2009

SmartPot: Creating a 1st Generation Smartphone Honeypot

Michael Freeman
Edith Cowan University

Andrew Woodward
Edith Cowan University

SmartPot - Creating a 1st Generation Smartphone Honeypot

Michael Freeman and Andrew Woodward
secau – Security Research Centre
School of Computer and Security Science
Edith Cowan University
mfreema0@student.ecu.edu.au

Abstract

This paper discusses an experimental method for creating a 1st generation smart-phone honey-pot with the intention of discovering automated worms. A Honeyd low-interaction virtual honey-pot is conceived as a possible method of discovering automated smart-phone worms by emulating the operating system Windows Mobile 5 and Windows Mobile 6, along with the available TCP/UDP ports of each operating system. This is an experimental method as there are currently no known malicious smart-phone worms. Honeyd emulates devices by mimicking the devices operating system fingerprint which is created by the unique responses each operating system sends to a discrete series of TCP and UDP packets sent by the network scanner Nmap. Honeyd uses the Nmap fingerprint database for how it should emulate these responses each operating system. A significant obstacle was discovered during the implementation of the Honeyd smart-phone honey-pot, as the format of fingerprints (2nd generation) utilised by Nmap are now different to the previous format (1st generation) which is utilised by Honeyd. Honeyd cannot make use of the new Nmap format of the smart-phone operating systems and thus a honeypot for smart-phones cannot be created. Future work forecasts the creation of a technique to convert the new Nmap format to one which can be utilised by Honeyd.

Keywords

Smartphone, honey-pot, worm, Honeyd, malware, Windows Mobile 5, Windows Mobile 6

INTRODUCTION

A Smartphone is a device which unifies the functions of a cellular phone, a personal digital assistant (PDA), a digital camera and a personal computer. Most Smartphones use proprietary operating systems (OS) such as Microsoft Windows Mobile, Google Andriod or Symbian OS. An advantage of using such operating environments is third–party applications can typically be installed onto these devices and multiple applications can be run simultaneously. A common feature of Smartphones is the ability for developers and regular users to produce applications to run on these devices. Most smart-phone operating system manufacturers also offer Software Development Kit's (SDK) (Schmidt & Albayrak 2008). These kits employ a selection of software that allows developers to create applications, based on factors including physical device platform, operating system, programming language utilised and available / compatible hardware. But this feature also provides a vector which can be used to infect smart-phones with malicious software.

Malicious software, otherwise known as malware, are programs created with a malicious intent and typically include viruses, Trojan horses and automated worms. Most malware, especially for smart-phones, utilise some form of user interaction for the malware to infect the device. As of 2006, there were more than 300 pieces of malware, including worms, viruses and Trojan horses that had been created for Smart-phones. Most of these target the Symbian Operating System. (Hypponen 2006) But as other Smartphone Operating Systems gain a larger market share, they will also become greater targets.

Potentially the most damaging type of malware are automated worms. This form of malware requires no user-interaction and infects devices through automated searching of victims and attacking vulnerabilities. The only known case of a smart-phone worm was one created by Becher, Freiling & Leider (2007). The purpose of their study was to discover the amount of time and effort it took to find vulnerability and create a worm for the Windows Mobile 5 Operating System. They created a prototype worm which spread without user interaction which was achieved in 13 weeks, but were unable to discover a current vulnerability to direct the worm to attack.

Furthermore, malware creators are expected to develop more sophisticated malware such as worms including. One such attack vector discussed by Maurer (2003) was that of so-called super-worms that could intelligently attack hosts based upon the operating system fingerprint such as created by Nmap to determine which exploits to attempt. While no malware of this type have currently been detected with malicious intentions, this paper presents a 1st generation service whose purpose will be to discover any automated malware for smart-phones, if it is created as predicted (Maurer 2003).

Honeypots and honeynets are automated tools which emulate a live network, but in reality are usually running on an enclosed offline environment. Honeypots have a number of uses including identifying unusual or malicious activity on a

network, identifying and capturing malicious software or malcode, used in network defence and network forensics. Honeypots can simply emulate a network environment and return operating system variables, or they can be as complex as emulating services or even re-directing to live servers.

The methodology used in this study is experimental in nature as there are no known automated smart-phone worms with malicious intentions which can be reverse engineered to discover attack vectors. Hence the assumption is made that automated worms search for victim devices based on the operating system and / or ports available (Becher, Freiling & Leider 2007). Our method involved creating a Honeyd (Provos 2004) low-interaction honey-pot which emulates Windows Mobile 5 and Windows Mobile 6 devices. To test the viability of this method, Nmap (Lyon 2008) network scanner was to be utilised to determine if the honey-pot correctly emulates the devices to a scanning system, like an automated worm may perform. Lastly our results are provided and future work is predicted.

METHODOLOGY

A method was conceived to create a system for the purpose of discovering the fourth category of Smart-phone malware, namely automated worms, or malware which requires no user interaction.

As there is currently no known automated malware for Smart-phones with the majority of current malware being rogue applications or imbedded in messages which use social engineering techniques to infect devices, it is difficult to determine or create a methodology which could be utilised to capture such malware. Initially, this paper submits a method which could be utilised to capture automated malware created for Smart-phones, based on the assumption that most automated forms of malware such as worms scan for potential victims based on their Operating System or try to infect all devices available.

This paper utilises the low-interaction honeypot concept. A honeypot is a networked system which does not serve any purpose. Any traffic to that system we can deem as suspect or malicious as there is no reason for traffic to be sent to that system (Hudak 2008). A low-interaction honeypot is merely an emulated device, which advertises certain ports but does not provide any services, compared to a high-interaction honeypot which is typically an actual device (Spitzner 2002)

The theory behind the creation of this 1st Generation Smartphone honeypot is that automated malware typically searches for victims based on the services or ports that are available. Detection of malware that attacks Smart-phone Operating Systems will enhance the capabilities researches have available to discover, capture and understand the models and attack vectors utilised by creators of this form of malware.

METHOD - A 1ST GENERATION SMARTPHONE HONEYPOT

This honeypot will only demonstrate network sniffer (Nmap) fingerprints of Smart-Phone operating systems and the available ports as real smart-phone devices running the chosen operating systems would. No further services of the smart-phone operating systems will be emulated. The function of this honeypot will be to discover if any malware has been created targeting systems such as its emulated environments.

The major component for this honeypot is a Honeyd configured honeypot running on an Ubuntu Virtual Machine emulating Windows Mobile (WM) 5 and WM6 Operating System environments.

Honeyd (Provos 2004) is a low-interaction virtual honeypot application that simulates computer systems at a network level namely emulating the computer systems TCP/IP stack in combination with network infrastructure and network routing emulation. Honeyd uses Nmap's fingerprint database as its reference for an emulated device or operating systems TCP and UCP behaviour. The purpose of Honeyd is to deceive malware and network scanners that it's emulated devices and operating systems are actually real devices.

NMap network scanner is used to test the viability of the Honeyd configuration. "Nmap ("Network Mapper") is an open source tool for network exploration and security auditing…(that) uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering (and) what operating systems (and OS versions) they are running" (Lyon 2008). The Nmap application includes a file of Operating System Fingerprints (nmap-os-db) which it uses to detect and display the host/s operating systems.

The advantage of using Honeyd as the honeypot software is it emulates different devices by responding to certain packets in the appropriate way to create the unique fingerprint (based on an nmap fingerprint) for that device type. Additionally, for each emulated device, we can list which ports should be available and in what state.

Even though Smart-phones typically use wireless networks or data-services provided by their mobile phone carrier, data is transferred based on the TCP/IP protocol stack. Hence to reduce overheads, this honeypot will be initially deployed on a wired network which will also have a wireless access point attached to the network. Further deployments of this Honeypot virtual machine could be used on devices with wireless and carrier data-services connectivity.

Creating the Honeypot

The first step was deciding which devices to emulate in the honeyd honeypot environment. Windows Mobile 5 (WM5) and Windows Mobile 6 (WM6) operating systems were chosen for this initial honeypot, as they are commonly used Smart-Phone operating environments, with 16.9% of the market in 2006 (Schmidt & Albayrak, 2008) and physical devices running these environments were available for testing. Additionally WM5 and WM6 emulators are available which could be utilised in Virtual Machines and integrated into Honeyd in future work and improvement of the smart-phone honeypot.

Two HTC Hermes 100 smart-phones, one with Windows Mobile 5 installed and another with Windows Mobile 6 installed, were scanned using NMap to discover if they had any open ports available and if their Operating Systems were detected.

Both smart-phones were configured to access a local wireless network and given a dynamic IP address. An advantage of the Windows Mobile operating Systems is the ease of discovering the devices current network settings including IP address.

A laptop, with NMap installed was also added to the wireless network. Several NMap scans were made of each device using the following commands to discover open ports.

- -sS TCP SYN scan

  This scan sends SYN packets, as if you are going to open a real connection and then wait for a response. A SYN/ACK reply indicates the port is listening (open), while a RST (reset) is indicative of a non-listener. If no response is received after several retransmissions, the port is marked as filtered. (Lyon, 2008)

- -sT TCP connect () scan

  For this scan, Nmap asks the host device operating system to establish a connection with the target machine and port by issuing the `connect` system call. This is the same high-level system call that web browsers, P2P clients, and most other network-enabled applications use to establish a connection. (Lyon, 2008)

- -sU UDP scan

  For this scan, NMap sends an empty (no data) UDP header to every targeted port. If an ICMP port unreachable error is returned, the port is `closed`. Occasionally, a service will respond with a UDP packet, proving that it is `open`. If no response is received after retransmissions, the port is classified as `open|filtered`. This means that the port could be open, or perhaps packet filters are blocking the communication. (Lyon, 2008)

- -O Operating System FingerPrint Scan

  This scan activates remote host identification via TCP/IP fingerprinting. Nmap sends a series of TCP and UDP packets to the remote host and the results of these packets are combined to create unique fingerprint. The fingerprint is then compared to the Nmap database of known Operating System fingerprints. (Valli 2003)

The output of the scans provided the following open ports:

```
Port:    137 UDP – Filtered | Open     NetBIOS - ns
Port:    138 UDP – Filtered | Open     NetBIOS - dgm
Port:    1034 UDP – Filtered | Open    Activesync - notify
OS:      Warning: OSScan results may be unreliable because we could not
find at least 1 open and 1 closed port
OS details: HP Compaq t5520 thin client (Microsoft Windows CE 5.00),
Microsoft Windows Server 2003 SP1, Microsoft Windows Mobile 6 Classic or
Microsoft Zune audio player (firmware 2.2), Microsoft Windows CE 5.0
(ARM), Microsoft Windows XP Professional SP2, Microsoft Windows XP SP2
```

Figure 1. *Windows Mobile 5 scan ouput.*

```
Port:    137 UDP – Filtered | Open     NetBIOS - ns
Port:    138 UDP – Filtered | Open     NetBIOS - dgm
OS:      Warning: OSScan results may be unreliable because we could not
find at least 1 open and 1 closed port
OS details: HP Compaq t5520 thin client (Microsoft Windows CE 5.00),
Microsoft Windows Server 2003 SP1, Microsoft Windows Mobile 6 Classic or
Microsoft Zune audio player (firmware 2.2), Microsoft Windows CE 5.0
(ARM), Microsoft Windows XP Professional SP2, Microsoft Windows XP SP2
```

Figure 2. *Windows Mobile 6 scan output.*

The UDP port 137 was found to be Filtered | open on both devices. Becher, Freiling & Leider (2007) discussed how every Windows Mobile device that connects to a Wireless network initially sends a series of NetBIOS datagrams on UDP port 137 to the network broadcast address. Their proof-of –concept worm utilised a vulnerability in the Windows Mobile network stack that allowed an infected device to capture NetBIOS messages and recover the IP addresses of devices joining the network. The infected device would then send a copy of itself to the discovered IP address. This method demonstrates that UDP port 137 could be one avenue that smartphone malware may utilise to discover and infect victims, which is why it will be emulated by our honeypot.

The second step was discovering NMap finger prints for WM5 and WM6. A honeypots aim is to deceive a potential attacker, be it automated malware or a human attacker. It attempts to achieve this deception through the use of Nmap fingerprints to emulate devices and networks (2003). It is assumed that the same fingerprints used by Nmap will be also used by Honeyd. Since no OS fingerprints for any Windows Mobile Operating Systems were discovered in the Honeyd nmap.prints file, the OS fingerprints shown below were added to the nmap.prints file. These fingerprint personalities were retrieved from the latest Nmap fingerprint 'nmap-os-db' file (Nmap OS Fingerprinting 2nd Generation DB 2009). The same file is included in the latest Nmap application.

Fingerprint Microsoft Windows CE 5.0 (ARM)
Class Microsoft | Windows | PocketPC/CE | specialized
SEQ(SP=77-81%GCD=1-6%ISR=92-9C%TI=I%II=I%SS=O%TS=0)
OPS(O1=NNT11|M5B4NW0NNT00NNS%O2=NNT11|M5B4NW0NNT00NNS%3=NNT11|M5B4NW0NNT00%O4=NN
T11|M5B4NW0NNT00NNS%O5=%O=)
WIN(W1=832C%W2=8340%W3=8200%W4=81D0%W5=0%W6=0)
ECN(R=Y%DF=N%T=7B-85%TG=80%W=0%O=%CC=N%Q=)
T1(R=Y%DF=Y%T=7B-85%TG=80%S=O%A=S+%F=A|AS%RD=0%Q=)
T2(R=Y%DF=N%T=7B-85%TG=80%W=0%S=Z%A=S%F=AR%O=%RD=0%Q=)
T3(R=Y%DF=N%T=7B-85%TG=80%W=0%S=Z%A=O%F=AR%O=%RD=0%Q=)
T4(R=Y%DF=N%T=7B-85%TG=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)
T5(R=Y%DF=N%T=7B85%TG=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q)
T6(R=Y%DF=N%T=7B-85%TG=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)
T7(R=Y%DF=N%T=7B85%TG=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q)
U1(DF=N%T=7B85%TG=80%IPL=B0%UN=0%RIPL=G%RID=G%RIPCK=G%UCK=G%RUD=)
IE(DFI=S%T=7B-85%TG=80%CD=Z)

Fingerprint Microsoft Windows Mobile 6 or Zune audio player (firmware 2.2)
Class Microsoft | Windows | PocketPC/CE | general purpose
Class Microsoft | embedded || media device
SEQ(SP=75-9B%GCD=<7%ISR=92-9C%TI=I%II=I%SS=S%TS=0)
OPS(O1=M5B4NW0NNT00NNS%O2=M5B4NW0NNT00NNS%O3=M5B4NW0NNT00%O4=M5B4NW0NNT00NNS
%O5=M5B4NW0NNT00NNS%O6=M5B4NNT00NNS)
WIN(W1=832C%W2=8340%W3=8200%W4=81D0%W5=81D0%W6=805C)
ECN(R=Y%DF=Y%T=80%TG=80%W=832C%O=M5B4NW0NNS%CC=N%Q=)
T1(R=Y%DF=N|Y%T=80%TG=80%S=O|Z%A=S+%F=AR|AS%RD=0%Q=)
T2(R=Y%DF=N%T=80%TG=80%W=0%S=Z%A=S%F=AR%O=%RD=0%Q=)
T3(R=Y%DF=N|Y%T=80%TG=80%W=0|805C%S=O|Z%A=O|S+%F=AR|AS%O=M5B4NW0NNT00NNS%RD=0%Q=
)
T4(R=Y%DF=N%T=80%TG=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)
T5(R=Y%DF=N%T=80%TG=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=Y%DF=N%T=80%TG=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)
T7(R=Y%DF=N%T=80%TG=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
U1(DF=N%T=80%TG=80%TOS=0%IPL=B0%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUL=G%RUD=G)

IE(DFI=S%T=80%TG=80%TOSI=S%CD=Z%SI=S%DLI=S)

Fingerprint HTC TyTN II (Kaiser) mobile phone (Microsoft Windows Mobile 6)
Class Microsoft | Windows | PocketPC/CE | phone
SEQ(SP=82-8C%GCD=<7%ISR=92-9C%TI=I%II=I%SS=S%TS=0)
OPS(O1=M1F4ANW1NNT00NNS%O2=M1F4ANW1NNT00NNS%O3=M1F4ANW1NNT00%O4=M1F4ANW1NNT00
NNS%O5=M1F4ANW1NNT00NNS%O6=M1F4ANNT00NNS)
WIN(W1=FFFF%W2=FFFF%W3=FFFF%W4=FFFF%W5=FFFF%W6=FFFF)
ECN(R=Y%DF=Y%T=80%TG=80%W=FFFF%O=M1F4ANW1NNS%CC=N%Q=)
T1(R=Y%DF=Y%T=80%TG=80%S=O%A=S+%F=AS%RD=0%Q=)
T2(R=Y%DF=N%T=80%TG=80%W=0%S=Z%A=S%F=AR%O=%RD=0%Q=)
T3(R=Y%DF=Y%T=80%TG=80%W=FFFF%S=O%A=S+%F=AS%O=M1F4ANW1NNT00NNS%RD=0%Q=)
T4(R=Y%DF=N%T=80%TG=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)
T5(R=Y%DF=N%T=80%TG=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=Y%DF=N%T=80%TG=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)
T7(R=Y%DF=N%T=80%TG=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
U1(DF=N%T=80%TG=80%TOS=0%IPL=B0%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUL=G%RUD=G)
IE(DFI=S%T=80%TG=80%TOSI=S%CD=Z%SI=S%DLI=S)

Figure 3. *2nd Generation Nmap Fingerprints*

With knowledge of the operating system fingerprint personalities and having identified which ports and services are available on WM5 and WM6 devices, the next step was installing Honeyd. In this instance, an Ubuntu Virtual Machine was created and a modified version of Honeyd was installed. This modified version of Honeyd has been created by Christian Kreibich to be used with his application Honeycomb.

Honeycomb (Kreibich & Crowcroft, 2003) is a pattern detection engine that monitors any network traffic that Honeyd receives and creates NIDS signatures for any patterns that regularly occur. It is assumed that any regular traffic that Honeyd receives is malicious in nature, as honey pots in general serve no other network purpose and should not be receiving valid traffic. The advantages to using Honeycomb include reducing overhead caused by using additional programs to perform the same task and it is integrated into Honeyd hence will not have any synchronisation issues. Additionally the creation of NIDS signatures could be very useful for detecting very new automated mobile malware and integrating the signatures into Network Intrusion Detection Systems on wireless networks to track the spread and effect of such malware. Mobile Intrusion Detection Systems such as Gibraltar (Jacoby *et al.*, 2006) could also utilise the signatures.

An updated copy of the "nmap.prints" file, which contains the operating system fingerprints used by Honeyd to emulate operating environments, was retrieved from another installation of Honeyd running the latest version. The reason for this modification is the assumption that the latest version should include the most recent discovered nmap fingerprints for new systems such as Windows Mobile5 and Windows Mobile 6.

The next step was creating the Honeyd configuration file to be used by the honeypot. The Honeyd configuration script was:

ANNOTATE "Microsoft Windows CE 5.0 (ARM)"
ANNOTATE "Microsoft Windows Mobile 6 or Zune audio player (firmware2.2)"
ANNOTATE "HTC TyTN II (Kaiser) mobile phone (Microsoft Windows Mobile 6)"

CREATE Windows Mobile 5
SET Windows Mobile 5 PERSONALITY "Microsoft Windows CE 5.0 (ARM)"
BIND 192.100.0.10 Windows Mobile 5
SET DEFAULT Windows Mobile 5 TCP ACTION RESET
ADD Windows Mobile 5 UDP PORT 137 OPEN
ADD Windows Mobile 5 UDP PORT 138 OPEN
ADD Windows Mobile 5 UDP PORT 1034 OPEN

CREATE Windows Mobile 6
SET Windows Mobile 6 PERSONALITY "Microsoft Windows Mobile 6 or Zune audio player (firmware2.2)"
BIND 192.100.0.11 Windows Mobile 6
SET DEFAULT Windows Mobile 6 TCP ACTION RESET

```
ADD Windows Mobile 6 UDP PORT 137 OPEN
ADD Windows Mobile 6 UDP PORT 138 OPEN


CREATE HTC
SET HTC PERSONALITY "HTC TyTN II (Kaiser) mobile phone (Microsoft Windows Mobile 6)"
BIND 192.100.0.12 HTC
SET DEFAULT HTC TCP ACTION RESET
ADD HTC UDP PORT 137 OPEN
ADD HTC UDP PORT 138 OPEN
```

Figure 4. *Honeyd Configuration File*

Lastly, after Honeyd was installed and configured, the honeypot was started with the following command

```
sudo honeyd -p /usr/local/share/honeyd/nmap.prints -f /usr/share/honeyd/honeyd.conf -I
eth0 192.100.0.10 192.100.0.11 192.100.0.12
```

A laptop was also added to the same network as the Honeyd honeypot and Nmap was installed to test if the OS emulated by the honeypot were correctly identified.

DISCUSSION

The major issue encountered during this research was that the operating system fingerprints (2nd generation) recovered from the latest version of Nmap were not compatible with Honeyd. Additionally, the operating system fingerprints did not have contemporaries (1st generation) within the nmap.prints file used by Honeyd to emulate operating system fingerprints. While both fingerprint systems work by sending a specific number of packets to query the TCP/IP stack, and create the unique fingerprint for the operating system based on how those packets are received and replied to, the format of the output is different between Nmap and Honeyd.

Honeyd uses the fingerprint format (Figure 5) described by Fyodor (1998), which was also the 1st generation fingerprint utilised initially by Nmap.

```
FingerPrint  IRIX 6.2 - 6.4
TSeq(Class=i800)
T1(DF=N%W=C000|EF2A%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(Resp=Y%DF=N%W=C000|EF2A%ACK=O%Flags=A%Ops=NNT)
T4(DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(DF=N%W=0%ACK=S%Flags=AR%Ops=)
PU(DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)
```

Figure 5. *1st Generation Nmap Fingerprint.*

Since the operating system fingerprints for the Smart-Phone environments were not compatible, Honeyd was not able to properly run, based on syntax errors produced by the 2nd generation Nmap fingerprints added to the "nmap.prints" file. When those additional fingerprints were removed from the "nmap.prints"file, Honeyd was unable to find the fingerprints requested in the configuration file and subsequently did not initiate.

Even if Honeyd was able to emulate the Smart-Phone fingerprints successfully based on Nmap scans, additional testing of the Honeyd configuration by known vulnerabilities would be limited by the fact there are currently no known Windows Mobile automated malware in the wild. This type of malware has been created by Becher, Freiling & Leider (2007) in controlled environments, thus there is an assumption that some form of specific malware will created in the near future, as Windows Mobile smart phones become more prolific and more of a target to malicious third parties.

CONCLUSION

A 1st generation honey-pot designed to specifically discover knowledge of the existence of automated worms targeting smart-phones is a possible concept. The methodology of discovering available ports on actual Windows Mobile 5 and Windows Mobile 6 smart-phone devices using Nmap network scanner and then emulating those ports in a Honeyd honey-pot seems viable. Yet the method of using Honeyd to emulate smart-phone operating systems including Windows Mobile 5 and Windows Mobile 6 cannot be tested for viability until a technique is created to adopt the format of 2nd generation Nmap operating system fingerprints to the format utilised by Honeyd. When this technique is proven, Nmap testing of the Honeyd honey-pot will be possible to further understand the abilities and attack vectors of smart-phone worms.

The next step will be researching and creating a technique that will allow a conversion of a 2nd generation Nmap fingerprint into the format utilised by Honeyd. This will be critical to achieving the aims of testing the concept and discovering automated smart-phone malware of the 1st Generation Smart-phone honey-pot. Consequently this technique should allow the addition of more Operating System fingerprints of other Smart-phone environments including devices running the Google Andriod and Apple iPhone Operating Systems to be added to further generations of smart-phone honeypots.

Secondly, once the issue highlighted by this paper is rectified, Virtual Machines running the Windows Mobile5 and Windows Mobile 6 emulators could be linked to the Honeyd Honeypot, allowing any detected malware to be observed performing its functions in a high-interaction environment. Such information could be used to reverse engineer new mobile malware and provide insight into the methods that malware writers will use attack Smart-phone devices and operating systems.

REFERENCES

Becher, M. Freiling, F.C. & Leider, B. (2007). On the Effort to Create Smartphone

Worms in Windows Mobile. Proceedings of the 2007 IEEE Workshop on Information Assurance. New York: United States Military Academy, West Point.

Chen, T. M. & Peikari, C. (2008). Malicious software in mobile devices. In Zhang, Y. Zheng J. & Ma, M (Eds.), Handbook of Research on Wireless Security (Vol.1., pp 1-10). IGI Global

Fyodor. (1998). Remote OS detection via TCP/IP stack fingerprinting. Retrieved June, 2009 from http://nmap.org/nmap-fingerprinting-article.txt

Guo, C. Wang, H.J & Zhu, W. (2004). Smart-Phone Attacks and Defenses. Paper Presented at The Third Workshop on Hot Topics in Networks, San Diego, CA.

Hudak, S. (2008). Automatic Honeypot Generation and Network Deception. Retrieved June 2009 from CiteSeerX - Scientific Literature Digital Library and Search.

Hypponen, M (2006). Malware goes Mobile. Scientific American Magazine. Nov 2006, 70-77.

Jacoby, G. Hickman, T. Warders, S. Griffin, B. Darensburg, A & Castle D. (2006).

Gibraltar – A Mobile Host-Based Intrusion Protection System. In proceedings of the 2006 International Conference on Security & Management (pp. 207-212).

Kreibich, C. & Crowcroft, J. (2003). Honeycomb: Automated NIDS Signature Creation using Honeypots. Retrieved June 2009 from http://www.icir.org/christian/publications/honeycomb-poster-sc2003.pdf

Lyon, G. (2008). Nmap Network Scanning. Sunnyvale, CA: Insecure.com LLC

McAfee. (2009). Mobile Security Report 2009. Santa Clara, CA

Maurer, J (2003). Internet Worms: Walking on Unstable Ground. Sans Institute. Retrieved June, 2009 from http://cnscenter.future.co.kr/resource/security/ virus/1229.pdf

Nmap OS Fingerprinting 2nd Generation DB. (2009). Retrieved June 2009 from Nmap.org/svn/nmap-os-db

Provos, N. (2004). A Virtual Honeypot Framework. In Proceedings of 13th USENIX Security Symposium, (pp. 1–14).

Schmidt, A.D, & Albayrak, A. (2008). Malicious Software for Smartphones. *Technical Report: TUB-DAI 02/08-01*. DAI-Labor, Berlin.

Spitzner, L. (2002). Definition and Value of Honeypots. Retrieved June 2009 from http://www.windowsecurity.com/whitepapers/Honeypots_Definitions_and_Value_of_Honeypots.html

Valli, C. (2003). Honeyd – A OS Fingerprinting Artifice. Paper presented at the 1st Australian Computer, Network & Information Forensics Conference, November 2003, Perth, Western Australia.

COPYRIGHT