*I. J. Computer Network and Information Security*, 2012, 10, 63-75
Published Online September 2012 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2012.10.07

# A Survey: Recent Advances and Future Trends in Honeypot Research

Matthew L. Bringer, Christopher A. Chelmecki, and Hiroshi Fujinoki
Department of Computer Science, Southern Illinois University Edwardsville, Edwardsville IL 62025, US
matt.bringer@gmail.com, chris.chelmecki@gmail.com, hfujino@siue.edu

***Abstract*** — This paper presents a survey on recent advances in honeypot research from a review of 80+ papers on honeypots and related topics mostly published after year 2005. This paper summarizes 60 papers that had significant contribution to the field. In reviewing the literature, it became apparent that the research can be broken down into five major areas: ① new types of honeypots to cope with emergent new security threats, ② utilizing honeypot output data to improve the accuracy in threat detections, ③ configuring honeypots to reduce the cost of maintaining honeypots as well as to improve the accuracy in threat detections, ④ counteracting honeypot detections by attackers, and ⑤ legal and ethical issues in using honeypots. Our literature reviews indicate that the advances in the first four areas reflect the recent changes in our networking environments, such as those in user demography and the ways those diverse users use new applications. Our literature reviews on legal and ethical issues in using honeypots reveals that there has not been widely accepted agreement on the legal and ethical issues about honeypots, which must be an important agenda in future honeypot research.

***Index Terms*** — Honeypots, Alarm system, Computer hacking, Computer crime, Computer security

## I. Introduction

Since their introduction at the end of the 90's, honeypots have evolved in diverse directions to cope with various new security threats against not only security defenders but also novice users in the Internet today. The recent changes, including those in hardware, software and even user demography, have been rapid enough to require a new survey especially on the recent challenges to and evolutions in honeypots.

As can be seen in the work by Bailey [1] and Holz [2], the changes that had significantly influenced the recent advances in honeypot research are, adoption of new types of network applications, geographical diversity in Internet user population, growth in underground attackers' communities, and advances in networking hardware.

There have been a few surveys on honeypot research. Seifert [3] provided a survey on different types of the existing honeypots focusing on their purposes and advantages. Fu and Porras provided a brief survey on specific types of honeypots. Fu's work includes a brief survey on the existing techniques and methods used for detecting Honeyd virtual honeypot to develop effective countermeasures against its detections [4]. Porras's work contains a survey on the existing challenges in wide area data collection using honeypots and other methods [5].

On top of the contributions by those existing surveys, this survey intends to provide an organized view of the recent achievements in honeypot research to cope with the rapid changes mentioned above. This survey focuses on investigating the recent evolutions in honeypots, as well as understanding the current trends and the directions of future honeypot research.

The rest of this paper is organized as follows. Section II contains summaries of the literatures that focus on new types of honeypots, and outlines the recent evolutions in honeypots. Section III contains summaries of the literatures that focus on improving utilization of output data from honeypots. Section IV is for those that focus on helping configurations of honeypots. Section V features the literatures that focus on the battle between attackers and security administrators in detection and counter-detection of honeypots, followed by Section VI for legal issues related to using honeypots. Section VII is the conclusion of this survey, followed by a list of the literatures summarized in this survey. For simplicity, each literature is referenced by the first author.

## II. NEW TYPES OF HONEYPOTS

To cope with the recent changes in the Internet, such as the advent of new popular network applications, wide adoption of wireless networking devices, introduction of high-speed subscriber link technologies to every household, and diversity in users' demography in terms of culture and legal systems, new types of honeypots have been proposed and introduced. This section focuses on the recent evolutions of honeypots by describing the new types of honeypots recently proposed.

**Adachi** proposed "BitSaucer", a low-interaction and high-interaction hybrid honeypot to achieve lower resource requirements in low-interaction honeypots and the ability to emulate full responses in high-interaction honeypots [6]. The key in the solution is the proxy running in each host, which is a daemon responsible for generating virtual hosts and redirecting network traffic. Each virtual host emulates full system in high-interaction honeypots, while on-demand invocations of such virtual host minimize resource consumptions, since high-interaction honeypots are automatically invoked only

 

when network traffic that requires such high-interaction honeypots arrive at a host.

**Alberdi's** solution allows honeypots to monitor actual malicious activities by bots, worms, and viruses without letting them leave the honeypots [7]. Alberdi proposed "redirection kit", which redirects outgoing attacks, such as messages bots used to coordinate attacks, to other honeypots to prevent the malicious attacks to other production servers through honeypots at the same time it prevents detection of the honeypots by the attackers. Using this mechanism, the bot masters still believe that their bots communicate with hosts outside of the network, while they are actually communicating with another honeypot in the same network.

**Alosefer** developed a low interaction client-side honeypot, "Honeyware", for detecting malicious web servers [8]. Alosefer tested Honeyware against 94 URL's he collected in advance (84 malicious, 10 benign). Honeyware detected 83 of the malicious URL's. Capture-HPC detected 62 malicious URL's, counted 23 as benign and misrecognized the remaining 9. Since Honeyware is a low interaction honeypot, the data collected by it must be processed by an external processing engine, which takes time. Honeyware averaged 1 minute per URL where Capture-HPC took 17 seconds. Alosefer concluded that high-interaction and low-interaction honeypots should be integrated to take the advantages of the both in future honeypot design. Such integration should take the advantage of a low interaction honeypot, which is easily installed but requires external data analyses while high-interaction honeypots have opposite properties.

**Anagnostakis** focused on the trade-off problem between the accuracy (false-positive and false-negative) in high-interaction honeypots and the breadth of the coverage in anomaly detection [9]. To solve the trade-off, Anagnostakis proposed "shadow honeypots" that are real production network applications but contain honeypot codes embedded in them. All incoming requests to a server running the shadow honeypot will be executed just as if they were executed by a production server. However, the embedded honeypot codes always monitor the behaviors of each request. When one is confirmed malicious, any activity made by the attackers will be rolled back. Each incoming request is first processed by an anomaly detection, which is configured intentionally with high false positives. The requests classified malicious will be forwarded to the shadow honeypots, while those determined innocent will be directly forwarded to a production server. If the shadow honeypot determines a request to be innocent, it forwards the request to the production server.

**Bailey** integrated low and high-interaction honeypots to solve the trade-off problem between width of honeypot coverage (the advantage of low-interaction honeypots) and behavioral fidelity (the advantage of high-interaction honeypots) [10]. Wide coverage means different types of network traffic (different addresses, ports, applications, etc), while high behavioral fidelity means the details in the information collected by honeypots (a sequence of activities, responses made by attackers, etc.). Bailey proposed use of multiple low-interaction honeypots as sensors to collect information about network traffic to increase the width of coverage. If low-level honeypots observe anything related to threats, they hand-off the sessions to one of the high-interaction honeypots. Using this architecture, the number of high-interaction honeypots in a network can be reduced, while multiple, possibly a large number of, low-interaction honeypots widens coverage.

**Das** proposed a solution to mitigate denial of service attacks by hiding production servers behind an access gateway, called "Active Server (AS)" [11]. Each AS works as a required gateway to reach a production server. Each AS authenticates its clients and once a client is authenticated, a path is opened between the client and a server. If an AS does not authenticate a client, it behaves as a honeypot, trapping the client there. If a client has access to multiple ASes, the client can be authenticated by any AS. Since the authentication of all clients at an AS must happen prior to access to the server, it will prevent DoS attackers from clogging the path from an AS to the protected server. This will save the legitimate clients who have access to only one AS to reach the protected server.

**Ghourabi** argued that there was no high-interaction honeypot for securing routers, which have been targets for exploits on the existing routing protocols, such as RIP, OSPF, and BGP [12]. Ghourabi proposed the client-side honeypots for securing routers since such client-side honeypots could actively search for attackers to routers, which would allow early detections of new and unknown attacks to routers. Ghourabi developed a high-interaction client-side honeypot, using existing software tools, such as Quagga for actively sending messages to remote routers based on particular routing protocols, to examine if the routers have been compromised. The responses from remote routers were captured and manually processed using Wireshark. The authors tested their high-interaction honeypot for the two test cases that exploit RIP and OSPF routing protocols to show that the honeypot could potentially detect such attacks to routers.

**Jiang** investigated the weaknesses in high-interaction honeypots that are implemented using a virtual machine (VM) system [13]. Since VM's are complex software systems, they are vulnerable to exploits on bugs and security holes, such as possible miss-configurations. Such vulnerabilities mean high risk of VM's, including honeypots' running on top of them. Jiang proposed use of two sensors to monitor high-interaction honeypots. Internal sensors are running within the honeypot and record invocations of system calls and their responses, including the information about which processes make which system calls. External sensors are introduced to cover the weakness in internal sensors; internal sensors are vulnerable to compromises by attackers. External sensors intercept the data within the traffic in and out of a honeypot to monitor internal sensors.

**Khattab** developed a solution to mitigate impact of denial of service (DoS) attacks to the production servers

 

[14]. In the solution, honeypots and production servers are continuously shuffled within a network. Honeypots trap attackers, which prevented, reduced, and delayed the impacts from the DoS attacks. The production processes automatically became honeypot when they migrate to other hosts. This solution will be effective when the majority of the incoming requests are DoS attacks. The most significant weakness, as the authors admitted, is that, since *k* out of *N* hosts in a network must be allocated to honeypots, the solution will not be effective for flash crowds or when DoS attacks are minority in the incoming requests. This method will be an effective solution for a large scale distributed DoS attacks.

**Kreibichi** proposed a prototype honeypot to automatically generate signatures for intrusion detections without hard-coding any clue in advance to achieve zero-day detections of unknown malware [15]. The core of the prototype consists of connection tracking and a new signature creation algorithm, which are implemented in a plug-in module hooked to the code in an existing low-interaction honeypot. The connection tracking performed two-stage tracking, which classified connections in the connection-establishment phase and in the connected phase to utilize the fact that a large number of connection requests usually appear at a host, but not many of them actually complete connection set up – thus it reduced the number of connections that are carefully monitored. The signature creation algorithm then performed protocol-level and application-level analyses on each suspicious connection to generate signatures of malware based on their activities and correlations in them.

**Lauinger** warned that the concept of honeypot is being abused by attackers in recently prevailing instant messaging (IM) applications [16]. The technique, called "social engineering", performs physhing attacks in IM by hijacking sentences human users create in real-time. A high-interaction honeypot, which is set up by an attacker, first invites two human users to an IM session, one at a time. Then, it establishes a connection between the two human users and the honeypot relays the messages as a hidden middleman between them. When the honeypot relays the messages, it monitors the ongoing message exchanged between the human users and modifies the messages on the fly. Using this technique, the honeypots can create messages that look like those created by actual human users.

**Li** invented a new way of integrating spam-trapping for online banking systems using honeypots to cope with the problems in the existing anti-phishing techniques [17]. In the existing systems, after spam traps capture spam emails from physhers, the human administrators analyze them and launch a client-side honeypot to the physhing sites to examine whether they are physhing sites or not. This causes significant delay in responses that lets attackers detect such investigations. The proposed honeypots, called "phoneybots", are distributed honeypots that are specially designed for monitoring bank transactions using fake accounts to trap behaviors of physhers. It automates the process of accepting physhing requests, responding to the physhing sites, and analyzing their potential dangers. Li also analyzed optimal distributions of their honeybots to minimize the risk of counter-detections by physhers.

**Nazario** proposed a honeypot called "PhoneyC", to extend existing honeypots in two directions [18]. The first is to make honeypots "active", which means client-side honeypots. The second is the dynamic web content parser to interpret binary dynamic contents, especially client-side scripts, such as JavaScript, VB Script, and even Active-X controls. Since it is extremely difficult to understand binary expressions of those client-side scripts for human security administrators, the binary content parser assists human administrators to discover malicious web servers. Integrating the two extensions to web applications, active client-side honeypots become "web clawers" that visit a large number of web servers to automatically detect malicious web servers. As a result, PhoneyC was able to detect many malicious script/control activities during experiments.

**Portokalidis** designed a honeypot particularly to slow down dissemination of new, and thus unknown, malware, such as worms, viruses, and bug exploits [19]. The honeypot, called "Argos", automates monitoring, detecting, and generating signatures of new unknown malware for intrusion detections. Argos was designed to improve the weaknesses in the existing systems, such as Minos and Vigilante. For example, Minos does not generate a signature for intrusion detections, while Vigilante does not protect OS kernel. Argos performs dynamic taint analysis, which keeps track of data delivered from a network connection so that Argos raised a flag if it is executed in a local host. When Argos detects a suspicious taint data, it also dynamically inserts assembly codes, called "shellcode", into the process to extract detailed information about the process and/or to modify the execution paths of the process so that the process is slowed down or trapped in an infinite loop to minimize its harm.

**Prathapani** designed a honeypot to detect black hole routers in wireless mesh networks, which have at least two different paths between any two routers in a network [20]. Black hole routers announce bogus best paths to attract traffic through it, simply to drop all attracted network traffic. Prathapani's solution consists of feedback module, alert module, and router module. The router module sends a message to the feedback module through queried routes and a known route. The feedback module responds to the messages sent by the router module. If the router module does not receive a response, it determines that the tested router is a black hole attacker. The alert module sends notifications to the rest of the network to avoid the attacker. The detection technique proposed by Prathapani is essentially the honeypot detection technique used by attackers [21], but Prathapani used it for detecting attackers.

**Rowe** proposed fake honeypots to repel attackers from production hosts and developed a mathematical model that estimates the effect of the fake honeypots to maximize their effects [22]. Fake honeypots are dummy honeypots in a sense that they look like a honeypot, but

 

they are not performing any real feature typical honeypots perform. The goal of fake honeypots is to repel attackers from a production network by intentionally exposing themselves to attackers. The solution counter-utilizes the fact that many attackers avoid network hosts where honeypot is running nearby. However, too obvious appearance of fake honeypots may let attackers counter detect them to be fake, while too subtle indication of a honeypot will not maximize their effects. A mathematical model was introduced to maximize the effect of the fake honeypots, using some parameters, such as the probability of a system being a honeypot, the benefit expected by an attacker from compromising a production host, and the cost for compromising a host.

**Webb** developed a high-interaction honeypot to cope with currently popular social spamming [23]. Social spamming is sending spam messages to innocent human users in social networking services to guide them to malicious web sites. The proposed high-interaction honeypot analyzes "friend invitation requests" spams in MySpace. Webb created 51 dummy personal profile accounts in MySpaqce. Running behind each dummy page, the honeypot waited for incoming friend invitation requests from spammers, downloaded the spammers' profiles, recorded their origin network addresses, and identified the spammers' geographical location for further analyses. Webb found that the most popular spamming targets were Midwestern states, and that the most popular home of the spam profiles was California. As much as 57.2% of the spam profiles used the same "About Me" content. Spam profiles used thousands of URL's and various redirection techniques to funnel to a handful of destination web pages. Webb also found that the spamming behaviors in spam profiles follow distinct temporal patterns.

**Zhuge** proposed a new honeypot, called "HoneyBow", to automatically detect and capture malware, such as viruses and worms, without requiring human security experts manually investigating output data from honeypots [24]. A component in HoneyBow, MmFetcher**,** detects the modifications of files by comparing their initial MD5 hash after it intentionally lets malware modify its files. When any modification is detected, the process that made the modification is captured as malware and MmFetcher restores the initial copy of the files. Another component, MmWatcher, monitors system calls that perform file creation and modification, which triggers intrusion detection. Finally, MmHunter monitors code being executed like a debugger to detect malware's suspicious activities.

## III. UTILIZING HONEYPOT OUTPUTS

Since the introduction of honeypots, honeypots have been a domain of network security experts. The main reason behind this has been that understanding and utilizing the raw data collected by honeypots requires thorough analysis skills, which also requires comprehensive expertise in network protocols, applications, network hardware equipment, operating systems, and even user management. Moreover, those skills require experience in network security administration. These prerequisites have inhibited the adoption of honeypots as a popular tool for enhancing network security, despite their potential. In the last five years, much has been done to ease and even automate the raw data processing, and to integrate advances in other related areas, such as intrusion detections, data mining, expert systems, artificial intelligence, and even game theories. This section focuses on summarizing such recent advances in honeypots.

**Chen** integrated intrusion tolerance into network security forensics using honeypots, which Chen called "dynamic forensics" [25]. The solution makes sure that data gathered for forensic analysis is reliable even if those attacks have tried to modify the data. Its key component is the intrusion detection that monitors threats. When the threat level is high, the forensic system is dynamically activated. If the threat warning is validated by the dynamic forensic system, the traffic is automatically redirected to the honeypot. If attacks progress beyond a certain point, all traffic is cut off to avoid modifications of the collected data by attackers. The attacks are then analyzed and the signatures will be extracted to detect future attacks. An agent, called "Evidence Collection Agent (ECA)", keeps logs and data for all alerts as well as the firewall/load balancer which acts as a redirector when the alert level exceeds a threshold.

**Cooke** studied the possible threats to honeypots by bots [26]. Although honeypots are tools to detect and prevent bots, Cooke's survey on bots revealed that the recent advances in bots have been posing serious threats to honeypots. Cook performed experiments in which a honeypot was installed to observe that the honeypot was repeatedly compromised by attacks from bots, sometimes by two bots at once. These stunning findings imply that the recent bots are strong so that special honeypots to protect honeypots are needed. Cooke proposed "super honeypots" that are honeypots for honeypots. Cook designed the super honeypot in the following way. First, deceptive honeypots that intentionally let bots infect and compromise them are set up. Then, the super-honeypot monitors the deceptive honeypots to capture, learn, and prevent such bots.

**Dantu** proposed a new architecture, which detects worms by monitoring the rate of their outgoing connections. The new architecture slows down the worms by throttling the rate of creation of new outgoing connections based on a closed feedback loop control [27]. The primary contribution by this work is in designing, tuning and performance-testing of the closed-feedback-loop control for throttling outgoing connections. Dantu applied Proportional, Integral, and Derivative (PID) algorithm which throttles outgoing connections to a designated set point for reducing spreads of worms. The authors found that the algorithm significantly slowed down the spread of worms by approximately five times. With containment (killing infected processes and blacklisting hosts) using multiple-feedback loops and intelligently queuing the connections, spreads of worms

 

could be stopped with a significantly fewer number of hosts infected.

**Fairbanks** proposed a new method to overcome a limitation in data collection within Linux Virtual File System (VFS) and possible modifications of files by attackers [28]. The limitation is the inability to identify file names from the i-nodes. When forensic analyses need to identify the file names whose i-nodes raised some warning for possible security anomalies, there is no easy way to identify their file names since their linkages are unidirectional (only from filenames to their i-nodes). The proposed method solves the problems by altering the VFS code in the Linux kernel in such a way that the kernel saves a data structure called "dentry", which contains the backward link from an i-node to its filename. Saving the dentry to a remote file through serial connection also can be used to analyze what files attackers changed or executed even if they alter log files to cover their attacks.

**Francois** applied intersection graphs to detecting coordination in distributed attacks that are collaboratively performed by a group of remote hosts that have different source addresses while there is no clear overlap in their message contents [29]. Intersection graphs were used to identify responsible nodes, called "central nodes", in hidden coordination in distributed attacks by profiling communication patterns in seemingly unrelated nodes. The authors compared the results from the intersection graphs between distributed honeypots and network telescopes to conclude that distributed honeypots were better in recognizing the source addresses in hidden coordinated attacks, while network telescopes were better in detecting target network services by such coordinated attacks when attacks were not frequently performed.

**Hoepers** designed and implemented a standard protocol to exchange output data from heterogeneous honeypots [30]. The standard protocol also aimed to coordinate activities in heterogeneous honeypots. The protocol was proposed to integrate many distributed honeypots and coordinate them for better and quicker detections of network security threats. In designing the standard, Hoepers identified properties and designs that should be satisfied by such a protocol. Hoepers proposed inter-operability, open systems, syntax that describes semantics, object-oriented encoding, and future extension of data analyses in the standard protocol.

**Krasser** argued the importance of visualization of data collected by honeypots to assist analysis on the data to cope with the two current problems in the data analyses. The first problem is that it is extremely labor intensive or hard for detecting anomalies from huge volume of data collected by honeypots, while the second problem is that investigations on data require special skills, experiences and training [31]. Krasser visualized time lines, packet size, distributions of the source IP address, protocol types, duration of each connection and local ports accessed, with additional visualization options of live packet-capture display, animated playback feature, manipulation by a pointing device for interactive capability, animated scatter-plots, and three-dimensional display features with zoom and pan.

**Mohammed** proposed Double Honeynet and Principal Component Analysis (PCA) to enhance accuracy in signature generations for polymorphic worms, which is a difficult task since polymorphic worms continuously modify themselves to avoid their precise fingerprinting [32]. The proposed system catches a worm in the first honeynet, the worm is then allowed to infect other systems in the second honeynet. The worm can move back and forth between the two honeynets, evolving as it goes. Each version of the worm is captured as the worm repeats infection between the two honeynets. All versions are analyzed using PCA to produce a signature that can be used to detect the polymorphic worm using intrusion detection systems.

**Narvaez** assessed their hypothesis on a weakness in high-interaction honeypots implemented on a virtual machine (VM) using experiments [33]. Narvaez hypothesized that such high-interaction honeypots should have less success in identifying websites that host malware in drive-by-download attacks if malware are able to detect VM implementations of honeypots. The authors set up two systems, a honeypot on a VM and another on a full system (Capture-HPC was used for both). The honeypots were connected to the same set of known malicious websites and compared for their effectiveness in detecting the malicious web sites. The results suggested that use of a VM in implementing a high-interaction honeypot would not negatively affect the likelihood of counter-detections by attackers, which was one of the concerns McCarty raised in his paper [21].

**Newsome** developed a solution called "polygraph" to cope with polymorphic worms in automating their signature generations with low false negatives and low false positives while using network flows that may contain noise to generate the signatures [34]. The solution extracts a signature that consists of multiple disjoint content substrings for a polymorphic worm, instead of a particular fixed substring to achieve the accuracy. The solution first discovers tokens, which are the substrings that have to appear in a specific order or the special case of regular expression using clustering techniques. To efficiently extract signatures, the solution classifies the contents to three major sections of invariant, wild card, and code bytes data sections. The authors concluded that content-based filtering with effective signatures would be still useful, contrary to the wide-spread rumors that are skeptical about their usefulness.

**Raynal** described a procedure of information assurance forensics using honeypots [35, 36]. Raynal described the suggested forensic procedure in investigating server intrusion incidences to find their purposes and the techniques used by the intruders, as well as the tools they used. The proposed forensic procedure consists of network activity analyses, system and file analyses, and evidence gathering. What this paper implied is that the true challenge in using honeypots is in how we should develop systematic methods to utilize the data collected by honeypots to efficiently prevent known and unknown network security attacks, as well as the methods to disseminate such skills and experiences.

 

**Su** applied data mining techniques to automating discoveries of new attack patterns from data collected by honeypots [37]. The authors introduced the concept of "episodes" to identifying new unknown attack patterns. Episodes are defined as a series of events where the events are identified based on their basic patterns of serial, parallel, or complex events, which are a combination of serial and parallel events. The authors then applied two episodes mining algorithms of WINEPI and MINEPI proposed by Mannila [38] to the data collected by distributed honeypots, called "KFSensor". The authors demonstrated the capability of the data mining approach by discoveries of redundant and correlative episodes, which detected the Korgo, Shelp, and Sasser worms. The authors showed that the algorithms would be effective in finding short repetitive episodes that are generally innocuous.

**Tang** proposed "double honeypot" to effectively extract signatures for detecting polymorphic worms to achieve their zero-day detections [39]. It consists of inbound honeypots, outbound honeypots, and address translators. Tang also proposed a new method of data analysis applied to the data collected by the double honeypot, called "Position Aware Distribution Signatures (PADS)". The double honeypot counter-utilizes the fact that worms open outgoing connections. By monitoring such unexpected outgoing connections from an inbound to an outbound honeypots, worms can be identified easily. PADS was designed to increase the chances of detecting polymorphic worms by allowing possible variations in a signature, instead of "all fixed symbols" in the existing signatures. To control "variations" in each position in signatures, PADS uses the byte frequency distribution, which specifies what variations are how likely possible in each position in a signature string.

**Thonnard** developed a framework for efficiently detecting unknown attack patterns from honeypot raw output data [40]. This paper addressed the trade-off problem between the amount of effort security administrators have to invest to their investigations of raw output data and the quality in how successfully they can extract attack patterns from the output data. Thonnard applied a data clustering technique, and similarity distance to solving the trade-off problem in conjunction with graph-based approach. The graph-based approach used graphs to represent topological relations among the clusters, identified by their similarity distance. Thonnard discussed possible metrics that could be used for identifying clusters and calculating the similarity distance, as well as how they would make sense.

**Trivedi** analyzed instant messaging spams ("spims") using honeypots [41]. Trivedi set up honeypots as open proxies and analyzed spims they observed. Trivedi found that the purpose of the most of the spims was to guide innocent users to malicious web sites. To avoid detections, many spimers used different first-destination URL's, in which URL redirection was used to take the victims to malicious web sites. Many spimers used a technique, "randomized markup text", which added some extra characters to text lines in messages to avoid detections by intrusion detection systems. Trivedi's experiences demonstrated uses of honeypots for emerging network applications that cause new security threats.

**Wagener** proposed a model to verify how well a traffic monitoring system, such as honeypots, would be able to reassemble TCP sessions [42]. The proposed models quantified the confidence level on the data collected by honeypots especially when the data was used for network forensic analyses. Reassembling TCP sessions is a process of recognizing TCP sessions by identifying and grouping TCP packets that belong to the same TCP session to understand what is happening or has happened in the session. The focus was on quantifying errors that could occur during reassembles of TCP sessions. The types of errors Wagner's model quantified included neglected IP fragmentation rate, incomplete packet capturing rate, and misrecognized TCP session counts (and some more). Expected future work in the area would be thorough performance analyses and performance comparisons with existing verification models.

**Yegneswaran** studied the effectiveness in various analysis techniques to distinguish worms, bots, and anomalies caused by miss-configurations by human network administrators using the data collected by honeypots [43]. The authors described temporal source counts, arrival window adjustment, inter-arrival distribution analysis, destination-net scan foot printing, first destination preference, source-net dispersion analysis, per-source scanning profiling, and source lifetime analysis. Yegneswaran applied the techniques to each suspicious case observed at a honeypot and inferred the type of incidents for each case. Yegneswaran validated their inferences to conclude that their techniques were reliable.

**Zhao** designed a solution to prevent spreads of worms using honeypots [44]. The core of the solution is a model that predicts the propagation of worms. The model categorizes the state of each host as susceptible, infectious, or immune, to estimate the rates of infection, which determines the necessary actions to be taken for each host. The solution divides a network into parts, with honeypots deployed at some host computers in a network, called "hub nodes". In the model, Zhao developed a systematic method that optimizes a number of honeypots to be deployed, which will maximize the effect of stopping spreads of infections.

## IV. HONEYPOT CONFIGURATIONS

Another factor inhibiting popular adoption of honeypots is the difficulty in tuning honeypots. Honeypots are traps that monitor attackers' activities without being detected. Just like real traps, honeypots must be carefully configured to attract the right targets. Therefore, the honeypot operators must consider what activities should be monitored, whose activities should be monitored, when such activities should be monitored, and so on. If it is incorrectly configured, a honeypot not only fails to draw its prey, but also can be hijacked by attackers. Blindly capturing a huge volume of network

 

activities can also make subsequent data processing extremely difficult. Thus, it is extremely important that honeypots are configured to satisfy specific objectives. This section focuses on describing the recent significant contribution by honeypot research to ease and even automate honeypot configurations to maximize the effectiveness of honeypots for given objectives.

**Briffaut** developed the architecture of distributed high-interaction honeypots that monitor anomalies, especially possible abuses and hijacks on high-interaction honeypots, to automate frequent reinstallations of such illegally modified high-interaction honeypots [45]. Briffaut developed the architecture as a solution for a trade-off problem between low-interaction and high-interaction honeypots. The solution monitors the current status of each distributed high-interaction honeypot deployed in a network, frequently examines its status using intrusion detection and other tools, and automatically reinstalls a whole system if any anomaly is detected. The real challenge in the solution will be how to update "clean system image" while each system can sometime change in legitimate ways. This most probably requires logging every single activity in each high-interaction honeypot. Also, frequent reinstallations of the high-interaction honeypots will significantly increase their downtime, which can be another target of DoS attacks.

**Carroll** applied the concepts in the game theory to determining the optimal distribution of honeypots within a network [46]. Carroll classified four types of systems: normal production systems, fake normal systems (honeypots camouflaged as production systems), honeypots (not camouflaged), and fake honeypots (production systems camouflaged as honeypots). The authors developed models to achieve equilibrium between attackers and defenders using the four classifications. They used case studies to show how these models could be applied to distributions of honeypots. Although the models took a quite naive view of the defender-attacker relationship, these techniques could prove useful in distribution of honeypots, especially if combined with a dynamic honeypot distribution technique such as the one developed by Wang [47].

**Chen** offered advices and guidance on setting up a honeypot specifically for detecting and studying SQL injection attacks [48]. Chen argued that the honeypot should be highly interactive for all database related activities. Another suggestion was to have other non-production systems use the server to make the honeypot appears as real as possible. The system should allow attackers access up to the point of data manipulation. Monitoring and restricting the use of certain procedures were recommended since these could be used to alter the system. The authors also suggested that a proxy between the web and database servers could be used to stop certain SQL commands from reaching the database. The suggested design would use a honeynet to simulate a real network. This honeynet would forward all SQL injection attacks to a high interaction honeypot with a database server. This database should be populated with real-looking data. The authors also suggested using honey tokens, which are data that is not real but can be traced when it is used. The final consideration was to make sure that the database should not be too easy to access.

**Hecker** developed a system, "Honeyd Configuration Manager", for dynamic deployments of low interaction honeypots based on the needs detected by network scans [49]. The manager used Nmap to scan a network and gather details on the systems on the network, including operating systems and open ports. After the entire network is scanned, the configuration manager deploys and starts low-interaction honeypots based on the configuration files. The configuration files are introduced to allow administrators to set open ports and a network address to each honeypot, as well as to specify which server services should be emulated at each honeypot. The authors identified the possible future work for high interaction honeypots; passive network scanning to create a more realistic honeynet, dynamic modification of running honeypots, and overall improvements in the system emulation by the honeypots.

**Kohlrausch** described their experiences in analyzing the data collected by a honeypot testbed using Argos for detecting W32.Conficker worm [50]. The honeypot used "Dynamic Taint Analysis (DTA)", which marked each byte of data received from the network and traced its activities within a debugger. DTA was implemented by a combination of Argos honeypots, Snort intrusion detection system, and a debugger. Argos checked for unknown threats, while Snort was used to detect known attacks, each of which marked suspicious data for subsequent activity tracing by the debugger. The authors reported that the primary difficulty was in determining the meaning and contribution of each activity to the attacks due to a large volume of detailed data. Output from the honeypot also contained data about multiple unrelated attacks, which made the analysis even harder. Kohlrausch suggested that data output based on network flows would be a simple but an effective technique to significantly mitigate the difficulties.

**Spitzner** presented techniques for detecting insider threats using honeypots and honey tokens [51]. Spitzner pointed out that insider threats have challenges different from outsider attacks, as that the malicious insiders are given access to the system and are much more familiar with it. To help catch such malicious insiders, honeypots should be moved into the network and can take up all unused IP addresses. Also, because they are familiar with the system, all of the honeypots must be high interaction. The malicious insiders must be directed to the honeypots rather than hoping they come across them on their own. Attackers of this type are after information that they can use, therefore the honeypots must provide data that the attackers will want but do not need to know. This could include false business plans and design specifications. These false documents, as well as the password to log in such servers can be honey tokens.

**Wang** developed a method to dynamically distribute different types of honeypots to different places in a network [47]. The proposed method automatically adjusts distributions of high, medium, and low interaction

 

honeypots. The low interaction honeypots are basically sensors for known attacks, with the medium and high interaction honeypots acting as full systems to trap and analyze unknown attacks. The core of the solution is the formulas that optimize distributions of the three different types of honeypots. The system divided the network into four zones: outside the firewall, on the intranet, in the DMZ, and in a sub-network. As attacks are detected within the network, the system determines which zones, which level, and the number of honeypots to distribute using the formula on the fly. Through experiments, the authors showed that their system was capable of slowing down and stopping attacks at a greater rate than static deployments.

## V. HONEYPOT ANTI-DETECTIONS

Honeypots monitor and collect information about attackers' behaviors and their procedures for launching attacks. This is performed to prevent future occurrences of the attacks and even to collect forensic evidences for subsequent legal prosecution of the attackers. Thus, the true value in honeypots is in monitoring attackers' activities without being detected. Thus, from the attackers' point of view, detection of honeypots is vital and they have developed various, but systematic, honeypot detection methods, most of which are highly tuned to particular types of honeypots. During our literature review, we found that one of the most active front lines in recent research on honeypots is the battle between honeypot detection and anti-detection.

**Dornseif** demonstrated how honeynets could be compromised and controlled by attackers without any substantial logging as an antithesis to a wide-spread belief that honeynets are hard to detect and that attempts by attackers to detect and diable honeypots will be easily monitotred [52]. Dornseif analyzed the Sebek module, which are used in high interaction honeypots to log data, and described four methods to detect, disable, and circumvent Sebek, which showed that detecting Sebek was a fairly easy task against the wide-spread belief. Dornseif suggested to implement Sebek by a kernel patch instead of an add-on module to avoid detection. Using random or decptive names for symbols and variables and using decoy variables in high-interaction honeypots are also suggested as other techniques to avoid detections of honeynets.

**Holz** described major existing honeypot counter-detection techniques to conclude that high-interaction honeypots are facing serious challenges of counter-detection by attackers [53]. The authors described major known techniques to detect User Mode Linux (UML), VMware, Chroot, and Jails, as well as the execution mode of the processor (many VMwares run processors in supervisng mode) for examples performed by many attackers. The authors also discussed that the execution timing technique, as a general detection method for high-interaction honeypots, is effective in detcting processes running on top of a virtual machine. With those existing detection techniques, the authors conclueded that, for many honeypots running in the Internet today, detection is not a diffucult task any more. The authors also expected that some tools that automatically perform honeypot fingerpriting would be developed and distributed in underground coomunities so that it will be an easy task even for non-skilled attackers.

**McCarty** studied how a anti-honeypot tool, called Honeypot Hunter (developed by Send-Safe), worked, which was designed to help spammers detect honeypots that act as open proxy relays [21]. In testing potential honeypots, Honeypot Hunter created a local mail server and connected to a target system proxy. It then attempts to proxy from the target back to itself. If the target claimed this attempt was successful, but the mail server did not see the connection, then Honeypot Hunter knows it had likely found a honeypot. Honeypot Hunter also sent test emails through a target system and see if they were delivered. If they did not make it through, then the target was likely a honeypot.

**Mukkamala** studied detection techniques for low-interaction honeypots and honeypots running in virtula machines, which are basis for many high-interaction honeypots, by perfogming timing analysis, analyses on avilable server services, and TCP/IP fingerprinintg [54]. Timing analysis measured the response time from a server and detceted honeypots by slower response from honeypots than real production servers. Mukkamala suggested a threshold of $4.4 \times 10^{-4}$ seconds to distinguish between physical and virtual systems. Analyses on network servers tested the servers for any missing feature the servers should offer, utilizing the fact that many honeypots do not implement full features particular production servers implement or they do not fully implement responses from real servers. TCP/IP fingerprinting is similar to missing network server tests in that it detects honeypots by examing information in the TCP and IP hdeaders, utilizing the fact that the responses from honeypots and production server are not completely identical.

**Perdisci** proposed a method, called "noise injection", to defeat existing automated signature generators for worms [55], especially the one developed by Newsome [34]. Noise injection is a technique to intentionally confuse the signeture generator to prevent it from accurately recognizing the invariant bytes in polymorphic worms, by sending extra network traffic flows which are similar to the flows that contain the polymorphic worms, but without containing the invariant bytes. Perdisci demonstrated that by well crafting the noise, the signature generator proposed by Newsome failed to capture the invariant bytes in polymorphic worms even at a noise level (the ratio of noise and the invariant bytes in their counts) of 50%, while Newman initially claimed that the generator would tolerate up to 80% of noise. Perdisci's work suggested that emerging counter-attacking methods to existing honeypots and signature generations using honeypots are a major category in future honeypot research.

**Wang** studied advanced botnet designs to prepare for future attacks from botnets. In predicting the ways

 

botbnets would evolve in the future, Wang designed an advanced hybrid peer-to-peer botnet, which is harder to shut down, monitor, and hijack, compared with the current botnets [56]. The key of the proposed hybrid botnet is having two classes of bots, one called "servents" that are publicly accessible in the global Internet and the others called "clients", which are hidden from global access using private addresses or connected behind a firewall. By having some bots out of global access, the proposed botnet is designed to be hard to shut down even when some of the bots in a botnet are detected and their peer lists are obtained by security administrators. In coping with this type of botnets, the authors suggested to poison botnets by deploying a large number of honeypots, with static IP's, to saturate the peer-lists and gain as much detail as possible to shut down a botnet. Another possible solution would be submission of a large number of fake bots.

**Yegneswaran** compared the performance of four IP address shuffling methods of UBS (Uniform Block Shuffling), NBS (Non-Uniform Block Shuffling), PSS (Per Source Shuffling), and SGS (Source Group Shuffling) by mathematical analyses on shuffling interval, the size of data structure to keep track of address blocks for honeypots and production servers, number of live connections possible, and resilience against flooding DoS attacks [57]. The performance of the four methods was also measured and compared in packet delay, packet-loss rate, and connection disruption rate using experiments, which indicated no major significant difference in the four methods in the measurements.

**Zou** studied the techniques botmasters used to detect and avoid honeypots. The techniques described by the authors are based on an assumption that the honeypot administartors have liability constraint that they can not allow their honeypots to participate in attacks [58]. In one of the techniques, botmasters first send attacking messages to bots, some of which can be honeypots. The attacking messages should be the ones that command each bot to initiate some actions to a remote target host. Assuming that bots are supposed to make whatever attacking actions commanded by a botmaster, any host that fails to take part in the attacks must be a honeypot, which will be removed from the network, and thus avoided.

## VI. LEGAL/ETHICAL ISSUES ON HONEYPOTS

***Disclaimer*:** As noted in the articles that follow, there is little to no legal precedent concerning the use of honeypots. What follows is an overview of how the authors believe the laws apply in the United States. For valid legal advices, it is best to consult a lawyer.

Legal and ethical issues related to the use of honeypots and of the data collected thereby, have recently been serious concerns in the field of network administration. The essential problem is that there has not been consensus by the network community regarding what activities are considered acceptable and what are not. For example, since honeypots monitor and collect information about users, privacy issues have recently been a major concern. However, if we prohibit any monitoring activity using honeypots in order to maximize privacy, then honeypots cease to be relevant. The question is where to draw the line. Although recent discussions on this issue in the past five years have contributed to development of "quasi consensus", there are still many other issues on which we have yet to reach a consensus. This section focuses on the recent discussions of possible legal and ethical issues related to honeypots, and efforts to obtain consensus on the issues.

**Rubin** argued that researchers and security professionals should ensure that their methods are legal and ethical [59]. For example, network sniffing can be illegal under wiretapping laws, with the following exceptions: for the purpose of protecting rights and property, if the user is notified, or with specific direction from certain government agencies. Entrapment may prevent prosecution of a hacker, but it must be proved that honeypot owner went above and beyond normal practices to entice the hacker to commit a crime that they would not normally commit. Advice is also given to avoid attacking attackers; it is illegal especially if the machines that launched the attack belong to innocent users. A honeypot owner may be held liable if a honeypot's resources are used to commit crimes. Rubin also advises to be careful whenever performing research on viruses and worms: dissection, analysis, and creation of proof of concept viruses and worms involve risk of accidental release. Furthermore, effort must be made to disclose discovered vulnerabilities to a company long enough to develop patches before releasing them to the public, but not so long that the company is not put under pressure to fix the problem.

**Scottberg** discussed the ethical and legal dilemma that must be evaluated when operating a honeypot [60]. It is noted that since there is almost no legal precedent regarding the use of honeypots, most of the information presented is simply a well educated guess as to how the laws apply. Entrapment by honeypots typically only applies to law enforcement agencies. However, precautions and legal counsel should be taken if a system administrator intends to prosecute attackers. It is further stated that although privacy laws are not applicable to data stored on an illegally accessed system, the logging of data sent through a honeypot may be considered an illegal wiretap. The authors do not believe this will hold up in court given that since honeypots do not provide user accounts, they should not be bound by common carrier legislation. The last advice for a private honeypot operator is to review liability laws with their lawyer. Under existing liability laws, a honeypot operator could be prosecuted if they are found to be lacking in due diligence or even gross negligence if a compromised honeypot is used to attack other systems.

## VII. CONCLUSIONS

This paper summarizes our survey on the recent advances and the current trends in honeypot research. We recognized five major areas in the current honeypot research and summarized 60 papers published mostly

 

after year 2005 based on the five areas. The five areas are ① new types of honeypots, ② utilizing honeypot output data, ③ configuring honeypots, ④ honeypot anti-detection, and ⑤ legal issues in using honeypots.

Section II summarizes new types of honeypots, in which honeypots have been improved to cope with the recently emerging security threats. Another evolution is integration of honeypots with other security solutions, such as anomaly and intrusion detection. Table 1 summarizes the papers in this category.

| ① New types of honeypots | | |
|---|---|---|
| | Threat | Author |
| New threats | bots | Alberdi [7] |
| | denial of service | Das [11], Khattab [14] |
| | exploiting routers | Ghourabi [12], Prathapani [20] |
| | malware | Portokalidis [19], Zhuge [24] |
| | phishing | Lauinger[16], Li[17],Webb[23] |
| | attack web servers | Alosefer [8], Nazario [18] |
| | Purpose | Author |
| New concepts | trade-offing low/high hp's | Adachi [6], Bailey [10], Jiang [13], Mukkamala [54] |
| | anomaly detection | Anagostakis [9] |
| | intr. detection | Kreibichi [15] |
| | avoid hp hijacking | Jiang [13] |
| | fake honeypots | Rowe [22] |

Table 1 – Summaries of Section II

Section III focuses on advances in honeypots with regard to utilizing the data collected by honeypots. The majority of the literatures in this area focus on how to assist security administrators in investigating honeypot data for discovering both known and unknown security threats. Others discuss particular procedures to utilize honeypot output data for discovering and studying specific network security threats. Table 2 summarizes the contributions in this category.

| ② Utilizing honeypot output data | | |
|---|---|---|
| | Purpose | Author |
| Data analysis | for general threats | Dantu[27], Francois[29], Su[37], Krasser [31], Thonnard [40] |
| | for worms | Mohammed [32], Newsome[34], Tang [39], Kohlrausch [50] |
| | for bots | Yegneswaran [43], Zhao [44] |
| | for spams | Trivedi [41] |
| | Purpose | Author |
| Procedure | honeypot data contamination | Cooke [26], Fairbanks [28], Narvaez [33] |
| | forensic analysis | Chen [25], Raynal [35, 36] |
| | honeypot data integration | Hoepers [30] |

Table 2 - Summaries of Section III

Section IV focuses on the recent research regarding how to assist security administrators in setting up, customizing, and configuring honeypots for particular security administrative objectives. Table 3 summarizes the contribution in this category.

| ③ Configuring honeypots | |
|---|---|
| Purpose | Author |
| optimum deployment of distributed honeypots | Briffaut [45], Carroll [46], Wang [47] |
| dynamic deployment | Hecker [49] |
| detect SQL injection | Chen [48] |
| detect W32.Conficker | Kohlrausch [50] |
| malicious insiders | Spitzner [51] |

Table 3 - Summaries of Section IV

Section V summarizes the results of our survey on the problem of honeypot counter-detection by attackers and solutions to the problems. Table 4 summarizes the contribution in this category.

| ④ Honeypot anti-detection | |
|---|---|
| Topic | Author |
| major honeypot detection techniques | Holz [53], Mukkamala [54] |
| hijacking honeypots after their detections | Dornseif [52] |
| study of Honeypot Hunter | McCarty [53] |
| detection using bots | Wang [56], Zou [58] |
| denial of service to honeypots | Yegneswaran [57] |
| polymorphic worms to honeypots | Perdisci [55] |

Table 4 - Summaries of Section V

Section VI discusess the legal and ethical issues involved with honeypots and the data they collect. Although the number of papers published on these issues is still relatively small compared to the other issues in this survey, we considered the issues worth a section because of their importance. Table 5 lists the two papers summarized in this section.

| ⑤ Legal issues in using honeypots | |
|---|---|
| Topic | Author |
| importance of ethical and legal use of honeypots | Rubin [59] |
| ethical and legal dilemma in using honeypots | Scottberg [60] |

Table 5 - Summaries of Section VI

## REFERENCES

[1] Michael Bailey, Evan Cooke, Farnam Jahanian, Yunjing Xu, and Manish Karir, "A Survey of Botnet Technology and Defenses," Proceedings of the Cybersecurity Applications & Technology Conference for Homeland Security, March 2009, pp. 299-304.

[2] Thorsten Holz, "Learning More about Attack Patterns with Honeypots," Proceedings of Sicherheit 2006, February 2006, pp. 30-41.

[3] Christian Seifert, Ian Welch, and Peter Komisarczuk, "Taxonomy of Honeypots," CS Technical Report TR-06-12, School of Mathematics, Statistics and

Computer Science, Victoria University of Wellington, New Zealand. Available: http://www.mcs.vuw.ac.nz/comp/Publications/CS-TR-06-12 .abs.html (last accessed: August 17, 2012).

[4] Xinwen Fu, Wei Yu, Dan Cheng, Xuejun Tan, Kevin Streff, and Steve Graham, "On Recognizing Virtual Honeypots and Countermeasures," Proceedings of the IEEE International Symposium on Dependable, Autonomic and Secure Computing, September 2006, pp. 211-218.

[5] Phillip Porras and Vitaly Shmatikov, "Large-Scale Collection and Sanitization of Network Security Data: Risks and Challenges," Proceedings of the Workshop on New Security Paradigms, September 2006, pp. 57-64.

[6] Yu Adachi and Yoshihiro Oyama, "Malware Analysis System using Process-Level Virtualization," Proceedings of IEEE Symposium on Computers and Communications, July 2009, pp. 550-556.

[7] Ion Alberdi, Éric Philippe, Owezarski Vincent, and Nicomette M. Kaâniche, "Shark: Spy Honeypot with Advanced Redirection Kit," Proceedings of the IEEE Workshop on Monitoring, Attack Detection and Mitigation, November 2007. Available: http://spiderman-2.laas.fr/ METROSEC/monam.pdf (last accessed: August 17, 2012).

[8] Yaser Alosefer and Omer Rana, "Honeyware - Web-based Low Interaction Client Honeypot," Proceedings of the International Conference on Software Testing, Verification, and Validation Workshops, April 2010, pp. 410-417.

[9] K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, and A. D. Keromytis "Detecting Targeted Attacks Using Shadow Honeypots," Proceedings of the Conference on USENIX Security Symposium, August 2005, pp. 9-23.

[10] Michael D. Bailey, Evan Cooke, Farnam Jahanian, Niels Provos, Karl Rosaen, and David Watson, "Data Reduction for the Scalable Automated Analysis of Distributed Darknet Traffic," Proceedings of the ACM SIGCOMM Conference on Internet Measurement, October 2005, pp. 239-252.

[11] Vinu V. Das, "Honeypot Scheme for Distributed Denial-of-Service," Proceedings of the 2009 International Conference on Advanced Computer Control, January 2009, pp. 497-501.

[12] Abdallah Ghourabi, Tarek Abbes, and Adel Bouhoula, "Honeypot Router for Routing Protocols Protection," Proceedings of the International Conference on Risks and Security of Internet and Systems, October 2009, pp. 127-130.

[13] Xuxian Jiang and Xinyuan Wang, "Out-of-the-Box Monitoring of VM-Based High-Interaction Honeypots," Proceedings of the International Conference on Recent Advances in Intrusion Detection, September 2007, pp. 198-218.

[14] Sherif M. Khattab, Chatree Sangpachatanaruk, Daniel Moss, Rami Melhem, and Taieb Znati, "Roaming Honeypots for Mitigating Service-Level Denial-of-Service Attacks," Proceedings of the International Conference on Distributed Computing Systems, March 2004, pp. 328–337.

[15] Christian Kreibichi and Jon Crowcroft, "Honeycomb – Creating Intrusion Detection Signatures Using Honeypots," ACM SIGCOMM Computer Communication Review, vol. 34, no. 1, January 2004, pp. 51-56.

[16] Tobias Lauinger, Veikko Pankakoski, Davide Balzarotti, and Engin Kirda, "Honeybot, Your Man in the Middle for Automated Social Engineering," Proceedings of USENIX Symposium on Networked Systems Design and Implementation, April 2010. Available: http://portal.acm. org/citation.cfm?id=1855697.

[17] Shujun Li and Roland Schmitz, "A Novel Anti-Phishing Framework Based on Honeypots," Proceedings of eCrime Researchers Summit, October 2009, pp. 1-13.

[18] Jose Nazario, "PhoneyC: A Virtual Client Honeypot," Proceedings of USENIX Workshop on Large-Scale and Emergent Threats, April 2009, pp. 1-8.

[19] Georgios Portokalidis, Asia Slowinska, and Herbert Bos, "Argos: an Emulator for Fingerprinting Zero-Day Attacks," ACM SIGOPS Operating Systems Review, vol. 40, no. 4, October 2006, pp. 15-27.

[20] Anoosha Prathapani, Lakshmi Santhanam, and Dharma P Agrawal, "Intelligent Honeypot Agent for Blackhole Attack Detection in Wireless Mesh Networks," Proceedings of IEEE International Conference on Mobile Adhoc and Sensor Systems, October 2009, pp. 753-758.

[21] Bill McCarty, "Anti-Honeypot Technology," IEEE Security & Privacy, vol. 2, no. 1, January 2004, pp. 76-79.

[22] Neil C. Rowe, E. John Custy, Binh T. Duong, "Defending Cyberspace with Fake Honeypots," Journal of Computers, vol. 2, no. 2, April 2007, pp. 25-36.

[23] Steve Webb, James Caverlee, and Calton Pu, "Social Honeypots: Making Friends with a Spammer Near You," Proceedings of the Conference on Email and Anti-Spam, August 2008. Available: http://citeseerx.ist.psu.edu/ viewdoc/summary?doi=10.1.1.150.588 (last accessed: August 17, 2012).

[24] Jianwei Zhuge, Thorsten Holz, Xinhui Han, and Wei Zou, "Collecting Autonomous Spreading Malware using High-Interaction Honeypots," Proceedings of the International Conference on Information and Communications Security, December 2007, pp. 438-451.

[25] Lin Chen, Zhitang Li, Cuixia Gao, and Lan Liu, "Dynamic Forensics based on Intrusion Tolerance," Proceedings of IEEE International Symposium on Parallel and Distributed Processing with Applications, August 2009, pp. 469-473.

[26] Evan Cooke, Farnam Jahanian, and Danny McPherson, "The Zombie Roundup: Understanding,

 

Detecting, and Disrupting Botnets," Proceedings of the USENIX Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop, July 2005, pp. 39-44.

[27] Ram Dantu, Joao W. Cangussu, and Sudeep Patwardhan, "Fast Worm Containment Using Feedback Control," IEEE Transactions on Dependable and Secure Computing, vol. 4, no. 2, April-June 2007, pp. 119-136.

[28] Kevin D. Fairbanks, Ying H. Xia, and Henry L. Owen III, "A Method for Historical Ext3 Inode to Filename Translation on Honeypots," Proceedings of the IEEE International Computer Software and Applications Conference, July 2009, pp. 392-397.

[29] Jérôme Francois, Radu State, and Olivier Festor, "Activity Monitoring for Large Honeypots and Network Telescopes," International Journal on Advances in Systems and Measurements, vol. 1, no. 1, 2008, pp. 1-13.

[30] Cristine Hoepers, Nandamudi L. Vijaykumar, and Antonio Montes, "HIDEF: a data Exchange Format for Information Collected in Honeypots and Honeynets," INFOCOMP Journal of Computer Science, vol. 7, no. 1, March 2008, pp. 87-96.

[31] Sven Krasser, Gregory Conti, Julian Grizzard, Jeff Gribschaw, and Henry Owen, "Real-Time and Forensics Network Data Analysis using Animated and Coordinated Visualization," Proceedings of IEEE International Conference on Systems, Man and Cybernetics Information Assurance Workshop, August 2005, pp. 42-49.

[32] Mohssen M. Z. E. Mohammed, H. Anthony Chan, Neco Ventura, Mohsim Hashim, Izzeldin Amin, and Eihab Bashier, "Detection of Zero-Day Polymorphic Worms Using Principal Component Analysis," Proceedings of International Conference on Networking and Services, March 2007, pp. 277-281.

[33] Julia Narvaez, Chiraag Aval, Barbara Endicott-Popovsky, Christian Seifert, Ashish Malviya, and Doug Nordwall, "Assessment of Virtualization as a Sensor Technique," Proceedings of the IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering, May 2010, pp. 61-65.

[34] James Newsome, Brad Karp, and Dawn Song, "Polygraph: Automatically Generating Signatures for Polymorphic Worms," Proceedings of IEEE Symposium on Security and Privacy, May 2005, pp. 226-241.

[35] Frederic Raynal, Yann Berthier, Philippe Biondi, and Danielle Kaminsky "Honeypot Forensics Part I: Analyzing the Network," IEEE Security and Privacy, vol. 2, no. 4, July 2004, pp. 72-78.

[36] Frederic Raynal, Yann Berthier, Philippe Biondi, and Danielle Kaminsky "Honeypot Forensics Part II: Analyzing the Compromised Host," IEEE Security and Privacy, vol. 2, no. 5, September 2004, pp. 77-80.

[37] Ming-Yang Su, "Internet Worms Identification through Serial Episodes Mining," Proceedings of the International Conference on Electrical Engineering /Electronics Computer Telecommunications and Information, May 2010, pp. 132-136.

[38] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo, "Discovery of Frequent Episodes in Event Sequences," Data Mining and Knowledge Discovery, vol. 1, no. 3, 1997, pp. 259-289.

[39] Yong Tang and Shigang Chen, "Defending against Internet Worms: a Signature-based Approach," Proceedings of IEEE INFOCOM, vol. 2, March 2005, pp. 1384-1394.

[40] Oliver Thonnard and Marc Dadier, "A Framework for Attack Pattern's Discovery in Honeynet Data," Digital Investigation, vol. 5, no. 1, September 2008, pp. 128-139.

[41] Aarjav J. Trivedi, Paul Q. Judge, and Sven Krasser, "Analyzing Network and Content Characteristics of Spim using Honeypots," Proceedings of the USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet, June 18, 2007, pp.1-9.

[42] Gérard Wagener, Alexandre Dulaunoy, and Thomas Engel, "Towards an Estimation of the Accuracy of TCP Reassembly in Network Forensics," Proceedings of the International Conference on Future Generation Communication and Networking, vol. 2, December 2008, pp. 273-278.

[43] Vinod Yegneswaran, Paul Barford, and Vern Paxon, "Using Honeynets for Internet Situational Awareness," Proceedings of the ACM/USENIX Workshop on Hot Topics in Networks, November 2005, pp. 240-243.

[44] Narisa Zhao and Xianfeng Zhang, "The Worm Propagation Model and Control Strategy Based on Distributed Honeynet," Proceedings of the International Conference on Computer Science and Software Engineering, vol. 3, December 2008, pp. 868-871.

[45] Jérémy Briffaut, Jean-François Lalande, and Christian Toinard, "Security and Results of a Large-Scale High-Interaction Honeypot," Journal of Computers Special Issue on Security and High Performance Computer Systems, vol. 4, no. 5, May 2009, pp. 395-404.

[46] Thomas E. Carroll and Daniel Grosu, "A Game Theoretic Investigation of Deception in Network Security," Proceedings of the International Conference on Computer Communications and Networks, September 2009, pp. 1-6.

[47] Haifeng Wang and Qingkui Chen, "Design of Cooperative Deployment in Distributed Honeynet System," Proceedings of the International Conference on Computer Supported Cooperative Work in Design, April 2010, pp. 711–716.

[48] Thomas M. Chen and John Buford, "Design Considerations for a Honeypot for SQL Injection Attacks," Proceedings of IEEE Local Computer Networks, October 2009, pp. 915-921.

[49] Christopher Hecker, Kara L. Nance, and Brian Hay, "Dynamic Honeypot Construction," Proceedings

 

of the Colloquium for Information Systems Security Education, June 2006, pp. 95-102.
[50] Jan Kohlrausch, "Experiences with the NoAH Honeynet Testbed to Detect new Internet Worms," Proceedings of the International Conference on IT Security Incident Management and IT Forensics, September 2009, pp. 13-26.
[51] Lance Spitzner, "Honeypots: Catching the Insider Threat," Proceedings of the Computer Security Applications Conference, December 2003, pp. 170-179.
[52] Maximillian Dornseif, Thorsten Holz, and Christian N. Klein, "NoSEBrEaK - Attacking Honeynets," Proceedings of IEEE International Conference on Systems, Man and Cybernetics Information Assurance Workshop, June 2004, pp. 123-129.
[53] Thorsten Holz and Frederic Raynal, "Detecting Honeypots and Other Suspicious Environments," Proceedings of IEEE International Conference on Systems, Man and Cybernetics Information Assurance Workshop, June 2005, pp. 29-36.
[54] S. Mukkamala, K. Yendrapalli, R. Basnet, M. K. Shankarapani, and A. H. Sung, "Detection of Virtual Environments and Low Interaction Honeypots," Proceedings of IEEE International Conference on Systems, Man and Cybernetics Information Assurance Workshop, June 2007, pp. 92-98.
[55] Roberto Perdisci, David Dagon, Wenke Lee, Prahlad Fogla, and Monirul Sharif, "Misleading Worm Signature Generators Using Deliberate Noise Injection," Proceedings of IEEE Symposium on Security and Privacy, May 2006, pp. 15-31.
[56] Ping Wang, Sherri Sparks, and Cliff C. Zou, "An Advanced Hybrid Peer-to-Peer Botnet," IEEE Transaction on Dependable and Secure Computing, vol. 7, no. 2, April-June 2010, pp. 113-127.
[57] Vinod Yegneswaran, Chris Alfred, Paul Barford, and Jin-Yi Cai, "Camouflaging Honeynets," Proceedings of IEEE Global Internet Symposium, May 2007, pp. 49-54.
[58] Cliff C. Zou and Ryan Cunningham, "Honeypot-Aware Advanced Botnet Construction and Maintenance," Proceedings of the International Conference on Dependable Systems and Networks, June 2006, pp. 199-208.
[59] Bradley S. Rubin and Donald Cheung, "Computer Security Education and Research: Handle with Care," IEEE Security & Privacy, vol. 4, no. 6, November-December 2006, pp. 56-59.
[60] Brian Scottberg, William Yurcik, and David Doss, "Internet Honeypots: Protection or Entrapment?," Proceedings of International Symposium on Technology and Society, August 2002, pp. 387-391.

**Mr. Matthew L. Bringer** is currently a graduate student in the Department of Computer Science at Southern Illinois University Edwardsville, Illinois, since September 2009. His research areas include theory and techniques in network security for e-commerce. Mr. Bringer is currently working for his Master's degree at SIUE.

**Mr. Christopher A. Chelmecki** is currently a graduate student in the Department of Computer Science at Southern Illinois University Edwardsville, Illinois, since September 2009. He participated in multiple research projects related to network security in the department and is a co-author for multiple academic papers with Dr. Fujinoki. His research areas are in the area of network security for e-commerce.

**Dr. Hiroshi Fujinoki** received the PhD degree from Department of Computer Science, University of South Florida, Tampa, FL, US in August, 2001. Currently he is working as an associate professor at the Department of Computer Science, Southern Illinois University Edwardsville. His research interests include routing in computer networks, performance analysis in network protocols, and network security. He was a member of KML (Knowledge Management Lab), which was a technical advisory committee for US Transportation Command at Scott AFB, IL.

 