This Network is Infected: HosTaGe - a Low-Interaction Honeypot for Mobile Devices

This Network is Infected: HosTaGe - a Low-Interaction Honeypot for Mobile Devices

Emmanouil Vasilomanolakis†, Shankar Karuppayah†§, Mathias Fischer†, Max Mühlhäuser†

†Telecooperation Group,
Technische Universität Darmstadt - CASED
first.last@cased.de

§National Advanced IPv6 Center (NAv6),
Universiti Sains Malaysia (USM),
Penang, Malaysia

ABSTRACT

In recent years, the number of sophisticated cyber attacks has increased rapidly. At the same time, people tend to utilize unknown, in terms of trustworthiness, wireless networks in their daily life. They connect to these networks, e.g., airports, without knowledge of whether they are safe or infected with actively propagating malware. In traditional networks, malicious behavior can be detected via Intrusion Detection Systems (IDSs). However, IDSs cannot be applied easily to mobile environments and to resource constrained devices. Another common defense mechanism is honeypots, i.e., systems that pretend to be an attractive target to attract malware and attackers. As a honeypot has no productive use, each attempt to access it can be interpreted as an attack. Hence, they can provide an early indication on malicious network environments. Since low interaction honeypots do not demand high CPU or memory requirements, they are suitable to resource constrained devices like smartphones or tablets.

In this paper we present the idea of *Honeypot-To-Go*. We envision portable honeypots on mobile devices that aim on the fast detection of malicious networks and thus boost the security awareness of users. Moreover, to demonstrate the feasibility of this proposal we present our prototype *HosTaGe*, a low-interaction honeypot implemented for the Android OS. We present some initial results regarding the performance of this application as well as its ability to detect attacks in a realistic environment. To the best of our knowledge, *HosTaGe* is the first implementation of a generic low-interaction honeypot for mobile devices.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: [Security and Protection - Invasive software]; C.2.0 [**Computer-Communication Networks**]: General: Security and Protection

Keywords

Mobile Honeypot; Malware; Security; Android

1. INTRODUCTION

Recent security reports indicate an increase in sophisticated cyber attacks [13]. With the advancements in mobile devices (smartphones, tablets, etc.) as well as the increased number of available wireless networks many challenges arise from the security perspective. People tend to utilize unknown, in terms of trustworthiness, wireless networks in their daily life. They connect to these networks, e.g., airports and coffee shops offering Internet access, without knowledge of whether they are safe or infected with actively propagating malware.

From conventional networks, additional defenses like IDSs [4] and dynamic firewalls are known for the detection of malicious behavior. However, these defenses cannot be applied easily to resource constrained mobile devices. Moreover, IDSs as purely passive monitoring components may not alone be sufficient in protecting against cyber attacks.

Honeypots are special systems whose value lies in being probed, attacked or compromised [12]. Honeypots provide a more active line-of-defense compared to passive intrusion detection. Furthermore, they are also able to provide additional knowledge regarding recent malware techniques and trends. Honeypots can be classified with respect to the level of interaction that is offered to the attacker. A low-interaction honeypot simulates network operations, usually at the TCP/IP stack, while high-interaction honeypots are real systems that are vulnerable and need to be carefully safeguarded to avoid that they are compromised. In this paper we focus on low-interaction honeypots since they require low resources and thus are suitable for mobile devices. Popular low-interaction honeypots in conventional networks are for example *Nepenthes* [3] and *Dionaea*[1]. However, we still lack honeypots that are tailored specifically for deployment in mobile environments.

For this reason, we propose the idea of *Honeypot-To-Go*: lightweight, low-interaction, portable, and generic honeypots for mobile devices that aim on the detection of malicious, wireless network environments. As most malware propagate over the network via specific protocols, a low-interaction honeypot located at a mobile device can check wireless networks for actively propagating malware. We envision such honeypots running on all kinds of mobile devices, e.g., smartphones and tablets, to provide a quick assessment

[1]http://dionaea.carnivore.it

on the potential security state of a network. In addition, a honeypot-to-go may serve as a tool to increase the security awareness among users and allow them to check publicly available wireless networks for signs of malicious activity before actually using them. Moreover, such honeypots can also assist network administrators analyzing the health of their networks on-the-go.

The contribution of this paper is twofold. First, we introduce the concept of Honeypot-To-Go, i.e., portable honeypots running on mobile devices to detect malware spreading via wireless networks. Second, we introduce our prototype, *HosTaGe*[2], which stands for Honeypot-To-Go, and that is a low-interaction honeypot for the Android OS. *HosTaGe* has been specifically designed with usability in mind to increase the security awareness of users when connecting to wireless networks.

The remainder of this paper is organized as follows: In Section 2 we present generic requirements for low-interaction honeypots for mobile devices. Based upon the requirements, Section 3 summarizes the related work in this area. Section 4 presents the architecture and the features of *HosTaGe*, our mobile honeypot. In Section 5 we provide a detailed discussion of our prototype (with respect to the requirements from Section 2) and present some initial evaluation results. Moreover, we discuss current limitations and future work. Finally, Section 6 concludes the paper.

2. REQUIREMENTS FOR A MOBILE HONEYPOT

In the following, we describe basic requirements for a mobile low-interaction honeypot.

- **Visibility:** In order to attract malware and malicious users, honeypots must emulate vulnerable services at the respective ports. This requires the honeypot to listen for incoming connections at those ports. However, it is not always possible to constantly listen to all ports due to limitations of mobile devices such as resources and also security policies of mobile Operating Systems (OSs). However, for those ports that are being listened, the honeypot application must respond according to the protocol specification in order to avoid raising any suspicions.
- **Usability:** A mobile honeypot needs to be developed by keeping ordinary users as the main users in mind. This is an important requirement since one of our main goals is to promote security awareness. Therefore, the mobile honeypot should not only be useful to security specialists and network administrators, but also ordinary users. This can be achieved via a user-friendly graphical interface and an easy setup. Nevertheless, *advanced* modes should also be provided for expert users.
- **Security and Containment:** The developed application needs to focus on security and containment aspects to ensure the security of the user device itself. Special care and attention needs to be given to prevent malware and adversaries from compromising the user's device to launch any kind of attack.
- **Resource Utilization:** Resources such as power, memory and network bandwidth are usually scarce at mobile devices. Hence, special care needs to be taken to avoid unnecessary computational overhead.
- **Extendability and Interoperability:** A mobile honeypot should be easily extendable by new protocols, for vulnerability emulation, to the existing system. Additionally, the honeypot should be able to submit statistics to third-party entities, e.g., to security incident monitors or servers.

[2]http://www.tk.informatik.tu-darmstadt.de/en/research/secure-smart-infrastructures/hostage

3. RELATED WORK

Honeypots have been studied extensively over the recent years [12]. The low-interaction class especially exhibits a variety of proposals and implementations, e.g., [3, 10, 11, 9, 6]. As mentioned before, low-interaction honeypots emulate only network operations and therefore require low resources for their deployment.

Early work that considered honeypots and mobile devices focused only on Bluetooth communications [5, 17]. The recent advances on mobile devices, their interconnectivity as well as their popularity created a whole new ecosystem for honeypot researchers.

Existing work in this direction [8, 15, 7, 14] focuses on the detection of mobile-specific malware. Specifically, Mulliner et al. were the first to discuss the idea of a honeypot for smartphones, by providing initial ideas, challenges and an architecture for their proposed system [8]. Moreover, in [15, 14] Wahlisch et al. provided insights from the deployment of a honeypot on a mobile network. However, their honeypot is not specifically crafted for mobile devices, but rather existing Linux-based desktop honeypots deployed in mobile networks. Furthermore, in [7] the idea of nomadic honeypots has been introduced. The authors focus on mobile-specific attacks and also require their system to collect a large amount of personal information.

In addition, most honeypot proposals are strictly focused on implementing novel detection methods. Hence, user-friendly solutions are not the primary focus and usually target only security professionals as the main users. However, the idea and the benefits of creating user-friendly honeypots is getting more attention lately. In [2, 1] the "Honey@home" is proposed, where organizations and people can participate by deploying a honeypot that reports to a large-scale centralized honeypot monitoring system. Nevertheless, this approach does not include mobile devices and there are no clear benefits for the end-user to participate.

In contrast to existing or ongoing work that is focusing on mobile-specific attacks, our idea is to develop a user-friendly honeypot that can run out-of-the-box on mobile devices. This benefits the users by providing an indication of the security state of their network and thus fosters their interest and motivation to utilize this application.

4. HOSTAGE MOBILE HONEYPOT

In this section we present our prototype *HosTaGe*, that stands for Honeypot-To-Go. First we introduce the underlying architecture of *HosTaGe*, followed by a detailed description of each module and its functionalities.

Figure 1 depicts the architecture of *HosTaGe*. The application is written in Java and supports all Android's API levels from 8 (v.2.2 Froyo) to 18 (v.4.3 Jelly Bean). It runs on

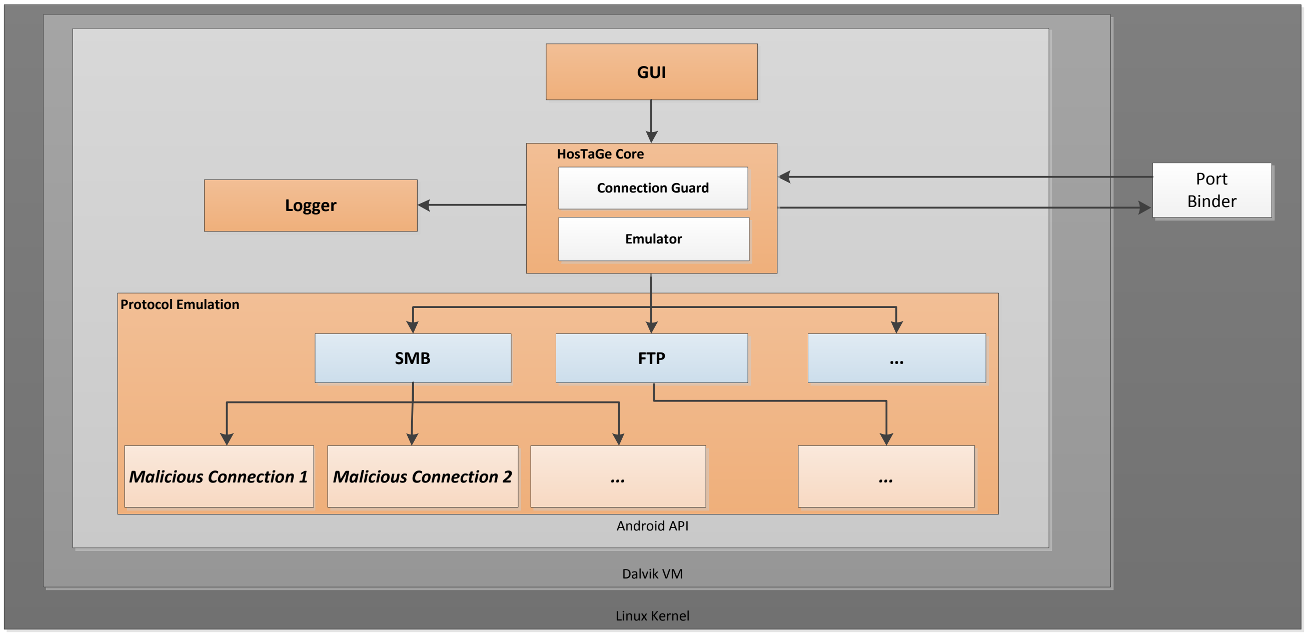


Figure 1: HosTaGe System Architecture

top of the Dalvik Virtual Machine environment and consists of several modules that are closely interconnected, namely *HosTaGe Core*, *Logger*, *Port Binder* and the *Graphical User Interface (GUI)*.

When the application is executed, *HosTaGe Core* runs in the background as a *Service* and activates the *Emulator* submodule. The submodule then emulates[3] the selected protocols by listening to incoming connections in the respective ports that are associated to each of the protocols. In order to bind the sockets with the ports, the *Port Binder* which runs on the Linux Kernel level is called by *HosTaGe Core* with the port that it should listen on. After that, the respective module listens for incoming connections. Upon receiving a connection request, *HosTaGe Core* alerts the *Emulator* submodule which in return calls the *Logger* to record all activities that are being observed. At the same time, the user is notified of the activities that are being detected via the *GUI*. A more detailed description of the related modules are explained below.

4.1 HosTaGe Core

The core provides an interface for the activation or deactivation of the implemented emulated protocols. It also sends status reports to the GUI to provide the user with connection information. It consists of two sub-modules called *Emulator* and *Connection Guard*.

Emulator.

This sub-module is responsible for the emulation of protocols in *Protocol Emulation*. It executes multiple threads, i.e., each thread for every selected protocol that listen to incoming malicious connections to the respective ports and activates the *Logger* module for logging all activities at this port.

Each of the emulated protocols can accept several simultaneous connections. They listen for connection requests on the specified protocol port and start a *Connection Handler* for every incoming request. Each *Connection Handler* communicates with one single client. It provides a basic protocol interface according to the specifications of the respective protocol. The binding of the ports is handled by the *Port Binder* module, as described in Section 4.4.

For a low-interaction honeypot, it is important to have a broad selection of available protocols to emulate. For this reason we support most of the protocols that are in use for malware propagation activities: SMB, FTP, HTTP and Telnet. Moreover, other protocols, e.g., HTTPS and SSH, are currently under development. Besides that, a simple mechanism for adding additional protocols, through a XML-parser, is also under development.

Connection Guard.

With respect to various security concerns explained in Section 2, this sub-module prevents that the hosting device gets compromised. Furthermore, the *Connection Guard* is also responsible for blocking incoming connections when it suspects that the device is under attack, e.g., due to a Denial of Service (DoS) attack. In such a case it limits the maximum allowed incoming connections, from the same source IP and/or the same destination port. Besides that, established connections are also terminated after some time.

4.2 Logger

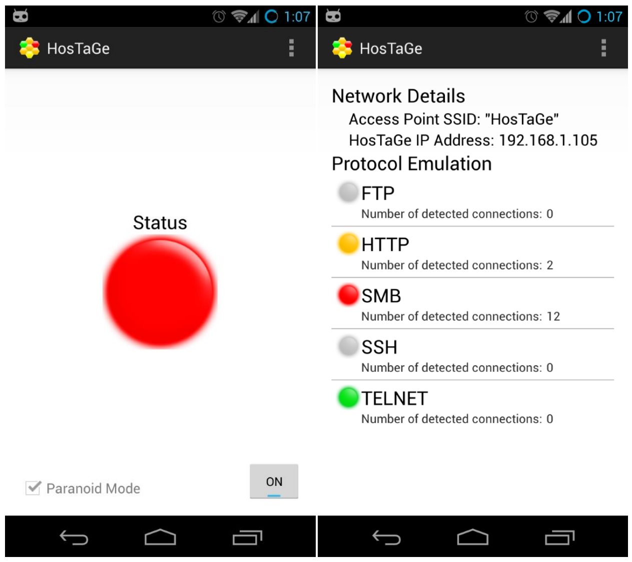
The application supports different formats for the gener-

[3]User can select which protocol(s) to be emulated

ated log files. It also supports exporting the logs to a plain text file and/or to a *SQLite* database. In addition, for interoperability reasons *HosTaGe* can produce logs in the JSON [4] format for further processing of the alert data by other third-party applications.

4.3 Graphical User Interface

HosTaGe provides an usable GUI to ease the understanding of the underlying application to the users. The default view of the GUI offers an overall view to users by providing a *network health condition status* in a single glance. However, if necessary, users can also choose the advanced view.



Network Details
Access Point SSID: "HosTaGe"
HosTaGe IP Address: 192.168.1.105
Protocol Emulation
FTP
Number of detected connections: 0
HTTP
Number of detected connections: 2
SMB
Number of detected connections: 12
SSH
Number of detected connections: 0
TELNET
Number of detected connections: 0

Figure 2: GUI of HosTaGe, Left: Default View, Right: Advanced View

The application itself offers three different functional modes to meet the users' needs via the GUI:

- Default Mode: This mode is activated with only the SMB protocol simulation enabled. This decision takes in consideration the fact that a large amount of malware utilizes mainly this protocol for their propagation. For instance, the worm W32.Sality.AE, i.e., listed among the most popular malware in 2012 [13], makes use of the SMB service for its propagation. As such, *HosTaGe* can easily detect the most common hostile environments (if present), without the need of additional protocols. Users may enable additional protocols via the *advanced mode.*

- Advanced Mode: This mode allows the user to choose additional protocols that needs to be emulated from the list of available protocols.

- Paranoid Mode: This mode enables the emulation of all available protocols simultaneously.

HosTaGe features a network security health indicator as shown in Figure 2 that provides the user an indication of the security status of the network. Moreover, the application also alerts the user by indicating previous known statuses (whenever possible). Different colors indicate the network security health condition of the currently connected network:

- *Green:* Indicates that no malicious activity has been reported (as of yet) in the particular network, even in the past.

- *Yellow:* Indicates that (at least) a malicious activity has been reported in the past from the particular network.

- *Red:* Indicates that the particular network has just been diagnosed as infected/malicious.

- *Grey:* Indicates that *HosTaGe* (or its respective protocol emulator) is turned off.

4.4 Port Handling

Based on the security policies of Android OS, only system-signed applications are allowed to have access to privileged network ports (below 1024). Therefore, there are usually no third-party applications that are capable of accessing the privileged ports. Although this is a reasonable justification from a security perspective, it creates challenges for the development of our mobile honeypot. As mentioned in Section 2 a basic requirement for such an application is to be visible to malicious users and malware. Hence, access to many privileged ports, e.g., 80 and 445, is required.

In order to achieve this we implemented a small program (see Figure 1), in native C, cross-compiled for the Android OS. This program binds a port, passed to it as an argument, and sends the file descriptor back to the caller through a UNIX domain socket. The application calls this program with super-user-privileges and receives the file descriptor of the bound port. It uses its own extended version of a Java Server-Socket, which constructs a socket from a file descriptor. This way access to the required ports is achieved.

However, if an existing running service is already bound to the target privileged ports, *HosTaGe* reports to the user that it is unable to create a socket connection. Users need to disable or terminate the particular service before *HosTaGe* is able to activate its protocol service.

The drawback of the aforementioned solution for having access to the privileged ports is that *HosTaGe* requires a rooted Android device to operate properly. However, since we envision the usage of our application for both IT specialists and ordinary users, we are currently investigating other possible solutions in order to overcome this limitation.

5. DISCUSSION

In this section we present some initial results for *HosTaGe* with respect to the requirements defined in Section 2. In more details, we studied its effectiveness in detecting attacks (*visibility*) as well as its ability to handle large number of connections (*security and containment*) besides the application's power utilization (*resource utilization*).

Attack Handling.

In order to examine the effectiveness of our proposal we deployed *HosTaGe* in an isolated testbed as shown in Figure 3. For this we connect several clients to a wireless access point: a Linux-based system running a Dionaea honeypot, a Windows XP SP3 machine that was infected with a variety

[4] http://www.json.org

of malware (more details can be found in the Appendix), and a mobile device with the following specifications:

- Device : Galaxy Nexus
- CPU: 1.2 GHz dual-core ARM Cortex-A9 (ARMv7 rev 10 [v7l])
- RAM: 693 MB
- OS: Android 4.2.2 (Jelly Bean)
- Mod-Version: CyanogenMod-10.1.2-maguro
- Linux-Kernel: 3.0.31

HosTaGe successfully detected all the malware's attempts to propagate and the results also corresponds to the reports from the Dionaea honeypot. Moreover, we also executed several port scans in the network using the Nmap network scanner[5], to test its resilience and effectiveness towards large number of port scans. *HosTaGe* successfully handled all these scans and remained functioning without any sign of misbehaviour in the presence of large number of connection requests.

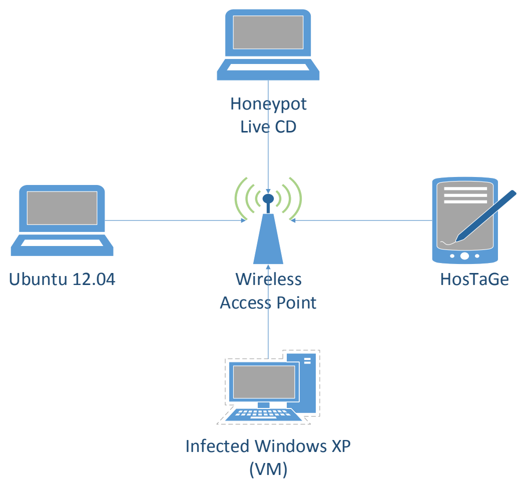


Figure 3: Attack Testbed Architecture

Power Utilization.

While the *Attack Handling* experiment indicates the feasibility of *HosTaGe*, one of the basic requirements of any mobile application is to provide a substantive power utilization. We profiled the performance of *HosTaGe* using an Android application called PowerTutor[16]. PowerTutor provides measurements of power utilization of an application on Android OS. We compared the utilization of *HosTaGe* with other frequently used applications such as WhatsApp[6], Facebook [7] and AVG Free AntiVirus[8].

In order to measure the power utilization of *HosTaGe*, we deployed *HosTaGe* (running in the background) in a network and conducted an automated script testing. We utilized the same device as described in the attack detection analysis (above). The script automates random number of protocol connections, between none to five connections every 30 seconds, to the *HosTaGe* device (emulating attack) for a total duration of 60 minutes.

We compared the power consumption of *HosTaGe* (with script automation) to the other Android applications which were executed in the background. The resulting comparison graph is shown in Figure 4. We observed *HosTaGe* performing reasonably well considering the amount of attacks handled by it throughout the testing duration.

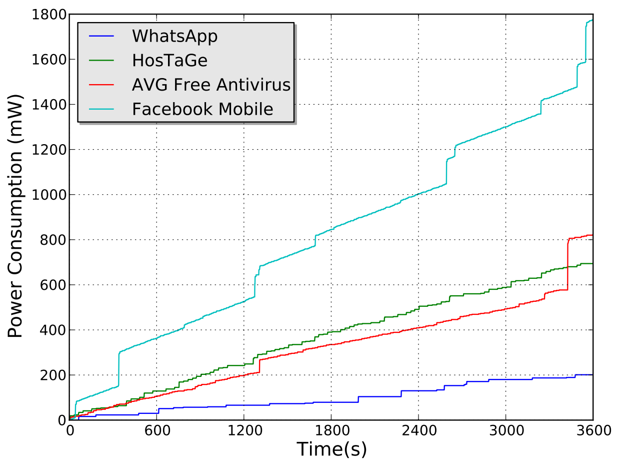


Figure 4: Accumulated Power Consumption Analysis

Limitations and Future Work.

There are still some limitations of *HosTaGe* that are currently being studied and improved. For instance, the requirement of a rooted Android device is limiting the potential user set. Future work will attempt to overcome this limitation. Moreover, additional effort is required to improve *HosTaGe*'s GUI as well as for generic performance and power utilization improvements.

A successful detection triggered by *HosTaGe* depends on the specific behavior of the malware and the timing the device joins the network. Some malware attempt to propagate by scanning IP addresses incrementally for available hosts in the subnet range. If *HosTaGe* is issued an IP address which has already been scanned by the malware, it is unable to detect the malicious activity unless the malware repeats this activity. This malware-specific behavior also influences the amount of time that *HosTaGe* needs to be active in a network before being able to successfully detect an existing malware propagation (if any). However the aforementioned limitations do not apply for malicious users trying to scan or compromise systems over the network.

In terms of new features, adding a *phone home* capability to *HosTaGe* for submitting coarse grained alert data to a centralized monitor is preferable. This could provide us the ability to create a map of open wireless networks with indications of their security status. Moreover, *HosTaGe* devices can also learn about security status of other networks from participating devices. However, this functionality exhibits a

[5]http://www.nmap.org
[6]http://www.whatsapp.com
[7]https://www.facebook.com/mobile/
[8]http://www.avg.com/eu-en/antivirus-for-android

variety of challenges, e.g., dealing with IP addresses under NAT and possible privacy concerns of the users. Moreover, in such a scenario GPS-data, if applicable, could be also utilized.

6. CONCLUSION

In this paper we proposed the idea of Honeypot-To-Go, which is a lightweight low-interaction portable honeypot for mobile devices that aims on detecting malicious wireless network environments. A Honeypot-To-Go may be used by ordinary users that are concerned about the trustworthiness of a wireless network they are about to connect, or even by security professionals and network administrators that require a brief security analysis of their networks on-the-go. We presented our prototype *HosTaGe* along with initial results, indicating its potential on the mobile platform and the feasibility for ordinary users. To the best of our knowledge, *HosTaGe* is the first implementation of a generic low-interaction honeypot for mobile devices.

7. ADDITIONAL AUTHORS

Mihai Plasoianu, Lars Pandikow and Wulf Pfeiffer (Technische Universität Darmstadt, email: `first.last@stud.tu-darmstadt.de`).

8. REFERENCES

[1] S. Antonatos, M. Locasto, and S. Sidiroglou. Defending Against Next Generation through Network / Endpoint Collaboration and Interaction. In *3rd European Conference on Computer Network Defense*, pages 131–141. Springer US, 2009.

[2] S. Antonatos, E. P. Markatos, and K. G. Anagnostakis. Honey @ home : A New Approach to Large-Scale Threat Monitoring. In *ACM workshop on Recurring malcode*, pages 38–45. ACM, 2007.

[3] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling. The nepenthes platform: An efficient approach to collect malware. *Lecture notes in Computer Science*, 4219:165–184, 2006.

[4] B. I. A. Barry and H. A. Chan. Intrusion Detection Systems. In *Handbook of Information and Communication Security*, pages 193–205. Springer Berlin, 2010.

[5] A. Galante, A. Kokos, S. Zanero, and P. Milano. BlueBat : Towards Practical Bluetooth Honeypots. In *IEEE International Conference on Communications*, pages 1–6. IEEE, 2009.

[6] J. Gobel. Amun : Automatic Capturing of Malicious Software. Technical report, University of Mannheim, 2010.

[7] S. Liebergeld, M. Lange, and C. Mulliner. Nomadic Honeypots : A Novel Concept for Smartphone Honeypots. In *Workshop on Mobile Security Technologies (MoST'13), in conjunction with the 34th IEEE Symp. on Security and Privacy.*, 2013.

[8] C. Mulliner, S. Liebergeld, and M. Lange. Poster : HoneyDroid - Creating a Smartphone Honeypot. In *IEEE Symposium on Security and Privacy (S&P)*, 2011.

[9] E. Peter and T. Schiller. A Practical Guide to Honeypots. Technical report, Washington Univerity, 2011.

[10] N. Provos. Honeyd : A Virtual Honeypot Daemon. In *DFN-CERT workshop*, 2003.

[11] C. Seifert, I. Welch, and P. Komisarczuk. HoneyC - The Low-Interaction Client Honeypot. In *NZCSRCS*, 2007.

[12] L. Spitzner. Honeypots : Catching the Insider Threat. In *Computer Security Applications Conference*, number Acsac, pages 170–179. IEEE, 2003.

[13] Symantec. Internet Security Threat Report. Technical Report April, 2013.

[14] M. Wählisch, T. C. Schmidt, A. Vorbach, C. Keil, J. Schonfelder, and J. Schiller. Design, Implementation, and Operation of a Mobile Honeypot. Technical report, 2013.

[15] M. Wählisch, S. Trapp, C. Keil, J. Schönfelder, T. C. Schmidt, and J. Schiller. First Insights from a Mobile Honeypot. In *ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication*, pages 305–306. ACM, 2012.

[16] Z. Yang. PowerTutor - A Power Monitor for Android-Based Mobile Platforms. Technical report, University of Michigan, 2012.

[17] K. Zolfaghar and S. Mohammadi. Securing Bluetooth-based payment system using honeypot. In *International Conference on Innovations in Information Technology (IIT)*, pages 21–25, Dec. 2009.

APPENDIX

The malware used for the attack handling section were downloaded from the Open Malware Archive[9], operated by Georgia Tech Information Security Center. Table 1 gives insights of the malware family as well as the MD5 hash of each particular malware that was used.

Malware Family	MD5 Hash
Win32/Themida	682d6c26a81c5f62e9bb02349c804995
Worm/Generic.AHB	e56d66fa40e3c2097b8824953a5250cb
Downloader.Generic10.BNH	9523e99c4d4267f813c97e795b33480e
PSW.Generic8.GKS	7fe658710e4904e0ee2148ddee02b8e9
PSW.Generic8.HBB	d29a563bdca54d8ca381ea75ff619b2d
PSW.OnlineGames3.AOPH	41b3419b105afe85a38a3ca2765c53fd
Worm/Delf.HA	2e83efa9412ada82f527005d44281792
Worm/Delf.LC	3947c2e31e87ae4a6d5c81b3cee14b15
Win32/Prepender.J	21dc5b946d7a09999f205410ad8f080e

Table 1: Malware Deployed In The Testbed

[9]http://oc.gtisc.gatech.edu