Use of Deception to Improve Client Honeypot Detection of Drive-by-Download Attacks

Barbara Endicott-Popovsky[1], Julia Narvaez[1], Christian Seifert[2], Deborah A. Frincke[3], Lori Ross O'Neil[3], and Chiraag Aval[1]

[1] University of Washington
4311 – 11th Avenue NE, Suite 400, Seattle, Washington 98105, USA
{endicott,jnarvaez,chiraaga}@u.washington.edu
[2] Victoria University of Wellington
School of Engineering and Computer Science, Victoria University
PO Box 600, Wellington 6140, New Zealand
christian.seifert@gmail.com
[3] Pacific Northwest National Laboratory
902 Battelle Boulevard, Richland, WA, USA
{deborah.frincke,lro}@pnl.gov

Abstract. This paper presents the application of deception theory to improve the success of client honeypots at detecting malicious web page attacks from infected servers programmed by online criminals to launch drive-by-download attacks. The design of honeypots faces three main challenges: deception, how to design honeypots that seem real systems; counter-deception, techniques used to identify honeypots and hence defeating their deceiving nature; and counter counter-deception, how to design honeypots that deceive attackers. The authors propose the application of a deception model known as the deception planning loop to identify the current status on honeypot research, development and deployment. The analysis leads to a proposal to formulate a landscape of the honeypot research and planning of steps ahead.

Keywords: deception, counter-deception, honeypots, drive-by-downloads, cyber-attacks.

1 Introduction

With increasing reliance on computer networks, important expected security concepts—confidentiality, integrity and availability—are under constant threat: 1) personal information, such as names/credit card numbers, is stolen; 2) office desktop computers are compromised into sending e-mail spam; and 3) risk of power grid outages caused by denial-of-service attacks on SCADA systems [1] might escalate.

A particularly insidious type of online attack has emerged in recent years, which targets clients through malicious servers that deliver an attack as part of the server's response to a client request. As the web browser requests content from a web server, the server returns a malicious page that launches a so-called drive-by-download attack on the browser. If successful, the web server pushes and executes arbitrary programs on the client machine.

Security devices called high-interaction client honeypots are able to find these malicious web pages by driving a client to visit web pages and make an assessment as to whether the page launches an attack. However, if the malicious server can first identify the client as a honeypot, it could choose not to launch attack code, rendering the client honeypot ineffective. Attacker counteracts are exemplified by articles on honeypot detection, in which several ways to fingerprint honeypots are introduced [2].

These researchers have concluded that the use of detection techniques in drive-by attacks necessitates the inclusion of deception techniques in client honeypots. With an understanding of the anti-detection techniques used by malicious servers, this paper proposes deception methodologies designed to develop client honeypots that elude detection. As the adversary improves in sophistication, so do the defenders.

2 Background

"A honeypot is a security resource whose value lies in being probed, attacked, or compromised". Even though the notions of honeypots were originated in the early 1990's, only recently commercial products have been developed and papers have been published [3]. The concepts of honeypots were formulated in 1990/1991 with the work of Clifford Stoll's ”The Cuckoo's Egg" and Bill Cheswick's "An Evening With Berferd" [4]. The use of honeypots and decoys as a deception in the defense of information systems was related by Cheswick, Bellovin, D'Angelo and Glick, in 1991 [5] in the paper "An Evening with Berferd In Which a Cracker is Lured, Endured, and Studied." The paper is a chronicle of how the researchers offered a “bite” to a cracker, the traps used to lure and detect him, and the chroot “Jail” the researchers built to watch his activities [6].

Types of honeypots can be differentiated by their ability to interact with an attacker. Systems that emulate vulnerabilities and allow limited interaction with the attacker are low-interaction honeypots. Systems that are vulnerable and allow interaction with the attacker at all levels are high-interaction honeypots [7]. Another differentiation is between physical honeypots, which run on physical machines, and virtual honeypots, which run on virtual machines [2].

As a result of attackers exploiting vulnerabilities in client programs (such as browsers), honeypots have evolved to simulate the behavior of a human and analyze how such behavior is exploited by an attacker [2].

2.1 Client Honeypots

A client honeypot consists of three components: the queuer, visitor, and analysis engine (Fig. 1 illustrates components). This client is controlled by a visitor component which interacts with potentially malicious web servers. Information about what server to interact with and the data to be sent to the server is created by a queuer component, for example a web crawler, that generates server requests. Lastly, the analysis engine assesses whether the server is malicious or benign.

The visitor component maps to high- and low-interaction client honeypots. The former allows the honeypot system full functional interaction. As the client interacts with the server, the system monitors for unauthorized state changes, such as file

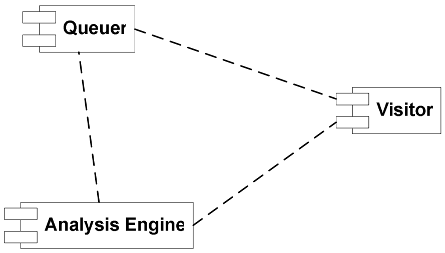

Fig. 1. Client Honeypot Component Diagram

modifications or process adjustments that would indicate a successful attack [8,9]. The latter signifies that the functionality of the client is limited, typically by using emulated services. Because no active exploitation occurs, the low-interaction client honeypot inspects the response directly using signatures, heuristics, and security predicates to detect attacks [9, 10, 11].

Given that honeypots are deceptive by nature, there is a wealth of wisdom to be gained from the study of deception theory in other sciences, such as social science.

2.2 Deception

The Longman Dictionary of American English defines Deception as "An act of deceiving." Deceiving is defined as "To cause someone to accept as true or good what is false or bad [13]." Multiple studies and theories of deception have been proposed. Cohen states that "Deception exploits errors in cognitive systems for advantage. It is achieved by systematically inducing and suppressing signals entering the target cognitive system [5]."

Bell and Whaley studied the general theory of deception and types of deception [14]. They argue that there are two levels of basic deceptive methods found in nature: hiding and showing. Humans consciously use these two methods found in nature.

Hiding, level one, is divided into three parts: masking, repacking and dazzling. Masking: the real thing is hidden by blending with the background, integrating itself with the surroundings, or seeking invisibility. Repacking: the real thing is perceived in various ways, as dangerous, harmless or irrelevant. Dazzling: ultimate problem of what to do when masking and repacking do not work and the attacker knows the victim is there. The qualities of the object might be changed as to confound [14].

Showing, level two, is divided into three parts: mimicking, inventing and decoying. Mimicking: a replica of reality is created by selecting one or more characteristics of the real in order to achieve an advantageous effect. Inventing: the false is presented through the creation of an alternative reality, e.g. the false document appears to be real, but it is not. Decoying: gives an additional alternative pattern, increasing its certainty [14]. The work performed by honeypots fits within these levels and categories of deception.

3 Problem: Detecting Honeypots

The design of honeypots faces three main challenges: deception, counter-deception and counter-counter-deception [15]. a) Deception problem: how to design honeypots that look like normal computer systems. b) Counter-deception problem: techniques used to identify if a computer is a honeypot. Objectives of counter-deception include the appraisal of whether an attacker can detect a honeypot, and the identification of whether the data collected from such a honeypot are misinformation. c) Counter-counter-deception: how to design honeypots that make attackers think that they are real systems [15].

4 Analysis

Honeypots are used to research and to prevent, detect, and respond to attacks. For research purposes, honeypots collect information on threats, which can be used for trend analysis, identification of new tools or methods, and attacker identification [16]. In this section, the authors focus the analysis on the research purpose of honeypots.

4.1 Deception

Before launching an attack, adversaries collect information about the host operating system and services running. Learning about the operating system allows attackers to understand what vulnerabilities the host might have. Learning about the services and versions facilitates planning of a route of attack [2]. Researchers value the knowledge of how the adversary breaks into a target machine and honeypots enable them to do that. The type of honeypot used varies according to the intended victim of attacks, which can be targeted attacks or targets of opportunity.

Targeted attacks are directed to targets of choice, which are organizations with high value information resources. For these targets of choice, production honeypot file servers could be used to provide falsified information to a human attacker who analyzes information given out by the honeypot [4]. Creating fake file systems is a form of mimicking and inventing [15]. Spitzner proposes the use of honeytokens, which are digital information entities, not computers. Any interaction with them is an unauthorized interaction. This form of honeypot is also useful to detect, identify and gather information about the malicious insider threat [17].

Targets of opportunity attacks can use multiple deception techniques, e.g. honeypot farms, in which honeypots are services. All the traffic coming to the production server is re-routed to pass through honeypots that are locally or remotely located. The honeypots need to emulate the production systems. In the event of detecting malicious activity, this can be logged, trapped, and traced back [18]. Roaming honeypots are mechanisms that allow the locations of honeypots to be unpredictable, continuously changing, and disguised within a server pool, from which a subset of servers provides services and the rest of the server pool is idle and acts as honeypots [19].

Client honeypots simulate the behavior of a human and actively search for attacks and malicious content on the Internet [2]. The level of interaction between client honeypots with servers can be low or high. Low-interaction client honeypots use a simulated client in place of a browser and assess the malicious nature of a server via static analysis such as signatures. High-interaction client honeypots interact with servers and assess the malicious nature of the server based on state changes [7].

Significant development of client honeypots is expected for web clients, the most critical of the cross-platform vulnerabilities in the SANS Top 20 list. Honeypots for newer applications such as VoIP and SCADA may become widespread [7].

4.2 Counter-Deception

Malware is increasingly more sophisticated. Developers of malware aim to make it undetectable. New offensive techniques are adopted once they are made public and quickly adapted to face new defensive techniques [21].

Examples of counter-deception are found in publications in Phrack magazine describing methods to detect, disable, and defeat Sebek [1] [22] in an attempt to avoid malware collection and hence malware analysis.

A trend has emerged in which malware uses evasion, e.g. the Agobot botnet family uses polymorphism as an obfuscation mechanism [20]. Malware is able to detect whether it is running in a virtual machine and change its behavior, e.g. a specimen discovered by Intelguardians [12], the worm Conficker, the Storm worm [24], and Agobot [23]. Examples developed by security researchers include Nopill [26], Vmdetect [27], Redpill [28], Scoopy Doo [2], and VMwareTools [29]. Scientific literature on the topic of detecting honeynets includes *NoSEBrEaK - Attacking Honeynets*, by Dornseif et al. who demonstrate methods to control honeynets [22].

In [30] two broad groups of strategies for detecting deception were identified: strategies based on detection of evidence of deception in the environment, and inspection for signs of deception in the information within the environment.

Honeypot detection methods usually exploit discrepancies between the real systems and honeypots [2]. Provos and Holz discuss several techniques to detect low-interaction and high-interaction honeypots. Realistic looking low-interaction honeypots need to deceive network scanning tools. High-interaction honeypots need to simulate an entire operating system environment. The deceiving nature of physical high-interaction honeypots can be concealed; however, honeypots running in virtual environments have additional challenges as virtualization is detectable [2].

Methods of virtualization detection exploit logical discrepancies, resource discrepancies and timing discrepancies. a) Logical discrepancies evaluate semantic differences in the interfaces of real and virtual hardware. b) Resource discrepancies evaluate the resources that the virtual machine shares with its guests, such as CPU cycles, physical memory and cache footprint. c) Timing discrepancies evaluate the variance in latency, relative differences in the latency of any two operations, and the behavior of these latencies over time [31]. The main reason for these discrepancies is that the virtual machines were designed to provide fidelity, performance and safety, but not transparency [2].

Several methods suggest themselves for detecting client honeypots. a) Observing click rate and dwell time could identify a client honeypot tasked with identifying malicious web pages as fast as possible. b) Referrer evaluation is another mechanism that identifies client honeypots based on their navigational characteristics. c) Another possible identification means is the network location of the incoming requests. These techniques are applicable to both low- and high-interaction client honeypots.

[1] Developed by the Honeynet Project [25], Sebek is a tool for collecting forensic data from compromised high-interaction honeypots [2].

There are other techniques that are specific to this type of client honeypot. High-interaction client honeypots could be identified by rendering checks. As a page is loaded, an adversary could check whether the page is actually displayed.

A low-interaction client honeypot likely appears like a regular browser. Using the header fields of entire requests can uncover this deception. Header order and data formatting might also give the deception away or the TCP/IP track can be analyzed with the passive OS fingerprinting tool p0f [32].The header lines and values of an http request can be analyzed and compared to a fingerprint database to identify a given web browser using browserrecon [33]. Further, low-interaction client honeypots are light weight, stripped-down versions of the browser. An adversary can discern it by calling functionality that is present in a full-fledged browser, but not in a low-interaction client honeypot [34]. Because low-interaction honeypots simulate a system and do not provide a complete operating system environment to the adversary, they can be detected more easily than high-interaction client honeypots.

4.3 Counter Counter-Deception

These researchers refer to counter counter-deception as the analysis of attackers' counter-deception techniques that result in new deception designs. The changes occurring in Sebek's code after publication of *Advanced Honey Pot Identification* [22], describing methods to defeat Sebek, is an example of counter counter-deception.

Counter counter-deception focuses on two main areas: creation of defenses and understanding how attackers work and think. The authors believe that this understanding will lead to improvements in honeypot research and development, applying deception techniques. Seifert, et. al., proposed a taxonomy of honeypot systems that facilitates the understanding of honeypot technology by presenting a faceted classification that addresses six areas of honeypot study: interaction level, data capture, containment, distribution appearance, communication interface and role in multi-tier architecture. This taxonomy offers a framework for describing honeypot research [35]. The values for each area [35] are shown in Table 1.

Table 1. Honeypot Taxonomy

Category	Interaction Level	Data Capture	Containment	Distribution Appearance	Communication Interface	Role in Multi Tier Architecture
Values	-High -Low	-Intrusions -Events -Attacks -None	-Defuse -Block -Slow Down -None	-Distributed -Stand-Alone	-Software API -Network IP -Non Network Hardware IF	-Client -Server

The authors argue that the systematic application of Bell and Whaley's theory of deception, using the taxonomy of honeypots, facilitates the identification of potential research gaps. According to Bell and Whaley, even though most cheating is done intuitively, the complex process to plan and design a deception can be depicted in a Deception Planning Loop. Deception falls in categories within two levels, hiding and showing [14], as shown in Table 2.

Table 2. Deception Levels and Categories

Level	**Hiding**	**Showing**
Category	- Masking - Repacking - Dazzling	- Mimicking - Inventing - Decoying

These categories give a spectrum of characteristics or charcs [14] (e.g. taxonomy of honeypots) to be used during the deception. The ruse is the process of selecting the appropriate categories of cheating and subsequently the characteristics to create a cover or effect. Ruses fall in categories: unnoticed, benign, desirable, unappealing and dangerous. The ruse creates a cover or effect for the attacker to accept the illusion. The planning of the deception aims at anticipating the illusion; however, the illusion depends only on the perception of the target audience [14].

Bell and Whaley describe the *Deception Planning Loop* as:

> *…Fashioning a RUSE from CHARCS that are projected by a selected CHANNEL as an EFFECT or COVER that, if successful, created an ILLUSION made up of the perceived CHARCS that is, therefore, a successful stratagem supporting the Deception Goal and hence the Strategic Goal* [14].

The deception model varies for attacks focused on targets of choice, and attacks focused on targets of opportunity executed with automated tools such as worms [4]. It also varies according to the type of honeypot. High-interaction honeypots are examples of mimicking users, browsers, and active content. Low-interaction honeypots are examples of decoying.

For example, the deception goal of a high-interaction client honeypot is to "look" like a human user and be attacked. The ruse is to mimic the human behavior by navigating on the Internet (charc/channel) and interacting with servers using a web browser (charc/channel). The operating system and applications have a degree of known vulnerabilities that are controlled according to the empirical experiment (charc/channel). If successful, the malicious server will have the illusion that the client honeypot is an actual user and will execute the attack.

The definitions of new approaches to develop honeypots are examples of different ruses. Some new approaches to develop honeypots have been formulated. For example, Vukasin Pejovic et al. conducted an initial investigation and implementation steps for the deployment of honeypots as an independent hardware device with the incorporated honeypot behavior [36].

These researchers argue that for the future development of honeypots, the results of a deception plan should retro-feed the *Deception Planning Loop*, making the definition of charcs and channels an ongoing process. For example, attackers frequently use compromised computers to spread attacks. To prevent these attacks, using deception techniques, honeypots control the data leaving them. E.g. Sebek and other Gen II honeynets impose a hidden limit to the number of outbound connections [37]. Lessons learned from some experiments are useful when planning the deception. For instance, Rowe and Goh observed increasing number of attacks after the system went down and came back up. This analysis suggests that keeping an existing long-used IP address and responding normally to packets might lead to a decrease in the number of attacks [38].

The authors believe that counter counter-deception in the development of client honeypots, in addition to a technical approach, should be complemented by a political and social approach to learning about trends and alerts in the attacker community. This is part of a framework to study malware, attackers' behavior and attack trends.

The Honeynet Project deployed the Global Distributed Honeynet project, with goals such as global deployment of more high-interaction honeynets, and cross referencing of incident data for correlation against historical forensic databases [39].

The NOAH Project, funded by the European Commission, is a three year project that intends to gather and analyze information about the nature of Internet cyber attacks [40]. Its *Honey At Home* implementation project extends its network to homes and small businesses [41]. It will develop an infrastructure to detect and provide early warning of attacks to expedite countermeasures to combat them.

Fred Cohen proposes the creation of a set of red teaming experiments in which attackers as well as defenders are studied, to understand how attackers work and think, and the effects of defenses on attackers [42]. Moreover, to isolate the effects of deception, he proposes the creation of control groups, and experiments with double blind data collection [42].

5 Conclusions

The determination of the current status of honeypot research and deployment by using deception theory can help identify which areas of honeypot technology research are priorities. This would be part of a framework to analyze malware, attacks and attackers' trends.

Stating the strategic deception goals, studying the feasibility of application of deception techniques available in the social sciences, becoming aware of what technology is available and the research status of such technology, and assessing the level of accomplishment of goals, would guide the depiction of the honeypot research and deployment landscape in order to indicate future research direction.

These researchers believe that aggregation, sharing and analysis of data captured with honeypots help describe the status of attacks and attacker trends. Adopting a taxonomy of honeypots enables the research community to agree on the object of study and facilitates needed communication.

References

1. Finisterre, K.: The Five Ws of Citect ODBC Vulnerability CVE-2008-2639 (2008), `http://www.milw0rm.com/papers/221`
2. Provos, N., Holz, T.: Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Pearson Education, Boston (2008)
3. Tan, P., Kumar, V.: Discovery of Web Robot Sessions Based on their Navigational Patterns. Data Mining and Knowledge Discovery 6(1), 9–35 (2002)
4. Spitzner, L.: Honeypots: Tracking Hackers. Addison-Wesley, Boston (2003)
5. Cohen, F.: The Use of Deception Techniques: Honeypots and Decoys. In: Bidgoli, H. (ed.) Handbook of Information Security, vol. 3, pp. 646–655. John Wiley & Sons, Chichester (2006)

6. Cheswick, B.: An Evening with Berferd in which a Cracker is Lured, Endured, and Studied (1991), `http://www.cheswick.com/ches/cv/main.html`
7. Riden J., Seifert C.: A Guide to Different Kinds of Honeypots. Security Focus (2008), `http://www.securityfocus.com/infocus/1897/3`
8. Seifert, C., Steenson, R.: Capture - Honeypot Client. Honeynet Project (2006), `https://projects.honeynet.org/capture-hpc`
9. Moshchuk, A., Bragin, T., Gribble, S.D., Levy, H.M.: A Crawler-based Study of Spyware on the Web. In: 13th Annual Network and Distributed System Security Symposium, The Internet Society, San Diego (2006)
10. Seifert, C., Welch, I., Komisarczuk, P.: HoneyC - The Low-Interaction Client Honeypot. In: NZCSRCS, Hamilton (2007), `http://www.mcs.vuw.ac.nz/~cseifert/blog/images/seifert-honeyc.pdf`
11. Ikinci, A., Holz, T., Freiling, F.: Monkey-Spider: Detecting Malicious Websites with Low-Interaction Honeyclients. In: Sicherheit, Saarbruecken (2008)
12. Carpenter, M., Liston, T., Skoudis, E.: Hiding Virtualization from Attackers and Malware. IEEE Security & Privacy 5(3), 62–65 (2007)
13. Longman: Dictionary of American English. Longman, White Plains (1983)
14. Bell, J.B., Whaley, B.: Cheating and Deception. Transaction Publishers, Edison (1991)
15. Rowe, N.C.: Measuring the Effectiveness of Honeypot Counter-Counterdeception. In: Proc. of the 39th Hawaii International Conference on System Sciences, vol. 6, p. 129c. IEEE Xplore (2006)
16. The Honeynet Project: Know Your Enemy: Learning About Security Threats. Pearson Education, Boston (2004)
17. Spitzner, L.: Honeypots: Catching the Insider Threat. In: Proc of the 19th Annual- Computer Security Applications Conference, pp. 170–179. IEEE Xplore (2003)
18. Lakhani, A.D.: A dissertation on Deception Techniques Using Honeypots. Information Security Group, Royal Holloway, University of London
19. Khattab, S.M., Sangpachatanaruk, C., Moss, D., Melhem, R., Znati, T.: Roaming Honeypots for Mitigating Service-Level Denial-of-Service Attacks. In: Proc. of the 24th ICDCS 2004, pp. 328–337. IEEE Computer Society, Washington (2004)
20. Barford, P., Yegneswaran, V.: An Inside Look at Botnets. In: Christodorescu, M., Jha, S., Maughan, D., Song, D., Wang, C. (eds.) Advances in Information Security, vol. 27, pp. 171–191. Springer, US (2007)
21. Harris, S., Harper, A., Eagle, C., Ness, J.: Gray Hat Hacking. McGraw-Hill, New York (2007)
22. Dornseif, M., Holz, T., Klein, C.N.: NoSEBrEaK - Attacking Honeynets. In: Proc. of the IEEE Workshop on Information Assurance and Security, pp. 123–129. IEEE Xplore (2004)
23. Dittrich, D.: VMWare Detection?. Virus.org (2004), `http://lists.virus.org/honeypots-0411/msg00044.html`
24. Zdrnja, B.: More Tricks from Conficker and VM Detection. SANS Internet Storm Center (2009), `http://isc.sans.org/diary.html?storyid=5842`
25. Spitzner, L.: Sebek. The Honeynet Project, `http://www.honeynet.org/project/sebek`
26. Quist, D., Smith, V.: Detecting the Presence of Virtual Machines Using the Local Data Table. Offensive Computing, `http://www.offensivecomputing.net`
27. Lallous: The Code Project. Detect if Your Program Is Running Inside a Virtual Machine (2005), `http://www.codeproject.com/KB/system/VmDetect.aspx?display=Print`
28. Rutkowska, J.: Red Pill or How to Detect VMM Using (Almost) One CPU (2004), `http://invisiblethings.org/papers/redpill.html`
29. Kato, K.: VM Back, `http://chitchat.at.infoseek.co.jp/vmware/vmtools.html`

30. Santos, E., Johnson, G.: Toward Detecting Deception in Intelligent Systems. In: Proc. SPIE the International Society for Optical Engineering, vol. 5423, pp. 130–141. SPIE, Bellingham (2004)
31. Garfinkel, T., Adams, K., Warfield, A., Franklin, J.: Compatibility Is Not Transparency: VMM Detection Myths and Realities. In: Proceedings of the 11th USENIX workshop on hot topics in operating systems (2007), http://www.usenix.org/event/hotos07/tech/full_papers/garfinkel/garfinkel_html
32. Zalewski, M.: The New p0f: 2.0.8 (2006-09-06), http://lcamtuf.coredump.cx/p0f.shtml
33. Ruef, M.: Browserrecon Project, http://www.computec.ch/projekte/browserrecon
34. Hoffman, B.: Circumventing Automated JavaScript Analysis. In Black Hat USA, Las Vegas (2008), http://www.blackhat.com/presentations/bh-usa-08/Hoffman/Hoffman-BH2008-CircumventingJavaScript.ppt
35. Seifert, C., Welch, I., Komisarczuk, P.: Taxonomy of Honeypots. Technical Report CS-TR-0. School of Mathematical and Computing Sciences. Victoria University of Wellington (2006)
36. Pejovic, V., Kovacevic, I., Bojanic, S., Leita, C., Popovic, J., Nieto-Taladriz, O.: Migrating a HoneyDepot to Hardware. In: The International Conference on Emerging Security Information, Systems, and Technologies, pp. 151–156. IEEE Xplore (2007)
37. Rowe, N.C.: Deception in Defense of Computer Systems from Cyber Attack. In: Janczewski, L.J., Colarik, A.M. (eds.) Cyber Warfare and Cyber Terrorism, pp. 97–104. IGI Global, Hershey (2008)
38. Rowe, N.C., Goh, H.C.: Thwarting Cyber-Attack Reconnaissance with Inconsistency and Deception. In: Proc. of the IEEE Workshop on Information Assurance United States Military Academy, West Point, NY, pp. 151–158. IEEE Xplore (2007)
39. Watson, D.: GDH Global Distributed Honeynet. The Honeynet Project (2007), http://www.ukhoneynet.org/PacSec07_David_Watson_Global_Distributed_Honeynet.pdf
40. European Network of Affined Honeypots: About NoAH, http://www.fp6-noah.org/about
41. European Network of Affined Honeypots: honey@home, http://www.honeyathome.org
42. Cohen, F.: The Use of Deception Techniques: Honeypots and Decoys, http://all.net/journal/deception/Deception_Techniques_.pdf