in creating hakin9 magazine you can join our authors or betatesters today!

All you need to do, is to send an email to:

**[editors@hakin9.org](mailto:editors@hakin9.org)**

and give us a brief description of your field of interest.

**We look forward to hearing from you!**

# A Bits' Life

Have you ever wondered what makes all these devices around you alive? I might have to give you a bit of bad news – this is not a black magic of any kind, neither any supernatural powers, not even the Jedi Force; it is just a simple set of interesting ideas, well described with a language called science and technology.

As some of the most stubborn representatives of our species noticed long ago that any physical phenomenon has some specific attributes that can be described through the use of mathematics and abstract modeling. So if the abstract information is related somehow with our physical reality, then it should also be possible for information to interact with a real world. Do you think this is possible?

Many great minds across ages have tried to accomplish this goal – to create machines that could help people to perform boring and time consuming tasks. Tasks can be automated with mechanical manipulation of physical objects and controlled by information processing. It was not so long ago, however, when the first computer was built and it required great knowledge and technical skills to use. But the first digital machines had dramatic impact on our civilization, so let's take a look at how these machines work...

## Zeros and Ones

All digital equipment use numeric data representation. This means that any information must be stored in its memory as a set of numbers. Changing these numbers changes the data. Numbers can be represented in many different forms – for instance we use Arabic or Roman notation in our everyday life – and it is important to remember that the main goal is to symbolize some abstract meaning.

Symbols and abstract models on the other hand should be as simple as it's possible in implementation and use, and the more intuitive the better. That is why selecting the proper numeric system was so important. It allowed for the construction of information processing machines, and a little later their revolutionary miniaturization.

Please keep in mind that today extremely sophisticated devices can fit easily into pocket, what ten or twenty years ago was still only a science fiction dream.

In a binary system, the rules of logic apply. At first it is hard to imagine that everything in a digital world is based on the good old logic of truth and false, but as we will soon understand, it is powerful enough to accomplish even the most sophisticated tasks. The smallest portion of information is called a BIT, that can be set to 0 (false) or 1 (true), nothing more. Combining bits together into more advanced binary data can represent increasingly complex information. Any transformation on this type of data must conform to the Boolean Logic principles. There are four logic functions; NOT (negation), AND (multiplication), OR (sum) and EXOR (sum modulo 2). These can be arranged into a combination of operations to accomplish any requested mathematical operation.

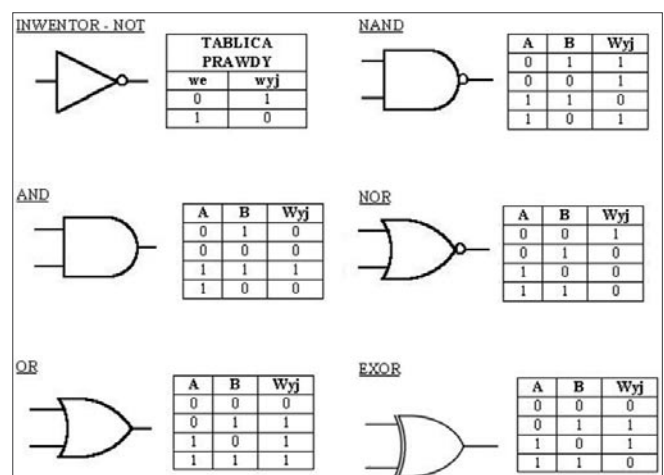


Figure 1. Basic logic gates and their function

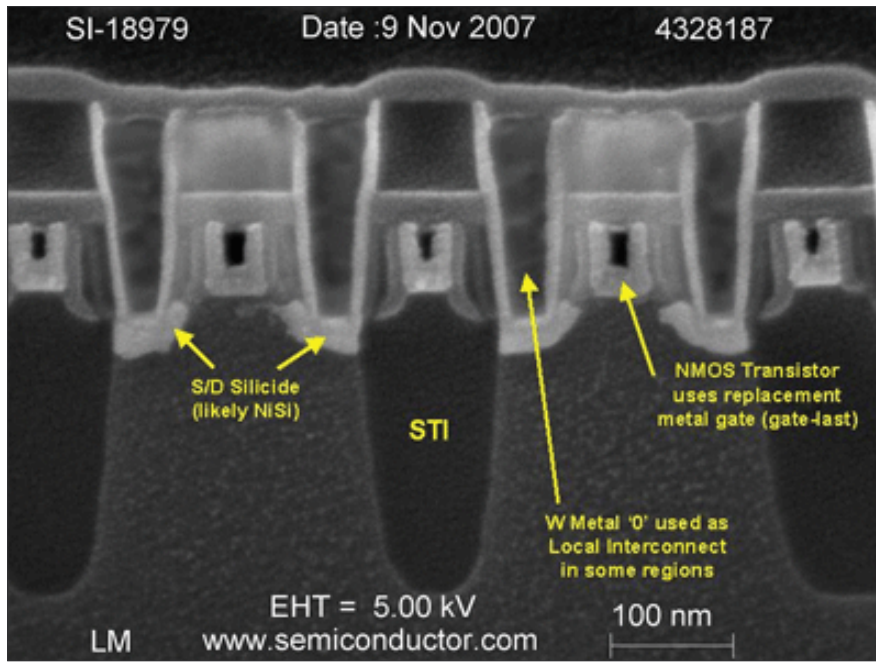


Figure 2. FET structure

At this point you are probably suspecting that all of these four basic logic functions require only very simple machines to operate. You are right! Each one of them has assigned a device called a *logic gate*, that is an electronic component that performs an operation on electrical signals. Interconnecting hundreds of thousand or even millions of these simple devices into small integrated circuits that can fit into pocket is only a matter of knowledge, resources and determination to create new technology...

**Logic gates**

You may be wondering how physical Logic Gates are built? They use conventional semiconductor electronic elements. The so called *active elements* that are used to interact with signals are called transistors. Signals are usually in a form of current flowing through (semi)conductive elements, or voltage measured between element pins. All elements can be miniaturized and sealed in a small outline package of an integrated circuit. Nowadays a CMOS (Complementary MOS) technology is dominating semiconductor industry,

mainly because of its simplicity and flexibility at low cost – only two MOS (*Metal-Oxide Semiconductor*) transistors of opposite type are required to build simplest NOR gate (this is also where its name comes from).

Transistors cannot be further divided into smaller elements, so this is the most basic three pin active element in electronics. Two of its pins are used to flow the signal, and the third one can control the signal. There are two main families of transistors; amplification and keying, and these perform similar functions. One of them is called BJT (*Bipolar Junction Transistor*) and the second one is FET (*Field Effect Transistor*). MOS transistors belong to the FET family and they are called MOS-FET with type N and P. N-MOS-

FET (or simply NMOS) is turned on when there is a *true* condition (also called *high logic state*, or simply *1*) at its input. P-MOS is turned on when there is a *false* condition (also called *low logic state* or simply *0*) at its input. It is important to note that *false* means no voltage or no current flow, while *true* means voltage or current flow. This way the virtual world touches the real world – by controlling the single bunch of electrons that represent our information...

Even one N-MOS and one P-MOS can build the inverter – which is nothing more than a device that performs a NOT function. Four transistors can build AND, OR, XOR logic gates respectively. A Flip-Flop requires more transistors, but is essential to *remember* the logic values and it is placed usually at the input and output of the bigger logic device to separate internal calculations progress and results from the outside world.

**Devices**

Now that we have the logic gates and flip-flops, we can dare to build more sophisticated devices, such as functional blocks, arithmetical units, control units, communications blocks or even whole microprocessor

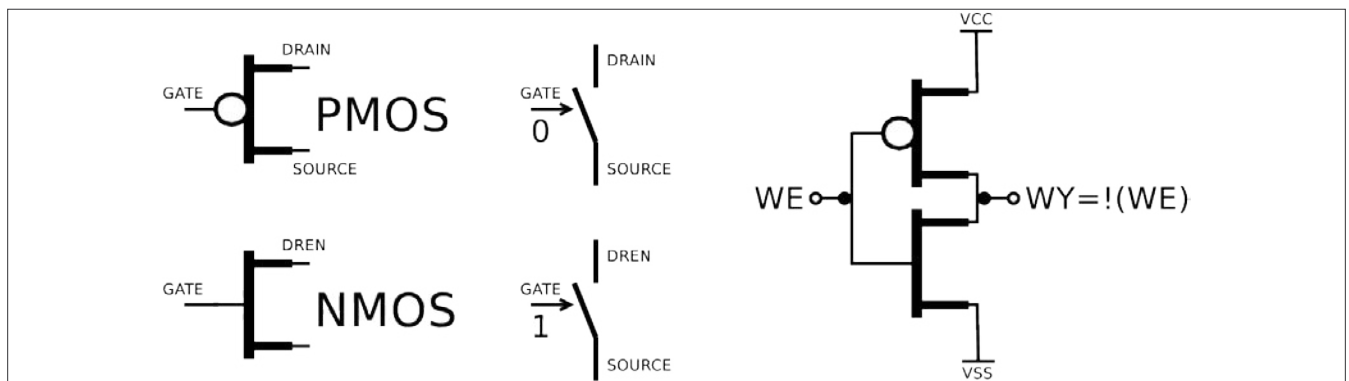


Figure 3. Schematics and the work principle of the CMOS inverter

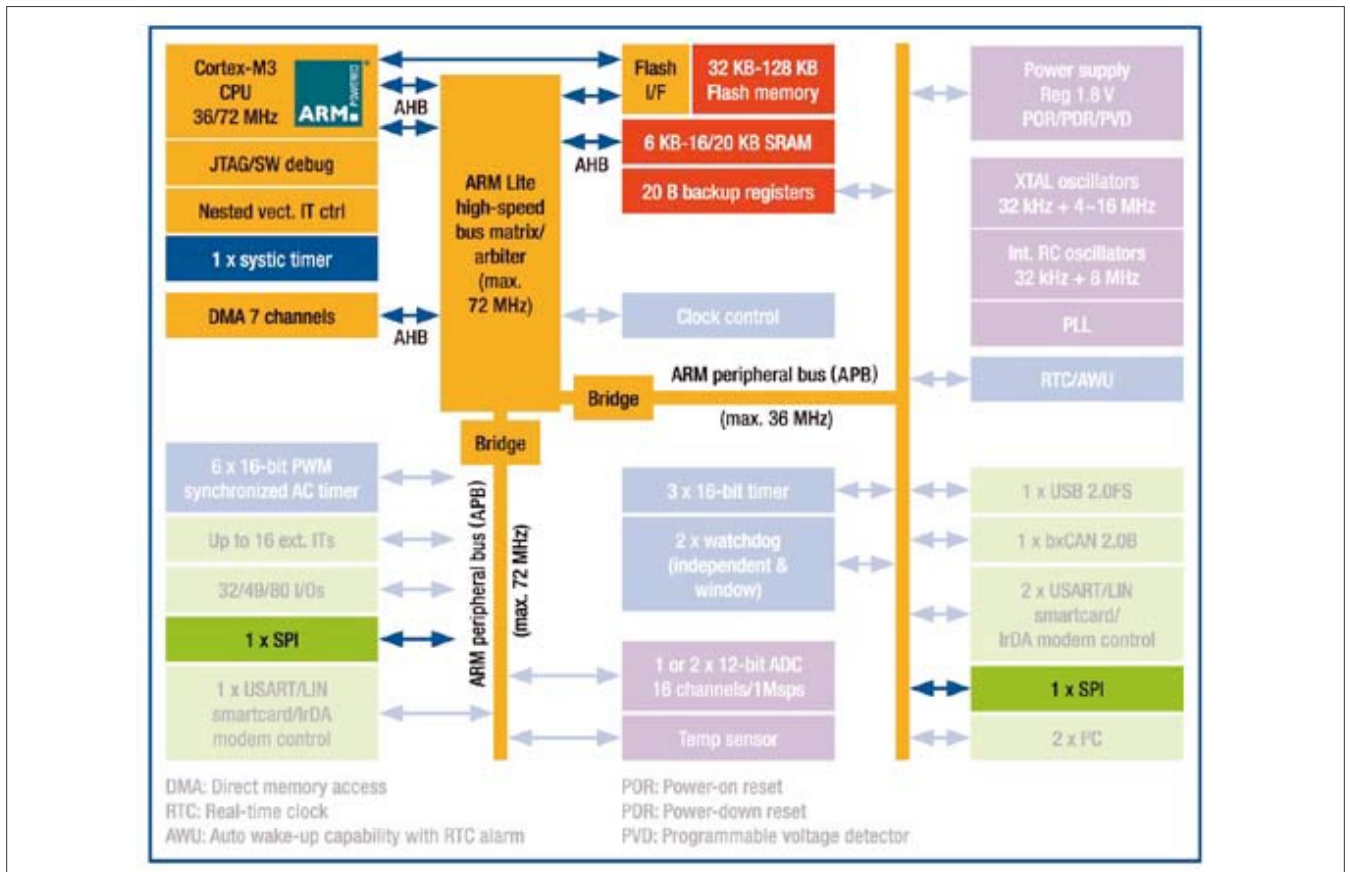


Figure 4. An example architecture of the ARM CORTEX microcontroller

systems. At first, when the semiconductor technology was young, these blocks were manufactured as small outline *Integrated Circuits* (IC) and then mounted on a bigger board with other elements. Boards were connected together and then cased to become a working device.

Because the technology is still being developed by the greatest minds across the globe, the basic semiconductor elements are becoming smaller and smaller. Nowadays, the average IC contains millions of transistors and it is possible to fit almost the entire functional device inside a single chip using SoC (*System-on-Chip*) technology. It is only a matter of imagination to speculate what further miniaturization can bring to us...

### The Limits

On the other hand you be asking yourself if there are any boundaries of the miniaturization – how small can our devices can be – is it really possible to create memory with unlimited capacity? There are in fact some limitations and they are mainly constrained by the laws of physics of the world that we live in. We are all made of atoms, waves, quarks, strings, or anything else that might be proven and used as a working theory of physics applied to the field of engineering. Theoretically at the most basic level (the atom) the operations cannot be changed, unless the laws of physics are changed :-)

### The Architecture

Modern microprocessor based devices are manufactured by interconnecting the various pre-designed physical elements (hardware) with the software controlling the device (firmware). Hardware is designed by the engineers and the software is written

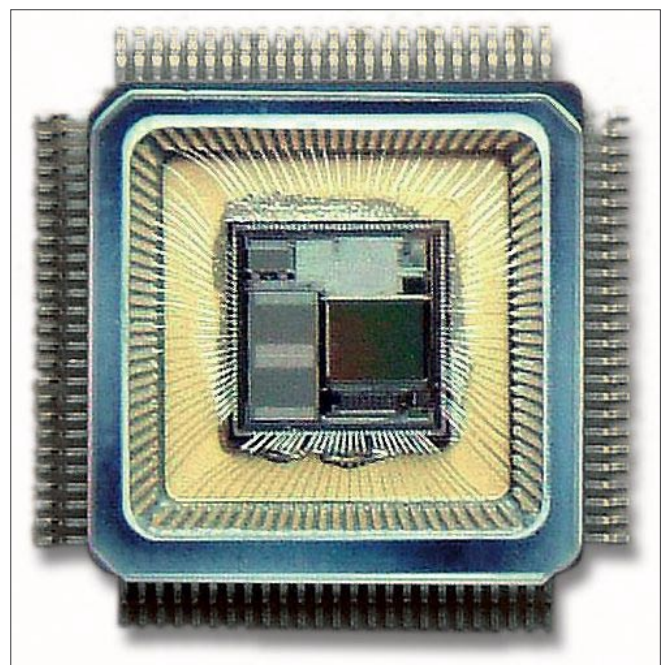


Figure 5. Integrated circuit set on a casing

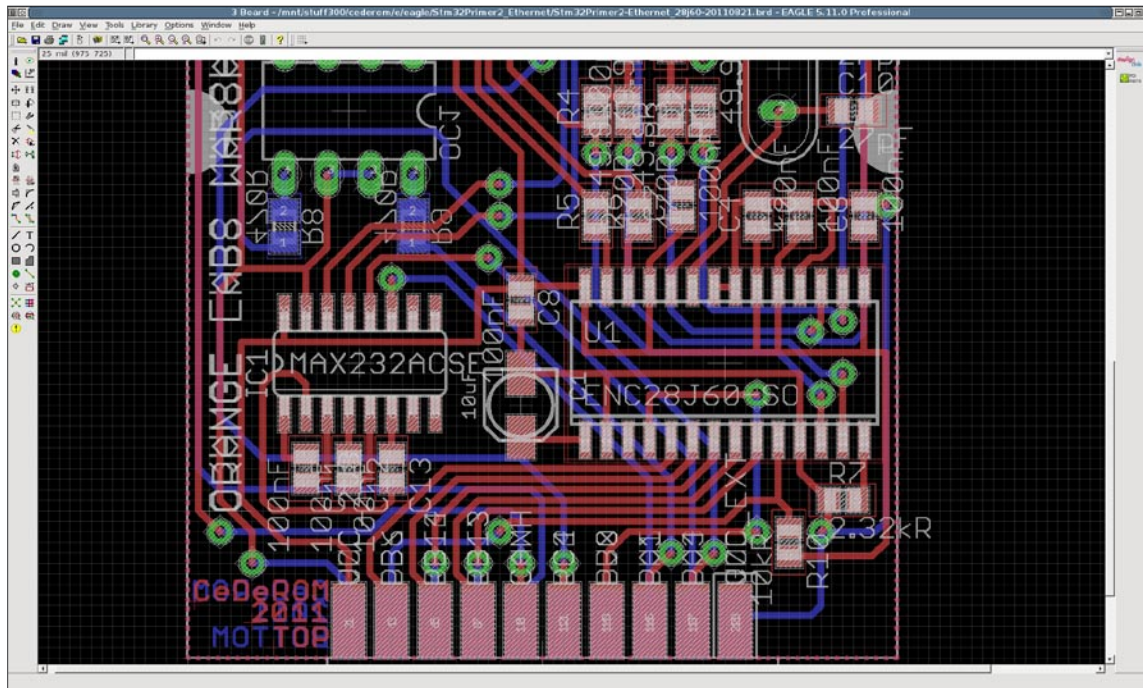


Figure 6. Computer Aided Design simplifies design and verification process

by the programmers. These are usually well organized teams of highly skilled individuals using professional (and very expensive) tools that assist their work, such as CAD (*Computer Aided Design*) or SDK (*Software Development Kit*), sometimes even dedicated Operating Systems (OS).

This approach, where virtual bits control the behavior of physical elements is now a reality of our digital world and most of us are surrounded by devices and embedded systems in our everyday life.

### How intelligent are zeros and ones?

Even most advanced devices are not intelligent by themselves (for now) and cannot make decisions on their own. The CPU (*Central Processing Unit*) is a brain of the digital system, but it can only execute instructions that are stored in memory. The set of instructions is a program, written by a software developer; perform some specific task by directing the work of a piece of hardware – information gathering and processing, performing calculations, decision making, etc. Sending zeros and ones from one place of hardware to another in a controlled manner, produces a physical information flow that makes the system work, with no struggle, no fear, (hopefully) in the same exact manner.

Electronic components do not spend time wondering or guessing. The program will be executed exactly the way that the programmer has written it, so the programmer is responsible for handling any possible situation that may occur. This is very important, as humans often forget about the automatic actions caused by instinct; many situations that seem to be obvious for humans are completely out of reach for the unaware

machines. Because machines have no instinct, they will blindly execute any command found in their program memory. Using this logic it is then possible to find a weak spot in the device or the program code to create a situation where an unhandled exception will change the program execution or make a system crash resulting in device behavior totally different than expected. The device cannot defend itself in a way other than it has been instructed previously with knowledge of what can happen and how to react. This is why, at best, any device is as intelligent as its creator.

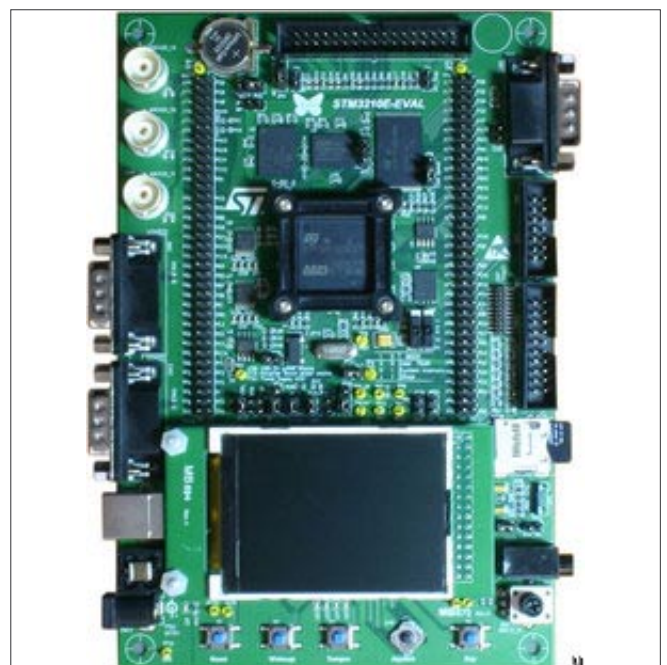


Figure 7. Ready to program electronic hardware

## Errors

It is human nature to make mistakes, so it is extremely important to remember this little truth during the design process. Every detail of an initial idea should be well considered, planned and verified. Starting development with no goal, planning ad-hoc, or changing ideas in the middle of the project – none of these will bring good results. A bad idea will follow you until the end of its days, but a good idea brings constant satisfaction and possible further solutions if properly developed and implemented. For example, let's take a look at the binary system – not only was it good enough to allow computers to work, but also brought a dynamic growth of the semiconductor industry and the information technology that has revolutionized our civilization.

This is why errors are so important; simply put they can ruin everything and must be taken into account during all phases of the project, from design to final validation. Errors can not only make the whole project unreliable, but also cause serious security issues leading to information leakage, *denial of service* (DoS), identity manipulation, and other nasty issues. If a implementation bug is the problem, then it can be fixed, but in case of design flaw, it may be necessary to replace a component or if the bug is bad enough, the entire system. There are many people around the world that search for this kind of flaws in all nature of software/firmware. Exploited bugs and vulnerabilities allow attackers to obtain classified information. While the motivation of these individuals differ, the goal is the same, obtain information that someone else wants kept secure.

## Control

How can we master this huge world of tiny bits? Why there are so many traps at each and every step? The reason is that the information systems are becoming even more complex every day. On the one hand we want them to be smaller, cheaper, faster and more reliable, but in the same breath we demand increasing levels of functionality. The new functionality can be very useful for end-users, but also bring new challenges for the developers. Today's mobile digital equipment contains more computational power and memory than the old supercomputers. The modern microchip is not only a CPU, but also all possible performs communications, processing and storage. All this is becoming hard to control by an average end user, not to mention the developer teams.

## Standards

Without a common standard and project hierarchy, none of the modern devices could exist and work correctly. Just as we have used a few transistors to build a logic gates, logic gates can be used to build complex functional blocks, and these blocks can again create even more

complex blocks. Blocks are connected with each other with a bus of some standard, so information flow can take place – serially (one bit after another), or in parallel few bits at time, according to the designed protocol.

There are groups of engineers designing these blocks in a standard way, so other groups can reuse them in their designs. It is not obvious for an average user that his/her device was designed by more high skilled individuals and cost thousands more than the final cost of the product. This build process can be described as a layered cake, where at the bottom there is a silicon mask for an physical IC, then there are transistors, then logic gates, then functional blocks, microprocessors, and some abstract structures using mathematical modeling. All this conforms to a standard, so at any level of abstraction one person can understand other person's work. For instance some information can be considered a set of bits, a memory location, or high-level programming language *int* type variable written in C or Python *class* that implements simulation or processing.

## Careful birth

Starting from the first idea for a device, through a design sketch on a paper, then to computer design software, prototyping a PCB or even IC mask, the exciting first build, slow verification, software development and final testing – throughout the entire process the quality of work must be closely monitored.

There are professional tools for this purpose that helps in verifying the project at different stages. For instance an Integrated Circuit mask usually contains some additional elements that only exist to determine that the production process was successful. The first run is performed in a laboratory environment to strictly test the electrical characteristic and make sure that the device will function as intended. Software development is coordinated and then modules are verified by special test patterns. All this additional work is added to avoid mistakes – by human error or technology process inaccuracy – that could result in construction errors and device malfunction.

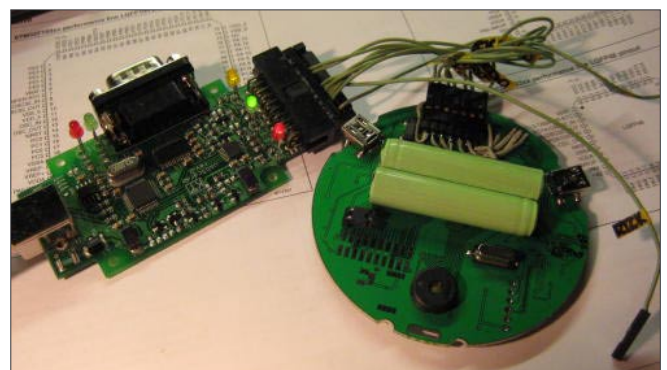


Figure 8. First steps of a new electronic device

### Testing Tools

It would be difficult if not impossible to perform all of the necessary testing by hand. This is why specialty tools were created to assist in this process, and to make work even simpler some of these tools are being used by other tools. One of the well-known tools is the IEEE1149.1 testing standard, also known as JTAG (*Joint Test Action Group*). It defines some of the standard electrical connectors used by the device (TDI, TDO, TMS, TCK signals) that allow sending bit streams to look into internal registry map, memory contents, functional blocks status, even program execution, with no other physical interference. It is very useful to test and service a system with no need to unmount or disassemble components. JTAG logical signals are generated by computer software that is connected by a dedicated physical JTAG interface attached between a computer and device being tested. Electrical signals are then passed to a TDI (*Test Data Input*) pin, to flow the command over the internal structures, and produce the result on the TDO (*Test Data Output*) pin. Test Data is a special variation of zeros and ones grouped in a series of commands that target device can execute with use of the ICE block (*In Circuit Emulator*). JTAG standard must be supported by the examined device's internal components and the dedicated computer software being used. Elements on the circuit board can be connected one after another creating more progressively more complex test chains. More and more new devices conform to JTAG standard, which promotes test standardization, which can be very useful by increasing yield, lowering the final product price and dramatically shortening the time-to-market period.

A relatively new alternative to the JTAG standard is

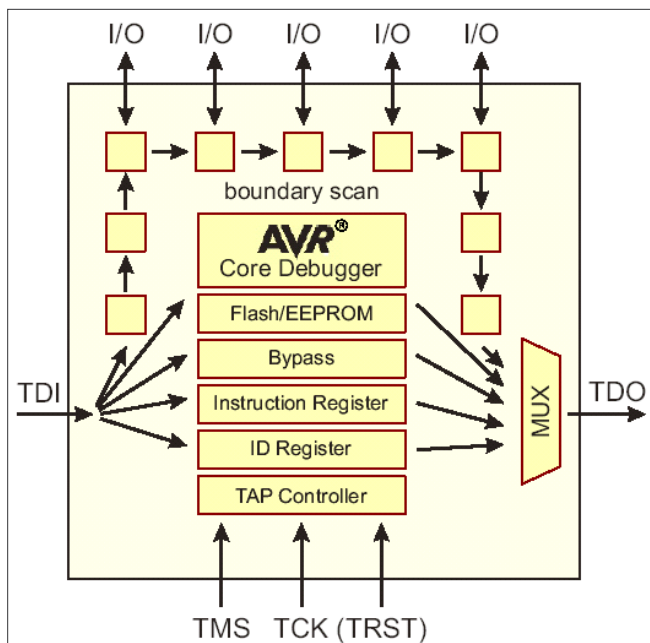


Figure 9. The JTAG organization in AVR microcontroller family

the SWD (*Serial Wire Debug*) that was introduced and implemented in their ARM–Cortex devices. SWD allows access to the internals of the micro-controller system in a packet based request–acknowledge–response manner, which is different than JTAG state machine. SWD is electrically equivalent with the JTAG, but it uses only two data pins (instead of 6), so it is possible to make it part of the service connector. Almost every data cell of a device can be accessed and verified with use of dedicated low–level access tools of this kind. Unfortunately, this seemingly perfect situation has its disadvantages. Perhaps the biggest problem is that after production this test interface is left functional and freely available for an end-user. Even is the connector is physically disabled or hidden they can be found with some effort and patience, creating a priceless security flaw in the system for an attacker.

### Practice

In the IT Security Section (Marcin KUZIA) at Middleware and Application Platform Services Division (Sebastian GRABOWSKI) of Orange Labs Warsaw/ Poland (Krzysztof KOZŁOWSKI, Daniel PIECHOCKI, Andrzej KOWALSKI), a small group of specialists work on improving computer and telecommunication systems security. We develop and verify security of various devices, using all methods and tools available. We constantly learn new techniques and invent new methods. New products are new challenges for us.

One of our current research tasks is to work out the full potential of a JTAG interface, to perform functional analysis of the firmware program code and look for



Figure 10. Most advanced computer users, developers and designers, so called hackers, are interested mostly in principles of work and weak points of their computer systems



**Figure 11.** Practical low-level access to embedded system prototype hardware using JTAG/SWD debug port

potential vulnerabilities left by the manufacturer, such as information leak and attack susceptibility – their exact source, mechanism, and impact.

We support Open-Source; we have created LibSWD – a first in the world open framework to operate on Serial Wire Debug bus implemented in ARM-Cortex cores that can be integrated with existing tools such as UrJTAG and OpenOCD (work in progress). We cooperate closely with manufacturers, developers and engineers of computer and telecommunication systems in order to verify the solutions before they hit the market. All of this is to protect our clients and make the network a better place.

### To bit or not to bit

The digital world of Bits is greater and more complex than it might seem at first glance. All this is possible due to the binary system – zeroes and ones – a brilliant idea that enabled the existence of many different computer systems and dynamic development of information technologies. Machines make our lives easier and do the most boring or time consuming tasks for us. However, they are not yet as intelligent and autonomous as their creators, so the applications are still limited. On the other hand the digital world gives us an opportunity to develop our skills and create better life for all. We can use each one of the bits just as we like, breaking barriers of our

### TOMASZ BOLESŁAW CEDRO

*Orange Labs Warsaw (Polish Telecom R&D). Tomasz Bolesław CEDRO, born 23 December 1982 Warsaw/Poland. Finished Electronic Technical High School in Kielce, BSc MSc at Warsaw University of Technology. Senior R&D Specialist at Orange Labs Warsaw (TP R&D). Interests: electronics, biocybernetics, embedded systems, Unix/BSD, demoscene, extreme sports, diamond way buddhism.*



### References

- <http://www.google.com/>,
- <http://www.wikipedia.org/>,
- <http://www.ieee.org/>,
- <http://www.itu.int/>,
- <http://www.3gpp.org/>,
- <http://www.tp.pl>,
- <http://www.orange.com>,
- <http://www.elektroda.pl/>,
- <http://www.gnu.org/>,
- <http://www.freebsd.org/>,
- <http://www.cadsoftusa.com/>,
- <http://www.cadence.com/>,
- <http://www.altium.com/>,
- <http://www.intel.com/>,
- <http://www.atmel.com/>,
- <http://www.st.com/>,
- <http://www.ti.com/>,
- <http://www.maximic.com/>,
- <http://www.infineon.com/>,
- <http://www.nxp.com/>,
- <http://www.analog.com/>,
- <http://www.elka.pw.edu.pl/>,
- <http://www.imio.pw.edu.pl/>,
- <http://www.ise.pw.edu.pl/>,
- <http://www.ire.pw.edu.pl/>,
- <http://www.edw.com.pl/>,
- <http://www.ep.com.pl/>,
- <http://www.wnt.pl/>,
- <http://mikom.pwn.pl/>,
- <http://www.rm.com.pl/>,
- <http://urjtag.sf.net/>,
- <http://openocd.sf.net/>,
- <http://libswd.sf.net/>,
- <http://stm32primer2swd.sf.net/>,
- <http://hackaday.com/>.

imagination, just by pressing a key. We can and should learn from their inventors – a smart approach, great imagination, humility and persistence in creating new solutions. The way we think of a problem, how we define a task – all this has tremendous impact on the final result. The better the approach, the better the solution. Well, we might not invent a new zero and one again, but a world of bits is a very flexible material, that if properly used can realize an unlimited number of brilliant ideas. Bits are here to serve us, with no fear, no effort, always the same exact, good way.

### MARCIN ARMAND KUZIA (MCSE, CISSP)

*Head of IT Network & Security Skill Center in Orange Labs Warsaw. Conference, seminars, lectures and cooperation animator with industrial, academic and research sites in Poland and Europe. Interested in intelligent buildings and independent energy sources.*

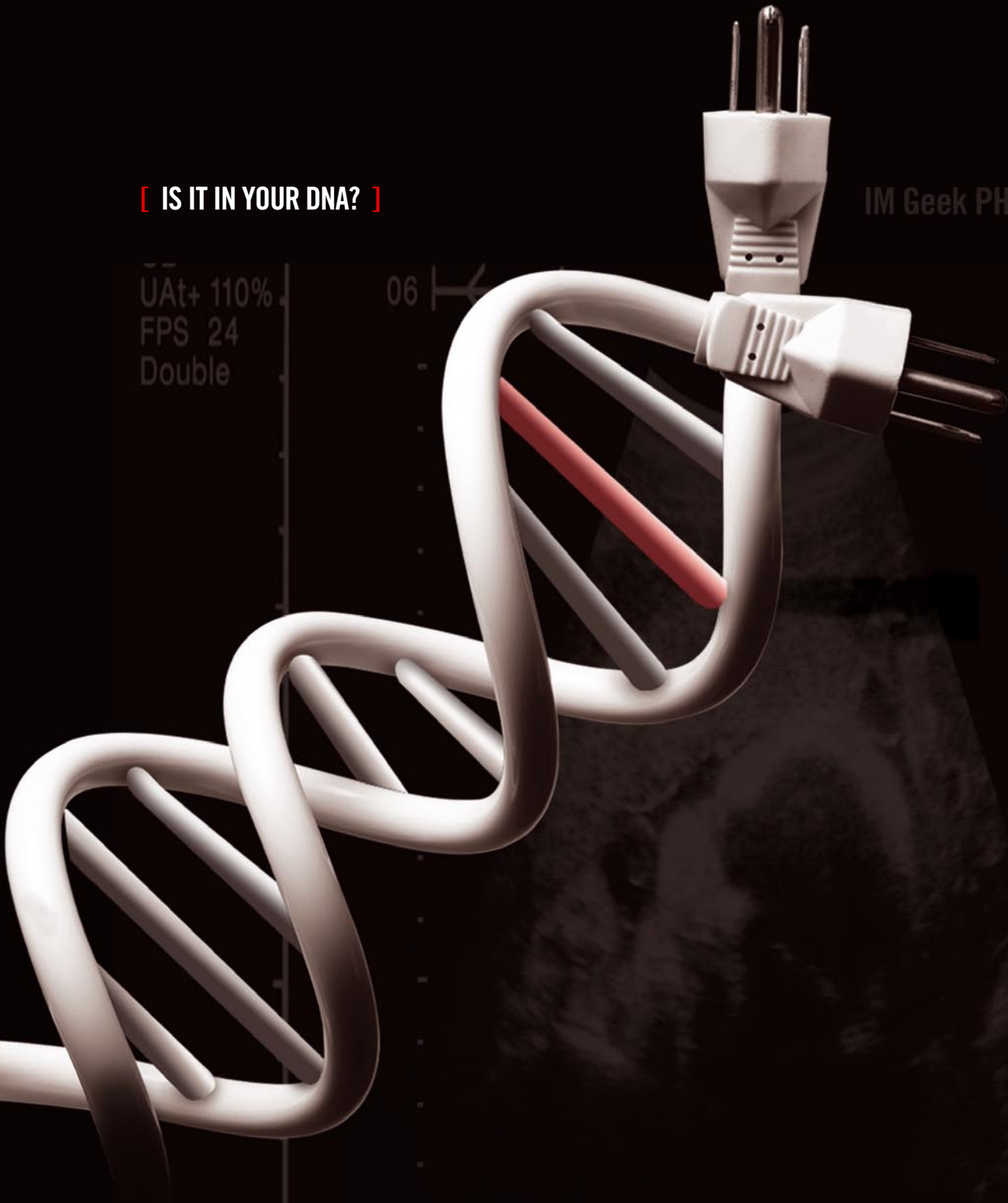




[ IS IT IN YOUR DNA? ]

IM Geek PH: 877.UAT.GEEK

UAT+ 110%  
FPS 24  
Double



[ 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



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

[www.uat.edu](http://www.uat.edu) > 877.UAT.GEEK

## Protect your important data before it is too late!

---



No-one likes to consider the worst case scenario, but are you prepared for a loss of all computer assets?

Computers are easily replaced, but your critical data and files are something money can't buy.

zebNet offers powerful, easy-to-use and leading backup solutions for all major web browsers and email clients which are designed to protect you as much as possible.

With a backup solution from zebNet you will always be protected from the worst case scenario at an affordable price, starting at just \$9.99

Visit [www.zebnet.us](http://www.zebnet.us) to be protected!

### Highlighted features at a glance:

- Fast and reliable backup and recovery
- Self-restoring backup files
- Backup reserve copies
- Backup to any FTP server
- Scheduled backups on a regular basis
- Data migration between different computers
- Support for portable editions of your web browser/email client
- Create a portable edition of your web browser/email client
- And many more



### Exclusive limited-time offer for you as a Hakin9 subscriber:

Get a **50% DISCOUNT** off any zebNet backup solution you wish by simply entering the discount code "**Hakin9**" in our store at [www.zebnet.us](http://www.zebnet.us)

zebNet backup solutions are available for Microsoft Outlook, Windows Live Mail, Microsoft Internet Explorer, Mozilla Firefox, Mozilla Thunderbird, Mozilla Seamonkey, Google Chrome, Opera, Apple Safari, Postbox and Incredimail.

For any questions you may have, please get in touch with us directly at [info@zebnet.us](mailto:info@zebnet.us) or visit [www.zebnet.us](http://www.zebnet.us)