Fraunhofer SIT

The Fraunhofer SIT Malware Analysis Laboratory

Establishing a Secured, Honeynet-based Cyber Threat Analysis and Research Environment

Martin Brunner, Michael Epah, Hans Hofinger, Christopher Roblee, Peter Schoo, and Sascha Todt

Fraunhofer-Institute for Secure Information Technology SIT

Contact: peter.schoo@sit.fraunhofer.de

September 22, 2010

Abstract

This report presents the design of a malware collection and analysis environment for operation in an enterprise context, to facilitate empirical improvements to malware detection and Internet early warning research. In addition to reviewing the overall system design, we describe the operational security measures employed to ensure the secure and isolated handling of malware and other malicious content. The environment uses honeypot, honeynet, and virtualisation technologies to collect and discreetly analyse the behaviour of malware samples circulating on the Internet. To avoid potential liability and damage to the enterprise, our design must minimise the inherent risks of cross-infection to local IT systems and third parties imposed by the collection and handling of malware. Our malware collection and analysis pipeline is based on a novel combination of existing open-source technologies that together offer a highly flexible experimental platform, while fulfilling our operational security obligations. Through careful technical and administrative measures, we are able to sufficiently minimise, if not undermine, the risk of undesired impact to both enterprise and third party systems.

Experiences with our environment as a research tool will be presented in a future paper.

Description of the **NES research department**

The *Network Security and Early Warning Systems* (NES) research department of the Fraunhofer-Institute for Secure Information Technology SIT is researching and developing solutions for the secure operation of networks and network infrastructure services. One focus of this group is IT early warning, which requires significant, in-depth knowledge of IT systems security defences, and consequently up-to-date information on the techniques used by attackers. The analysis of malware and attack techniques contributes to early warning research.

1 Introduction

1.1 Motivation for Malware Analysis

The majority of the volume of contemporary Internet-based attacks can be attributed to botnet-connected malware infecting victim host systems and using them to perform various malicious activities, such as (D)DoS, spam delivery, identity theft, and illegal content hosting. There is a major need to track and understand the rapid evolution of these pervasive threats. Most existing measures that claim to improve the security of IT systems (firewalls, IDS, etc.) suffer from the critical disadvantage of only addressing known vulnerabilities and exploits. In the ongoing race between attacker and defender, this means that the latter can only react to new threats statically. Since the attackers will inevitably have a lead over defenders, timely intelligence on novel threats is essential for improving the state of the art in intrusion detection methods. This requires regimented collection and examination of threat samples, such as malware binaries, service vulnerability exploit shellcodes, and malicious emails that spread malware or trick users into revealing sensitive data. Acquiring sufficient quantities and varieties of this real-world attack data is a major challenge and a fundamental part of our work. This report describes the considerations taken into account by the NES group of Fraunhofer SIT during the design process of a research environment for these collection and analysis tasks.

1.2 Design Goals and Risk Analysis

Without mitigating precautions, the collection and analysis of malware can expose both the analysing party and Internet-reachable third parties to compromise. This is especially the case with unknown malware, whose behaviour might pose a risk to every system either directly or indirectly connected to the malicious binary capture and execution environments.

Dedicated and specially designed deception systems, called *honeypots*, are widely considered useful to capture such network-initiated attack behaviours and malware samples without exposing production systems to potential compromise (for a detailed description see section 2). Unfortunately, Internet-exposed honeypot systems are also vulnerable to compromise. This infection process may be intended or desired with *high-interaction* honeypots, which consist of production-like systems for malware collection. Even with *low-interaction* honeypots, however, which only emulate specific vulnerable services, the possibility of malware

or a human attacker exploiting vulnerabilities in the honeypot software or the underlying host system itself cannot be entirely excluded. A compromised low-interaction honeypot could therefore be used as a launching point to attack other systems connected to the Internet, as with any other infected system. Fortunately, this risk can be minimised through appropriate precautionary measures.

While security incidents that only affect the organisation handling collected malware may lead to internal consequences, they might be much more far-reaching if an external organisation is affected. The severity of damage to third parties may range from loss of money or intellectual property to, in the worst case, the loss of life. The analysing entity could ultimately be held civilly and criminally liable for such damages. To prevent such incidents and minimise the risks involved, the individual threats introduced by such a malware collection and research environment must be carefully evaluated. This has been implemented for the Fraunhofer SIT malware analysis laboratory and resulted in the definition of the following requirements for this facility:

- No connections between the enterprise's and the malware collection and analysis networks (*isolation*).
- No malware connections to the Internet (neither initiated nor received) outside the scope of our control during the collection phase (*controllability*).
- No interaction between malware and third parties (attacker or non-attacker controlled) during the analysis phase (*side effect free*).

Isolation requires that no system operating within the malware collection and analysis network be allowed to connect to the internal network of the Fraunhofer SIT. This can be guaranteed by using independent (physically separated) network equipment, thereby eliminating the possibility of direct damage to the institutional network. Furthermore, the malware collection and analysis network must have its own independent Internet connection. Connections to the institutional network could only be initiated via the Internet, and would be handled as any other potentially malicious connection attempt by production security devices protecting the internal Fraunhofer SIT network.

Vulnerability emulation via the above mentioned low-interaction honeypots (see section 2) is currently the preferred collection mechanism to minimise the risk of system compromise. During the collection and analysis workflow, it is often necessary to retrieve subsequent stage malware binaries from the Internet after the initial exploit of an emulated vulnerability (which would normally initiate the download). Thus, it is unreasonable to simply block all outbound connections. Meanwhile, every initiated connection must be treated as potentially malicious. Instead of downloading a subsequent malware binary, the exploited (emulated) service could be used to attack other systems (e.g., launch an SQL injection attack or participate in a (D)DoS attack). Fine-grained control of all connection attempts from or to the malware collection environment is required to minimise the risk of this abuse. This constraint is defined in the *controllability* requirement.

The *side effect free* requirement concerns the regulation of connections initiated from or to malware during analysis. In many cases, a malicious executable will attempt to infect additional systems or try to establish backchannels to external command and control servers to request further instructions. Such instructions could contain commands to initiate attacks against third parties. Since it is often difficult to interpret the communication between malware and external control servers, especially if the data stream is encrypted, reliable automated differentiation between harmless and malicious communications is impractical. To prevent attacks against third party systems, it was decided that no communication between the analysis environment (i.e., systems running collected malware) and the Internet would be allowed.

2 Malware Collection

One widely-applied malware capture and analysis technology with the described functionality, is dedicated specially-administered systems (physical or virtual), independent of any production system, called *honeypots*. Likewise, *honeynets* are formed by two or more honeypots operating in a network, and enable the emulation of a more diverse set of platforms, services and vulnerabilities. The deployment and operation of a honeynet helps to achieve the objectives outlined in subsection 1.2.

A research honeynet facilitates the collection, monitoring and understanding of the technologies and tactics currently employed by both malware and human attackers. Analysis of the collected malware can provide valuable insights into emerging attack behaviours, enhance conventional intrusion detection methods, and enable research into novel detection and prevention techniques (e.g., machine learning algorithms). In research honeypots that are completely isolated from all enterprise production systems, it can be safely assumed that no legitimate communications is ever initiated from the Internet to the honeypots. This is a key advantage of the technology, as every incoming connection attempt can be assumed malicious, maximising the information density of the collected traffic. This principle leads to highly valuable intelligence that is difficult to acquire within ordinary production networks [1,2] [10].

Honeynets are often deployed within enterprise networks to detect and protect against attacks aimed at the organisation's production systems and networks. The technical and logistical issues involved in establishing such a production honeynet, however, are beyond the scope of this paper.

Apart from data analysis, the operation of a honeypot is based on three fundamental requirements, namely the central

- collection,
- logging and
- control

of all network traffic initiated from the honeynet. There are two basic categories of honeypot systems that differ in the range of interactivity provided to the attacker:

[1] http://old.honeynet.org/papers/honeynet/
[2] http://old.honeynet.org/funds/honeynet_sponsorship.pdf

Low-interaction honeypots emulate potentially vulnerable systems or services. Since an attacker's activities are limited to emulated services, the operation of such a honeypot (or honeynet) poses a minimal risk to reachable networks. As low-interaction honeypots are easily revealed as such by human attacker or sophisticated malware, they are primarily suited for intelligence gathering during the first steps of an attack and for capturing associated malware samples (e.g., Conficker propagation through NetBIOS service).

High-interaction honeypots are real (non-emulated), non-production systems that simulate production uses by offering attackers interaction with real operating systems, services and data. The target audience comprises skilled human attackers and sophisticated or zero-day malware. Due to its complexity, the operation of a high-interaction honeypot is much more time-consuming than with a low-interaction one. When connected to a real network (intranet or Internet), they can also pose a significantly higher risk of undesired attacks against internal assets or third parties, as the interaction between attacker and honeypot is unpredictable and much less restricted (e.g., malware can directly manipulate the OS and establish remote connections).

There are also many combinations and subtypes of the above mentioned honeypot categories.

To comply with legal and liability responsibilities, high-interaction honeypots are not deployed within this environment. Instead, as the focus is on malware collection and analysis, it is based on low-interaction honeypots. These basic conditions lead to the design of the environment described in the following chapter.

3 Environment: Hardware and Software

This chapter describes the system design of the malware collection and analysis environment, which integrates various hardware and software components in a specialised networking topology. The network is partitioned into multiple security zones to properly isolate collection, analysis and management functions in the laboratory. Potentially dangerous behaviours are strictly contained to satisfy the previously detailed security precautions and assure a safe and reliable operation of the honeynet system. A sketch of the network design is presented in figure 3.1. For consistency, we refer to this overall environment comprising honeypot, management and analysis components as a notional *honeyfarm*.

3.1 Hardware

The honeyfarm's hardware components are selected to support a smooth operation of the malware collection and analysis software and to enable granular control of all inbound and outbound network traffic.

3.1.1 Perimeter Router and Firewall

The honeyfarm's LAN is connected to a dedicated Internet link through a configurable router/firewall device, which provides the following perimeter protections (among others) from outbound and inbound attacks:

- configurable firewall with stateful packet inspection
- NAT routing to block unsolicited inbound connections
- static routing for custom honeynet routes
- administration, blocking and logging of outbound connections from specific IP ranges (e.g., honeypot machines)
- port-based isolation between DMZ, management, and WAN zones

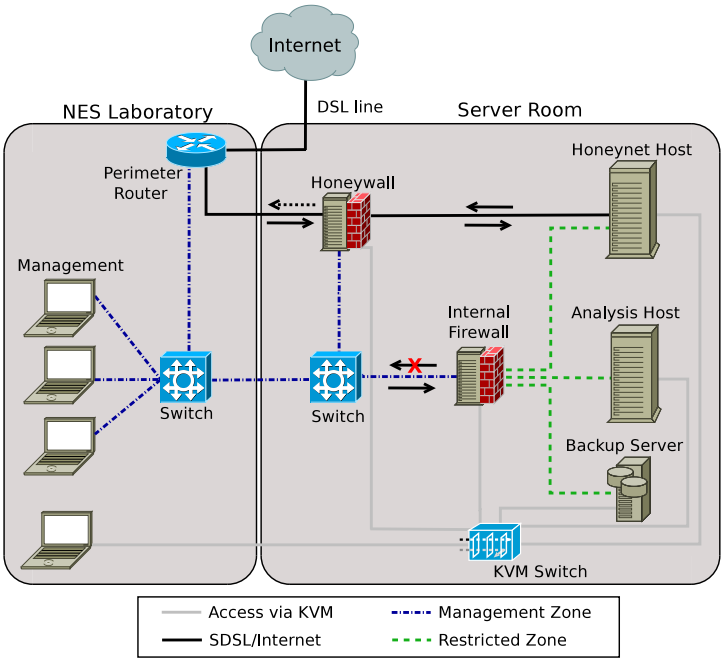

Figure 3.1:
Malware analysis laboratory (honeyfarm) network diagram depicting each system's location and security zone. The honeynet host's Internet access is controlled by a honeywall. The second (internal) firewall enforces the separation between management and restricted security zones, only allowing inbound connections to the latter. This firewall also controls communication within the restricted zone by functional role.

3.1.2 Honeywall

To reach the honeynet host (see Fig. 3.1) after passing through the perimeter router and firewall, packets must pass through the honeywall, a dedicated system running specialised data control software (see subsection 3.2.1). This system is equipped with three network interface cards (NICs) - one each for incoming and outgoing traffic, and one for management. To avoid IP address assignment, the honeywall operates merely as a network bridge and is therefore not addressable above the data link layer. This protects it from any possible network cross-contamination or compromise, as it is transparent to the honeyfarm LAN at the network layer.

3.1.3 Host Machines

The analysis and honeynet hosts utilise virtualisation extensions to separate and manage different instances of honeypots and analysis environments efficiently. The *Ether* [4, 3] malware analysis platform, described in section 4, was selected to facilitate dynamic analysis.

3.1.4 Internal Firewall

The internal firewall system (see "Firewall" in Fig. 3.1) is a standard server with four network ports. As a multi-homed firewall, it is responsible for segregating the different security zones in the honeyfarm. Its fundamental role is to protect management and third party networks by blocking outbound connections from the honeynet, analysis and backup hosts in case they were ever compromised. As shown in Fig. 3.1, the internal firewall only allows connections to be initiated from the management zone to the restricted zone. The firewall also restricts how systems within the restricted zone may communicate with each other:

- Honeynet host may only send logs and time synchronisation requests to the backup server
- Analysis host may initiate connections to and pull collected files from the honeynet host on a specific port (analysis pipeline). It may also send logs and time synchronisation requests to the backup server
- Backup server may initiate connections to and pull files from the honeynet and analysis hosts on a specific port (backup)

3.1.5 Backup and Repo Server

The backup and repo server (see "Backup" system in Fig. 3.1) is a dedicated file server to store archive, analysis, and software package data. Its primary role is to serve synchronous backups for the honeynet, analysis and honeywall systems. Additionally, it serves time synchronisation (NTP) and remote system logging for the entire honeyfarm. To miminise the backup server's attack surface, it only performs *pull-based* backups of the honeynet and analysis hosts, instead of the more conventional *push-based* backup (see subsection 3.1.4). Backups of the honeywall system are push-based, as the internal firewall allows inbound connections from the management zone.

3.2 System Software

This section describes software used for the honeyfarm's malware collection and analysis processes, and its respective dependencies.

3.2.1 Honeywall

The honeywall manages and audits all connections between the honeynet host and the Internet (see Figure 3.2) and facilitates the central collection, logging and control of all network traffic initiated from the honeynet.

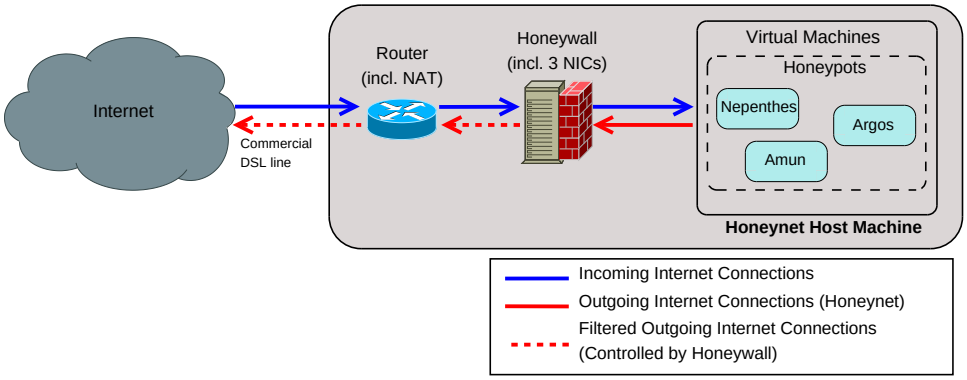

Figure 3.2:
Sub-diagram of the honeyfarm's malware collection components, depicting possible network flows to and from the Honeynet Host. The honeywall strictly controls in and outbound traffic. On behalf of the captured malware, the Nepenthes (low-interaction honeypot) fetch module can download further malware parts, if necessary.

The *Roo*[1] honeywall is a stand-alone system with the following built-in security mechanisms:

- hardened Linux OS
- iptables (firewall encompassing packet filtering and routing)
- snort_inline (NIDS/NIPS)[2], (cf. section 3.3.1)
- support for dedicated management interface

By default, the honeywall system blocks all outgoing traffic initiated by honeypots on the honeynet host, preventing it from being abused as a launching point for subsequent attacks in case of compromise. Outbound traffic originating from the honeynet host is restricted as described in section 3.3.1.

3.2.2 Host Machines

The analysis and honeynet hosts are standard Debian systems hardened with firewalls, anti-virus software, host-based intrusion detection and prevention software, and the principle of least-privilege. Malware analysis is performed on the *Ether* framework [3], which leverages hardware virtualisation extensions and resides completely outside the virtual target OS. This allows for the transparent (and therefore higher fidelity) monitoring of malware executables. *Ether* requires a patched Xen 3.1 hypervisor running on 64-bit CPU architecture.[3]
The system's integrity is furthermore monitored as described in section 5.3.

[1] http://old.honeynet.org/papers/cdrom/roo/index.html
[2] http://snort-inline.sourceforge.net/oldhome.html
[3] http://ether.gtisc.gatech.edu/source.html#prereqs

3.3 Honeypot Software

Various methods exist to collect malicious content from the Internet. In order to avoid effects to third party systems, we are limited to those designed for safe and isolated collection.

3.3.1 Low-interaction Honeypot

Low-interaction honeypot systems like *Nepenthes* [2] emulate known vulnerabilities of Internet-connected services and operating systems, without providing direct access to the underlying system. The limited interactivity provided to the attacker minimises the risk of infection of the actual honeypot host itself. Since low-interaction honeypots involve far less risk and are more easily deployed and maintained than high-interaction honeypots, they are often used to monitor unused IP space within production networks to asses risks to operational systems and provide early warnings to system administrators.

Nepenthes specifically facilitates our research with its:

- automated capture of worms, bots and other exploits pushed from the open Internet
- optional automatic downloading of malware from the Internet through fetch modules (see below)
- ability to collect novel threat payloads
- malicious shellcode decoding and handling
- extensibility through development of custom vulnerability modules

If a vulnerability is exploited in a way that is not mapped to a vulnerability module, the interaction fails. However, these attempts are logged to facilitate future manual analysis or vulnerability module development.

Nepenthes can further handle multi-step infection schemes in which shellcode is injected into a vulnerable service: its *Shellcode Parsing Module* is responsible for extracting the download location of the next-stage malware. This information is then passed to its *Fetch Module*, which can automatically download this malware sample. When an emulated service is "infected", these outgoing connections are often used to download the next malware stage through ordinary or benign commands (e.g., HTTP GET, etc.). To bypass the firewall, which blocks outbound traffic by default, the Fetch Module operates on a separate virtual machine (VM) on the honeynet host with a different IP address. However, to block malicious outbound connections targeted at third party systems (i.e., non-attacker controlled), the honeywall implements attack detection and prevention through the *snort_inline* IDS, capable of dropping all inbound and outbound packets deemed malicious with respect to its ruleset. Therefore, we can filter all of the outgoing traffic following specific and regularly-updated rules.

Although Snort provides generic rules to detect web application attacks (e.g., SQL injections), more sophisticated attacks that use obfuscation techniques may circumvent these generic Snort rules. To prevent such attacks that may result in damage to a third party system, additional Snort rules will be implemented to generally detect and block more sophisticated attacks, such as SQL injections.[4]

To summarise, low-interaction honeypots like the Nepenthes platform, in combination with the Roo honeywall, are especially well suited for safe malware collection and analysis, since they:

- do not require installation or execution of any vulnerable services or applications - they are merely emulated
- run on a Linux host, not infectable by exploits targeting the emulated Windows services
- passively monitor the network
- strictly control outgoing communications
- offer a built-in IPS (snort_inline) to block outbound attacks against third parties
- quarantine collected malware samples in a secure location
- are easily deployed on a VM for tighter control and compartmentalisation

Apart from the Nepenthes, other low-interaction honeypot systems are also suitable for use in our honeyfarm. Alternatives such as Nepenthes' successor *Dionaea*[5], *Amun* [5], and *HoneyTrap*[6] are also promising solutions for our safe malware collection needs and will be evaluated.

3.3.2 Spamtrap

A spamtrap is a special type of honeypot designed to collect (or divert) spam email and linked content, such as attached malware and embedded malicious URLs. As spamtraps operate passively and only accept incoming email, no communications are initiated to third party systems. Our spamtrap is deployed within a virtual honeypot on the honeynet host. It receives and processes all unsolicited emails sent to email addresses for a specially registered domain. Email addresses for this domains are posted to various open Internet forums and web sites as bait to induce harvesting by spammers or potential attackers. The assumption is that 100% of the messages received by the spamtrap honeypot are either advertising spam, malicious, or both. The objective of this effort is to lure as much maliciously-tainted email as possible, such as phishing attacks containing malware attachments or links to malicious web sites. The extracted malicious

[4] http://www.securityfocus.com/infocus/1768
[5] http://dionaea.carnivore.it/
[6] http://honeytrap.carnivore.it

content is subsequently analysed, profiled and catalogued by the analysis host, along with the samples retrieved from other honeypots.

Specific objectives include:

- acquisition and maintenance of a constantly-evolving malware corpus for research
- tracking of emerging trends in email-borne attacks for early warning
- automated characterisation and classification of malicious URLs (e.g., for fast flux service networks)

4 Analysis Pipeline

We now review the chain of analysis components and notional workflow in the honeyfarm. A detailed description of each component beyond the scope of this paper, and can be obtained from the references.

The full end-to-end analysis pipeline consists of the stages and components depicted schematically in Fig. 4.1 and is summarised in the following subsections.

Step 1: Capture and Storage

Malware samples are captured from the Internet through the described honeypots (e.g., spamtrap, low-interaction honeypot as described in section 3.2) and quarantined in a separate encrypted filesystem (see section 5.1.2).

Step 2: Transfer and Cataloging

Malware samples are copied to the analysis host from the quarantine filesystem via pull mechanism. All files transfers are logged and tracked with MD5 fingerprints.

Step 3: Static Analysis

Each recovered sample is administered a regime of static analysis tasks, resulting in initial surface-level malware profiles. Static analysis does not require the execution of malcode, and is the least risky form of analysis. It is easily and safely carried out in various environments. The following standard analysis tasks are executed (described in [6]):

- file fingerprinting
- virus profiling (e.g., with AV engines)
- static disassembly
- string extraction (unpacked executables)
- packer detection and tagging

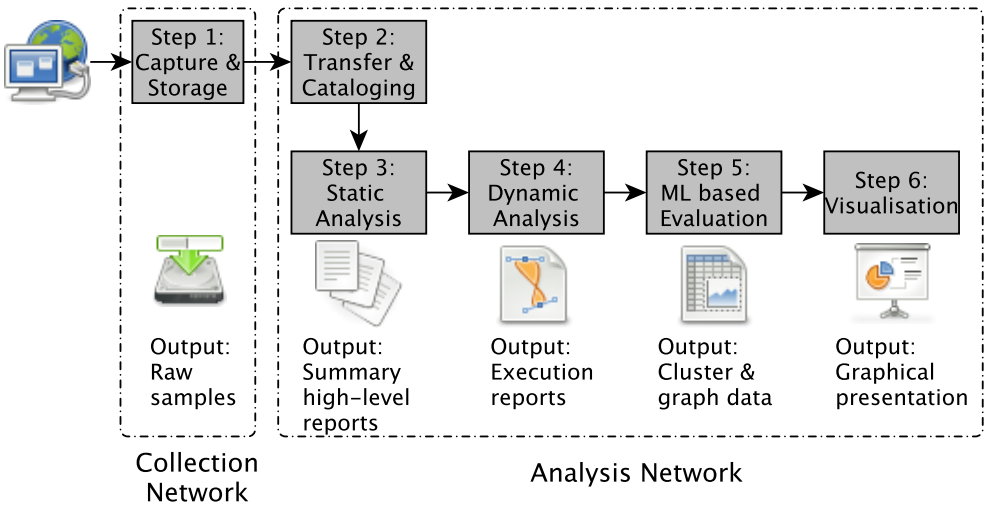

Figure 4.1:
Analysis pipeline for malware samples processed in the honeyfarm. Each subsequent step leads to a finer grained analysis and facilitates discovery of the relationships between investigated malware samples.

Step 4: Dynamic Analysis

This is the most critical stage in the pipeline, as it entails actual execution and live examination of malware. It is therefore conducted offline and restricted to sandboxed execution environments. The sandboxes are standard Xen VM (unprivileged domains) running unpatched operating systems (e.g., Windows XP), and serve as victims for the injected malware.

Some common types of dynamic analysis, also described in [6], include:

- process monitoring (track post-unpacking runtime behaviours)
- debugging (e.g., IDApro[1])
- network monitoring through packet capture

To impede malware from discovering that it is monitored, we use a combination of the Xen virtual machine monitor and the *Ether* [3] open-source transparent analysis framework. The latter resides in part in the userspace of Xen's managing domO and in part in the hypervisor, but not in the virtual machine that is used for investigation. Thus the Ether controller enables fine-grained malware tracing directly at the host level and without instrumentation within the virtual guest systems.

Since malware sandboxes are generally configured with limited or no connectivity to the Internet, they do not allow malware to receive necessary updates or to exercise their full life cycles. To partially mitigate this weakness while maintaining isolation from external networks, we must emulate certain requested services.

[1] http://www.hex-rays.com/idapro/

This is solved through a combination of the *honeyd* [7] and methods for sinkholing and emulating Internet services. These systems enable a limited, and tightly controlled amount of *virtual* Internet connectivity to sandboxed malware.

Honeyd is a software daemon that creates multiple virtual honeypots to simulate a network running different operating systems and services. It can instrument darknets by emulating address space (via arpd) and by providing emulated services via custom scripts. Although honeyd is often used as a low-interaction honeypot to detect or deter adversaries on the open Internet, we use it to emulate anticipated Internet resources requested by malware during dynamic analysis (e.g., DNS, WWW, SMTP, IRC) [7, 8].

Additionally emulating generic services of the Internet is intended to trick malware into believing that it is online and can commence its normal program, otherwise prevented by its obfuscation mechanisms.

In summary, the automated experimental sequence of dynamic analysis

1. transfers and launches the malware samples on victim sandboxes
2. records detailed malware execution behaviours (e.g., system calls, fine-grained instructions, opened network sockets)
3. serves and captures any network requests through Internet emulation

The malware is allowed to run for a specified amount of time before the sandbox is halted and its disk image either discarded or saved for post-mortem forensics. The resulting captures and reports form the basis of a solid understanding of the malware's behaviour without observing its full operation, which would require Internet connectivity or a high-interaction honeypot.

To comply with the third (side effect free) operational requirement, executed malware is never allowed to access the "real" Internet in any way during dynamic analysis.

Step 5: Machine Learning-Based Evaluation

The resulting Ether-based dynamic analysis runtime reports are processed by a suite of machine learning algorithms to derive better understanding of the corresponding malware samples' relationships and lineages. They are also used to identify novel classes and variants of malware. The algorithms cluster similar (perhaps unknown) malware behaviours or classify new samples by assigning them to known groups of malware. The pipeline currently uses the Malheur [9] tool to perform these machine learning tasks.

Step 6: Visualisation

Information gleaned from the above analysis is visualised in constantly-updating, user-intuitive graphs and charts. A novel strain of emerging malware behaviour could, for example, manifest itself conspicuously as an independent cluster, expediting both operational analysis and research. Our system currently leverages the capabilities of the Guess graph visualisation framework [1] and the Google Motion Charting API. [2]

[2] http://code.google.com/intl/de-DE/apis/visualization/documentation/gallery/motionchart.html

5 Management & Precautions

5.1 Data Management

5.1.1 Data Types

The processing and storage of various types of operational and experimental data within the honeyfarm requires well-defined data management and handling procedures. Expected *data types* generally fall into the following categories:

- *Network traffic traces (pcap)*:
  may contain both benign and malicious communications
- *Network summary data*:
  may describe both benign and malicious flows, sampled pcap, etc.
- *Stand-alone malicious binaries (malware)*:
  this type of data is independent from any encapsulating network traffic
- *Network application layer payload*:
  independent of its containing network traffic (e.g., emails, web pages, multimedia files), whereas much of this could be malicious
- *Benign binaries*:
  used as control group in experiments, such as those from a standard Windows installation or legitimate application
- *Log and alert data*:
  refers to structured data generated by sensors and applications (e.g., IDS, firewall, syslog, SMTP)
- *Honeypot virtual machine disks, snapshots, and states*

5.1.2 Storage and Processing

The stored and processed data types pose different security risks to the honeyfarm. Their handling is defined separately as follows: Most data resides on the dedicated storage server (Backup Server in figure 3.1). Malware files are stored on a dedicated encrypted partition (hosted on an external USB drive) to segregate it from other data. The external USB drive(s) hosting the malware are physically connected to the honeynet host as direct attached storage (DAS) and

Table 5.1:
Anticipated types, maximum accumulated volumes, and storage of data handled in the honeyfarm

Data Type	Volume (approx)	Storage Location
Network traffic trace	TBs	backup server
Network summary	10s of GB	backup server
Malware	100s of MB	encrypted USB drives
Benign binaries	100s of MB	backup server
Network application layer	10s of GB	backup server
Log & alert data		backup server
Virtual machine images	100s of GB	local disks

are mounted via the guest operating system running the honeypot. The transfer to the dedicated analysis host machine is conducted via a pull mechanism. Thus, the collected malware is only stored on the USB drive and the analysis host machine. Apart from the USB drives, all described machines and devices implement state-of-the-art measures to ensure data security (such as dedicated RAID and logical volumes). This strong separation avoids possible mixing of benign and (potentially) malicious data throughout processing. Table 5.1 summarises the data types and their handling.

5.1.3 Backup and Archiving

To assure the segregation of benign and malicious content over the lifetime of the processed data, the issue of long-term storage must also be covered. This specifically entails archiving collected and analysed malware, as well as (benign) configuration and log data. The basic backup strategy is defined as follows to achieve the separation and archiving of both malware and benign data:

- one external storage device acts as Direct Attached Storage (DAS) with a corresponding RAID configuration, used to backup the storage server (hosting all the benign honeyfarm data)
- a second external DAS with a corresponding encryption setup backs up the USB drives storing malware

5.2 Role Management

An audit-proof operation of the honeyfarm requires a thorough and systematic tracking of all changes performed, especially configuration changes and the handling of malicious binaries. This is not only vital for operational security (e.g., in

case a host system is itself compromised), but also for the reproduction of (legitimate) modifications to the system, such as configuration changes or software updates. Therefore, all user accounts on all honeyfarm systems must comply with the following requirements:

- Disable root logins and enforce sudo usage for privileged tasks to avoid working directly as root.
- All accounts are personalised.
- Shared credentials may be used in case of sensitive privileged user accounts. For example, the password can be split into two or more parts, with each user knowing only one.
- Documentation of the list of users and their assignment to specific roles and permissions.

5.3 Change Management

Changes to honeyfarm systems may negatively impact compliance with the three requirements defined in section 1.2, especially if firewall configurations are modified. Therefore, all modifications must be carefully implemented and communicated to all honeyfarm staff and management. A consistent change management strategy needs to be followed. Tracking system changes in this context refers to changes to both configuration and binary data (i.e., result of updates or security fixes). Auditing important configuration files (e.g., the contents of /etc) is accomplished through automated, scheduled, differential backup of these files to a version control system. Changes to these configuration files are justified, documented and made available to authorised users in contingency.

The *AIDE* [1] host-based IDS and file integrity checker tracks all changes to both configuration and binary data. A revision control system, such as *Subversion* [2] (configured with local repositories) is used to track historical changes to configuration files. This allows all modifications to be tracked by honeyfarm staff, who receive daily email reports. Additionally, critical system alerts requiring immediate attention are emailed asynchronously to all staff.

To minimise the risk of a honeyfarm host compromise, all systems must be kept at the most recent security patch level possible. Since a permanent connection of the honeyfarm systems to an external package repository poses a potential risk to third parties, it was decided to use an offline update mechanism. Updates are installed by copying necessary packages to the honeyfarm systems using the management hosts, which are protected by the internal firewall depicted in figure 3.1.

[1] http://sourceforge.net/projects/aide/
[2] http://subversion.apache.org/

5.4 Special Precautions

5.4.1 Use of WLAN

In addition to the physical separation between honeyfarm and enterprise networks, there remains the risk of WLAN-capable components connecting to the enterprise or other WLANs and causing damage there. Although the enterprise WLAN is only accessible to authenticated users, the wireless cards were physically removed from all WLAN-capable systems to prevent inadvertent connections.

5.4.2 Incident Response

The exposure to all known attack vectors has been minimised by the state-of-the-art means as described earlier. We now describe the remaining attack scenarios and their possible countermeasures.

In the unlikely event that the honeynet host is compromised, special precautions have been taken to prevent an attacker or malware from abusing the honeyfarm to damage other networks. The honeywall blocks all connection attempts from the honeynet host and all honeypots (running as VMs), except for the VM hosting the malware download fetch module. This is the only VM allowed to initiate outbound connections.

To compromise the honeynet host, an adversary would need to find exploitable vulnerabilities in the entire software stack to gain root-level host access. Despite the unlikelihood of a successful attack against the corresponding services, the honeynet host and all of its honeypots are monitored using a host-based IDS, with configuration changes and critical alerts regularly emailed to honeyfarm staff.

Since the honeywall is configured as an IP-less bridge, it is not directly addressable from the honeypot IP range. The only way to manage the firewalls and honeywall is with dedicated systems in the honeyfarm management network.

We are not aware of a scenario that would allow an attacker to compromise a honeypot system to an extent where we would be unable to regain control. Nevertheless, the wiring of all nodes has been designed such that the link between the honeynet host (or the entire honeyfarm) and the Internet can be physically disconnected at any time by all staff, without needing access to the server room hosting the honeyfarm systems.

6 Administrative Compliance

We now revisit the security and administrative issues involved with operating the described malware collection and research environment, and summarise how they are addressed by the implemented technical measures. To comply with the enterprise's mandate to eliminate liability exposure, our design provides maximum assurance that it will not adversely impact local or third party systems.

As discussed earlier, the following three technical requirements were derived from the central mandate (see subsection 1.2):

- No connections between enterprise and honeyfarm networks (*isolation*)
- No malware connections to the Internet (either initiated or received) outside the scope of our control during the collection phase (*controllability*)
- No interaction between malware and third parties (attacker or non-attacker controlled) during the analysis phase (*side effect free*)

Isolation requires a permanent physical separation between all honeyfarm components and enterprise networks or unrelated IT resources. These enterprise systems are also protected from direct or indirect exposure to captured malware artifacts (e.g., binaries, scripts, configuration files). The use of isolated network hardware in a locked laboratory and server room, dedicated Internet connectivity, and encrypted malware storage provide this separation. WLAN separation is likewise guaranteed through hardware and software modifications to all wireless-capable devices operating in our honeyfarm. These precautions minimise the risk of inadvertent data connections between enterprise and honeyfarm networks to the maximum practicable extent.

We are further considering a network access control (NAC) solution to prevent devices connected to other networks from initially joining the honeyfarm, or to proactively disconnect honeyfarm devices as soon as they connect to another (unauthorised) network.

Controllability applies to the honeyfarm's interface with the open Internet. Exceptionally tight control is required over all connections between the honeyfarm and the Internet during malware retrieval. Therefore, our design only allows low-interaction honeypots (e.g., Nepenthes) to connect to remote servers and retrieve malware. Since these low-interaction honeypots operate under deterministic control and do not actually execute malware, they comply with the controllability requirement. The honeywall provides additional network-level control over all connections, simultaneously protecting third parties from the honeynet

and preventing undesired inbound connections to the honeynet itself. The honeywall only allows connections to and from specifically designated VMs (e.g., low-interaction and spamtrap honeypots) on the honeynet host. The internal firewall eliminates additional pathways between the honeynet and the Internet or other honeyfarm systems, in the case of compromise.

The **Side effect free** requirement concerns the handling, execution and analysis of collected malware executables. No malware is allowed to connect to the Internet and communicate with third parties, whether they be attackers or innocents. Malware can either be downloaded (pulled) by the low-interaction honeypot or pushed to the spamtrap. In both of these scenarios new malware samples are immediately quarantined from the collection mechanisms for analysis, which is performed on separate VMs or servers. All subsequent malware execution and dynamic analysis is conducted in off-line sandboxes isolated from real networks (all services are emulated), preventing connections to third parties on the Internet. The honeywall provides an additional barrier preventing connections to third party systems at the network level (see *Controllability*, above). Finally, the internal firewall enforces separate security zones for the analysis host and other systems, eliminating additional pathways to the Internet in the case of compromise to any of the restricted zone systems.

7 Summary and Conclusion

This report provides an overview of the Fraunhofer SIT Malware Analysis Laboratory and its operation as a secured, honeynet-based cyber threat analysis and research environment. In summarising the applied technologies and countermeasures, we have described our implementation of the maximum precautions necessary for the safe and productive operation of a research honeyfarm. The resulting security level of the research environment actually exceeds that of ordinary operational networks, and therefore poses less risk to systems or third parties than other properly administered systems. Thus, the risk of liability for damage to third party systems is generally lower than that with conventional enterprise IT systems, which are inherently vulnerable to compromise.

The presented malware collection framework and analysis pipeline are based on a novel combination of existing open-source tools and technologies. They provide a highly flexible experimental platform while fulfilling our operational security obligations. We expect to release a follow-up report describing practical experiences, sensor improvements, and useful malware analysis methodologies in the future, with reference to this technical report.

Bibliography

[1] Eytan Adar. Guess: a language and interface for graph exploration. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 791–800, New York, NY, USA, 2006. ACM. 19

[2] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F.C. Freiling. The Nepenthes Platform: An Efficient Approach to Collect Malware. In Diego Zamboni and Christopher Krügel, editors, *RAID*, volume 4219 of *Lecture Notes in Computer Science*, pages 165–184. Springer, 2006. 13

[3] Artem Dinaburg, Paul Royal, Monirul Sharif, and Wenke Lee. Ether: malware analysis via hardware virtualization extensions. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, pages 51–62, New York, NY, USA, 2008. ACM. 10, 12, 17

[4] Yaozu Dong, Shaofan Li, Asit Mallick, Jun Nakajima, Kun Tian, Xuefei Xu, Fred Yang, and Wilfred Yu. Extending xen with intel® virtualization technology. *Intel Technology Journal*, 10(3):193–203, 2006. 10

[5] J. Göbel. Amun: A python honeypot. Technical Report TR-2009-008, Laboratory for Dependable Distributed Systems, University of Mannheim, Germany, 2009. 14

[6] K. Kendall and C. McMillan. Practical malware analysis. In *Black Hat Conference, USA*. f13-labs.net, 2007. 16, 17

[7] Niels Provos and Thorsten Holz. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*, chapter Honeyd – The Basics, pages 105–134. Addison Wesley Professional, 2007. 18

[8] Niels Provos and Thorsten Holz. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*, chapter Honeyd – Advanced Topics, pages 135–162. Addison Wesley Professional, 2007. 18

[9] Konrad Rieck, Philipp Trinius, Carsten Willems, and Thorsten Holz. Automatic analysis of malware behavior using machine learning. Technical Report 18, Berlin Institute of Technology, 2009. 18

[10] Ryan Talabis. Honeynet learning: discovering it security. *SIGCSE Bull.*, 38(2):110–114, 2006. 7