MOBILE HACK
MOBILE SECURITY TESTING

CROWD-SOURCING FOR IT SECURITY
— NEW PENTESTING BUSINESS MODEL
BUILDING YOUR OWN PENETRATION TESTING APPLICATION
DO YOU TRUST YOUR PENTESTING TEAM?
Dear Readers,

How do you feel when you read yet another piece of news about yet another tabloid journalist hacking into yet another celebrity/politician/accident victim/etc. mobile and extracting confidential information from their voice mail (The News of The World thank you very much for making the news)? I believe that people who use their mobiles like their cars (you don’t have to know what’s under the bonnet to know how to drive it – so the vast majority of us) are perplexed at the idea that their precious secrets can be disclosed so easily and their indispensable mobile friends can be hacked into by nameless agents, lurking somewhere out of their sight. But how a hacker feels? I believe – offended, because how can you call trying out a 4-digit code (which is most likely 1,2,3,4, or the year of the user’s birth, or something equally impenetrable) till you find the correct sequence? I might be a bit biased here, but I find calling it brute-forcing a bit of an overstatement.

Thus, we’ve decided to devote our September edition to mobile security, seen, as always, from a pentester perspective. The mobile apps market is growing rapidly, and so are attempts of compromising its security. Nowadays everyone can be a „hacker”, as we have already mentioned, but securing yourself from a real threat is another pair of shoes. And what better way of managing security issues than penetration testing?

The centerpiece of this issue’s focus is Aditya K Sood’s Breaking Down the i"{Devices}, concentrating on data testing, decrypting and mobile apps developers „wrongdoings”, who sometimes tend to disregard security issues at a scale which can be described as at least inappropriate, taking into consideration the expanding market. Cory Adams will encourage you to Act Like a Criminal while Leveraging Android Malware for Improved Penetration Testing Results, Bill Mathews will share his views on Attacking the Mobile Infrastructure, and Devesh Bhatt will take you Inside Android Applications, concentrating on manifest configuration. Some general points of Mobile Application Security Testing will be presented to you by Iftach Ian Amit.

There are of course other articles worth looking at in this issue of PenTest Magazine. I can definitely recommend Arthur Gervais’ New Penetration Business Model – the idea behind his Hatforce project, based on crowd-sourcing. It might be another step in the field of IT security, surely worth looking at and taking further.

Enjoy your reading
Sebastian Buła
& Penetration Test Magazine Team
### POINT OF VIEW

**Isn’t Social Engineering the Safest Form of Pentesting?**
by Ankit Pratek

One might argue over this, but for a student and a budding pentester like me, this is the truth and holds water. Social engineering won’t call your work illegal unless you harm someone personally or cause some financial loss. Plus, since you don’t have certifications at competitive prices, no one even wants you to be a certified Social Engineer at that unaffordable price.

---

**Trust Pentesting Team. Do you?**
by Rishi Narang

With the advent of security and its counterpart, a large share of vulnerabilities has been due to human errors in the software lifecycle. These errors have either crept in mistakenly, or the loop holes have been intentionally inserted with ‘malicious’ intentions.

### FOCUS

**Breaking Down the i*{Devices}**
by Aditya K Sood

Smartphones have revolutionized the world. The online world is grappling with severe security and privacy issues. The smartphone applications require an aggressive approach of security testing and integrity verification in order to serve the three metrics of security such as confidentiality, integrity and availability.

---

**Act Like A Criminal**
by Cory Adams

What, act like a criminal? That would usually be considered bad advice, but having an understanding of how cyber criminals conduct business will lead to better penetration testing results. In-depth malware analysis will reveal criminals’ tactics, techniques, and procedures. These can be utilized to generate improved penetration testing abilities by allowing the tester to view the target as a would-be intruder does.

---

**Mobile Application Security Testing**
by Iftach Ian Amit

Thriving vendor marketplaces (such as iTunes and the Android store) encourage the rapid development and deployment of mobile applications to consumers and businesses alike. Additionally, alternative 3rd-party download and install markets open up as software writers seek opportunities, outside the walled gardens provided by the mainstream stores.

### (NEW) STANDARDS

**New Penetration Testing Business Model**
by Arthur Gervais

Today everybody can become a hacker. The knowledge spreads all over the Internet. A lot of hackers are showing their know-how by sharing the results of their attacks. Why do not use this knowledge through crowd-sourcing in order to globally improve the security? Starting from this fundamental idea, a business model has been developed by Hafforce.

### HOW-TO

**Building Your Own Pentesting Application**
by Dhananjay D. Garg

Although even today web browsers serve the primary purpose of bringing information resources to the user, they no longer represent a software application with bare bones support for just HTML. Today, web browsers like Mozilla Firefox come with the support of add-ons, which are small installable enhancements to a browser’s foundation.
Isn’t Social Engineering the Safest Form of PenTesting?

If it’s permitted, registered and certified, it’s pentesting, and if it’s not, it’s just plain words scary hacking.

One might argue over this, but for a student and a budding pentester like me, this is the truth and holds water. Social engineering won’t call your work illegal unless you harm someone personally or cause some financial loss. Plus, since you don’t have certifications at competitive prices, no one even wants you to be a certified Social Engineer at that unaffordable price.

As a learner I don’t think any of the two should be your main concerns. Just knowing the password and some browsing using it should be enough for an encouragement. I can get someone drunk and get his passwords rather than doing phishing and other stuff. Getting picked up by girls from a bar and then using their laptop or desktop with an excuse to check my mails is what I have been doing lately. The fun part is to discover the lover’s files and saved passwords... Okay, maybe I am not being picked up by girls in the bar, but they do give me their laptops to use the Internet (not in the bar of course, well the bar was supposed to sound cool). Anyways, other moves are: offering my laptop to others to change passwords or login into any account. Some smart ones check the anti-virus inclusion list to track keyloggers, some trust me, others have not heard about firefox addons, or the changed script that enables storing all passwords without offering to remember.

Trojans haven’t helped me much, nor has any exploit from Metasploit that I know of (some 3 or 4), except for my own virtual machine which has no anti-virus. Accessing other PCs myself than accessing it remotely has so far worked pretty good for me. I’m often filled with guilt that I make friends just to add them to my stolen passwords list... But that’s a different story, let’s not get there. Watching desktop screens of your friends at night and clicking their picture remotely at that very moment aren’t on the list of the most interesting things, but one still might enjoy doing it for fun and, of course, learning. But try not to go for the easy way, which is implanting the .pdf in your friend’s laptop, who uses an older version of Adobe Reader. Removing my device from my friend’s Facebook was the coolest correction that I’ve done so far (oh, try Konqueror, it impressed me). Getting the phone number to stay in touch is easy, then updating Facebook status from that number is so much fun, thanks to the websites the names of which can’t be disclosed here.

Moving on, the only method I’ve found to protect my own Facebook wall from sms spoofing is by not sharing my phone number with anyone. Sms spoofing is so easy, simple and free a non-geek can do it. Against caller id spoofing, those who can crack astrisk aren’t idle enough to try me, so I feel pretty much safe. I am not so sure if Facebook knows they have this vulnerability, since it’s still on the go. I really hope they buy this issue.

Upon being caught when the secret was somehow revealed to people, saying that I was pentesting your
I took a course for International Certification assuming it will make it easy for me to get permission from authorities to practice with them, but my trainers were doing fraud in the name of that false certification, so now I have even lost that hope too, humph! I’m looking forward to platforms like Hatforce, thanks to Arthur (see this issue). As a Non-Certified Infosec Pro, Social Engineering is what I feel best to practice and with positive results it’s always encouraging. And again, nothing illegal has been done so far, and none to be happening in the future either.

ANKIT PRATEEK, RHCE,CISP

In the next issue of HAKIN9 magazine:

Hacking Apple
Available to download on September 30th

Soon in Hakin9!


If you would like to contact Hakin9 team, just send an email to en@hakin9.org. We will reply a.s.a.p.
Trust Pentesting Team. Do You?

With the advent of security and its counterpart, a large share of vulnerabilities has been due to human errors in the software lifecycle. These errors have either crept in mistakenly, or the loop holes have been intentionally inserted with ‘malicious’ intentions.

The last decade witnessed millions of small or critical vulnerabilities and most of them duly fixed, mitigated or remediated, but what about the human link, the human mistakes, the human intentions. It can never be fixed but early detections and a keen eye can save you from unintentional handing the secret keys to a thief.

In my professional and personal experience, there have been very few clients and customers who are actually aware of what is happening during their pentest phase. They are aware about the vulnerabilities reported; the calls and explanations presented by the pen-testing team, but are oblivious to the network facts and access rights. Most of clients have a strict objectives mentioned in the contract. These objectives include guidelines that refrains a pentester from DOS attacks on service or system, persistent threats, intrusive attacks or code executions etc. if the system is live and production as this can result in disruption of their services. Wherein, if the system is a dummy clone, such genre of attacks can be permitted by the client in controlled conditions. But how many customers actually verify the attempts by the pentesting team through the logs – system as well as the network logs.

Pentesting team has limited timeslots or has limited time windows to perform such assessments. On a standard note, a client should always make a note of the IP addresses allowed for the pentester, and exempt it on perimeter security (if really needed) else, keep the rest of security postulate on its toes. IT team should always check the logs and look for anything that is beyond the scope of pentesting contract like,

- Check the resources being accessed via the application and/or server logs.
- Check the internal and/or public IP addresses being accessed via the network logs.
- Any discrepancy in the logs reflecting the pentesters’ IP address should result in blocking that IP address till a satisfactory explanation is provided by the team.

In the worst case scenario, if the attacker (hidden under a pseudonym) renders his services to a firm wearing a white hat and steals database information, source code, or even the credentials etc. Later, even if the vulnerability has been mitigated, he still possesses critical information at his disguise. If the logs show that some of the critical files have been dumped during the pentesting phase, a client can (and should)
always raise an eyebrow if this is a production system. Any pentesting phase if not intrusive needs a POC (proof of concept) and not a valid attack to hamper the services. Vulnerability can be proven with the fact that it can allow an attacker to download a file, but no actual need of accessing every critical file (passwords, configuration etc.) on the server just to prove your point. Client should mention that in the contract – During intrusive pentests, the team is ONLY allowed to download/access ‘index.html’ or (any low criticality files) as POC.

Customers can also deploy certain measures to prevent themselves from such disguised attacks,

- Prefer pentesting on dummy cloned servers where liberal perimeter network checks are in place (may be a different LAN/VLAN all together)
- Appropriate segregation of resources – critical vs. in-scope logistics
- Continuous monitoring of logs – network/perimeter as well as server and/or application and validate the accessing of files.
- If production system, try to keep critical files encrypted. It can enable a pentester to demonstrate a vulnerability (if any) without compromising the confidentiality or integrity.
- Interview the responsible team on their daily activity and which resources did they accessed and why?
- No modification in the production files existing on the server. Client can validate this with simple modification time checks on its servers.

The idea behind a successful pentesting assessment is to either validate the security controls against the thinking patterns of cyber criminals, or to get a green check on the compliance controls during audit cycles. But, in no manner will a client appreciate the data being compromised due to an attacker camouflaged under the white cloak of its team members.

RISHI NARANG

Rishi Narang is an Information Security Professional, and is Cyber Psychology enthusiast. He can be reached via Twitter (@rnarang).
This paper sheds a light on the behavioral testing and security issues present in Apple’s iOS devices and applications. Primarily, this paper revolves around penetration testing of iPhone device and its applications. The paper does not discuss the iPhone application source code analysis and reverse engineering.

**Mach-O Format and iPhone Architecture**

Mach-O is the primary file format that is used for running applications and programs on Apple devices. This format is stored as an application binary interface on the respective MAC OS X operating system. Mach-O provides support for intermediate (debug) and final build (released) of the binaries. This is quite helpful in debugging as MACH-O format supports both dynamic and statically linked code files. Mach-O format is basically divided into three main components stated as header structure, load structure and data structure. The header structure explicitly specifies the environment information of the binary which is required by the kernel to differentiate between the code execution on different processors and architectures. Load structure comprises of the various segments which define the byte size and memory protection attributes. When the code is executed dynamically, the segments map the desired bytes into virtual memory as these segments are always aligned with the virtual memory pages. Data structure contains various sections of data which are mapped through the segments defined in the loader structure. Usually, there are text and data segments. For example: considering an Objective C, there are segments defined as `__OBJC__` which are private to the Objective C compiler. The internals of Mach-O format can be read here [1]. Figure 1 shows the generic layout of iPhone architecture.

The application binaries (Mach-O) format are encrypted in nature when these are retrieved from the Apple store. In order to perform source code analysis these files are required to be decrypted by the process of reverse engineering.

---

**Penetration Testing Like a Hacker**

Smartphones have revolutionized the world. The online world is grappling with severe security and privacy issues. The smartphone applications require an aggressive approach of security testing and integrity verification in order to serve the three metrics of security such as confidentiality, integrity and availability.
DFU and Jail breaking

Device Firmware Upgrade (DFU) allows the user to restore factory defaults by restoring operating system. However, this mechanism has been exploited with robust exploits in order to thwart the inherent sandbox. The complete process is applied under jail breaking which allows the attackers to circumvent the Data Protection Framework (DPF). As a result of this, it becomes easy to run Objective C code on the device as the entire application layer protection gets demolished by jail breaking process.

Jail breaking [2] is an important step from testing perspective if the application has to go under the reverse engineering process. As the designed application runs only on the required architecture (ARM), it is ineffectual to try to run these applications on the local machine or even on the simulator for reverse engineering purposes using GDB. The best way is to perform jail breaking (exploits are available in public) and then debug that application in order to perform implicit analysis. It is advisable for normal and regular work purposes to avoid jail breaking from testing purposes and the penetration tester or reverse engineer has to go to the core to perform legitimate testing steps. Jail breaking requires specific steps to be performed as discussed here [3]. One should be sure of Jail breaking exploit which performs tethered [4] or un-tethered jail breaking (depends on the type of bootrom version is used), which might impact the running state of your IPhone. After jail breaking, requisite utilities such as GDB, GCC etc must be installed. For example: using red snow to perform a jail break loads only SSH as the main utility for rooting iPhone. Once Cydia is installed, use the default repository to install other tools. Cydia has aptitude package downloader. Always choose the developer or hacker profile in Cydia while jail breaking.

Scrubinizing IPhone Backup Process

The IPhone device follows a defined backup process. However, with the change in IOS and IPhone versions, there are certain developments that have taken place in creating backup files. Generally, when an IPhone is attached to the computer and iTunes is running, the software by default performs some backup operations and creates some backup file. These files are used extensively for forensic investigations and contain fruitful information. A new entry is created every time at the location: /Users/<your user name>/Library/Application Support/MobileSync/Backup/, C:\Documents and Settings\USERNAME\Application Data\Apple Computer\MobileSync\Backup; when a device is synced with the laptop. The back path may vary depending upon the operating systems. The backup directory contains some index files, property files (plist) and manifest (mbdb, mbdx) files. This might vary with different IOS versions. The primary files that need attention are manifest files having extensions such as mbdb and mbdx. Simply editing these files might give you some clear test information but that is incomplete in nature. The best way is to use IPhonebackupdb.py [5] script in order to process the files automatically.

Fuzzing Endpoints Communication

Every application requires end point communications which involves both port 80 and port 443. Certain large size applications work in a distributed environment. It means that the application does not communicate using single end point; rather data is transacted from multiple end points by using dual channel. The fuzzing should encompass different scenarios as discussed below:

- For HTTP specific communication fuzzing, use any HTTP proxy such as Charles, Web Scarab, Fiddler and Burp to perform fuzzing of different implicit parameters in the request. As discussed earlier, server might fetch data from other nodes in the communication channel. Thus, all the nodes must be verified against the type of data. When evaluating a Client-Server IPhone application, both the server and client should be tested (fuzzed). Always verify whether both end points are secure. It has been noticed that most of the vendors are implementing input validation only at Client-side, and, as a result, are vulnerable to injection attacks when request is directly issues to the server.
- For aggressive testing, fuzzing local files (Application specific files) can be used to instrument the application for automated brute force purposes, as well as abusing of other application specific features.
- For effective penetration testing, raw HTTP agent such as cURL should be used to send custom queries to different endpoints from the jail broken iPhone in order to utilize the device resources and session rather setting a network proxy and routing traffic out of it. This is a very effective step for fuzzing end points. This step actually ensures the nature of malicious applications attacking legitimate application on IPhone device.
- All the end points must be verified against the strength of deployed SSL. One should not concentrate on testing the single end point for SSL stability.

The end point fuzzing is very critical to determine the strength of application from a behavioral perspective.
IPhone Data Leaking and Verification

IPhone is prone to data leakage and privacy issues. The robustness of application design and legitimate iOS can prevent information leakage from the device and the system. From a penetration testing perspective, the data leakage and privacy issues should be dealt strongly. There are certain specific set of information that must be verified. It includes:

Every device including iPhone comes with a Unique Device Identifier (UDID) which is binded to the particular user when the device is sold in the market. It means UDID provides the information about the environment of the user and developers or hackers can use this identifier to query other system information. The main concern relates to the user monitoring which includes browsing habits. It is strongly recommended that every application must be tested against the usage of this identifier and its transaction with third party services.

Other critical information include Personal Identifiable Information (PII), Online Social Network ID (OSN-ID) and Location Identifiers (LID). Listing 1 shows the insecure implementation of user private information in an inappropriate manner.

A detailed study about the online social privacy leakage has been conducted here [6] which show exactly how mobile platforms reveal sensitive information.

Tracing Properties (Plst) Files

The files with the extension plist are used in every IOS device to store preferences in the form of strings. The format of these files varies with MAC OS versions. It is possible that one encounters plist files in binary format or XML format. The binary format is chosen to parse the strings for robust application performance. If the files are present in the XML format, then it is fine otherwise it is advisable to convert the binary format to XML format using inbuilt Xcode tool (Property List Editor) or open source utilities (psutil) as presented in Figure 2.

Tracing IPSW and IPA Files

Files with extension IPSW (iPad/iPhone Software) are the software update files that are used to upgrade program and iOS firmware images on apple devices. Basically this is a software updating package which bundles as a zip file. It is easy to change the .ipsw extension to .zip extension and then unzipping it to get the files in an unbundled state. These files are located at Mac OS X: [user]/Library/iTunes/iPhone Software Updates/ and Windows XP: \Documents and Settings\[username]\Application Data\Apple Computer\iTunes\iPhone Software Updates\. The unzipped archive contains .DMG files which is an updated version .IMG. The .DMG files are encrypted in nature and can be differentiated based on the size of DMG. If the size of DMG file is large, then it must be a root file system file otherwise these are ramdisk files. In order to decrypt DMG files, one can use VFDecrypt [7] tool. This tool might give an issue in certain version of windows and CPU architectures. In addition to that, one can use iDecrypt [8] tool as presented in Figure 3.

In order to decrypt it appropriately, DMG key for specific device is required. Luckily, it is provided with tool and all the keys are listed in XML file as presented in Figure 4 .

Files having IPA (iPad/iPhone Application) extension are termed as the updated versions of APP files. The IPA files store the application package for iPhone devices. Generally,
these files are derived from DRM and are encrypted in nature. IPA files can only be installed on the iOS devices because it contains an ARM binary in it which only runs on RISC architectures. As stated earlier, Jail breaking is required because these applications are not allowed to run on X86 architectures.

This fact should be taken into account while performing analysis of IAP files.

SSL Certificates Testing for CA’s

iOS devices should be tested aggressively for deployed root certificates. Certificate testing should be a part of the iOS penetration testing strategy. It is highly advisable to follow the below stated facts while testing SSL certificates in iOS devices.

- Self signed certificates for iOS devices should be avoided as it is a big problem in iPhone devices because it raises an alert about the integrity of certificate. This fact can be exploited for non-legitimate purposes. Further, weak ciphers over SSLv3 should be disallowed.
- Certificate Chain validation is a big problem in iPhone. Some of the iOS devices do not validate the intermediate parties for SSL channels and result in broken communication. This could be a problem for the server because it has to validate all the chains. Some of the browsers take care of it individually, but iOS devices fail to do it. This flaw has already been detected by the Trustwave [9] in iOS devices which is a result of basic constraints certificate extension. The basic constraint has already been exploited a lot in the wild and many SSL flaws have been detected. For iOS devices, the SSL chain should be validated explicitly and certificates should be used from legitimate CA providers. Figure 5 shows the basic constraint extension in use by the single CA authority.
- For general verification, SSLv2 should be avoided completely and there should be no support for backward compatibility. The protocol context should be verified for both SSL/TLS. Always be sure that all the data fetched by the application must be over SSL. Three tools such as cURL, OpenSSL and SSLScan must be used collaboratively to scrutinize the strength of SSL. All the SSL endpoints should be tested for SSL renegotiation vulnerability [10].

Apple Push Notification (APN) Testing

iOS devices greatly reside on the push technology
which is an internet based communication mechanism to send notification to the devices. When the device initiates a connection to the server and data is sent back to the client, the server does not terminate the connection in full and remains in open state for the listening events. Figure 6 shows the broad view of push notification technology.

All the applications performing push notifications should be tested appropriately. If the APN is implemented in an insecure manner, it results in information disclosure and this technology can be used to trick users to conduct differential attacks [11] such as phishing attacks. The application should be tested for the following security metrics

- APN should be implemented over SSL.
- Avoid sending sensitive information over APN message.
- APN is meant only for notification messages. While penetration testing, it should be verified that APN is not modifying any messages


**Location Tracking and Server Side Design**

iPhone location tracking is a default phenomenon which is used to track the location of the device. Generally, this functionality has been provided for the applications to use the location directives of the user. Every IPhone enables this service by default. One might have encountered push notification messages (as discussed earlier) which raise a flag about the tracking of device when a new application is installed on the device. From penetration testing point of view, following metrics should be checked in order to determine the state of location services for a particular application.

- Always look forward for the consolidated.db (data point's information) file which is used to store location preferences of the device pertaining to particular applications. This file provides information such as latitude, longitude and timestamps for a specific location. As we know most of the data is stored in the form of sqlite database. The primary file can be mapped by looking for string RootDomain
- Verify whether the location data is present in the clear text. By default, the data is cached continuously in the form of data points which are easily readable using SQLite browser [16] and SQLite manager [17]
- It depends on application design but the preferable choice is to encrypt the data while storing on client side. This is because whenever a device is connected to the computer, all the information is transferred to the computer as a backup so that it can be synced later on.
- The location of consolidated.db can be found at paths as presented in Listing 2.

One might encounter `manifest.mbdb` and it is necessary not to read these files directly in Microsoft Excel because it results in gibberish data. It is advisable to use an appropriate editor to get the information. For general information, one can also use TraceYou [13] and IPhoneDisk [14] applications as an extra step to validate things. Further, there is another way to transfer consolidated.db file to KML as discussed here [15].

This step should be included in the penetration testing of any application developed for IOS platform. However, an appropriate channel should be established between the device and the server on and the server

### Listing 2. Location tracking default path

- Jailbreak IPhone - /var/root/Library/Caches/locationd
- Computer - iTunes: ~/Library/Application Support/MobileSync/Backup/mumble1/mumble2

Mumble is used to set the device identifier for iPod, iPad and IPhone.
should verify the identity of the device so that requisite data must be transferred to the legitimate device in order to avoid modification and the interception of the data.

Client Side Logging, Caching and Session Data (Snapshots, Clipboard) – The Real Mess

Client side data handling and storage is one of the major fallacies in the online world. No doubt, it requires for caching and optimization but at the same time the inherent design has to be robust from security point of view. Well, that's not a case in the real world. This problem has become worst with devices like IPhone and iPad. Most of the IPhone applications handle client data in an insecure manner. The applications are designed in following ways.

- A number of applications perform client side logging by default which is inbuilt with no explicit settings.
- A heavy set of applications provide an implicit option in the IPhone settings to enable or disable the client side logging.

Some of the applications try to provide control in the user hands to handle client side logging. This is a really faulty practice. The developer must consider the rationality behind full client side logging. This practice should be done at the server side. From the penetration testing point of view, most of the client side logs are located in the on the path as presented in Listing 3.

Listing 3. Client side log storage paths

\Applications\[Application Name]\Documents\*.log
\Applications\[Application Name]\Library\Caches\*.log

Figure 7 shows the information extracted from Yahoo Messenger application client side logs. One can easily find the contact list, email addresses and session identifiers for a particular user. This information varies from application to application. Another anatomy is these applications hardly provide functionality of disabling client side logging which is a really erroneous design implementation.

Further, the configuration XML files used by these applications; for example: Skype provides a complete contact list of users with Skype id’s. Other information include voicemail logs, voicemail files such as *.dbb and *.dat files. The client side storage includes SQLITE database to be maintained which can be easily decompiled to find the type of information stored in it. Figure shows the contact list leveraged by the Yahoo messenger IPhone application.

This shows how well the designers are handling data storage in IPhone applications. If you want to give a try, analyze all the messenger applications for IPhone and scrutinize yourself the reality of these applications. For banking and financial applications, penetration tester must look from scratch and the type of information that is cached and logged on the user device. Apart from client sides logging, following tests should be conducted on the IPhone application.

- Every application generates a snapshot on quitting. While testing, one must ensure that there is no sensitive information stored in the snapshot files. Usually, the snapshots are replaced every time an application quits. Scrutinize the snapshots directory in the IPhone application folder.
- Verify that if com.apple.UIKit.pboard object is used to support clipboard activities. This object should be private and must no store sensitive information in it for a longer duration.
- Scrutinize the keyboard folder on the IPhone device in the application default path in order to verify the presence of keyboard cache in the system.

Figure 7. Yahoo messenger IPhone application – client side logs

Figure 8. Contact list information – Yahoo messenger IPhone application – SQLITE
• Always dive deeper into the nature of applications that save sensitive data insecurely (e.g. passwords, local identifiers and etc. not within the Keychain). Even though the data is not transmitted over the air, it is something that possible be exported via a backup-restore procedure and result in data leakage as discussed earlier.

Digging inside iPhone Management Tactics
As the iPhone is an integral part of enterprise environment, it must have some sort of enterprise management features which can be sued to protect the privacy of user to some extent and can act as a backup support if something wrong happens. Following tests should be conducted in order to understand the deployed policies and implemented policies.

Scrutinizing Profiles
Enterprise IOS device management is handled using iPhone configuration utility [18]. Basically, it provides a standard set of settings to be applied for every iOS device in the enterprise. There are other configuration profiles that take care of the Mobile Device Management (MDM), Simple Certificate Enrolment Protocol (SCEP) and PKCS credentials. Usually, the profiles are deployed by a web server though USB using MDM interface. All the deployed profiles should be tested and one must verify the type of profile that is enforced. Profiles that are small provide more granular control from a security point of view.

Policies Verification – Active Sync
Policies on iOS can be enforced used Microsoft Active Sync related to password policies, encryption, and timeout parameters etc. One should verify the deployed policies through ActiveSync. For example: Active Sync is used to apply remote locking and wiping of the device. Primarily, it is used to provide the similar set of functionality that blackberry enjoys. From the enterprise security point of view, remote wiping should be applied to every iOS device.

Mobile Device Management
This capability provides third party service providers to build web servers in order to manage iOS (or other mobile). Deploying MDM results in more centralized environment and effective management of iOS devices through maintaining configuration profiles. From a testing perspective, MDM structure should be tested in a robust manner. If the MDM is deployed at the enterprise level, it should be configured to detect jail broken phones on the fly.

OWASP Top 10 Mobile Risks
All the interactive applications should be tested against OWASP top 10 mobile vulnerabilities [19] apart from the listed issues above. However, considering the architecture of iPhone architecture, it is hard to confine the penetration testing in these top 10 vectors but it definitely provides a broad classification of vulnerability risks.

References
• IOS 4.3.2 Jailbroken Already Using PwnageTool. Here’s A Complete How-To Guide, http://www.redmondpie.com/jailbreak-4.3.2-IOS-iPhone-4-with-pwnagetool-4.3.2-bundle-how-to-tutorial/ [3]
• Apple Push Notification (Generate APNS SSL Certificate), http://www.developers-life.com/apple-push-notification.html [12]
• IPhone Tracking, http://www.geekosystem.com/IPhone-tracking/ [14]
• Consolidated.db to KML Win converter, http://meshx86.wordpress.com/2011/04/24/IPhone-users-iOS-4-aretracked/ [15]
• SQLite Database Browser, http://sqlitebrowser.sourceforge.net/ [16]
• IPhone Configuration Utility, http://support.apple.com/kb/dl8851 [18]
classes which should be verified in every penetration testing project. This set of issues has been derived on the same benchmark of web applications but in reality there is a difference in security testing due to architecture and deployment environment of the applications. In any case, the top 10 mobile risks should be incorporated into the methodology of iPhone testing.

During the course of this paper, it has been shown that there are a lot of developments that have taken place in the iOS world and testing should be executed accordingly. In the past, iPhone testing has been done in relation to specific scenarios, but nowadays iPhone applications require more aggressive testing to ensure security.

Acknowledgement
I would like thank Itzik Kotler (CTO, Security Art) for reviewing the paper and providing deep insight into the iPhone penetration testing.I would also like to thank Dr. Richard J Enbody for providing continuous support in doing security research.

Conclusion
The world is changing fast due to mobile revolution. This paper deliberated upon the iPhone architecture from perspective of penetration testing. The architecture plays a crucial role in developing security testing methodologies. In this paper, iPhone detailed security testing vectors have been discussed which include testing of data at rest, decrypting files and insecure design practices followed by the application developers. For a full matured security assessment of iPhones, all the discussed vectors should be tested appropriately so that secure applications can be developed.

ADITYA K SOOD
Aditya K Sood is a Senior Security practitioner, researcher and PhD candidate at Michigan State University. He has already worked in the security domain for Armorize, COSEINC and KPMG. He is also a founder of SecNiche Security Labs, an independent security research arena for cutting edge computer security research. He has been an active speaker at industry conferences and already spoken at RSA , HackInTheBox, ToorCon, HackerHalted, Source. TRISC, AAVAR, EuSecwest , XCON, Troopers, OWASP AppSec US, FOSS, CERT-IN (07)etc. He has written content for HITB Ezine, Hakin9, ISSA, ISACA, CrossTalk, Usenix Login,Elsevier Journals such as NESE,CFS. He is also a co author for debugged magazine.
Act Like A Criminal

Leveraging Android Malware for Improved Penetration Testing Results

With the emergence of the Android Operating System (OS) into the mobile market, nation state hackers and criminals alike are actively conducting attacks against the OS and its users for information gathering and financial gain. A high reward tool in an attacker’s arsenal is malicious software, also known as malware, which allows information to be gathered and extracted from targeted mobile devices. It is commonplace to rely on Anti-Virus (AV) as a basis for the mobile security model and trust that AV will identify malware. If AV does successfully detect malware most people will simply remove the malicious software and think that the threat has been mitigated, with no work left to be done. This is a flawed and incomplete approach that does not leverage the intelligence that can be gained from analyzing the malware sample.

Analysis can provide security professionals further insight into attack details such as: the intent, was this a targeted attack, persistence mechanism, propagation technique, etc. Analysis can also supply attack professionals the same information, allowing for the attack to be replayed during penetration testing. This data provides a developer with the ability to extract interesting pieces of malware samples to be repurposed and used in new attack weapons. The reason most defensive and offensive professionals do not conduct in-depth malware analysis is twofold. First, they do not properly understand the benefits of doing so and secondly they lack the knowledge necessary to do so. (well, maybe they have limited time, but I am biased and believe everyone should conduct malware analysis.) Both of these issues will be addressed with solutions comprised of the benefits for conducting Android malware analysis and details to setup an Android malware analysis environment.

Why target Android?
The Android OS is here to stay and with so many Android users out there utilizing the Android software, a large attack base is provided to attackers.

The Android OS is here to stay and with so many Android users out there utilizing the Android software, a large attack base is provided to attackers.
When vulnerabilities are found in the Android OS, Google will usually address them in a timely manner; however, the carriers are responsible for providing updates to the customized software running on their hardware. The process is very slow and in some cases non-existent. This leads to having different versions of Android, known as fragmentation.

Google seems to have built the Android OS to afford users and developers great flexibility providing a type of control. With this control should come a necessary respect for responsibility. Technology users have never been very responsible when it comes to understanding and properly using this control. A common example of this is how the Android OS displays to the user all the permissions an application is requesting, allowing the user to make an informed decision about whether or not to install the application. The problem is that most users lack the necessary knowledge to understand what is being displayed. Unfortunately, this usually results in users selecting install no matter what (Figure 1). This lack of knowledge provides an attacker a huge advantage over users and makes the job of those in the security field that much more difficult. The entire system was built around allowing users access to the Market and source code, which is great from a developer and user standpoint; however, for an attacker this is good for business.

**How to target Android?**

Let me start by saying that exploit development is extremely difficult. With that being stressed, in comparison with other targets, Android is considered by some a softer target because of the points stated previously. There are a few different attack vectors available for targeting Android. Among the most important are the traditional telephony system, Webkit (browser engine), and the Android Market.

Targeting the traditional telephony system provides a means for attackers to target the underlying OS. Though this article will not discuss telephony hacks. Traditional telephony systems attacks are well documented and more a part of history than emerging technology. If attacks against the telephony system are being conducted at this time, they probably won’t be talked about too much because they could be very valuable. It is important to understand that the telephony portion of the system can provide an attacker a remote means of accessing the phone. With that being said the next attack vector that is currently being targeted and exploited is Webkit.

As stated earlier, Google decided to provide the source code for the Android OS. This allows an attacker to conduct source code audits to locate and exploit vulnerabilities. Webkit is an open source browser engine utilized by Android. Webkit is no stranger to vulnerabilities as it is on Android devices and its code is available for auditing. What is one of Android’s biggest positives is also one of its biggest negatives. The fact that the code is open allows for development of more applications which leads to more users and ultimately a bigger market share. However, this also allows easy code auditing and vulnerability analysis by attackers. Many vulnerabilities have been identified in Webkit leading to exploitation as updates are slow from vendors, which means these vulnerabilities are likely to still be prevalent.

The final attack vector to be discussed in this article is the Android Market and 3rd party application stores. The Android Market provides a place for Android users to browse and download applications for their mobile device. To target Android users, an attacker simply needs to place a malicious application, usually embedded with a legitimate application, on the Market. Repackaging is a very common tactic in which a malware writer takes a legitimate application, modifies it to include malicious code, then republishes it to an app market or download site. The repackaging technique is highly effective because it is often difficult for users to tell the difference between a legitimate app and its repackaged doppelganger. In fact, repackaging was the most prevalent type of social engineering attack used by Android malware writers in the first two quarters of 2011, (Lookout, 2011). This approach...
allows an attacker the ability to sit back and let the victims go to him. Using the illustration by Lookout Mobile Security (see Figure 2) an example of the process malware authors use to conduct attacks can be better understood.

The Android Security Model will only be discussed briefly as it is a subject that really requires an article of its own. The Android OS runs on a Linux-based kernel and each application has its own User ID and Group ID. Because the kernel sandboxes applications from each other, applications must explicitly share resources and data. They do this by declaring the permissions they need for additional capabilities not provided by the basic sandbox. Applications statically declare the permissions they require, and the Android system prompts the user for consent at the time the application is installed, (Android Developers Guide, 2011). These permissions are defined in

![Diagram of malware attack process]

**Figure 2. Example attack process**
AndroidManifest.xml. An important point to understand is that simply locating a vulnerability, developing an exploit, and triggering the exploit does not guarantee full access to the Smartphone. However, even minimal permissions may allow an attacker the ability to collect valuable information about the victim.

Since not all penetration testers have the time to conduct vulnerability analysis and write an exploit it is important to note that other options are available that allow a quicker assessment of the target environment. At DefCon 19, Zimperium unveiled the Android Network Toolkit which is available from the Android Market free of cost (there is a $10 corporate version). The toolkit allows a penetration tester the ability to scan the network, check for un-patched phones, and fire older exploits, with the ultimate goal of arbitrary code execution. Another resource available to a penetration tester is the availability of free and useable exploits from public repositories such as Exploit Database (Figure 3) http://www.exploit-db.com. With these access methods available to penetration testers, why even bother analyzing Android malware?

**Why analyze Android malware?**

This article is not stating that malware analysis is necessarily part of the pentesting discipline. Rather, this article is trying to point out the importance of using other disciplines to increase the effectiveness of the penetration testing. Many in the security field are quick to undervalue the study of malware, unless it reveals the initial attack vector as they are mostly concerned with how the penetration occurred. Once they identify the attack vector, apply countermeasures, and clean up after an attack they move on without ever conducting in-depth analysis of any malware placed on the network. This often occurs due to one or a combination of reasons such as: limited manpower, time constraints, and lack of a malware analysis skill set. While these are all valid problems, not reviewing malware is a flawed approach as malware can reveal more than some in the security field understand.

When conducting a penetration test to evaluate the security of a network or an aspect of the network, why wouldn’t you test the security in a way that it is actually going to be tested? Regardless of the network, the security mechanisms in place will eventually be tested by an attacker. The attack may not be targeted, in fact it may be an attempted drive-by exploit, a user falling for a phishing attack, or simply a user downloading an application with a malicious payload. Malware analysis provides a means of understanding the attack vector, the intent of the attack, possible persistence mechanisms, its ability to propagate through the network, and sophistication level. All of this intelligence can be leveraged to build improved tests to be used during a penetration test to ensure that networks are being tested in a realistic manner.

Malware analysis will not usually provide the initial attack vector, but it can reveal how the attacker operates and this is a very important piece of the puzzle. Malware analysis can provide details into all of the functionality previously listed and insight into how attackers are evolving as security measures are put in place to mitigate threats. Google is aggressive about removing malicious applications from the...
Market (which is a huge attack vector) so attackers have developed ways to extend the life of an attack. Tim Wyatt of Lookout writes Lookout has identified a new Android Trojan, GGTracker, which is automatically downloaded to a user’s phone after visiting a malicious webpage that imitates the Android Market. The Trojan is able to sign-up a victim to a number of premium SMS subscription services without the user’s consent. This can lead to unapproved charges to a victim’s phone bill.

Another advancement in malware is a new threat to devices running Google’s Android mobile operating system is an advance on earlier Android Trojans examined by CA Security that unleash payloads which log incoming and outgoing call details and durations in a text file, according to researcher Dinesh Venkatesan. These provide examples of how the malware is growing in sophistication and is only a sign of things to come as security becomes tighter. The information gained here by thorough malware analysis is vital in understanding what threats are present today and allows penetration testers the ability to replay cutting edge attacks to ensure the end customer is protected.

Table 1. Android malware analysis tools

<table>
<thead>
<tr>
<th>Name of Tool</th>
<th>Tool Functionality</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dex2Jar</td>
<td>Disassembler</td>
<td><a href="http://code.google.com/p/dex2jar/">http://code.google.com/p/dex2jar/</a></td>
</tr>
<tr>
<td>Dedexer</td>
<td>Disassembler</td>
<td><a href="http://dedexer.sourceforge.net/">http://dedexer.sourceforge.net/</a></td>
</tr>
<tr>
<td>Small/Baksmali</td>
<td>Assembler/Disassembler</td>
<td><a href="http://code.google.com/p/small/">http://code.google.com/p/small/</a></td>
</tr>
<tr>
<td>ApkTool</td>
<td>Decompiler</td>
<td><a href="http://code.google.com/p/android-apktool/">http://code.google.com/p/android-apktool/</a></td>
</tr>
<tr>
<td>Java Decompiler</td>
<td>Decompiler</td>
<td><a href="http://java.decompiler.free.fr/">http://java.decompiler.free.fr/</a></td>
</tr>
</tbody>
</table>

How do you Analyze Android Malware?
Analyzing a piece of Android malware can be less complicated than analyzing other types of malware. This is because the analysis environment is rather simple to set up and the Dalvik Executables (.dex) can be decompiled to a readable language. To begin pick the OS of your choosing (the following instruction will successfully build an environment for Windows XP SP3 32-bit). Since Android applications are written in Java, download and install the JDK from: http://www.oracle.com/technetwork/java/javase/downloads/index.html After the installation of the JDK, the Android Software Development Kit (SDK) can now be downloaded and installed. (Note: the JDK, not just the Java Runtime Environment is necessary for proper installation of the Android SDK) The Android SDK can be found at: http://developer.android.com/sdk/index.html.

Once the Android SDK has been successfully installed, navigate to the Android SDK and Android Virtual Device (AVD) manager, select Available Packages and install the SDK for the version of Android desired (see figure 4). Next, a virtual device must be created using the AVD manager. This can be done by selecting a name (just for user reference) and selecting a target, which will be a version of Android that you installed the SDK for.

Rather than using an actual phone to analyze the malware which will, in turn, likely infect the phone, an emulator provides the same functionality while running safely in the virtual analysis environment. The emulator inside the analysis environment mitigates the risk of analyzing the sample and can save time over connecting to hardware. To start the emulator: open a command prompt, navigate to the android-sdk\platform-tools directory and run the following command:

`Emulator-arm.exe --avd <Name of AVD created>`

If successful, then the emulator window will appear (Figure 5). (Note: The emulator can be slow and may take a while to appear.) At this point a simulated
Smartphone running the version of Android you select is active within the analysis environment; now the malicious application can be loaded. This is accomplished using Adb and issuing the following command:

```
adb.exe install <sample.apk>
```

(Note: Replace sample with the title of the malware sample you are analyzing.)

The following table (Table 1) is a list (not comprehensive) of free tools available to Android malware analysis to aid during the examination of a malware sample.

Many in the security field view malware analysis as the reactive response to an attack, but the opposite approach can be taken to help mitigate damages prior to this. Penetration testers can analyze or use malware analysis results to understand what an attacker is after, persistence mechanisms, propagation techniques, and advanced methods being utilized. This intelligence allows penetration testers the ability to replay real world attacks and ensure the highest quality results are provided to the customer.

CORY ADAMS

Cory Adams has been in the information security field for over 7 years. He is currently a Reverse Engineer with a Fortune 100 company. He specializes in malware analysis as well as vulnerability analysis. Follow Cory on twitter @SeedyAdams.

COMMENT

We are open for suggestions and discussion. Don’t hesitate to comment on the articles which you’ve read in this issue. Share your opinion on the subject matter brought up, back up or confront the point of view of the author. The best comments will be published on our site and in our next issue.
Having your software purchased and downloaded by millions of people worldwide has long been the holy grail of mobile software developers, but it also attracts the attention of fraudsters who recognize the accessibility and lack of security features of these platforms. The mobile platform opens several attack avenues for malicious software and opportunities to defraud victims due to its lax control mechanisms, and lack of standardization of the user experience offering. Therefore, mobile applications should be designed, developed, and tested having security in mind, much like web applications that handle sensitive information.

The design and development of mobile applications is significantly different to that of traditional client-servers or web applications. Mobile applications should take into account both the environment (platform, libraries, capabilities), together with major differences in end-user expectations. Mobile users demand a simple user experience (in terms of details), and often require completely different business processes compared with other interaction channels.

Security Challenges
There are two main security challenges to mobile applications that stem from their usage and limitations:

- Insecure Connections
- Simplified User Experience

Insecure Connections
Mobile devices are used in a number of unknown and often insecure connection profiles (from public Wi-Fi, through rogue cells that proxy communication). This makes them vulnerable to simple attacks not considered in the threat modeling of a traditional web application. Additionally, insecure communications are often used to overcome platform limitations and design considerations such as: battery consumption profiles, processing speed, and communication overhead.

Insecure communications for mobile applications expose several exploitation avenues (including local and remote), and enable fraudulent application creation using extremely simple tools and techniques that are freely available in the market. This not only puts the end user at risk of data loss, but also allows attackers an easy access path into the organization that provides services through the mobile applications. Any foreign code that runs on the mobile platform has the potential to alter the user experience and manipulate the locally stored data as well as the data in transit. Thus fraudsters gain a prime opportunity to conduct their attacks.
Simplified User Experience
The user interface provided by mobile applications differs wildly from other interfaces provided for end users. It aims to provide a simpler and more interactive experience. Many times it actually changes the application logic behind the business process with the potential to undermine the integrity of even the most robust existing software processes.

Mobile Application Secure Design and Development
There are a few ways to deal with these challenges when approaching a secure mobile application testing project:

• Mobile Application Penetration Testing
• Fraud/Usability Testing

Mobile Application Penetration Testing
Mobile applications should be evaluated and tested against attacks that take advantage of the exposed exploitation avenues – including local and remote.

In order to even start such testing, a mobile application testing lab is required – which usually consists of a development PC for disassembling and simulating the application runtime in a contained environment, a mobile device (preferably rooted, in order to allow closer inspection of the application in its native environment), and a network that would simulate both the WiFi as well as the cellular connectivity.

The mobile applications and its corresponding server-side components should be tested as part of the penetration test, which covers both traditional issues (such as SQL injections and OWAPS top 10 vulnerabilities), together with custom fuzzing of any proprietary protocols, full analysis of the communications, and any logical issues in the application design and implementation.

One major area of focus should be the seemingly trivial elements of the communication models used by the applications. Issues such as establishing secure communications over an encrypted channel are overlooked too often. Situations such as improper verification of certificate chains, and the lack of user notification along with a fail open approach that naively ignores such errors leave mobile applications open to man in the middle attacks that would raise many alarms on a web application.

An additional element that should be looked into is the kind of media that the application considers as trusted. Many times, sensitive applications do not consider WiFi connections any different than that of a 3G one. Moreover, in an attempt to provide a better user experience, switching from one medium to another while keeping the session alive is often implemented in such applications. Obviously, there are different weights to the trustworthiness of a

Figure 1. The different scopes in which mobile application security should deal with, as opposed to the common approach where the focus is set almost exclusively on the application itself
cellular connection and a WiFi connection (and different weights within WiFi connections – from ad-hoc networks, through publicly accessible internet connections, to trusted home or office connections). Such behavior should also be examined when testing the mobile application, as one of the threats to a mobile application would be to force a user into an insecure network – thus lowering some of the guards and native security features of a trusted network and enabling an attacker to take advantage of the accessibility of an open connection.

Fraud / Usability Testing

The kinds of attacks that a mobile platform allows differ from those used on PC platforms. They are sometimes more successful, as the user is less aware of the underlying components that allow the application to run. Examples such as local pharming, and exploiting mobile operating system capabilities (where a more secure alternative should have been provided by the applications) are only a few of such attacks that are currently known to be highly effective in the wild.

Also, due to the business process changes mandated by the simplified mobile user interface, it is vital to verify that such business processes are still valid in terms of their correctness and security.

A fraud/usability test should be conducted to find any loopholes or vulnerabilities in the processes or user interface handling.

When alternative business processes and workflows are chosen (to improve customer experience), it should be taken for granted that an attacker will gain access to such application workflows, so it is important to fully analyze any consequences and impacts of such access.

Alternative business flows are very common when testing mobile applications, and as testers we often find these supported through the same servers that provide the standard web user with business logic.

Mapping out these services also allows an additional attack vector to be exposed – not only affecting mobile users, but every user of the system (especially when dealing with financially related processes).

Conclusion

It is important not to keep the focus/scope too narrow when approaching mobile application testing. As mobile applications are literally greater than the sum of their components: Having the actual mobile platform to take into account, the usability issues that take precedence too often, the backend services that expose a lighter business process (often erring on the side of making it easier for users to authenticate or authorize activities), and the lack of transparency to the user in terms of the underlying security features it has grown accustomed to (such as an indication that the communication channel is encrypted at all), make mobile application testing a challenge of its own, and our jobs much more critical in pointing out gaps that would later turn out to be critical for the application author.

IFTACH IAN AMIT

Iftach Ian Amit brings over a decade of experience in the security industry, and a mixture of software development, OS, network and Web security expertise as Vice President Consulting to the top-tier security consulting firm Security-Art. Prior to Security-Art, Ian was the Director of Security Research at Aladdin and Finjan. Ian has also held leadership roles as founder and CTO of a security startup in the IDS/IPS arena, and a director at Datavantage. Prior to Datavantage, he managed the Internet Applications as well as the UNIX departments at the security consulting firm Comsec.

Ian is a frequent speaker at the leading industry conferences such as BlackHat, DefCon, Infosec, Hacker-Halted, FIRST, BruCon, SOURCE, ph-neutral, and many more.

ADVERTISE WITH US

Does your company provide penetration testing services? Would you like to advertise them in our magazine? If you represent a company and contribute an article to the magazine, we will provide you with free advertising space – depending on the character and the quality of your contribution. The more contributions you make, the more advertising opportunities you get – like your logo on the cover, or a special issue of the magazine (which our subscribers will download for free), devoted exclusively to your company.
Penetration Testing?

We’re better than this.

Visit our website to find out how and why!

hurricanelabs.com

Contact us today
888.276.4106
sales@hurricanelabs.com
I would like to point out that I am by no means an expert in mobile devices or their management infrastructures. This article was as much a learning experience for me as a writing project. I chose, deliberately to not make this a terribly technical article and more of a how to approach this article because I think sometimes in our industry we get hopelessly lost in the this will be so cool that we forget the this is the right, practical approach. Hope you enjoy.

As penetration testers we often times get mired in trying to craft attacks and finding 0-days when we should be fixating on our jobs, that is to provide an assessment of the security posture of a given system with practical scenarios. Though I see the value in crafting new attacks, I'm not sure it's the job of a traditional penetration tester but that's another article. It's hard enough to resist that temptation when dealing with web applications and Windows systems that have been around forever and are pretty well understood but throw in something new and our geek buzzers start buzzing overtime. Whenever we’re asked to test some new thing, in this case a mobile infrastructure, out come the compilers and debuggers. We should start by asking ourselves the most boring question possible, is this stuff really THAT different than what we’re used to?

Mobile smart phones and tablets do have a few key differences that I wanted to outline:

- They are by and large single user systems with root or admin restricted by default
- They run specialized operating systems but rely heavily on web interactions
- Often they aren’t controlled or managed by IT, users bring in their personal phones for business use (we’re not focusing on these)
- Tablets (well the iPad anyway) are quickly becoming a great way to work from conference rooms, meetings, etc. They are really a hybrid between smart phone and a laptop.

Now before we dig too much deeper I want to say that I’m not going to focus too much on attacking the phones/tablets themselves, there is quite a bit of research and work being done in those areas already and I doubt I could add much to it. I have always taken a more practical approach to penetration testing (right or wrong), I start with the simplest, widest reaching techniques first then move out to the more difficult methods of attack. I’m not discounting direct phone attacks I just find them to be more of a pain.

We will explore a few philosophies for attacking a mobile management infrastructure. The article will cover the differences in testing mobile stuff vs “everything else” as well as reusing some of the things you know to demystify the mobile world.
than they’re worth. First you have to find a phone to target which is usually on a person, hope they have it on and then hope it’s vulnerable to something you’re prepared before. It’s pretty difficult to craft an attack in just a few seconds as the target is walking by. I digress, the real gold is in the management infrastructure of these devices (where it exists) because most likely it contains all the information in the phone anyway. It’s also probably a much easier and more practical target.

I’m also not going to focus on any one management infrastructure as I would like to keep it generic enough to apply to as many as possible. As I looked at these various management tools most of them seemed to have a few things in common. First, they’re almost all web based with a database backend, does that sound new, exciting or cutting edge? I hope not. That’s right though, most of these cutting edge, high end management infrastructures are simple web apps. Do we need to break out our compilers and start composing custom attacks yet? Probably not. Let’s look at a few ways to approach the problem, without doing anything crazy.

**Attacking the Front**

Now that we have determined most of these are basically web apps let’s look at where we can hit this infrastructure the hardest, the management interface. If you have either been a penetration tester or a web application developer, I’ve been fortunate (I think) to have been both, then you know a dirty not-so-secret secret. Developers, administrators and IT management do not take management interfaces terribly seriously. If it’s an *inside the firewall* test you are nearly guaranteed to find a few open admin interfaces typically with default credentials. Of course I’m certain this won’t happen with anything as important as a mobile management infrastructure but just in case let’s continue our attack on the front.

Theoretically every web application that interfaces with a database has a SQL Injection (SQLi) vulnerability of some sort. Bold statement? Not really, just based on years of experience, I’ve met very few with no exploitable vulnerability. Fast forwarding though let’s say our management infrastructure has a SQLi vulnerability and we can insert records, let’s look at all we can do with that. First and foremost we can probably enroll our own phone and figure out what the management infrastructure does to a phone. Second we will be able to push our own malicious code to the entire enterprise. From a penetration testing perspective it’s not going to get much better than that. Fortunately I’m sure all of these various infrastructures have undergone many rounds of security testing and hence it just won’t be this easy.

Moving on.

**Attacking the Middle**

A few things I noticed while taking my tour of the various management suites (aside from how *cute* all their names are) is that almost without exception they all included some sort of *enterprise app store* though they gave them various names. This piqued my interest for several reasons outside of just attack vectors though having an app store front-end presents us with the same vulnerabilities as the management system’s admin interface. This one is interesting from a purely logistical perspective because I’m curious who is doing *quality control* on the apps getting pushed out. Can anyone submit an app? Most of the vendor website’s weren’t very clear on this matter and I was on a tight deadline. At any rate, the workflow in these systems would be very interesting to analyze. Like it or not (I don’t particularly) but Apple’s policy of app review before app store submission probably catches most malware. More companies should take note for their *enterprise* mobile apps and adopt a similar policy I think.
At any rate, I think these enterprise app stores make excellent points of attack if you can get them to do what you want. That’s a lot of if’s because again I’m certain these things are tested beyond normal industry standards, wait, they’re not? Of course not, it’s that dirty little not-so-secret secret again, you can’t really restrict access to these app stores since every phone has to have access (this is the hard part of true mobility) and they’re web applications, once again, so same SQLi rules apply. This middle layer provides another great entry into the infrastructure, again if you can get it to do what you want.

Attacking the Mobility
One aspect of this that I was initially trying to avoid but really can’t be is what I’ll call the provider influence. Most enterprises, and if you’re not sick of that word by the end of this article there is something wrong with you and I’ll tell you why at the end, choose a single provider, in the US it’s Verizon, AT&T and a bunch of others. An interesting attack vector involving the vendor but is probably slightly harder would be an attack on the provider’s Over-the-Air (OTA) update system. Again unless you want to end up in prison this should remain a proximity attack on known target you are authorized to test and not attempted on the provider as a whole.

This could be used to push out an update of your own ROM or firmware, etc. This is advanced attacking stuff and not something I would typically recommend as I suspect the other methods would work just fine. I would use this one as sort of a last resort but it is a major flaw I don’t see considered in most of these enterprise management vendors. The provider influence might be the number one issue in the way of your mobile infrastructure security strategy. They tend to not like security much for reasons far beyond the scope of this article.

Attacking the Proprietary
I and almost every security guy I know have pretty firmly embraced Open Source software, again, for reasons beyond the scope of this article. Simply put, a system must be able to be examined to be considered secure. It has to stand up to scrutiny. Every mobile management infrastructure I came across (I narrowed it down to three initially for this article) proudly touted its proprietary nature as providing extra security. When I hear this as a penetration tester I immediately start salivating, closed systems that say things like we’re enterprise and closed so we’re better than everything else just get me going. I asked for demo copies from all three but all three said
they would be happy to give me a walk through but that installable demos weren’t available at this time. Again, a big red flag to me, especially when they all had a try it now button on their website. That’s what I get for being honest I guess. Anyway, the point is when a vendor touts their closed and proprietary nature and claims that as a feature you should usually run the other way screaming or buy something else. The funniest part of this is I had easy access to an installable trial copy of Exchange (the granddaddy of mobile management with ActiveSync, etc) and people say Microsoft isn’t easy to work with. Exchange wasn’t what I was looking for. Moving on.

Attacking the Backend (of the user)
So far we’ve attacked the front, back and various mid-sections but we’ve left the so-called backend alone. Only because I feel that these sorts of fire bomb attacks in a penetration test is cheating because often they’re too simple and involve very little actual skill. All you need to do is trick the user into installing a piece of custom software and bam, game over. Send them an email forged to be from their IT director or boss, install this or you’re fired or something more eloquent, wait for them to ignore the permissions warning most phones give you and then sit back and collect data. Sounds easy? It is, but it’s not much fun and it only proves that humans are often the weakest link. Everyone knows that already, get a real job. Sorry, that’s just a pet peeve of mine.

The key difference here is that often these mobile phones aren’t linked to any mobile management infrastructure so you don’t have a choice in attack vectors, it has to be the phone. Users would mostly never consider using their personal laptop for work but often they’ll choose to use a personal iPhone over a company provided BlackBerry (come on you would too) so as a result that iPhone isn’t the property of the company and isn’t managed by IT. This creates a management nightmare for sure and one that can be driven home more sharply by a proper penetration test. Imagine getting control of an iPad or phone that has VPN access into a corporate network but is personally owned by the user. That would probably turn a lot of executive heads and kick a mobile security program into high gear I would hope.

Bonus: Using Your Smart Phone to Perform a Penetration Test
I’m not terribly sure how practical this is but now you can even use your mobile phone as penetration test equipment. http://gitbrew.org/debdroid/ will show you the way. I plan to try this someday when I have some mobile gear to spare but again I’m not really sure a mobile phone is beefy enough to handle some of the tools folks use, we’ll see though.

The bottom line when pen testing a mobile management infrastructure is to really stick with what you know and aim for the largest payoff. Don’t try to create some new hardware device plus write software for a two week engagement when you can utilize a simple SQLi attack and own all the mobiles in the enterprise. Simpler is usually better.

Proprietary and closed systems usually make very rich targets for simple attacks and that boys and girls is what usually makes them enterprise. Enterprise is typically just another word for closed and as I’ve said they are also usually very brittle systems that have not been adequately scrutinized. If you have ever done one of those penetration tests where the clients says now stay away from that system, it will crash if you just port scan it, that system is usually considered Enterprise. Now go forth and test the mobile world, let me know what types of simple attacks worked for you and I’ll reprint them here (fully credited of course).

References
• How To Hack a Mobile Phone: http://video.google.com/videoplay?docid=-8987396544207653068
• How Phone Hacking Worked and How to Make Sure You’re not a Victim: http://nakedsecurity.sophos.com/2011/07/08/how-phone-hacking-worked/

BILL MATHEWS
Bill Mathews is co-founder and lead geek of Hurricane Labs, an information security firm founded in 2004. You can reach Bill @billford on Twitter and be read other musings on http://blog.hurricanelabs.com
FOCUS

ToneLoc and Load

Useful For a Pentester?

When on average it takes less than half an hour to bypass the security of many voicemail systems and the rewards can be over £250,000 for a weekends work, it’s no wonder that phreaking telephone systems is enjoying a resurgence.

Written off by many as Old Hat or Lo Tech and definitely Belonging to the 1980’s does the Wardialler still have a place in the modern pen testers toolkit?

I would suggest that this question is best answered by someone that is currently suffering from a Theft of Service attack against their PBX and is haemorrhaging cash at £30 to £40k per day.

The attack may not be new, the technology may have been around for many years but it is still very effective and increasingly popular.

Wardialling originally was the practice of dialling all of the telephone numbers in a range in order to find those which were answered with a modem. These days it is probably more accurate to say that the goal is to classify all the responses as accurately as possible, in fact if you visit the web sites for the last two war diallers in my brief timeline you will see that both make a point of saying that they can classify / attack PBX and voicemail systems.

Wardialling first came into the spotlight in the 1983 film War Games where David Lightman, the hacker, uses a wardialler, appropriately called the war games dialler to unwittingly accesses WOPR, the supercomputer, which is programmed to predict possible outcomes of nuclear war and he nearly starts World War III. As with most things to do with computing the original name just has to be shortened so the war games dialler became the war dialler.

1993(ish) ToneLoc: (http://downloads.securityfocus.com/tools/auditing/pstn/tl110.zip) Short for Tone Locator was created by Minor Threat and Mucho Maas. It is DOS-based but also runs on Win95+ platforms. It dials numbers, and saves the login session to be viewed later.

1995 THC-Scan, the worlds most used cross platform wardialler was released and approximately 10 years later THC-Scan evolved into THC-ScanNG (Next Generation). Once again van Hauser created a masterpiece; TSNG was distributed if you have a pool of 1000 modems – no problem! One master server could control a vast array of zombies allowing the war dialling to be controlled remotely. TSNG can be downloaded from http://thc.org/thc-tsng/.

1998 Sandstorm (now NIKSUN) released PhoneSweep the Corporate War Dialler. PhoneSweep offered a safe platform (no hackers using it to distribute Trojans) which utilised a GUI interface running under Windows 95. PhoneSweep is still available (commercially) today. PhoneSweep offers three distinct modes Connect, Identify or Penetrate and is capable of classifying phones, faxes and modems in a single call utilising their patented Single Call Detect methodology. Additional product information is available from http://www.niksun.com/product.php?id=17.

2001 SecureLogix release version 3 of their Telesweep Wardialler, Telesweep offers both passive: (the first call into a number is in voice mode – no tones are sent)
and active mode where tones are sent immediately. Telesweep, like PhoneSweep, also has a penetration mode where it can carry out a dictionary attack against any suitable target that it locates. Telesweep is available for download from http://www.securelogix.com/modemscanner/index.htm.

The big problem with modem based wardiellers is that they are slow, a single modem being capable of around 40 calls per hour at best, so it can take a long time to dial through a complete number range.

iWar

iWar: https://www.softwink.com/iwar/ The intelligent War Dialler: Once again we have a system which is capable of controlling multiple modems, however iWar goes further, iWar is capable of supporting IAX trunks. This not only allows you to scan without any additional hardware but it also opens up the world of caller ID spoofing; additionally the use of IAX trunks gives an increase in call throughput.

WarVOX

WarVOX: http://warvox.org/ this is a rather unusual dialler in as much as there is no modem support at all; all calls are carried over IAX trunks. Now we get some serious throughput a couple of trunks giving access to 40+ concurrent trunks would enable you to dial through a 10,000 number range within 3 hours. Another unusual aspect of WarVOX is that it uses the audio stream to classify the call thereby allowing it to identify Voicemail boxes, IVRs, and PBXs, as well as modems and fax machines.

If you don’t have access to an IAX trunk, don’t worry. I have WarVOX running on a virtual machine with an Asterisk PBX (http://www.asterisk.org/asterisknow/) on another. The Asterisk PBX working as a bridge between IAX and any other type of trunk you like. Or even WarVOX via IAX to Asterisk; Asterisk via SIP to your corporate PBX – now you can call out over your office telephone lines.

What are we detecting?

Obviously modems, potentially offering an unsecured method of accessing a network but I am more interested in telephony than data, so what do I get?

Modems, remote access maintenance modems, to be more specific, many of which will have default or very basic passwords, with these I potentially have access to the telephone system programming interface. Additionally these remote maintenance modems may offer access to the administration interfaces of the entire telephony estate including such goodies as voicemail and IVR servers, centralised management servers and contact centres; the list is very nearly endless.

IVR and voicemail systems; many telephone system users, including administrators, are extremely lax with their password management, making voicemail an interesting source of both corporate and personal information; additional services may also be available through voicemail and IVR ports. On one 700 user system I audited 54 users had 1234 as a passcode a further 37 had 1111. Another company’s corporate policy was that the voicemail passcode MUST be the last four digits of your telephone number.

DISA (Direct Inward System Access) this is a facility found on many different types of telephone system, although some manufacturers secure it by giving it a different name! DISA is a facility whereby you dial into the corporate telephone system, usually via a Direct Dial Number and then you are presented with secondary dial tone and in many cases this ‘secondary dial tone’ is totally unprotected. Once you have tone you dial 9 followed by any valid number and if you are connected, which 99 times out of 100 you will be, pwned!! You can now dial in and out of the system and mostly you can dial anything you want be it mobile, international or premium rated numbers.

Why attack telephone systems?

Disclosure of information

Data disclosed without authorisation. Examples include both eavesdropping on conversations and unauthorised access to voicemail messages.

Data modification

Data altered in some way by reordering, deleting or modifying it. For example, an intruder may change billing information, or modify system tables to gain additional services. Equally with VoIP traffic it may be possible to inject additional data into the audio stream; so there you are explaining to your partner how late you have to stay at work while the hacker injects audio from a night club.

Unauthorised access

Actions that permit an unauthorised user to gain access to system resources or privileges. For example A group of Chelsea Football Club fans hacked into Manchester United’s Phone system and replaced their normal out of hours message with chants of We are the champoins [1].

Denial of service

Actions that prevent the system from functioning in accordance with its intended purpose. A telephone system may be rendered inoperable or forced to
operate in a degraded state; for instance if you have four telephone lines with a call on each you can’t answer a fifth call (TDOS – Telecommunication Denial Of Service), equally if your VoIP system is busy dealing with a flood of SIP Invites it may not be able to process any calls (DOS or DDOS – Distributed Denial Of Service).

**Traffic analysis**
This is information gathering, for example call records (SMDR Output from a PBX) from which inferences can then be drawn.

For example, an intruder that observes a high volume of calls between a company’s legal department and lawyers specialising in acquisitions, could conclude that the company is about to expand by purchasing another business.

**The Terrorist Threat**
The French authorities that studied the terrorist attack on a Madrid commuter train in 2004, investigated whether the bombers hacked into the telephone exchange of a bank near Paris as they were planning their attack [2].

**Theft of service**
Toll fraud – (toll fraud), probably the most common of motive for attackers.

Toll fraud is a popular attack because it is a high profit low risk enterprise. Compromised systems are daisy chained together with Dial Through services (in much the same way as a hacker would use multiple proxies) to hide the origin of the attack. The PBX at the end of the chain is then used to dial premium rate numbers and the attacker takes the profit from these numbers.

**VoIP versus Modem Diallers**
The main advantages of VoIP diallers over Modem diallers are:

**Throughput**
If you customer has a number range of 100 numbers it is possibly bearable to test them with a modem as it will only take a few hours but if your customer base includes any major companies with say a 1000+ number range then you are going to be looking at days to scan the range with a single modem or an hour with something like WarVox.

**Fingerprinting**
The ability to detect and identify a greater variety of endpoints. With the current generation of war diallers you can achieve much more than just identifying dialling and modem tones. When you analyse calls with WarVox you are presented with both signal and spectrum graphs and although WarVox does not currently have the ability to automatically group or audio fingerprint the data once future versions contain this functionality the possibilities will be phenomenal. Automatically identifying the PBX by correctly fingerprinting the embedded voicemail system for a start! Potentially all you would need would be reference recordings of standard voicemail / IVR prompts.

And anything which tests faster and fingerprints more accurately MUST make life easier.

Of course life never just gets easier; there must also be a downside:

**Complexity**
You now have to understand about configuring an IAX trunk instead of popping in a modem driver disk. You will also have to understand the complexities of setting up call timers; if you make the call too long you may alert your customer to your testing by leaving them a lot of silent voicemails. You will also have to cope with a wider variety of output and until audio fingerprinting is available that could be time consuming and in the case of WarVox, that could be a lot of MP3s to listen to.

**Expense**
The software may be free but the IAX trunk won’t be, so if you don’t have the customer base or the right type of customers for this type of service the IAX trunk may be too much to swallow.

**Cautionary note**
Not all ITSPs are happy to let you use a war dialler across their network, so it pays to check this out before committing to any form of contract.

**Some theoretical toll fraud figures**

**Voicemail Passcodes**
If a voicemail system has 4 digit pin numbers then the number range is normally 1000 to 9999 (people generally don’t seem to like using the numbers that begin with zeros) which equates to 8,999 permutations. If the system can accept twenty simultaneous inbound calls (20 port voicemail) and it takes approximately five seconds to input a PIN number, then the life expectancy of a four digit PIN code is approximately 38 Minutes:

\[ \frac{(8999 \times 5)}{60}/20 = 37.4958 \]

On single port voicemail the life expectancy increases to 12.49 Hours.
Projected Revenues
Let’s assume that you have a bank of Premium Rate telephone numbers (probably in a foreign country) where each number is capable of generating £5.00 per minute.

If you can gain access to the target telephone system and start it dialling out at 5 O’clock on Friday evening (as they finish work) and if the target company is fast enough to terminate your calls at 9 O’clock on Monday morning, then you have been generating calls for 64 hours.

Therefore we have 3840 minutes at £5.00 per minute = £19,200
But if you can generate 10 simultaneous calls = £192,000
Or maybe 20 simultaneous calls = £384,000
Not bad for a weekends work!!!!!!

Telecom fraud is estimated at US$52-60 billion per annum, globally and is currently growing ay 15% per annum [4]. (Communication Fraud Control Agency)
Telecom Fraud in the UK costs £1.3 billion annually and 40% of companies have experienced it [5].

CHRIS MCANDREW

After completing college with a HND in electrical engineering Chris worked for a number of years as a diagnostic engineer on avionic systems. Having been made redundant he started working for a telecom company while looking for a “proper” job and 27 years later, as he admits, he’s still on the lookout. He is currently employed as a specialist engineer by one of the largest privately owned telecom companies in the world where (amongst other things) he is technical lead for anything to do with hacking.

CONTRIBUTE

PenTest Magazine is a community-oriented magazine. We want IT security specialists and enthusiasts to work together and create a magazine of the best quality, attractive to every individual and enterprise interested in the penetration testing field.

If you are interested in being a part of our community – submit an article or bring up a subject you consider important and up-to-date. Are there any trends on the market you’d like to take a closer look at? Are there any tools or solutions worth reviewing or presenting to the community? Are there any touchy and controversial issues you feel have to be discussed in public? Then share your opinions with us.

If you run an IT security company, your contribution is the most welcome. Tell us about your solutions and advertise in the magazine for free, or have a special issue devoted exclusively to you. As long as you provide top-notch, non-commercial writings, we are always ready to cooperate and help your company develop with us.

Are you a student? We’re looking forward to you articles! Fresh attitude, opinions and beliefs of the young and budding IT security gurus are invaluable for us. You will give your career a great start when you write to a respectable IT magazine. Showing an issue with your name among the names of other authors – and often famous ones – will be your great asset during a job interview.

If you think you don’t have enough time to create an article from scratch, but feel interested in the magazine – become one of our beta testers. This way you will get the opportunity to look at a new issue’s contents before it’s even published, and your name, too, will appear in the magazine. If you feel the need to contribute and share your knowledge, but don’t have enough spare time for creative writing – beta testing is just for you.
Inside Android Applications

By the end of 2011, the number of Smartphone shipments around the world will explode to nearly 468 million units and the android operating system would have a fifty percent market share. This would increase the number of attacks on mobile applications and also the investment in securing the applications from the attacks.

The most important part of performing an application pentest for an android application is understanding the manifest configuration. Analyzing a manifest file is one of the most important and tedious task while performing a penetration testing assessment on the world’s most popular mobile Os.

Android is a privilege-separated operating system, in which each application runs with a distinct system identity. At install time, Android gives each package a distinct Linux user ID. The identity remains constant for the duration of the package’s life on that device. On a different device, the same package may have a different UID; what matters is that each package has a distinct UID on a given device.

Every android application must have an AndroidManifest.xml file in its root directory. The manifest presents essential information about the application to the Android system. High-level permissions restricting access to entire components of the system or application can be applied through the AndroidManifest.xml. The manifest file does the following:

- It describes the components like the activities, services, broadcast receivers, and content providers that the application is composed of. These declarations let the Android system know what the components are and under what conditions they can be launched.
- It determines which processes will host application components.
- It declares which permissions the application must have in order to access protected parts of the API and interact with other applications.

Figure 1. AndroidManifest.xml natively obfuscated

Figure 2. Decoding apk application file
• Download the following tools
  • apktool-install-windows-file
  • apktool-file
  • Unpack both to your Windows directory.
  • Now copy the APK file also in that directory and run the following command in your command prompt (see Figure 2):
    `apktool d app.apk ./app_decrypted`
  Here app.apk is your Android APK file
  • This will create a folder `app_decrypted` in your current directory. Inside it you can find the `AndroidManifest.xml` file in decrypted form and you can also find other XML files inside the `app_decrypted/res/layout` directory.

The manifest contains juicy information like permissions, intent filters, and lots more. A typical manifest file is shown below (see Figure 3).

Some of the important configuration settings to look for while analyzing a manifest file: Table 1.

### AndroidManifest.xml file plays a very important role in analyzing the security of Android mobile applications. The file is of great interest when analyzing system security because it defines the permissions the system and applications enforce.

Android packages are `.apk` file. For the test purpose you can download any android application and extract it and you will see the `AndroidManifest.xml` file which would be difficult to open (see Figure 1).

Below I have mentioned step by step methodology to open and review it.

**Table 1. Android malware analysis tools**

<table>
<thead>
<tr>
<th>Setting</th>
<th>What to check</th>
<th>Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>android:installLocation</td>
<td>If it is set to “auto”, The application may be installed on the external storage, but the system will install the application on the internal storage by default. If the internal storage is full, then the system will install it on the external storage. Once installed, the user can move the application to either internal or external storage through the system settings</td>
<td>Use &quot;internalOnly&quot; value for this setting.</td>
</tr>
<tr>
<td>android:protectionLevel</td>
<td>Characterizes the potential risk implied in the permission and indicates the procedure the system should follow when determining whether or not to grant the permission to an application requesting it.</td>
<td>Check if the value is set to &quot;normal&quot; or &quot;dangerous&quot;. If it is set to &quot;dangerous&quot;, check the permissions.</td>
</tr>
<tr>
<td>android:persistent</td>
<td>Whether or not the application should remain running at all times – “true” if it should, and “false” if not. The default value is “false”.</td>
<td>Applications should not normally set this flag. It should be set to “false”</td>
</tr>
<tr>
<td>android:restoreAnyVersion</td>
<td>Indicate that the application is prepared to attempt a restore of any backed-up data set, even if the backup was stored by a newer version of the application than is currently installed on the device.</td>
<td>Setting this attribute to true will permit the Backup Manager to attempt restore even when a version mismatch suggests that the data are incompatible</td>
</tr>
</tbody>
</table>

It also declares the permissions that others are required to have in order to interact with the application’s components.

It declares the minimum level of the Android API that the application requires.

It lists the libraries that the application must be linked against.

And moreover, it names the Java package for the application. The package name serves as a unique identifier for the application.

**Figure 3. Example of AndroidManifest.xml**
Almost daily we can see on the news that a new IT system has been attacked by hackers. Even if it is about Sony [1] or the CIA website [2], these attacks, harmful in 90% of the cases, show that behind there lies a competent community who has a high IT security potential. We ask ourselves then: Where do these hackers come from? Are they employed professionals? Do they act with a well-defined purpose, or are they just smart individuals who don’t know what else to do with their knowledge and free time?

The beliefs of a hacker may be not easy to understand and gloomy. A hacker’s profile can extend from a rogue high-school teenager to an experienced professional. While some hackers have the chance to fructify their knowledge in a legal environment, others gain their living following illegal activities. Nevertheless, they all share a common passion for IT security and they have an important potential.

As the modern cybercrime is continuously developing and turning into a financial motivating market, there is a strong need of reinforcements. We should give to every IT-security talented person the opportunity to show their skills and use them for a good cause. Why not use their passion in order to turn them to the right side.

Current situation

Over the last couple of years, an interesting trend is visible in the world of IT: large companies start paying money to people who find vulnerabilities within their products. For example, Mozilla has been rewarding people who found security weaknesses of their well-known browser [3]. Google is also running a very well paid bounty program for their chrome browser and their websites and are ready to pay important amounts of money [4]. Facebook also adopted this new trend and started at the end of July 2011 to reward vulnerability researchers [5].

A possible explanation for this recent action may be the fact that companies start to become aware of the potential skillfulness that hackers might possess. Consequently, the companies start to cooperate with the hacker communities, instead of taking legal action against them (like Sony did for example [6]).

Considering that the cooperation between hackers and companies can stand while there is enough benefit on both sides, the startup Hatforce came up with an idea.

The idea

Hatforce.com came up with an idea which can be called an open market crowd-sourcing platform for penetration tests. The principle is simple: using the worldwide hacker community in order to find vulnerabilities in every IT system possible (websites, servers, software, etc.) and reward them for the vulnerabilities they found.
The concept in five steps:

1. Clients go on the website hatforce.com and publish a penetration test. At this moment, they specify a fixed reward the testers will get if they submit a valid vulnerability.
2. Penetration testers select a penetration test they want to participate to.
3. Once they have found a vulnerability in the client’s product, the testers submit a detailed description on hatforce.com.
4. The client gets the vulnerabilities descriptions and approves them.
5. The client pays then to the testers the specified reward for each approved vulnerability.

Let’s analyse more in depth the different steps:

1. A client is an owner of one or more IT systems (a website, server, software, etc.) and can publish a penetration test request for one of his products on the website of Hatforce. For each published request, he has to specify a fixed reward per vulnerability and how many vulnerabilities he wants to pay. In this way, he is sure about how much he will pay at maximum and minimum for each test.

   The principle is simple: using the worldwide hacker community in order to find vulnerabilities in every IT system possible (websites, servers, software, etc.) and reward them for the vulnerabilities they found.

   For example, if the client wants to reward a vulnerability with 100€ and at maximum 3 vulnerabilities, he will pay:
   • 0 € if no vulnerabilities are found
   • 100 € if only one vulnerability has been found
   • 200 € if only two vulnerabilities have been found
   • 300 € if 3 vulnerabilities have been found

   Furthermore, the client is sure not to pay more than 300€ because he has specified that he wants to pay at maximum 3 vulnerabilities.

   The scope of a penetration test can be defined from the beginning. For example, if the client wants to test its website only for Injection and Cross-Site-Scripting vulnerabilities, he is able to limit the test by clicking on I define myself the target vulnerabilities and checking the check-boxes Injection and Cross-Site Scripting (XSS). Consequently, only the vulnerabilities which are related to these types of security issues will be rewarded.

   When the client has entered all the necessary information for the test, a contract between the client and the testers is established. This contract is very important as it legalizes the penetration action. In most of the countries a penetration test can only be carried out if the owner of the system agreed upon it. By accepting this contract, the client agrees to let the penetration testers attack his product within the established scope.

   After the acceptance of the contract, the penetration test request will be forwarded to an administrator of Hatforce. The administrator might eventually contact the client if there are some ambiguities related to his test request and in order to ensure that the client is requesting a test for a product he really owns. Once all is clear, the administrator approves the penetration test which will be consequently published on the website.

2. After being approved by the administrator, the penetration test is visible to all the registered testers.
of Hatforce. Before they can see any details about the test, the testers first have to sign the contract with the client. This contract also includes an NDA (non-disclosure agreement), so that the testers shall not divulge any information about the test to third parties.

Crowd-sourcing seems not so well suitable for white-box penetration tests on proprietary products, since the source code cannot be made available in public.

The testers are then able to start testing the client’s product. During the testing they should fill in a so-called test procedure.

The test procedure is basically a description of what the tester has done during his test. The client will be able to visualize the descriptions and see what kind of actions the tester has performed on the product. This gives the client the possibility to verify that no illegal hacking attempt has been conducted.

3. Once a penetration tester has found a vulnerability during a test procedure, he has to submit it on the website of hatforce.com. The client gets immediately informed about it and should test it.

4. The client should be able to easily test the vulnerability by following the description provided by the penetration tester. If no any other tester has found the same vulnerability before, the client has to pay the tester the specified reward.

5. If the client approves the vulnerability, he pays the established reward to the corresponding tester.

The market

Nowadays on the IT Security market, a penetration test costs approximately 200$ per hour and can become much more expensive, depending on the experience of the penetration tester. Usually the client has to give the most of information to the tester about its product, because it is too expensive for a professional penetration tester to collect all the information freely available by himself. Consequently, pure black-box tests are not conducted very often. Unfortunately, exactly these tests represent at best an attack by malicious persons.

On the other side, there is the white-box testing. White-box tests make sense when the penetration tester is highly skilled and able to analyse source code in order to find vulnerabilities. Crowd-sourcing seems not so well suitable for white-box penetration tests on proprietary products, since the source code cannot be made available in public. Nevertheless, white-box tests are possible for open source products and could maybe prevent people from hiding backdoors within them [7].

There exist already on the market a couple of platforms which also involve crowd-sourcing in their business model. One example is the Zero Day Initiative (ZDI) from HP which rewards security researchers for the vulnerabilities they find in very popular IT-products. However, their platform is different than Hatforce.

ZDI makes security researchers an offer for submitted vulnerabilities, while on Hatforce a reward for a vulnerability is entirely set by the client so that testers know in advance how much they will be rewarded. By letting the client to freely define the amount of money he is willing to pay per vulnerability, an evolving open market for pentesting is created.

Security testing is very expensive nowadays and consequently, private people or even small and middle sized companies usually do not want to invest in this domain. However, with the increasing amount of hacking attempts that have been conducted over the last couple of months by hacker groups like LulzSec, even the smaller companies will have to take action and test their products. Therefore, penetration testing should become accessible to all types of companies.

![Figure 2. Example of a test procedure description on Hatforce.com](http://pentestmag.com)
and not only to those that have the means to pay large amounts of money.

**How does Hatforce makes money**

Hatforce connects the clients who want to test their products and the testers interested in being rewarded for finding vulnerabilities.

In order to make the service the most usable for clients, Hatforce offers them consulting services. These consulting services are applied to the vulnerabilities found by the testers. When a tester describes a vulnerability description, the client has to verify the vulnerability. If it is a correct vulnerability and it respects the testing scope of the test, it will be approved and paid.

Knowledge and time is necessary to verify the technical details of a vulnerability. As a consulting service, Hatforce can test the vulnerability descriptions and say if the vulnerability has to be approved or not. For an approved vulnerability, Hatforce can advise the client on how to fix it. Furthermore, once a vulnerability has been fixed, Hatforce can perform a retest of the product and verify if the problem has been correctly closed.

As this system would not scale if a lot of clients start using the platform in the future, Hatforce considers subsequently involving some of the testers in its consulting system. A new category of testers would be created – the certified testers. The certified testers would agree on providing consulting services for the security vulnerabilities of a client’s product and they would also be entitled to approve or disapprove vulnerabilities.

**Difficulties and their solutions**

There are certainly several difficulties and limits that crowd-sourcing brings which are described in the following paragraphs.

**Is it possible to trust an unknown tester?**

In order to have access to a test, the tester engages himself in a contract with the client and accepts its terms and conditions. This ensures that his testing activity is entirely legal. Nevertheless, there are always villains of both sides (clients and testers) that are not eager to respect any rule. Therefore, Hatforce introduces an evaluation system for testers and clients. If a tester submits a vulnerability, the client is able to evaluate his work by giving him a +1 or a -1 followed by a compulsory comment that explains his decision. The same procedure is possible in the reverse way – a tester can evaluate in the same way the company.

**Can the tester be sure that the company will pay?**

It is possible that some clients don’t want to approve vulnerabilities or pay their testers. After the client has evaluated one of his submitted vulnerabilities, the tester is able to evaluate the company. If a client does not approve a vulnerability description, he should state why and the tester can then respond with an evaluation.

A client should be aware that if he has not enough positive evaluations (or too many negatives) then the testers will most likely not participate at his tests anymore.

**The Client will try to contact the testers of Hatforce directly**

A client may try to contact directly a penetration tester without going through the platform of Hatforce. However, it is in the own interest of the client to use the global knowledge of the community in order to find security problems. The more testers get to test a product of a client, the better it is for the client. If a company engages a tester in private, the costs will be comparable to a standard expensive penetration test and the client loses the benefit of the system of Hatforce.

**Somebody could request a penetration test for a website or server he is not the owner of**

Hatforce has to make sure that the client is really the genuine owner of the product he wants to have tested. Therefore, Hatforce will contact the client each time he sends a test request and will validate the ownership.

**Is a penetration test legal?**

A penetration test is generally illegal if there exists no specific agreement between the owner of the product to be tested and the tester. Therefore, a written contract is needed between the client and the tester. On Hatforce.com clients can use the classic contract which has been checked by a German lawyer or contact Hatforce and propose a customized contract.

The client should be aware that it is the best for him if he states the most clearly what he is awaiting from the testers and if he wants to forbid anything during the tests.

**Is it practical to request a crowd-sourcing penetration test on a production system?**

Google and Facebook are doing it now. Usually if someone finds a XSS vulnerability this does not stop a website from working. If someone tries to trigger a Denial of Service attack on a production system, or tries to exploit a buffer overflow, this might be a more serious
attack. Therefore, Hatforce recommends the client to make a test on a copy of their website in order to not disrupt any productivity.

Nevertheless, sometimes it is not easily possible to make a copy of a website, this demands resources and it does also change the value of the test since software and configuration files may be slightly changed. Consequently, every test should be planned carefully and the client should be aware of the associated risks. Data backups and intensive communication with Hatforce might help mitigate the risk at most.

Is crowd-sourcing replacing the standard penetration test?
Depending on the size of the company and many other factors, there may be that crowd-sourcing would not be suitable for a penetration test. For example, large companies will most likely want to employ somebody who will check their entire network infrastructure from inside. Crowd-sourcing cannot be applied to such demands. Nevertheless, the recent example of Mozilla, Google and Facebook shows that the crowd-sourcing principle has been accepted and is actively in use.

Future improvements
Hatforce is for the moment just starting its service. There are many improvements that can still be done and the community can help to refine the model to their needs.

Once the evaluation system will start to be used, clients will get the best testers depending on their needs and testers will be able to choose the good clients which have proven to pay the testers.

Hatforce is actually using its own platform to test its own website! There might be some vulnerabilities, although we hope there are not really easy ones, and testers will be rewarded for finding them.

Conclusion
An open market for crowd-sourcing for IT-security testing is a new and efficient method to quickly get as many testers as possible. If Sony had used this possibility with a reasonable reward per vulnerability, testers would have found the simple SQL injections that have been made possible to hack its websites. And Sony is just one of the most popular examples. Who knows how many easy and critical vulnerabilities are still out there, not only in the systems of big companies, but also in the products of small and middle sized companies.

Hatforce offers an efficient and competitive testing opportunity compared to standard penetration tests, since there is no money to be paid if no vulnerabilities are found. Furthermore, through crowd-sourcing, every hacker gets the opportunity to use his knowledge in a good way. Converting illegal hackers in white hats who help other people fixing the security vulnerabilities of their products is the best result which can be achieved through crowd-sourcing.

References
- LulzSec states that the website of CIA is offline, https://twitter.com/#!/LulzSec/status/81115804636155906, 14.06.2011 [2]

ARTHUR GERVAIS
Arthur Gervais is a passionate student of IT-Security, currently in Sweden at the Royal Institute of Technology – KTH in Stockholm. Since an early age, he is determined to work in the field of IT-Security. He is the founder of the startup Hatforce which offers crowd-sourcing for IT-security tests. He has won recently the “Best Student Award 2011” offered by the German Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik). In 2012 he will achieve his Master degrees from INSA de Lyon (France), KTH Stockholm (Sweden) and the Aalto University in Helsinki (Finland).
The information security landscape is rapidly changing. Are you ahead of the game?

With information security threats becoming more targeted and sophisticated, how can you and your organisation stay on top of the situation?

Find out at RSA® Conference Europe 2011 - the place for Europe's smartest information security professionals who want to discover the latest trends, technologies and threats affecting the industry. Benefit from:

- 70 educational track sessions
- Keynotes from industry thought leaders
- Interactive programmes
- Demonstrations from leading vendors
- Time to meet and collaborate with peers

Be educated. Be informed. Register now.

www.rsaconference.com/2011/europe/pen
Building Your Own Pentesting Application

Although even today web browsers serve the primary purpose of bringing information resources to the user, they no longer represent a software application with bare bones support for just HTML. Today, web browsers like Mozilla Firefox come with the support of add-ons, which are small installable enhancements to a browser’s foundation.

These add-ons when installed inside a browser can add additional functionality to the browser and this additional functionality can be used on the web pages that are viewed by the user.

The best part about these add-ons is that they enable third-party developers to add new features without interfering with the original source code of the host application. These add-ons are dependent on the services that are provided by the host application to register themselves. Thus, third party developers can update their add-ons without making any changes to the host application as the host application operates independently. These add-ons can serve for scatterbrained as well as for informative purposes like hacking, penetration testing, and more.

Pen Testing Add-ons

Tor
Tor: Experts always suggest that it’s best to hide your identity before getting involved in any security related operations. Tor allows user to maintain online anonymity. Tor basically has a worldwide network of servers that helps route the internet traffic and thus, disguise a user’s geographical location. The best thing about Tor is that it’s open-source and anybody can use Tor network for free.

• To setup Tor, you need to first download the Tor Browser Bundle from Link: https://www.torproject.org/download/download.html.en. This bundle will will ask your permission to extract a bundle of files to the location where Tor installer was downloaded.

• Now, Start Tor Browser. Once you’re connected to the Tor Network, the browser (Firefox 3.6.20) will automatically open up with a congratulations message that your IP address is now changed. For example, my IP address changed to 85.223.65.238, which is located in Netherlands.

WHOIS
WHOIS: Internet resources such as domain name, IP addresses or controller systems are registered in database systems. WHOIS is used to query the
name, IP address and server location. Right clicking on the flag will let you access additional information about the web site using external lookups such as DomainTools WHOIS, WOT Scorecard, McAfee SiteAdvisor and many more. You can also add additional lookups which you find necessary. Clicking on the flag icon will by default take you to Geotool, which Flagfox’s default action (see Figure 2) (Add-on Link: https://addons.mozilla.org/en-US/firefox/addon/flagfox/).

**Exploit Database Search**

Exploit Database Search: The Exploit Database (http://exploit-db.com/) is an archive of more than 15000 exploits and software vulnerabilities. This exploit database is a great place for information security researchers and penetration testers for getting an exploit’s information in plain text format.

Offsec Exploit-db Search: This add-on simply adds Offsec Exploit Archive search among other installed search engines in your Firefox. (Add-on Link: https://addons.mozilla.org/en-US/firefox/addon/offsec-exploit-db-search/).

**SQL Injection**

SQL Injection: Database applications are critical in today’s web scenario. If a database application is unable to filter out escape characters then it becomes very easy for malicious users to perform SQL code injection on a vulnerable application. Using this, a malicious user can gain access to the server and can delete or modify records. Recently websites like Kathmandu Metropolitan City, Metropolitan UK Police, Nepal Telecommunications Authority, BART Police Database and NASA Forum were exposed of the SQL databases for cognizing the data about the resource, assignees, registrants and administrative information.

Flagfox: This extension introduces a flag icon on the right hand side of your address bar (see Figure 1). This flag shows the web server’s physical location. Hovering over the flag will display information such as

![Flagfox in Firefox 6.0](image1)

Figure 2. Flagfox in Firefox 6.0

![Geotool lookup of a website](image2)

Figure 3. Geotool lookup of a website
injection vulnerability. Thus, SQL injection plays an important part in any pen testing routine.

**SQL Inject Me 0.4.5** This add-on comes from a leading information security firm—Security Compass. This add-on will test a website for SQL injection vulnerabilities by substituting HTML form values with crafted database escape strings that are used in an SQL injection attack. Although this extension will not try to expose the security of a website, it’ll look for database error messages in the page. Hence, just like a web vulnerability scanner, this extension will enumerate the possible entry points without intruding into the system. (Add-on Link: https://addons.mozilla.org/en-US/firefox/addon/sql-inject-me/).

- To use this add-on you need to go to Tools > SQL Inject ME > Open SQL Inject Me Sidebar.
- Once you’re at a login page or on a HTML form, you can test this add-on by clicking the ‘Test all forms with all attacks’ buttons in the sidebar to test that particular page (see Figure 5).

**Cross-site scripting**

Cross-site scripting (XSS): XSS vulnerability is usually found in web applications. In this attack, a malicious user crafts a URL of a vulnerable website in such a way that when the malicious code is executed then client’s session cookie is sent to the malicious user. This enables him to steal sensitive information from client’s account. The crafted malicious link can easily embedded a HTML document inside a frame using inline HTML frame tag `<iframe>…</iframe>`. Recently websites like Bing.com (MAPS), Google Appspot, Forbes, EC Council and Samba Web Administration Tool (SWAT) were exposed of the XSS vulnerability.

**XSS Me 0.4.4**: This tool works in the same way as SQL Inject Me. This add-on detects reflected XSS vulnerabilities and points out the possible entry points for an attack. This add-on shows the resulting HTML page as vulnerable only when JavaScript value `document.vulnerable=true`). XSS Me comes from SecCom Labs. (Add-on Link: https://addons.mozilla.org/en-US/firefox/addon/xss-me/).

**Access Vulnerability**

Access Vulnerability: Web servers can sometimes be affected by file access vulnerability where a malicious user uses a mere web browser to get unauthorized access to the files stored on the server. This vulnerability doesn’t allow the malicious user to delete, modify or create a file; the user can only read or copy the file from the computer. The malicious user gets access to the file by specifically requesting its name by using a non-standard URL for bypassing the file access controls of the server.

**Access Me 0.2.4**: Web applications affected by access vulnerability are tested with four different methods. File access requests are sent by using session...
removed method, HTTP HEAD verb (retrieve whatever information in the form of an entity without returning a message-body in the response), SECCOM verb, and a combination of session and HEAD/SECCOM. (Add-on Link: https://addons.mozilla.org/en-US/firefox/addon/access-me/).

User Agent
User Agent: User agent is basically a client side application like web browser or search engine crawlers. User agent strings store information like type of application, OS, and software version. This user agent string is detected by websites for adjusting the page design layout. Hence, user agent spoofing is done by web scrapers and spam bots for forcing certain server side contents to show up by hiding the browser’s identity. For example, Android browser uses HTML rendering engine – WebKit (KHTML) and so Android browser pretends to be Safari.

User Agent Switcher 0.7.3: This add-on by Chris Pederick helps change your browser’s user agent string to Internet Explorer, Search Robots (Googlebot 2.1, Msnbot 1.1, and Yahoo Slup) or iPhone 3.0. To access User Agent Switcher go to Tools > Default User Agent. (Add-on Link: https://addons.mozilla.org/en-US/firefox/addon/user-agent-switcher/).

Tamper Data
Tamper Data 11.0.1: Tamper data can effectively be used for testing web based applications. This add-on will allow you to intercept the HTTP(S) traffic between your computer and the Internet. You can track and modify HTTP(S) headers, POST and GET request parameters. (Add-on Link: https://addons.mozilla.org/en-US/firefox/addon/tamper-data/).

- Once you install Tamper Data, go to Tools > Tamper Data. This will open a log window. Click Start Tamper from the top menu to start tampering with the HTTP(S) requests. The log will start showing you all the subsequent requests after you start tampering. To see details of a request you need to select the item and double click it to see details of a request header.

Figure 8. User Agent Switcher Menu
Figure 9. Hackbar
Figure 10. Tamper Data Log Window
Figure 11. Tamper with request

Hackbar
Hackbar: Hackbar 1.6.1 is a simple but powerful penetration and security audit tool. Basically you put a link in the hackbar and then you have to select various suitable options from the drop down menu and then just execute the edited URL. Hackbar is capable of encrypting a text or link to its MD5, SHA-1, SHA-256 or ROT13 hash format. Hackbar also has an encoder-decoder which can perform Base64/URL/HEX encoding and decoding. SQL and XSS options of this add-on will help you add statements into your URL, like for example clicking on Union Select Statement under SQL will give the output: UNION SELECT 1,2,3,4,5,6,7,8,9,10. The other amusing uses are viz., string reverse, insertion of Lorem Ipsum text, fibonacci series and more. (Add-on Link: https://addons.mozilla.org/en-US/firefox/addon/hackbar/).
If you right click an item and select *Replay in browser*, then you can modify that item’s parameters like protocol, credentials, port, host, path and you can then click *OK* to see the URI getting replayed with the modified details that you entered.

- When a request is made through your browser, a pop-up prompts you to – tamper the request, submit it without tampering or abort the request.
- Selecting tamper will show you a tamper window where you can edit the data using the context fields, after that you can submit the modified values. Tamper data is particularly useful when username and password parameters are passed through an HTTPS request. These parameters show up in the tamper window, which allows you to modify by adding SQL Injection/XSS text to the username/email and password fields.

### Conclusion

It is quite simple for a pen tester to make a good pen testing application from a freely available browser like Firefox. The Firefox project was released back on November 9, 2004, thus, the number of add-ons that are available for the Firefox community is in large numbers. This makes it a lot easier for even beginners to use third party add-ons for converting their browsers to something as strong as a hacking application. Due to the large number of add-ons available for Firefox, one should also understand that there are a lot of alternate add-ons available for doing the same XSS, SQL Injection or HTTP requests modify / replay.

In the beginning we used Tor Browser Bundle because it’s portable, doesn’t store any history logs and you can relocate the browser and Tor to any computer you like without installing anything. Also, once you install all your important add-ons on the browser that is used by Tor Browser Bundle, the add-ons stays with the browser and you don’t need to install them every time you change systems. An older version of Firefox (Firefox 3.6.20) was used because most add-ons are compatible with this version and not with the latest 6.0 or 7.0b1 versions of the browser.

**ER. DHANANJAY D. GARG**

The author holds a Bachelor’s Degree in Electronics & Telecommunication. He likes working on projects related to information security. He holds a diploma in Cyber Law and Information Security & Ethical Hacking. As a freelance writer, he often writes on topics related to computer sciences. He has written articles for journals like PenTest Magazine, Data Center Magazine and Enterprise IT Security Magazine. He can be reached out at: dhananjaydgarg1989@gmail.com.
Security Art's Red Team service operates on all fronts on behalf of the organization, evaluating all information security layers for possible vulnerabilities.

Only Red Team testing provides you with live feedback on the true level of your organizational security.

Thinking creatively! That’s our approach to your test.

Security Art’s Red-Team methodology consists of:

1. Information and intelligence gathering
2. Threat modeling
3. Vulnerability assessment
4. Exploitation
5. Risk analysis and quantification of threats to monetary values
6. Reporting

Ready to see actual benefits from your next security review?

info@security-art.com

Or call US Toll free: 1 800 300 3909
UK Toll free: 0 808 101 2722
It's more than just a conference. It's the Convergence of the Best at a World Class Event

Jeremiah Grossman
Bruce Schneier
Philippe Courtot
George Kurtz
Charlie Miller

www.hackerhalted.com