VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

School of Mathematical and Computing Sciences
Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341, Fax: +64 4 463 5045
Email: Tech.Reports@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/research

Taxonomy of Honeypots

Christian Seifert, Ian Welch, Peter Komisarczuk
{christian.seifert, ian.welch,
peter.komisarczuk}@mcs.vuw.ac.nz

Technical Report CS-TR-06/12
June 2006

Abstract

In this paper, we present a taxonomy of honeypots. This taxonomy adheres to the characteristics defined by Lindqvist et al and Krsul. We describe how to assign honeypots to classes via step-by-step instructions. We include six classes as part of the taxonomy's classification scheme: interaction level, data capture, containment, distribution appearance, communication interface, and role in a multi-tier architecture. We applied the classification scheme to classify seven distinctly different honeypots: Google Hack Honeypot, Honeyclient, Honeyd, Honeynet, Honeytrap, KFSensor, and a network telescope. The classification successfully separated these honeypots into different classes. The overall classification provided insight into current honeypot technology. Functional gaps exist around containment of malicious activity and utilization of non-network hardware interfaces. The classification also assisted us in predicting honeypot technology of tomorrow. In particular, it pointed towards a possible future honeypot technology of low interaction client honeypots.

1 Introduction

A honeypot is a security device that is designed to lure malicious activity to itself. Capturing such malicious activity allows for studying it to understand the operations and motivation of attackers, and subsequently helps to better secure computers and networks. A honeypot does not have any production value. "It's a security resource whose value lies in being probed, attacked, or compromised" [16]. Since it does not have any production value, any new activities or network traffic that comes from the honeypot indicate that a honeypot has been successfully compromised. As such, a compromise is very easy to detect on honeypots. False positives, as commonly found on traditional intrusion detection systems, do not exist on honeypots.

Honeypots' origins can be traced far back to military concepts and usage, but first appeared in the area of computer security in the 80s. Stoll describes the hunt of a hacker in 1986 [17]. For the purpose of monitoring the intruder on the live systems, Stoll and his colleagues provided "bait", fake military reports, to lure the attacker into a particular area of their system. While this was not the honeypot that we know today, it was the first attempt of "catching flies with honey". The initial honeypot that made use of a simulated environment was described by Cheswick in his account of tracking the Dutch hacker Berferd in 1991 [10]. In the late 90s, attempts to lure and observe attackers moved to the mainstream with the introduction of various tools [11, 1, 6] and commercial products [5, 3].

The tools that have been developed are labeled differently from the term honeypot. Examples are Honeyclient, Honeymonkey, Honeywall, Honeynet, and Honeyfarm. Despite the different labels, which attempt to express a tool category or simply provide a name the specific tool, we refer to all of them under one term for the purpose of this paper. We refer to all as honeypot systems as they adhere to Spitzner's definition above.

In this paper, we present a taxonomy for honeypot systems. We believe a taxonomy for honeypots is important so they can be studied in the research community efficiently. It allows researchers to agree on the object of study, allows for focus of specific areas within the object of study, and provides a framework for communication of research around honeypots. As increased understanding results from this classification, it will lead to improved design and development of honeypot technology. Currently no sufficient taxonomy in the area of honeypots exists, which hinders the advancement of the honeypot research field.

In the next section 2, we review the definition of taxonomy and different types of taxonomies that exist. Further, we summarize the characteristics of a good classification scheme, which we apply to the classification scheme that we present in section 4. We review related work in section 3 and apply our classification scheme to existing honeypot technology in section 5. In section 5.8, we review the results of the classification and present the insights obtained from it followed by a discussion in section 6 around some issues we encountered during the classification. We conclude in section 7 and point out some future work in section 8.

2 Characteristics of Taxonomies

Taxonomy is the theoretical study of classification, but can also refer to the specific classification or the underlying principal of the classification. The classification scheme of a taxonomy consists of classes with values to which objects of study are mapped, depending on the objects' characteristics. The means of assigning the object to a class value is also referred to as classification. Classes of a classification scheme are usually arranged hierarchically, but may also be arranged in other relationship schemes. Classifications based on empirical data are called a posteriori classifications, whereas classifications that are created without empirical data are called a priori classifications. We will be creating our honeypot classifications based on data obtained through experimentation; as such, we will follow a posteriori classification.

Krsul and Lindqvist et al determined that any taxonomy should hold to specific properties [14, 13]. We have combined these properties in a comprehensive list that we adhere to for the taxonomy presented in this paper:

- Comprehensible - The taxonomy should be easily understood even by people without expert knowledge.
- Conforming - The terminology of the taxonomy should comply with commonly accepted terminology of the field of study.
- Useful - The taxonomy should gain insight in the field of study.
- Determinism -The method of assigning an object to a class should be clearly defined.
- Objectivity - Observable value of a class should lead to unambiguous classifications.
- Repeatability - Classifications should lead to same results independent of who performs the classification.
- Exhaustive - All classes, taken together, allow classifying all objects of study.
- Mutually Exclusive - A class value of one class does not influence classification of other classes, as classes do not overlap.
- Specificity - The values of a class must be unique and unambiguous.

3 Related Work

One purpose of honeypots is to detect intrusions. As such, honeypots closely relate to intrusion detection systems. Debar et al present a taxonomy of intrusion detection systems [12] which was later expanded upon by Axelsson [8]. They propose a classification on various aspects of the detection principle of an intrusion detection system, as well as system characteristics such as time of detection, granularity of data processing, source of audit data, response to the intrusion, location of data processing and data collection, security, and degree of inter-operability. If this taxonomy were applied to classify a high interaction honeypot, such as Honeynet, and a low interaction honeypot, such as Honeyd, it would result in the same classification for both as there is no differentiating class identified by Debar et al and Axelsson. The property of *usefulness* would not be met; therefore, this taxonomy cannot be applied to honeypots.

Undercoffer et al present an ontology for intrusion detection, the basis of which is a taxonomy for classifying attacks [18]. Attack consequence, location, and means are the main classes of the proposed taxonomy. Since honeypots deal with the same attacks that intrusion detection systems are raising alerts on, Undercoffer et al's ontology can also be applied to attacks experienced by honeypots. However, this ontology does not allow for classification of honeypots themselves.

Zhang et al presented the first taxonomy on honeypots [21]. The class *security goal* is the sole class of their taxonomy. There are four values presented for this class: prevention, detection, reaction, and research. A honeypot that acts as a decoy is classified with a security goal of *prevention.* It lures attackers to itself, therefore preventing attacks on the real production systems. A honeypot simply tasked with capturing and alerting on intrusions is classified with a security goal of *detection.* If such a system mirrors a production environment and its primary purpose is forensic analysis of the honeypot to find the cause of an attack and patch the vulnerability, it is classified with a *reaction* security goal. Last, *research* security goal is placed on any honeypots whose results are used to understand the tactics, motivation, and tools of attackers.

Zhang et al's taxonomy is insufficient as it solely focuses on the security goals of honeypots. It is too narrow to classify honeypot technology as it would fail to classify diverse honeypots differently. The taxonomy, as a result, would not increase insight into the study of honeypot technology. If the authors' intent was to only understand the security goal of honeypots, we believe their taxonomy fails several essential properties of taxonomies as outlined in section 2. The values within the class are not mutually exclusive. A honeypot with the security goal of detection might also fill the security goal of prevention. Further, the security goal is not easily observable as a deployed honeypot does not contain information about its security goal. As a result, the classification is subjective and is likely to differ as the object of study is classified repeatedly.

4 Classification Scheme

In this section, we present the classification scheme of our taxonomy of honeypots. With each class that our scheme contains, we describe the class with its possible values in common honeypot terminology, along with a methodology of how to assign the object of study to the class. The six classes for our classification scheme are interaction level, data capture, containment, distribution appearance, communication interface, and role in a multi-tier architecture. A few classes may allow an object to be mapped to multiple values of the class. This is not because the values are ambiguous or overlap, but rather because the honeypot implements several strategies. In case the honeypot should be assigned to one value only, we mention so in the class description.

Classification of a honeypot is performed by observing the honeypot's behavior based on test cases the taxonomist devises according to the methodology presented in this section. As a result, classification could differ depending on the test cases that are used by the taxonomist. We believe the likelihood of such a scenario is low, as the defined functionality of a system should lead to the creation of similar test cases to observe the behavior of a honeypot if accepted test case creation methodology is followed. As such, even though we are faced with some variance in test cases, the resulting classification should be identical. Further, as test cases are being created and shared within the research community, this variance is likely to decrease resulting in identical test cases and therefore identical classification.

Figure 1 depicts a graphical representation of the classification scheme. The nodes in solid boxes represent the classes, whereas child nodes with dashed boxes represent possible values for the class. A potential hierarchical relationship between classes would be depicted by the child node consisting of another class (solid box) in which the label of the edge to this child node would represent the selected value for the class. Although the graphical representation of the classification scheme supports hierarchical relationships, the current scheme does not contain any. Instead, the classification scheme is flat in which all classes apply for all objects of study.

4.1 Interaction Level

4.1.1 Description

Interaction level describes whether the *exposed* functionality of a honeypot is limited in some way, which is usually the case for honeypots that simulate services. Low and high are the two possible values for interaction level:

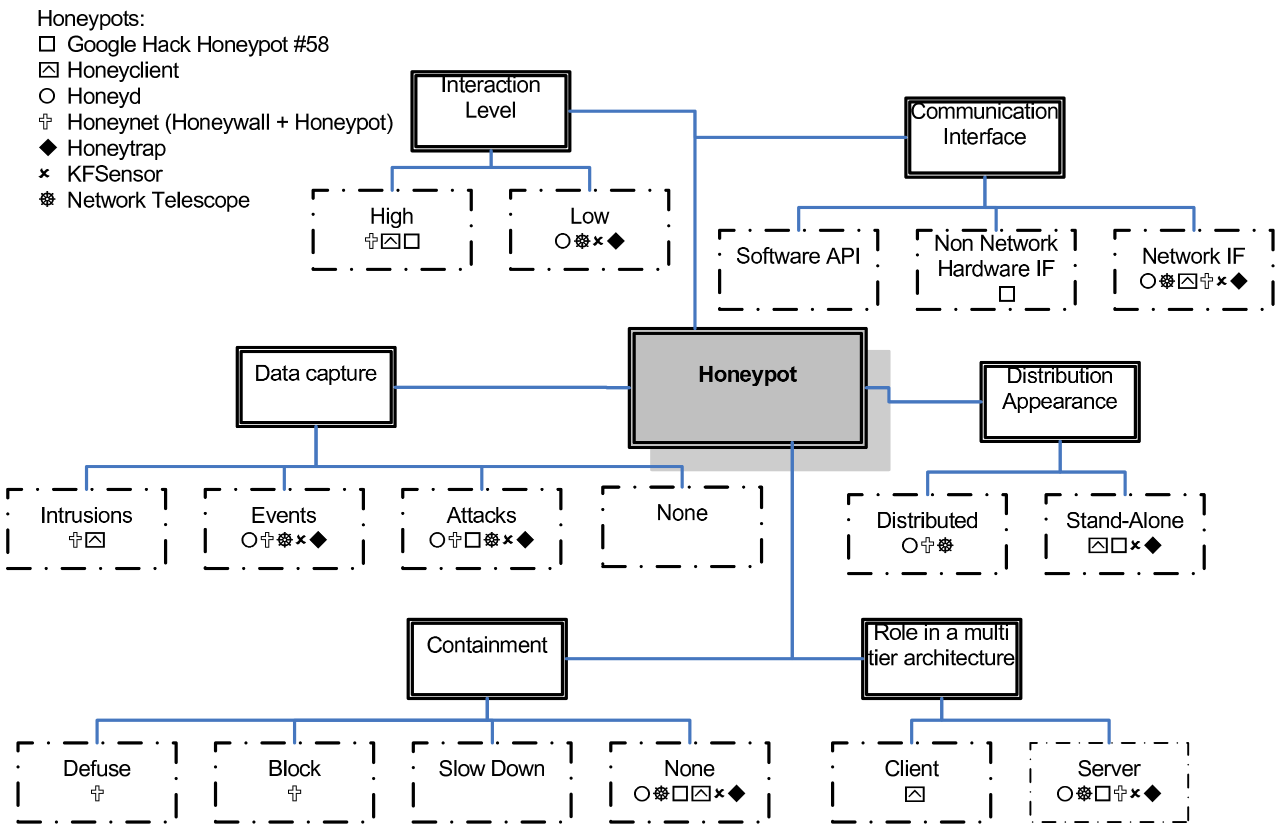

Figure 1: Honeypot Taxonomy

- High - High interaction level describes that exposed functionality of the honeypot is not limited.
- Low - Low interaction level corresponds to exposed functionality being limited. For example, a simulated SSH server of a honeypot is not able to authenticate against a valid login/password combination and allow for further interaction after successful authentication.

Some honeypots artificially limit their functionality in order to prevent malicious activity from spreading to other systems (see containment in section 4.3). Such limitations are not taken into account for describing the interaction level of a honeypot. Even though a honeypot might limit the functionality as part of its containment strategy, if no other functional limitations exist, the honeypot would still be classified with a high interaction level.

4.1.2 Classification Method

To classify the honeypot's interaction level, the taxonomist needs to demonstrate that the functionality that is available to the user of the system is limited. This is best done by comparing the exposed functionality of the honeypot to a corresponding non-honeypot system. Different functional behavior between these systems would sufficiently demonstrate functional limitation of the honeypot. This would result in a classification of a low interaction level. If the taxonomist were not able to demonstrate functional limitations of the honeypot, the honeypot would be classified with a high interaction level.

4.2 Data Capture

4.2.1 Description

Data capture describes the type of data a honeypot is able to capture. This class is not concerned with the technical aspects of the data that is being collected (tcpdump, log file, etc), but rather the type of data from an attack point of view. We borrow the possible values from the terminology used in MAFTIA taxonomy of intrusion detection systems and attacks [9]. The values for this class are events, attacks, intrusions, and none:

- Events - The honeypot collects data about something that has happened or took place; a change in state
- Attacks - The honeypot collects malicious activity threatening the security policy; a malicious external fault.
- Intrusions - The honeypot collects activity threatening the security policy that leads to a security failure i.e. a security policy violation; a malicious external fault that leads to failure.
- None - The honeypot does not collect events, attacks, or intrusions.

Whether the honeypot correctly tags the collected data along these lines is irrelevant for classification. Correctly labeling the collected data is left to the data analysis component of a honeypot, which is not part of the presented classification scheme. Since honeypots are able to collect several types of data, honeypot assignment can occur to multiple values of this class.

4.2.2 Classification Method

To determine the type of data a honeypot captures, the taxonomist needs to expose the honeypot to each type of data: an event, an attack, and an intrusion. Demonstration of collection of the data type by the honeypot or some supporting system belonging to the honeypot (e.g. a honeywall) would cause a classification of the corresponding data type. In case, collection of an event, attack, or intrusion cannot be demonstrated, the classification of data capture will be none. All three data types need to be examined for a complete classification.

4.3 Containment

4.3.1 Description

Attacks might spread from a honeypot to other systems. Containment classifies the measures a honeypot takes to defend against malicious activity spreading from itself. Possible values for this class are block, defuse, slow down, and none:

- Block - Attacker's actions are identified and blocked. The attack never reaches the target.
- Defuse - Attacker's actions are permitted, but are defused. The attack reaches the target, but is manipulated in a way that it fails against the target.
- Slow Down - Attacker is slowed down in his actions of spreading malicious activity.
- None - No action is taken to limit the intruder's spread of malicious activity against other systems.

Honeypots that do deploy defense strategies are likely to utilize multiple ones and, as a result, a honeypot can be assigned to multiple values of this class.

4.3.2 Classification Method

To classify the honeypot's containment strategy, the taxonomist needs to demonstrate which of the containment actions is applied by the honeypot. This is best done by attacking a target from the honeypot and from a corresponding non-honeypot and comparing the outcome of the attacks. Since some honeypots initiate the containment after some threshold is reached, the taxonomist needs to take into account the potential postponement of containment in his observations. In case statistically significant delays are detected on the outcome of attack, the honeypot is classified with a containment of slow down. In case the attack arrives at the target and is successful when originating from the non-honeypot, but is unsuccessful when originating from the honeypot, the classification of defuse containment is made. In case the attack arrives at the target when originating from the non-honeypot, but does not arrive when originating from the honeypot, the honeypot's containment is classified as block. If none of this behavior can be observed, none is the value selected for this class. For a complete classification, the taxonomist needs to try different attacks that solicit different containment responses from the honeypot.

4.4 Distribution Appearance

4.4.1 Description

Distribution appearance class describes whether the honeypot system appears to be confined to one system or multiple systems. We state "appearance" as some aspects of distribution might be simulated, such as one server behaving like several servers. Distributed and stand-alone are the two values for this class.

- Distributed - The honeypot is or appears to be composed of multiple systems.
- Stand-Alone - The honeypot is or appears to be one system. Note that if a honeypot is composed of multiple actual systems, but appears to be one system, it is classified as stand-alone.

Since the possible class values of distributed and stand-alone are binary and exclusive, only one value can be selected for this class.

4.4.2 Classification Method

To classify a honeypot's distribution appearance, the taxonomist has to show that one can interact with two or more distinct honeypots. (The term "interaction" does mean that the honeypot needs to respond with a reply once contacted. Recording of data on the honeypot would be sufficient to qualify as "interaction".) In case interaction is only possible with one system, the honeypot is classified as stand-alone, whereas in case interaction is possible with two or more systems, the honeypot is classified as distributed.

4.5 Communication Interface

4.5.1 Description

The communication interface describes the interfaces one can use to interact *directly* with the honeypot. Possible values are network interface, non-network hardware interface, or software API:

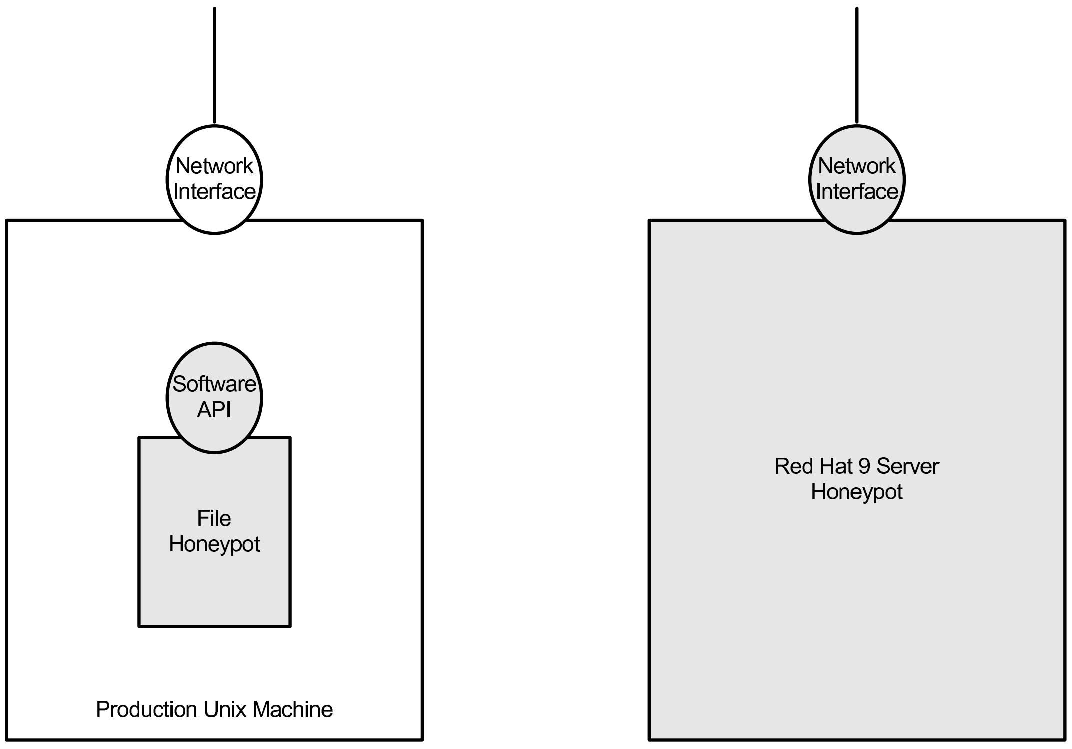

Figure 2: Honeypot Communication Interface

- Network Interface - The honeypot can be directly communicated with via a network interface.
- Non-Network Hardware Interface - The honeypot interfaces via hardware interfaces other than the network interface. Examples are printer port, CDROM drives, and USB connections.
- Software API - The honeypot can be interacted with via a software API.

It is important to note that this class does not classify the communication interface of the system the honeypot is running on, but rather the communication interface of the honeypot. Assume that a file placed on a legitimate machine is the honeypot. While the attacker might access this file via the network (via SSH), the communication interface in this instance is not the network, but rather software API. The network interface is the interface to the system the honeypot is running on, but the software API of the OS is actually the interface that allows for interaction with the honeypot. The communication interface of a high interaction honeypot, such as a RedHat9 server as part of a honeynet, which is accessible via the network, is network interface, because the receiving end of the network interface is the honeypot itself. Figure 2 illustrates these two examples. Data collection on the interface by the honeypot is a good indication that the hardware interface allows for direct interaction with the honeypot and that one is not interacting with an upstream system. A honeypot might allow for communication via various interfaces and, as a result, a honeypot can be assigned multiple values of this class.

4.5.2 Classification Method

To classify a honeypot's communication interface, the taxonomist needs to evaluate whether interaction of the honeypot is possible via each of the existing hardware interfaces. The honeypot is classified with a network communication interface once it has been demonstrated that a communication with the honeypot system is possible via a network interface and that data about the interaction is collected on the honeypot system. The same method applies to evaluate non-network hardware interfaces, such as a physical terminal. If the interface of a physical terminal interacts directly with the honeypot system and data about the interaction via this interface is collected by the honeypot, the physical terminal becomes an interface point to the honeypot. If hardware interfaces are not the means to directly access the honeypot, the honeypot is classified with a communication interface of software API.

4.6 Role in Multi-tier Architecture

4.6.1 Description

This class describes in what role the honeypot acts within a multi-tier architecture. The two possible values are server or client:

- Server - The honeypot is acting as a server. It is passively awaiting requests from clients.
- Client - The honeypot is acting as a client. It is actively initiating requests to servers.

The honeypot fills one role and as such can either be classified to fill the role of the client or the server.

4.6.2 Classification Method

To classify the role of the honeypot in a multi-tier architecture, the taxonomist needs to examine whether the honeypot accepts or initiates requests via its interfaces. Once the taxonomist has demonstrated that the honeypot acts upon requests via its interface (with a reply or some other action, such as data collection), the honeypot is classified with a server role. If the taxonomist cannot show such behavior of the honeypot, the honeypot is classified with a client role.

5 Classification of Existing Honeypots

In this section, we classify seven honeypot systems according to the presented classification scheme. We believe that classification of specific honeypots according to this method demonstrates that the taxonomy's classification scheme fulfills the characteristics of determinism, objectivity, repeatability, exhaustiveness, and specificity. Each step involved in classification is described in this section, so other taxonomists are able to follow our honeypot diagnosis and create identical classifications using the same procedure.

The honeypot systems we classify are Google Hack Honeypot No. 58 V1.1 [7], Honeyclient V0.1.1 [19], Honeyd, V1.5a [15], Honeynet (RedHat 9 with Honeywall Roo 1.0.189) [6], Honeytrap V0.6.1 [20], KFSensor V4.2 [2], and a network telescope. Those systems have been selected as they represent a broad spectrum of different honeypot technology. We anticipate that the classifications of these seven honeypots will differ and may provide us with insight in functional concentration or gaps within honeypot technology, which we review in section 5.8.

Since classification varies depending on the configuration of a honeypot, we include a description of the relevant configuration with a description of each honeypot. We configured the honeypots in such a way that they illustrate the most possible values of a class. We expect that this approach more clearly provides insight in the functional landscape of honeypots that results from the classification of all honeypots.

5.1 Google Hack Honeypot

The Google Hack Honeypot is an application honeypot that runs within a web server and detects information leakages resulting from exploitation of the Google search engine. Once information is accessed, it indicates that an attacker used malicious Google searches to find sensitive information on your site. Once the attacker accesses the sensitive information, the Google Hack Honeypot records information about time, file that was accessed, and source IP in a file.

We deployed Google Hack Honeypot No. 58, which simulates a password file, on top of an existing web server with legitimate content. The password file was linked from the main website via a hidden link. As a result, the password file could not easily be reached via the main website, but could be indexed by Google search engine. Once the attacker uses the Google search engine to find sensitive information, the attacker is directed to the Google Hack Honeypot password file indicating that the attacker has obtained sensitive information from our site.

5.1.1 Classification

- Interaction Level - We failed to devise a test case that demonstrated any functional limitations of the Google Hack Honeypot. While the Google Hack Honeypot simulates functionality (displaying a text file), it can not be distinguished from a real system due to its inherently limited functionality. We therefore classified Google Hack Honeypot's interaction level as *high*.

- Data Capture - We were not able to devise a test case that recorded data of a event on the Google Hack Honeypot. However, we were able to successfully demonstrate data capture of attacks by accessing the exposed file (passlist.txt). This attack, access to the exposed file, is recorded by the Google Hack Honeypot in a log. This results in a classification of *attacks*. We could not create a test case that demonstrates capturing data of a successful intrusion or malicious activity occurring after a successful intrusion, as accessing a bogus file does not represent an intrusion in the context of the honeypot system.

- Containment - We were not able to compromise the Google Hack Honeypot from which an attack could be launched against other systems due to the limited functional characteristics of this honeypot. As a result, we classified the honeypot with a containment value of *none*.

- Distribution Appearance - The Google Hack Honeypot was placed on a single web server. We were unable to show interaction with multiple systems. As such, the honeypot was classified with a *stand-alone* distribution appearance.

- Communication Interface - Attacking the Google Hack Honeypot occurs via the software API interface that is provided by the web server, because the network interface only provides access to the web server, but not to the actual Google Hack Honeypot. The fact that the Google Hack Honeypot does not record access to all files served by the web server is an indication of this circumstance. The network interface provides access to the web server, but the specific web server API provides access to the Google Hack Honeypot. No other hardware interface allows for communication with the Google Hack Honeypot. As a result, the honeypot was classified with a *software API* communication interface.

- Role in multi-tier architecture - We accessed the exposed file of the Google Hack Honeypot. The web server requests this file from the Google Hack Honeypot and serves it to the requesting client. The web server, in this instance, acts as a proxy between the client and the Google Hack Honeypot. We therefore classified the Google Hack Honeypot's role in a multi-tier architecture as *server*.

5.2 Honeyclient

Honeyclient is a proactive honeypot, which acts as an Internet Explorer client in search of malicious servers. Honeyclient makes HTTP requests to web servers based on a list of provided URLs. After the response is received, Honeyclient evaluates the integrity of the registry and a provided file list. Any modification to the registry or files would indicate a compromise. Honeyclient logs such behavior with information of the URL that caused it.

We installed Honeyclient on an unpatched version of Microsoft Windows XP running within VMware. We configured Honeyclient to monitor the file directory C:\ for additions, deletions, and modifications of files. We excluded files from monitoring that were created as part of the Honeyclient process (e.g. C:\cleanfile.dat).

5.2.1 Classification

- Interaction Level - We failed to devise a test case that would demonstrate any functional limitations of Honeyclient. We therefore classified Honeyclient's interaction level as *high*.

- Data Capture - We instructed Honeyclient to navigate to URL `http://www.heise.de/security/dienste/browsercheck/demos/ie/null/demos/mhtml.html`, which contained a non-malicious exploit demonstration of the mhtml code vulnerability (MS04-013). Honeyclient's log file indicated that a new file C:\browsercheck.exe was created. As such, Honeyclient successfully detected and recorded the *intrusions*, which was our classification for data capture. We failed to construct a test case that would demonstrate that Honeyclient could detect attacks or events.

- Containment - We were unable to identify a containment strategy of Honeyclient. Examination of the Honeyclient code showed no provisions that would limit the spread of malicious activity. As a result, the Honeyclient was classified with a containment value of *none*.

- Distribution Appearance - During its operation, Honeyclient actively initiated connections to servers. The source of those connections was always the Honeyclient IP address. As such, the honeypot was classified with a *stand-alone* distribution appearance.

- Communication Interface - During its operation, Honeyclient actively initiated connections to servers via the network interface. No communication was observed via the other hardware interfaces that existed on the system. As a result, *network interface* was the only classification for communication interface.

- Role in multi-tier architecture - We were unable to demonstrate that Honeyclient accepts requests, because Honeyclient does not expose any services, but rather is designed to initiate connections to remote services. As a result, the honeypot was classified as a *client* role in a multi-tier architecture.

5.3 Honeyd

Honeyd Virtual Honeypot (Honeyd) is a simple software program that is able to create multiple virtual honeypots on a network. Each virtual honeypot can be configured to expose several simulated network services, such as a Telnet, FTP, HTTP, or SMTP. Honeyd captures network traffic as well as any data recorded by the simulated services (e.g. ftp login name/password) in log files.

We configured Honeyd to simulate two Linux servers that hosted a simulated Telnet server. The two servers were configured with IP address 192.168.75.200 and 192.168.75.201. Honeyd was started with option -l, which caused all network traffic flowing to and from the virtual honeypots to be captured and recorded in a flow log.

5.3.1 Classification

- Interaction Level - We devised a test case to log in with a valid username/password combination. This test case was executed against a non-honeypot Linux system and against Honeyd. On the non-honeypot system, it led to a successful authentication, whereas on the honeypot, it resulted in a repeated display of the login screen. The test case demonstrated that the exposed functionality of Honeyd is limited. We therefore classified the Honeyd interaction level as *low*.

- Data Capture - We probed Honeyd with a SYN request on port 23 (Telnet), which resulted in a SYN ACK reply. Honeyd recorded the SYN request on port 23 in a flow log. This resulted in a classification of *events*. Further, we attempted to guess usernames and passwords against the Honeyd Telnet server. The honeypot recorded these attacks, resulting in a classification of *attacks*. We attempted to use a valid username/password combination, which did not result in a successful intrusion. We could not devise a test case that would result in a successful intrusion. As a result, the honeypot was not classified with intrusions data capture.

- Containment - We were not able to compromise the Honeyd honeypot from which an attack could be launched against other systems. This was due to the limited functional characteristics of this low interaction honeypot. As a result, the honeypot was classified with containment *none*.

- Distribution Appearance - We probed the honeypot with a SYN request on port 23 (Telnet) on IP address 192.168.75.200 and 192.168.75.201, which resulted in a SYN ACK reply for each probe. This demonstrated the ability to interact with two distinct systems resulting in a classification of *distributed* distribution appearance.

- Communication Interface - We probed the honeypot with a SYN request on port 23 (Telnet) via the network interface connected to the honeypot. The data of this probe was recorded by the honeypot. This resulted in a classification of *network interface* for this class. No other hardware interfaces existed on Honeyd that allowed for interaction with the honeypot. As a result, *network interface* was the sole classification.

- Role in multi-tier architecture - We probed the honeypot with a SYN request on port 23 (Telnet), which resulted in a SYN ACK reply for each probe, indicating acceptance of a connection request. As a result, the honeypot was classified as a *server* role in a multi-tier architecture.

5.4 Honeynet

The Honeynet is a network of real systems. This network is reachable via the Honeywall gateway, a stealth inline network bridge that closely monitors and controls the network data flow to and from the honeypots in the network. Data capture includes network traffic captured on the honeywall gateway, system event data captured in logs, and keylog data gathered by a stealth keylogger on the honeypot systems.

We configured the Honeynet with the Roo 1.x Honeywall and two standard RedHat 9 Linux servers (standard server installation with MYSQL database). The two servers were reachable via IP address 192.168.75.200 and 192.168.75.201. The Honeywall was configured to restrict TCP connections from the honeypot to 20 connections per hour. In addition, the Snort Inline module was enabled to defuse malicious outbound traffic before leaving the Honeynet. Hardware interfaces were disabled to disallow interaction of the honeypot via these interfaces as they were not monitored by the Roo 1.x Honeywall.

5.4.1 Classification

- Interaction Level - We failed to devise a test case that demonstrated any functional limitations of the Honeynet. We therefore classified the Honeynet interaction level as *high*.

- Data Capture - We probed the Honeynet with an ICMP echo request, which resulted in an ICMP echo reply. The honeynet recorded the ICMP echo request in a log. This resulted in a classification of *events*. Further, we attempted to guess usernames and passwords against the Honeynet's SSH server. The Honeynet recorded these attacks resulting in a classification of *attacks*. We guessed a valid username/password combination, which resulted in a successful intrusion. The Honeynet recorded the successful intrusion in a log. As a result, the third classification was *intrusion*.

- Containment - We demonstrated that the Honeynet followed a containment strategy of block and defuse. We executed 21 HTTP requests to a single web site on both the Honeynet and a non-honeypot system. On the Honeynet, 21 HTTP requests resulted in 20 responses whereas the non-honeypot received 21 responses. This indicated that the Honeynet executed a block containment strategy and resulted in a classification of *block*. Further, we attempted a root login to a MYSQL database target. From the Honeynet, this attack was received by the MYSQL server, but failed whereas on a non-honeypot, the attack succeeded using the same login parameters. The behavior demonstrated the containment strategy to defuse outgoing malicious activity by the Honeynet and resulted in a classification of *defuse*. We were not able to demonstrate that the Honeynet applied a *slow down* containment strategy. All attacks executed from the Honeynet and non-honeypot executed in approximately the same time.

- Distribution Appearance - We probed the Honeynet with HTTP request on IP address 192.168.75.200 and 192.168.75.201, which resulted in a HTTP reply for each probe. As a result, the honeypot was classified with a *distributed* distribution appearance.

- Communication Interface - We probed the Honeynet with HTTP request via the network interface connected to the honeynet. The data of this probe was recorded by the Honeynet. This resulted in a classification of *network interface*. No other hardware interfaces exist on honeynet that allow for interaction with the Honeynet. As a result, *network interface* was the sole classification.

- Role in multi-tier architecture - We executed a HTTP request, which resulted in a HTTP reply for each request by the Honeynet. As a result, the honeypot was classified as a *server* role in a multi-tier architecture.

5.5 Honeytrap

Honeytrap is a program that allows to observe known and unknown TCP network based attacks. It does not simulate pre-defined network services, but rather dynamically creates these services as requests via the network on unserviced ports are encountered. Once a service has been created by Honeytrap, it records all data being sent to the service. It optionally also can send a predefined response on a specified port.

We installed Honeytrap on a Linux machine with a single network interface on which no network services were configured. Honeytrap, as a result, would respond to any TCP requests bound towards this Linux machine and record the

5.5.1 Classification

- Interaction Level - We devised a test case to telnet to the system. This test case was executed against a non-honeypot Linux system and against the Honeytrap honeypot. On the non-honeypot system, it led to correctly prompting for the login, whereas on Honeytrap it resulted in several empty lines being displayed. The test case demonstrated that the exposed functionality of Honeytrap is limited in regards to standard functionality of a telnet server. We therefore classified the Honeytrap honeypot interaction level as *low.*

- Data Capture - We probes the Honeytrap with two SYN requests on port 22 (SSH), which resulted in one RST and one SYN/ACK reply. (Two packets are necessary because Honeytrap seems to fork a server request after it encounters a RST packet that indicates that no service is listening on that port). This resulted in a classification of *events.* Further, we used the Metasploit [4] framework to execute the "squid_ntml_authenticate" exploit with a simple shell payload against the Honeytrap honeypot. The honeypot recorded these attacks resulting in a classification of *attacks.* We could not devise a test case that would result in a successful intrusion. As a result, the honeypot was not classified with intrusions data capture.

- Containment - We were not able to compromise the Honeytrap honeypot which disallowed attacks to be launched from this honeypot. This was due to the limited functional characteristics of this low interaction honeypot. As a result, the honeypot was classified with containment *none.*

- Distribution Appearance - Honeytrap was installed on a single Linux machine. We were unable to show interaction with multiple Honeytrap honeypot systems. As such, the honeypot was classified with a *stand-alone* distribution appearance.

- Communication Interface - We requested a web page from the honeypot via the network interface connected to the honeypot using a browser. A web page was returned by the honeypot and the initial request was recorded in Honeytrap's attack log. This resulted in a classification of *network interface* for this class. No other hardware interfaces existed on Honeytrap that allowed for interaction with the honeypot. As a result, *network interface* was the sole classification.

- Role in multi-tier architecture - We requested a web page from the Honeytrap honeypot. Honeytrap returned a web page to the requesting browser demonstrating acceptance of a connection request. As a result, the honeypot was classified as a *server* role in a multi-tier architecture.

5.6 KFSensor

KFSensor is a commercial honeypot that is able to expose various legitimate network services, such as FTP and telnet servers, and illegitimate network services, such as backdoors, on a Windows system. KFSensor can be configured to simulate servers and record data that might result from interaction with the attacker via these exposed services (for example, username/password during a telnet session), but also can be configured to provide various levels of network responses on those network services (such as replying to a SYN request.)

We installed KFSensor on a Windows XP machine with one network card. We configured KFSensor to run Sim S on port 23 using a Linux Telnet server.

5.6.1 Classification

- Interaction Level - We devised a test case to telnet to the system by passing the username as part of a parameter on the command line. This test case was executed against a non-honeypot Linux system and against KFSensor. On the non-honeypot system, it led to correctly prompting for the password, whereas on KFSensor it resulted in a incorrect display of the login screen. The test case demonstrated that the exposed functionality of KFSensor is limited in regards to standard functionality of a telnet server. We therefore classified the KFSensor interaction level as *low*.

- Data Capture - We sent a ICMP echo request (ping) to the KFSensor honeypot, which caused a data event to be recorded by KFSensor informing us about the ICMP echo request and its source address. This resulted in a classification of *events*. Further, we attempted to guess usernames and passwords against the KFSensor Telnet server. The honeypot recorded these attacks, resulting in a classification of *attacks*. We attempted to use a valid username/password combination, which did not result in a successful intrusion. We could not devise a test case that would result in a successful intrusion. As a result, the honeypot was not classified with intrusions data capture.

- Containment - We were not able to compromise the KFSensor honeypot from which an attack could be launched against other systems. This was due to the limited functional characteristics of this low interaction honeypot. As a result, the honeypot was classified with containment *none*.

- Distribution Appearance - KFSensor was installed on a single Windows XP machine. We were unable to show interaction with multiple KFSensor honeypot systems. As such, the honeypot was classified with a *stand-alone* distribution appearance.

- Communication Interface - We probed the honeypot with a SYN request on port 23 (Telnet) via the network interface connected to the honeypot. The data of this probe was recorded by the honeypot. This resulted in a classification of *network interface* for this class. No other hardware interfaces existed on KFSensor that allowed for interaction with the honeypot. As a result, *network interface* was the sole classification.

- Role in multi-tier architecture - We probed the honeypot with a SYN request on port 23 (Telnet), which resulted in a SYN ACK reply for each probe, indicating acceptance of a connection request. As a result, the honeypot was classified as a *server* role in a multi-tier architecture.

5.7 Network Telescope

A network telescope is a monitoring device that occupies unused IP address space on a network. It captures all network traffic that flows to any of its address space, but does not generate any responses to such traffic. Network telescopes are usually not thought of as honeypots, but if we apply the honeypot definition of a security device whose value lies in being probed, attacked, and compromised, a network telescope certainly qualifies for this definition as it captures primarily probe as well as some attack data.

We deployed a network telescope on a local network using arppoison and tcpdump. We configured the network telescope to redirect traffic from the network block 192.168.75.2-192.168.75.100 and 192.168.75.111-192.168.75.254 to itself and to record all traffic in a tcp-dump log file.

5.7.1 Classification

- Interaction Level - We devised a test case to access the machine via SSH. This test case was executed against a non-honeypot Linux system and against the network telescope. On the real system, it led to display of the authentication screen, whereas on the network telescope it resulted in timeout on the client side since a SYN ACK package was never received to complete the three-way TCP handshake. The test case demonstrated that the exposed functionality of the network telescope is limited. We therefore classified the network telescope interaction level as *low*.

- Data Capture - We probed the network telescope with an ICMP echo request, which did not result in an ICMP echo response. However, the network telescope did record the ICMP echo request in a log. This results in a classification of *events*. Further, we demonstrated that the network telescope is also able to capture information about attacks. We attacked the network telescope with a SQL Slammer (which is contained in a single UDP packet). While this test did not result in a compromise or even a response, the network telescope did record the attack. We therefore classified *attacks* for this class as well. However, we could not devise a test case that would result in a successful intrusion. As a result, the network telescope was not classified with *intrusions* data capture.

- Containment - We were not able to compromise the network telescope from which an attack could be launched against other systems due to the limited functional characteristics of this low interaction network telescope. As a result, we classified the network telescope with *none* containment.

- Distribution Appearance - We probed the network telescope with ICMP echo requests on IP address 192.168.75.111 and 192.168.75.121, which did not result in an ICMP echo reply. However, the network telescope did record both ICMP echo requests in a log. As such, we successfully demonstrated interaction with two distinct systems and therefore were able to classify the network telescop with a *distributed* distribution appearance.

- Communication Interface - We probed the network telescope with an ICMP echo request via the network interface connected to the network telescope. The data of this probe was recorded by the network telescope. This resulted in a classification of *network interface*. No other hardware interfaces existed on the network telescope that allowed for interaction by other means. As a result, *network interface* was the sole classification for communication interface.

Category / Name	Interaction Level	Data Capture	Containment	Distribution Appearance	Communication Interface	Role in Multi Tier Architecture
Google Hack Honeypot No58	High	Attacks	None	Stand-Alone	Software API	Server
Honeyclient	High	Intrusions	None	Stand-Alone	Network Interface	Client
Honeyd	Low	Events Attacks	None	Distributed	Network Interface	Server
Honeynet	High	Events Attacks Intrusions	Defuse Block	Distributed	Network Interface	Server
Honeytrap	Low	Events Attacks	None	Stand-Alone	Network Interface	Server
KFSensor	Low	Events Attacks	None	Stand-Alone	Network Interface	Server
Network Telescope	Low	Events Attacks	None	Distributed	Network Interface	Server

Figure 3: Honeypot Classifications

- Role in multi-tier architecture - We probed the network telescope with a SYN request on port 22 (SSH), which did not result in a SYN ACK reply. However, it did record the SYN request in a log. As a result, the network telescope was classified as a *server* role in a multi-tier architecture.

5.8 Summary and Results

Figure 1 and table 3 summarize the classifications of all seven honeypots. The seven honeypots, which differ conceptually, were classified differently by our taxonomy. Honeyd and Honeynet are two honeypot systems that are usually cited as being very unlike one another. According to our taxonomy, classification differs in three of the six classes. Honeyclient was clearly separated by class role in multi-tier architecture from the other honeypots. The Google Hack Honeypot, the only application honeypot, was classified differently from the other honeypots via class communication interface. The network telescope, Honeyd, Honeytrap and KFSensor do not differ in classification. This is not surprising, as they mostly differ in their ability to capture attack data at different depths due to the different simulation capabilities.

Certain classifications did not occur with the seven honeypots at hand. Slow down containment is one of such classifications. Containment class is dominated by none for six out of seven classified honeypots. As such, it is not surprising that not all possible classifications were made. However, what is surprising is the fact that even some high interaction honeypots (e.g. Honeyclient) do not provide built-in means for containment of malicious activity. The Honeyclient documentation recommends installation of a firewall and usage of VMware, but these are optional measures left to the discretion of the user. On low interaction honeypots, it seems as though the products are confident that the limited functionality would contain an attack. However, we believe that danger exists with respect to the spread of malicious activity even on these low interaction honeypots, and that containment should be followed for certain low interaction honeypots (e.g. Honeyd.)

The other classification that fails to show up in our results is the non-network hardware communication interface. A honeypot that evaluates freely distributed CDROMs exploiting the autorun features to install malicious code (e.g. the Sony Rootkit) could be a honeypot with such a communication interface classification. Such a system might not appear in our list due to our failure to include such a honeypot in our taxonomy. However, a search has not resulted in any honeypot products of this kind. This might be because there is no need for such a device or because it could be easily custom built.

The Honeyclient classification warrants some further examination. As one takes this classification and varies the values for certain classes, one could infer client honeypot technologies that do not exist today, but could be valuable in the search of malicious servers. In particular, we believe a low-interaction honeypot would be of value in such a quest. Instead of having a fully functional client with a full operating system interacting with potentially malicious servers, one could imagine a lightweight, simulated client crawling the web for malicious servers. The responses could be examined with client side intrusion detection signatures. While such a setup would increase speed, it potentially reduces the ability of a low interaction client honeypot to detect attacks. This might be an acceptable tradeoff depending on the circumstances. Further, a distributed client honeypot would be worth exploring. Distribution seems to be accepted practice to increase effectiveness of honeypot servers, and might also be applicable to client honeypots. We expect to see such systems appearing in the future.

6 Discussion

In this section, we present some of the issues we encountered during the development of this taxonomy and the subsequent classification of the seven honeypots as well as our reasoning the decisions we made around these issues. The first issue we encountered was to find a precise definition and classification method for low and high interaction level. It is understood that a honeypot with emulated services is classified as a low interaction honeypot and a honeypot with real services is classified as a high interaction honeypot. However, classification can become difficult as an emulated service can behave identical to a real service. We have decided that if differentiation is not possible, it is irrelevant whether the service is emulated as the interaction of the honeypot with the attacker is identical. As such, we would classify such a system as a high interaction honeypot despite it being composed of emulated services. Google Hack Honeypot No.58 is an example of such a classification as we were unable to discern the difference of a real password list from an emulated password list.

Configuration dependency was the second issue we encountered during the classification of the seven honeypots. Classification of software, in general, is very dependent on the configuration of the software as it tends to drive its functionality. A flexible system, such as KFSensor, can be configured in a way it can be classified in one way (for example with a native server, it could be classified as a high interaction honeypot) or the other (for example with simple banners, it would be classified as a low interaction honeypot). If we would account for all configurations of a honeypot, it would lead to a classification that would make it difficult to differentiate honeypots and draw conclusions, identify gaps, and generalizations about honeypots. For example, the KFSensor would be classified as a high interaction *and* low interaction honeypot. As such, we have decided to configure the honeypot in a particular typical way and perform the classification on this particular configuration. We document this configuration to enable taxonomist to repeat the presented classification of the honeypots. It also allows taxonomist to add additional classifications of identical honeypots with different configurations.

Values of classes that seem to supersede each other was the third issue we encountered. For example, the distribution appearance class value *distributed* seems to supersede the value of *stand-alone* as a distributed honeypot could function as a stand-alone honeypot. We have addressed this issue by definition of the typical honeypot configuration. The resulting classification is unambiguous.

7 Conclusion

In this paper, we have developed a taxonomy for honeypots. Our taxonomy adheres to the characteristics outlined by Lindquist et al and Krsul. We described a methodology for classification that can be followed by each taxonomist in an objective, repeatable, and deterministic way. We believe the descriptions of the classes and their values are comprehensible and conform to common honeypot terminology. We have classified seven honeypots with the taxonomy and gained insight into some of the functional gaps that are currently presented by the existing honeypot technologies. Our taxonomy, therefore, has widened our insight of honeypots, which should lead to further exciting research in the field.

8 Future Work

Thus far, we have only classified seven honeypots with the developed taxonomy. Future research will concentrate on expanding classifications to include additional honeypot technology. We expect to gain further insights into honeypot technology from this work. Further, we acknowledge that the current classification scheme is a flat structure and does not allow for distinguishable classifications of certain groups of honeypots. This becomes particularly apparent when examining the classification of the four low interaction honeypots KFSensor, Honeyd, Honeytrap and the network telescope, which were classified identically. While conceptually, these honeypots are similar, we believe further subclasses can be developed that allow to classify differentiating characteristics of these honeypots. Future work aims at collecting additional data to expand the taxonomy in such a way.

References

[1] Back Officer Friendly. Available from `http://www.nfr.com/resource/backOfficer.php`; accessed on 6 June 2006.

[2] KFSensor - Advanced Windows Honeypot Server, Version 4.2. Available from `http://www.keyfocus.net/kfsensor/`; accessed on 10 July 2006.

[3] NetFacade. Available from `http://www22.verizon.com/fns/solutions/netsec/netsec_netfacade.html`; accessed on 2 June 2006.

[4] The Metasploit Framework, Version 2.6. Available from `http://www.metasploit.org`; accessed on 13 July 2006.

[5] CyberCop Sting, 1999.

[6] Honeywall CDROM Eyeore, 2003. Available from `http://project.honeynet.org/tools/cdrom/eeyore/download.html`; accessed 6 July 2006.

[7] The Google Hack Honeypot, 2005. Available from `http://ghh.sourceforge.net/`; accessed on 5 May 2006.

[8] Axelsson, S. Intrusion Detection Systems: A Survey and Taxonomy. Technical Report 99-15, Chalmers University, 2000.

[9] Cachin, C., Dacier, M., Daeak, O., Jublisch, K., Randell, B., Riordan, J., Tscharner, A., Wespi, A., and Wuest, C. Towards a Taxonomy of Intrusion Detection Systems and Attacks, 2001.

[10] Cheswick, B. An Evening with Berferd in which a cracker is Lured, Endured, and Studied. In *Winter 1992 USENIX Conference* (San Francisco, 1992), USENIX, pp. 163–174.

[11] Cohen, F. Deception Toolkit. Available from `http://all.net/dtk/dtk.html`; accessed on 6 July 2006.

[12] Debar, H., Dacier, M., and Wespi, A. Towards a Taxonomy of Intrusion-Detection Systems. *Computer Networks 31* (1999), 805–822.

[13] Krsul, I. *Software Vulnerability Analysis.* PhD thesis, Purdue University, 1998.

[14] Lindqvist, U., and Jonsson, E. How to Systematically Classify Computer Security Intrusions. In *IEEE Symposium on Security and Privacy* (Oakland, 1997), IEEE, pp. 154–163.

[15] Provos, N. Honeyd Virtual Honeypot. Available from `http://www.honeyd.org/`; accessed on 6 July 2006.

[16] Spitzner, L. *Honeypots: Tracking Hackers.* Addison-Wesley, Boston, 2002.

[17] Stoll, C. Stalking the Wily Hacker. *Communications of the ACM 31*, 5 (1988), 484–497.

[18] Undercoffer, J., Joshi, A., Finin, T., and Pinkston, J. A Target-Centric Ontology for Intrusion Detection. In *18th International Joint Conference on Artificial Intelligence* (Acapulco, 2004), Morgan Kaufmann Pub.

[19] Wang, K. Honeyclient, Version 0.1.1. Available from `http://www.honeyclient.org/`; accessed on 6 July 2006.

[20] Werner, T. Honeytrap, Version 0.6.1, 2006. Available from `http://honeytrap.sourceforge.net`; accessed on 1 July 2006.

[21] Zhang, F., Zhou, S., Qin, Z., and Liu, J. Honeypot: a Supplemented Active Defense System for Network Security. In *4th International Conference on Parallel and Distributed Computing Applications and Technologies* (Chengdu, 2003), IEEE, pp. 231–235.