Detecting Targeted Attacks by Multilayer Deception

Wei Wang, Jeffrey Bickford, Ilona Murynets, Ramesh Subbaraman, Andrea G. Forte and Gokul Singaraju

AT&T Security Research Center, New York, USA;
e-mail: {wei.wang.2, jbickford, ilona, ramesh.subbaraman, forte, gokul.singaraju}@att.com

Received 15 May 2013; Accepted 15 July 2013

Abstract

Over the past few years, enterprises are facing a growing number of highly customized and targeted attacks that use sophisticated techniques and seek after important company assets, such as customer data and intellectual property. Unlike conventional attacks, targeted attacks are operated by experts who use multiple steps to gain access to sensitive assets, and most of time, leave very few network traces behind for detection. In this paper, we propose a multi-layer deception system that provides an in depth defense against such sophisticated targeted attacks. Specifically, based on previous knowledge and patterns of such attacks, we model the attacker as trying to compromising an enterprise network via multiple stages of penetration and propose defenses at each of these layers using deception based detection. Due to multiple layers of deception, the probability of detecting such an attack will be greatly enhanced. We present a proof of concept implementation of one of the key deception methods proposed. Due to various financial constraints of an enterprise, we also model the design of the deception system as an optimization problem in order to minimize the total expected loss due to system deployment and asset compromise. We find that there is an optimal solution to deploy deception entities, and even over spending budget on more entities will only increase the total expected loss to the enterprise. Such a system

Journal of Cyber Security and Mobility, Vol. 2, 175–199.
doi 10.13052/jcsm2245-1439.224

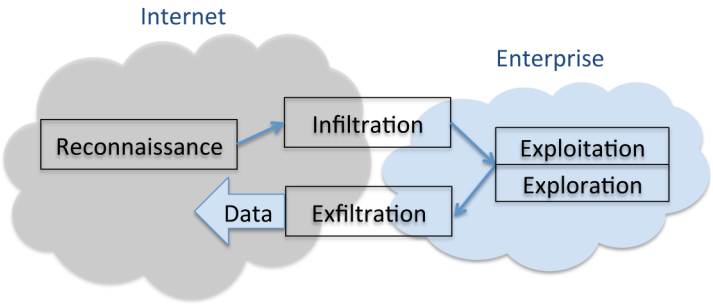

Figure 1 A multi-stage attack with layers of penetration.

can be coupled with existing detection techniques to protect enterprises from sophisticated attacks.

Keywords: Deception, honeypot, honeynet, optimization.

1 Introduction

Recent trends indicate that enterprises today face a growing threat of sophisticated attackers who seek to steal or compromise proprietary information and assets [19, 11, 2, 1]. These attacks are executed in multiple stages and each stage is highly customized for each targeted enterprise. Attackers typically leave few network and system level footprints in attempts to evade detection. They exploit zero-day vulnerabilities to deliver malware using a single event, such as a carefully crafted spear-phishing email, as the entry point into an enterprise network. Detecting such events by leveraging existing techniques is difficult, especially when social engineering techniques are used to infiltrate the target organization. Once an attacker has infiltrated an organization, detecting other phases of an attack, such as data exfiltration, is a very difficult process which requires correlation between multiple events, such as firewall, IDS and DNS logs. As attackers customize their attacks for individual targets, prior knowledge such as blacklists of malicious domain names, drop servers IPs and malware signatures may not be useful for the attack. Therefore, rapid detection is technically difficult and stopping an attack in the early stage is challenging.

Figure 1 illustrates a multi-stage attack with layers of penetration as an attack example of such. A typical pattern observed in these attacks is that an attacker first studies the targeted enterprise during a “reconnaissance”

phase, gathering information such as the organization background, resources and individual employees to initially target to launch the attack. By using social engineering techniques, such as a spear-phishing email, the attacker attempts to "infiltrate" into the enterprise by using a particular employee as the entry point. This typically requires an employee to fall victim to the social engineering attack, for example by following a web link or opening an attachment that contains some exploit and malicious payload. During this phase of "exploitation", the attacker penetrates a level deeper by gaining control of the employee's personal assets (such as email and personal computer). This may then be used to penetrate another level deeper into the enterprise through manual "exploration" of remote servers (hosting databases, proprietary algorithms, intellectual properties etc.), or to launch additional social engineering attacks against other employees who have access to the information that the attacker seeks to obtain. Some attacks may exploit and gain control of many different servers and machines during the exploration phase to gain a persistent foothold in the enterprise. Once an asset has been obtained, the attacker finally "exfiltrates" the data out of the enterprise network and the attack can be considered successful.

This pattern, as mentioned above, reveals that there are three layers of penetration – a human layer, a local asset layer, and a global asset layer. Each layer of penetration brings the attacker closer to the targeted information assets. The human layer is the information of an enterprise employee, which is researched and gained by an attacker in the reconnaissance activity. The local asset layer refers to the employee's local machine containing immediate assets and the global asset layer represents the assets hosted on servers accessed and shared by multiple users. To address this problem, we propose to apply a multilayer deception system in order to protect important assets within an enterprise network. That is, our proposed multilayer deception system provides *detection via deception* at these three layers. By detection via deception, we are referring to the idea of placing bogus facilities and resources within a network or file system such that when accessed, an alarm is triggered and the attackers presence is discovered. In order to trick an attacker to access the item, these bogus resources are generated to appear as valuable as normal. By providing deception at each stage of an attack, the proposed system greatly enhances the chance of detecting intrusions in an early stage.

In this paper, we introduce concepts of honey people, honey files, honey servers, and honey activity to defend the human, local, and global assets layers respectively. Honey activity can be considered as fake file system or network activity intended to prevent a sophisticated attacker from evading

bogus resources by observing actual user behaviors. We integrate all of these into a complete system that work cooperatively together to provide an in depth solution against targeted attacks. We also present an optimization based method to design a budget conscious defense solution which minimizes the expected loss due to asset compromise.

The following features of our work represent significant contributions towards the goal of detecting multi-stage attacks:

- Multiple layers of deception are designed specifically for each layer of penetration, so that each deception layer can increase the chance of detecting attacks at an early stage.
- By estimating the cost of deploying deception entities and asset values, proposed optimization model can find an optimal solution to intelligently allocate the budget on necessary deployment to minimize the overall expected loss. To our knowledge, this is the first proposal for an optimal system design based on the cost of deception.

The rest of the paper is organized as follows. Related work will be presented in Section 2. The details of our system are at described in Section 3. In Section 4, we describe a prototype implementation of honey files with honey activity. Section 5 presents our deception system design method using optimization and shows results for a specific example. Section 6 shows the future work to enhance the system implementation in the cloud and further study on system design optimization. Section 7 concludes our paper.

2 Related Work

This basic concept of deception has a long history of successfully being used in security. Generally a deception system consists of resources that appear to be a part of a network, but are actually isolated and monitored. The system will seem, to the intruder, to contain information or a resource of value to attackers, but such information is of no value in content. Because it is not a real entity, any actions on this resource are suspicious by nature [21, 3]. The concept of deception was firstly introduced by Clifford in his book [22], which described in detail how he hunted over months for a computer hacker who broke into a computer at the Lawrence Berkeley National Laboratory. By trapping the hacker in the decoy system, the hacker's actions were recorded and his behavior was successfully revealed and studied. By dedicating a chapter on "Traps, Lures, and Honey Pots" in their book [10], Cheswick and Bellovin discussed how to use unused services as decoys on firewalls and

during another chapter "An Evening with Berferd" they logged and interacted with a hacker in order to understand an attempted attack into their network.

The use of fake files as bait to detect malicious file accesses was first proposed in [25,26]. Such files can be set up by users and only those who are intimately familiar with file system can potentially avoid such a trap. Along the same line, Fiedler [13] proposed honeypots for database protection, where he described basic honeypot architecture to secure a SQL database server. This SQL database provides a honeypot trap for the intruder while still allowing the web application to run as normal. Alternatively, Bruce [20] proposed to embed macros in fake word or pdf documents to trigger alerts when files are opened. Younghee et al. [17] proposed a software-based decoy system that generates believable Java source code which, to an adversary, appears to be entirely valuable and proprietary. A honeynet is used to monitor a large and/or more diverse networks in which one honeypot may not be sufficient [7]. A honeynet can be utilized as a part of the network intrusion detection system. Instead of utilizing a variety of physical systems, Honeynet Project introduced virtual Honeynets to run honeypots on a single computer [18]. By using virtualization techniques, the Honeynet project runs several honeypots of multiple operating systems types on a single computer for analysis purposes. Project Honey Pot [15] employed a web based honeypot network which uses software embedded in web sites to collect information about IPs harvesting e-mail addresses for spam, bulk mailing and fraud. Most recently on the mobile communications front, Collin [16] implemented HoneyDroid, a smartphone honeypot for the Android operating system to catch attacks originating from the Internet, mobile networks, as well as through malicious applications; while Wang [24] introduced fake contacts on mobile devices to quickly detect messaging-based malware propagation in cellular networks. Brian et al. [8] proposed trap-based defense mechanisms and a deployment platform for addressing the problem of insiders attempting to exfiltrate and use sensitive information. Malek et al. [5] modelled user search patterns as well as touch interactions with decoy documents to detect deviations, indicating an attack.

Unlike all previous work that considers one flavor of honeypot as a system, we propose multiple layers of deception that work cooperatively. We focus on sophisticated and highly customized attacks against enterprises targeting their most valuable information assets via three levels of penetration. For each layer of penetration, we propose related methods of deception to detect such penetration. By considering multiple deception layers as one integrated system, we model the enterprise's expected losses when assets are

compromised, and formulate the problem of minimizing the overall expected loss using our deception system while meeting the budget constraints as an optimization problem. There is no previous work that considered such a systematic design.

3 Multilayer Deception System

3.1 Multilayer Deception System

In this section, we describe the key concepts of our proposed multilayer deception system consisting of Honey People (*HP*), Honey File with Honey Activity (*HFHA*), and Honey Servers with Honey Activity (*HSHA*). The framework can be naturally extended to more layers of deception if required, such as Honey Smartphone Contacts and Honey Databases. Figure 2 illustrates our proposed system with these mutliple layers of deception. The system is comprised of both real entities typically used by employees and deception entities used to detect an intrusion within the network. The alerts generated from accessing the deception entities will be sent to an analyst server, where an analysis process will handle the alerts. Throughout the rest of this paper we consider only three levels of deception, though in a full-fledged deployment we would rely on previous implementations of additional deception layers.

3.2 Honey People

The goal of Honey People is to protect employees against social engineering attacks coming from outside the enterprise, such as spear phishing messages containing a URL or malicious attachment. A HP is similar to the profile of a real employee but contains bogus identity and/or contact information (e.g. multiple email addresses). These HP can possibly be posted on public websites (corporate web-page and popular social network sites[1]) and on physical entities such as business cards, as shown in Figure 3. By this means, HP confuses an attacker with fake information so that when the attacker chooses a target to send a phishing message to, there is a probability that a HP is chosen. A message sent to a HP is forwarded to the analyst server to ascertain whether it was a penetration attempt. If not, the message can be forwarded to the actual recipient. Detecting phishing messages is out of scope of this paper but can be relied on existing techniques [6, 9].

[1] Subject to terms and conditions of the sites.

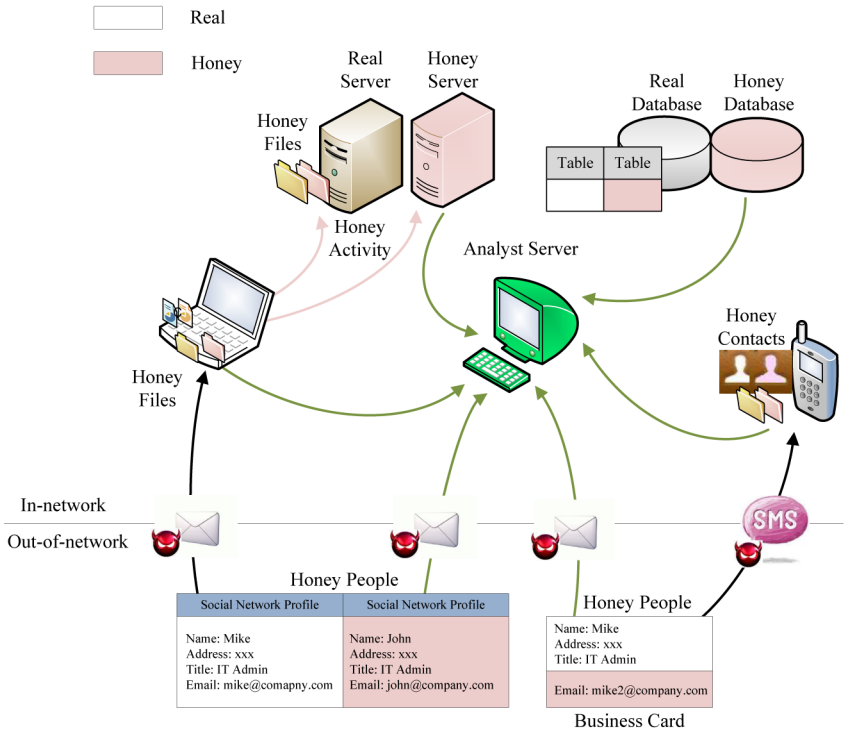

Figure 2 A multilayer deception system (H represents *honey entity* and R represents *real entity*).

The overhead at the analyst server is clearly dependent on the amount of actual emails the person receives. We believe that internally within a company, HP is not a scalable solution as employees will likely send each other a lot of emails, which is why in our system, HP is only used to protect against social engineering attempts from the outside. However, even then, for a given employee, HP may or may not be a feasible solution, or only be a partial solution depending on the employee's level of public exposure and the amount of external emails he receives. For example, a company's CEO's identity may be well known, rendering a bogus identity useless, but his email address may not, in which case HP via multiple email addresses may be feasible. On the other hand, a manager may have both a well known identity and contact address, so new clients can reach him, in which case HP is not feasible as a solution at all. In our overall system design in Section 5,

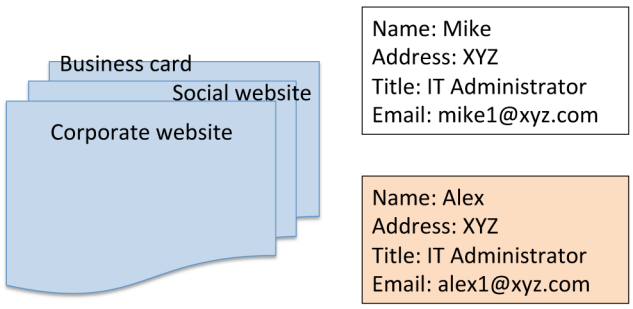

Figure 3 Honey people hosted on different publicly accessible medium. Mike is a real employee and Alex is a deception entity.

we capture this aspect of HP by having different costs for a HP solution for different employees.

3.3 Honey Files with Honey Activity

If the attacker manages to avoid HP defenses, an employee can potentially be successfully social engineered and infected with malware. After the attacker infiltrates the enterprise network and compromises a machine, he can begin the next stage of an attack, where local emails, files, and folders are explored, user name and password are sniffed, and even user daily activities are monitored. In the attempt that the attacker starts to compromise the employee's local assets, we introduce another layer of deception which utilizes honey files (HFs) with honey activity (HA) as a defense. Honey files are bogus files and folders that, to the attacker, are indistinguishable from real files and folders. Since the attacker, in our model, is extremely sophisticated, we assume he can obtain complete access to a machine and hence has the ability to monitor employee file and folder access behaviors. Thus, he can easily ignore files and folders that the employee never accesses, defeating the purpose of generic honey files. To obscure the view of the system, our honey files are augmented with local honey activity which updates file meta data (such as file size, name, date) to emulate actual employee accesses on real files.

In order to be convincingly realistic to the attacker, honey files should be created in separate, as well as the same, directories as the ones employees typically work with. Honey files should also be generated for files that contain evidence of attacks, such as log files. These files are typically modified or deleted by attackers in order to remove the evidence of their presence. Alerts

to the analyst server can be generated upon content related operations such as read, open, move, copy, and delete. An implementation of honey files with honey activity is described in Section 4.

3.4 Honey Server with Honey Activity

If the attacker is not caught while tinkering with an employee's local assets, he may then attack global assets that the employee has access to. At this layer, our system uses honey servers with honey activity to detect the attacker's presence. A honey server is a bogus remote server mirroring real remote servers in which the employee has access to. Again since we assume the attacker is sophisticated enough to observe an employee's remote access history, he can readily avoid traditional honey servers if they are never accessed. Hence, we augment user machines with remote honey activity to emulate user network behavior, such as connecting to a server with proprietary data.

Honey activity, with either honey files or honey servers, is a means to generate activities that appears to be generated by a regular user, and performs all tasks that a real user would perform. Honey activities need to be completely indistinguishable from a real user to avoid the possibility of malware distinguishing between HA and real user activities. There are two types of HA, local and remote honey activities. The local HA generates activities staying within the machine, such as creates, updates local honey files in order to make them correlated with real files. The remote HA generates activities going out of the machine which emulates user network behaviors such as connections to data servers. In the case where an attacker follows the local HA to access updated honey files, or remote HA to establish a SSH connection to a remote server, an alert will be triggered.

In order to differentiate between honey activity and an attacker's activity to a remote server, we need to define new algorithms. In particular, the honey server and honey activity module could agree on what network patterns the honey server expects to see as a result of the honey network activity. One such agreement could be the time pattern by which the honey activity module connects to the honey server. When an attacker tries to connect to the honey server at the wrong time, the honey server will be able to identify this network activity as originating from an attacker and thus trigger an alert to the analyst server.

3.5 Analyst Server

The Analyst Server is a center where alerts are received from different deception entities and analysis is applied to confirm or remove alerts. As we mentioned before, existing techniques can be applied to detect phishing emails or messages [9,6]. Since we capture the suspicious emails or messages with full content, a live sample of the malware can be potentially obtained from either the attachment or drive-by-download link. In such a way, we can analyze the malware itself. Different sophisticated detection schemes, such as behavioral analysis, automated URL browsing, and content based detection can be applied [4,12], to check whether a piece of malware is present on a web page. Once a phishing message is confirmed, a signature can be generated and applied in the network to block future delivery.

The analyst server is also a centralized place to correlate different alerts to increase the knowledge of a penetration attempt. For example, if alerts happen on honey file entities on two different machines, then it worth comparing host applications, logs and network traffic from these machines to identifying the cause of the similar anomalies. It also could be an alert that is triggered on a honey file entity and later on anoreother alert on a honey people entity. With some analysis, these two separate alerts could be correlated to the same attack campaign.

3.6 False Positives

In general, reducing the false positives is one of the most challenging tasks in deception systems. At one hand, deception entities should be indistinguishable from real entities in order to be convincingly realistic to the attacker. On the other hand, convincingly realistic honey entities may confuse legitimate users.

There are several ways to reduce false alerts triggered by legitimate operations on protected assets. Since people are creatures of various habits, normal routine activity from a user typically follows a detectable access pattern. In a case of an enterprise laptop, a user often goes to specific workspaces, use known applications and creates new files and updates known files for his own knowledge. Not so often, a user will go to an unknown folders and manipulate unknown files. Thus, one important aspect to help reduce false positives is employee awareness. Employees would be educated on the system and trained to not perform operations on unknown files, folders, and servers.

On the technical front, for honey files and folders, names can be differentiators between honey entities and real entities. When a file is registered

with the deception system, it can generate multiple honey files associated with a real file, with similar names for honey files but different identifiers. A secondary channel (such as SMS) can be utilized to deliver the identifier of the real file name. In the case of a machine being compromised, an attacker will not know the real file name unless he compromises the secondary channel at the same time. Automatic algorithms can also reduce false positives such as the time pattern described between the honey activity and honey server. A secondary channel can also be utilized to send alerts if honey entities are accessed, legitimate users can remove alerts if they operate on honey entities.

4 Implementation

Developing the multilayer deception system is ongoing work and as an initial proof of concept, we implement a system which can protect local assets using our deception approach. More specifically, the prototype focuses on protecting important files located on a user's machine. Based on a set of important documents (assets), typically located in various directories on a machine, we generate multiple *honey files* corresponding to a single asset. These files currently have the same name as the protected asset but with some identifier at the end. For example, a file called `secret.txt` will be transformed into multiple files e.g., `secret-1.txt`, `secret-2.txt`, `secret-3.txt`, etc. with `secret-2.txt` being the real file. This is currently a manual process and therefore the user must know which identifier corresponds to the actual asset. We are looking at automating the honey file generation process and maintaining a secondary channel in order for the user to identify the real asset.

When a honey file is generated, it is registered with a system level service which has the ability to monitor the file system and trigger *alert events* when a honey file is accessed. Due to the ubiquitous nature of the Windows XP operating system in enterprise corporations today, we built our prototype as a system level Windows service using C# and the .NET framework. Figure 4 shows the implementation of our system. Our *deception service* runs with administrator privileges and cannot be disabled by normal users. It is important to note that malware which exploits a vulnerability and gains administrator rights could disable our file monitoring service. If this occurs, the malware could access all honey files without triggering an alert event. To protect against these attacks, the deception service could be implemented using a hypervisor-based approach if the highest level of security is required.

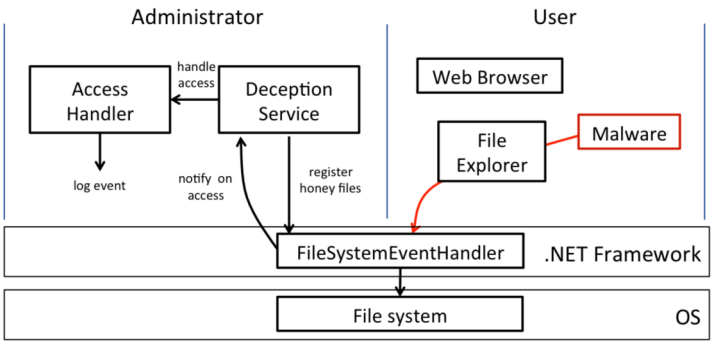

Figure 4 Kernel level deception service implementation.

The `DeceptionService` registers honey files with a `FileSystemEventHandler`, which invokes a handler function when files are accessed. When a honey file is accessed, this handler function passes the file off to an `AccessHandler` which maintains a queue of access events to process. While there are files within this queue, the `AccessHandler` thread currently logs the honey file access for later inspection. In practice, the `AccessHandler` can perform any task the multilayer deception system requires. In the full system implementation, we plan to notify the analyst server that the specified honey file was accessed. At this point, since we assume the user's machine is compromised, we can validate if the honey file was accessed accidentally or not through a secondary channel such as SMS. If the user determines that they did not intentionally access the honey file, the computer has been compromised and must be cleaned or replaced.

5 Deception System Design by Optimization

In practice, enterprises only have a finite budget to implement their security solutions. The available budget must be utilized in the most effective way possible. In this section we model the enterprise's expected losses when assets are compromised, and formulate the problem of minimizing the overall expected loss using our deception system while meeting the budget constraints as an optimization problem. Here, we focus on two deception layers which are honey people and honey files with honey activity, but the method can be easily extended to more layers.

5.1 Model

Consider an enterprise that has N_L local assets, N_G global assets, and M employees with different levels of direct access to these assets. Also, the employees are connected to each other via a social network and consequently, they have indirect access to all other employees' assets. The probability p_{ik}^{sn} represents the probability of user k being successfully compromised by the direct social connection from i to k if user i is compromised. We attempt to find the total probability Q_{ik}^{sn} introduced by all paths from i to k in the social network. Let node i be user i and the weight of a directed link from i to k be p_{ik}^{sn}. We then formulate this as a reliability problem in graph theory, where the objective is to calculate the reliability of node pairs in the graph, given every link may fail with a probability. A modified version of Don's algorithm [23] is applied to calculate Q^{sn} for all user pairs. Algorithm 2 recursively finds all successful paths from user i to k, which is plugged in the main algorithm 1 to calculate Q_{ik}^{sn}.

Data: Adjacency matrix for the social network A
Result: Q_{ik}^{sn} from user i to k
Initialize an empty set L, the max hop h_m;
while *h not exceed max hop h_m and A^h is not zero* **do**
    Find the paths between i and k by calculating A^h;
    Store paths in set L;
end
Initialize $Q_{ik}^{sn} = 0$;
Initialize an empty successful path set W;
if *L is not empty* **then**
    Find the shortest path l in L;
    Store path l in W;
    Execute algorithm SuccessPath(l, W, L);
    for *each path l in W* **do**
        $Q_{ik}^{sn} += \prod_{\forall link \in l}$ probability(link is success/fail);
    end
end

Algorithm 1: Calculate social impact Q_{ik}^{sn} from user i to k.

Suppose that the attacker attacks employee i with probability P_i, gets access to employee's asset j with probability p_{ij}^{ua} and gains access to employee k with probability p_{ik}^{sn}. Let L_{ij} be the loss if the attacker accesses asset j of employee i. Then the enterprise's expected total loss is defined as a sum of

Data: A path set l
Result: successful path set W
SuccessPath(l, W, L);
for *each link* l_k *in* l **do**
  Fail the link l_k;
  Generate a new path set L' from L by removing the failed link l_k;
  if *new path set* L' *is not empty* **then**
    Find the shortest path ll in the new path set L';
    Store the shortest path ll and failed link l_k in W;
  end
  SuccessPath(ll, W, L');
end

Algorithm 2: Function SuccessPath(l, W, L) to calculate the successful path set W.

direct and indirect losses, $T_{AL}(i) = D_{AL}(i) + I_{AL}(i)$, where the direct loss is represented as

$$D_{AL}(i) = P_i \left(\sum_{j=1}^{N_L} p_{ij}^{ua} L_{ij} + \sum_{j=1}^{N_G} p_{ij}^{ua} L_{ij} \right) \tag{1}$$

and the indirect loss introduced by social connections is

$$I_{AL}(i) = P_i \sum_{k=1,k\neq i}^{M} \left(\sum_{j=1}^{N_L} Q_{ik}^{sn} p_{kj}^{ua} L_{kj} + \sum_{j=1}^{N_G} Q_{ik}^{sn} p_{kj}^{ua} L_{kj} \right). \tag{2}$$

Q_{ik}^{sn} is the total probability of an attacker to compromise user k via all social paths from user i to user k.

As described in the early section, if a honey entity is attacked in an attempt to reach a real entity, an alert is raised revealing the attacker's presence. Thus, honey entities reduce the probability of an attacker compromising a real entity. We also note that social engineering could be one of the ways that an attacker can infiltrate an enterprise network. Thus, given that user i is protected by x_i^u (integer) number of honey people, the probability that the attacker compromises user i reduces to $P_i^H(x_i^u) = P_i(\alpha_1 f(x_i^u) + \alpha_2)$, where α_1 is the probability of being attacked via social engineering and α_2 by other ways ($\alpha_1 + \alpha_2 = 1$). Similarly if local asset j of a user i is protected by x_{ij}^a (integer) number of honey files with honey activity, the probability of asset j being compromised reduces to $p_{ij}^{uaH}(x_{ij}^a) = p_{ij}^{ua} g(x_{ij}^a)$, and for global assets

we have $p_{ij}^{uaH}(x_j^a) = p_{ij}^{ua} g(x_j^a)$. Functions $f()$ and $g()$ should be decreasing functions of their arguments to make sure that probabilities of compromise decrease when when entities are protected by honey people and honey files. Since the attacker is equally likely to pick any of the real corresponding honey people/files, we assume that $f(z) = g(z) = 1/(z+1)$ hereafter.

We would like to find the optimal allocation of HP(s) and HF(s) with HA that minimize the total loss in case of an attack. Therefore, the optimization problem can be formulated as:

$$\begin{aligned}
&\min \sum_{i=1}^{M} T_{AL}(i) = \\
&\min_{x_{ij}^a, x_i^u, x_j^a} \sum_{i=1}^{M} \left[P_i^H(x_i^u) \left(\sum_{j=1}^{N_L} p_{ij}^{uaH}(x_{ij}^a) L_{ij} + \sum_{j=1}^{N_G} p_{ij}^{uaH}(x_j^a) L_{ij} \right) \right. \\
&\left. + P_i^H(x_i^u) \sum_{k=1, k\neq i}^{M} Q_{ik}^{sn} \left(\sum_{j=1}^{N} p_{kj}^{uaH}(x_{kj}^a) L_{kj} + \sum_{j=1}^{N} p_{kj}^{uaH}(x_j^a) L_{kj} \right) \right]
\end{aligned} \tag{3}$$

subject to

$$\sum_{i=1}^{M} C_i^u x_i^u + \sum_{i=1}^{M} \sum_{j=1}^{N_L} C_{ij}^a x_{ij}^a + \sum_{j=1}^{N_G} C_j^a x_j^a \leq B$$

C_i^u is the cost of deploying a single HP on user i, C_{ij}^a is the cost of deploying HF on local asset j corresponding to user i's direct access to j, C_j^a is the cost of deploying a global honey asset and B is a budget on implementation of the deception system. Due to implementation restrictions, the variables, x_i^u, x_{ij}^a, and x_j^a, may need to satisfy additional constraints:

$$x_i^u \in \{0 \dots S_i^u\}, \quad x_{ij}^a \in \{0 \dots S_{ij}^a\}, \quad x_j^a \in \{0 \dots S_j^a\} \quad \forall i, j$$

where S_i^u, S_{ij}^a, and S_j^a are the (integer) upper bounds on the number of HPs and HFs respectively.

5.2 Example

The following example demonstrates how the optimization model works. Assume a data analytics company have ten employees ($M = 10$), that is CEO, manager, system administrator (SysAd) and seven data analysts (DAs) as in

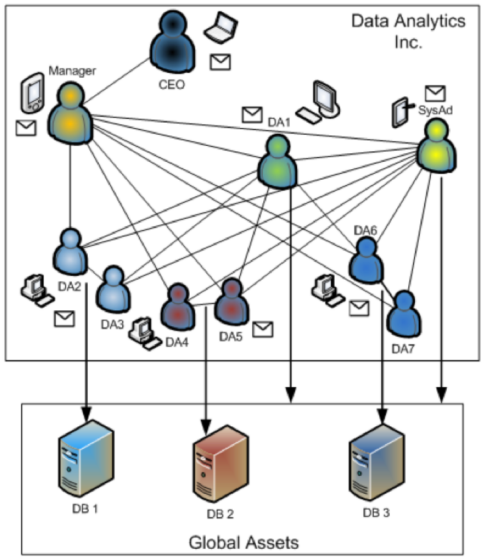

Figure 5 Local and global assets with social network connections.

Figure 5. Each employee has two local assets ($N_L = 2$): a personal computer and e-mail account. The company also has three global assets ($N_G = 3$) which are databases with different levels of access for different employees. The database 1 (DB 1) is more valuable than database 2 (DB 2), which in turn is more valuable than database 3 (DB 3). The employees have varying levels of (direct) influence on other employees in the social network. The employees and their (direct) access and (direct) influence profiles are described below.

- *CEO* has high value local assets; no access to global assets and among employees communicates only with the manager.
- *Manager* has moderate value local assets; no access to global assets; high social influence on other employees; and is highly influenced by the CEO, System Administrator and Data Analyst 1, but not others.
- *System Administrator* has low value local assets; moderate access to all global assets; high social influence on other employees, except CEO; and moderately influenced by the Manager, but not others.
- *Data analyst 1* has low value local assets; moderate access to all global assets; high social influence on the Manager and some the other Data Analysts; is highly influenced by the Manager and System

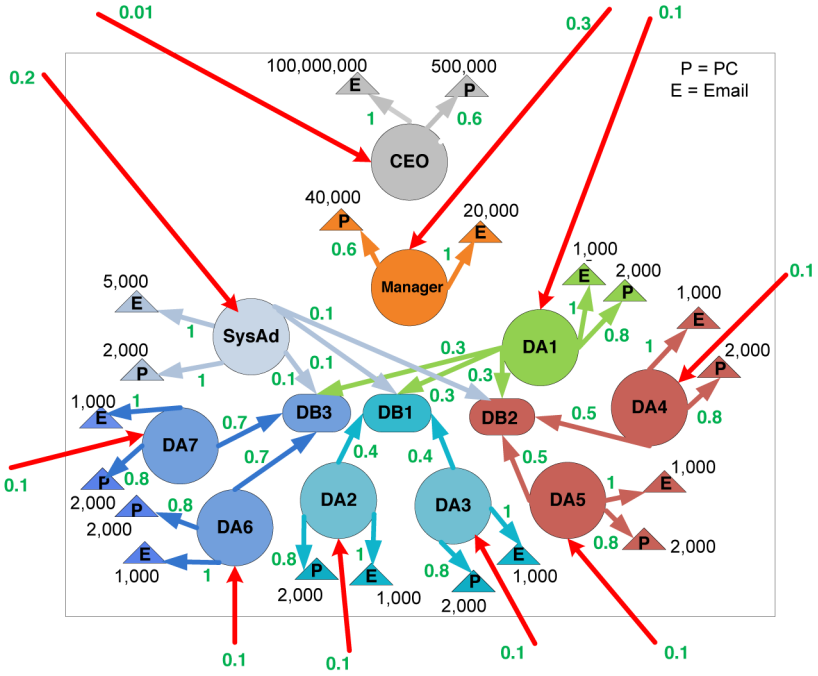

Figure 6 Local and global asset values with access probability p_{ij}^{ua}.

Administrator; and moderately influenced by some of the other Data Analysts.

- *Data analysts 2-7* have low value local assets; high access to some global assets; high influence over some employees and highly influenced by some employees.

Tables 1(a)–(f) show losses that an attacker causes by compromising each of the company employees, social influence of employees by their co-workers, attack probabilities and costs of honey people and honey files. Figures 6 and 7 represent the graph view of the parameters and values in the tables. The probability of attacks from social engineering is $\alpha_1 = 0.25$. Here, honey emails accounts and honey databases can be implemented in the system using honey files.

The optimal solution of the problem was found using Mathematica's Differential Evolution algorithm [14] for different budgets. This algorithm is a stochastic optimization method that minimizes an objective function by modelling the problem's objectives while incorporating constraints. Simil-

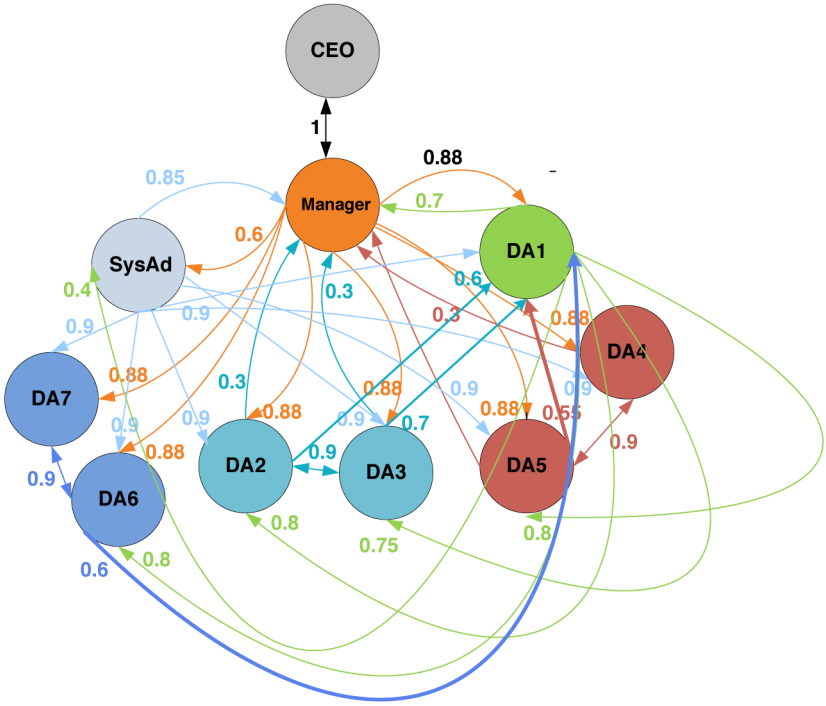

Figure 7 Social influence probability p_{ik}^{sn}.

arly to genetic algorithms, Differential Evolution algorithm is a population based algorithm using crossover, mutation and selection operators. The main steps of the Differential Evolution algorithm are initialization and evaluation followed by recursively repeating mutation, recombination, evaluation and selection steps until a termination criteria are met.

For practical reason, we limit the maximum number of honey email accounts and HP to be five, which are two conditions applied to Equation 3. There are several observations that we learn from the results.

1. If the company does not implement the deception system ($B = 0$), the expected total loss in case of attack is \$122 million. If there is a certain amount of the budget available, for example, the budget of the company on implementation of honey entities is $B = \$500{,}000$, the deception system decreases the expected loss from \$122 million to \$10 million. The optimal allocation of HP is shown in Table 2(a). This solution has the following explanation: the CEO, System Administrator and Manager

Table 1 (a) Losses in million dollars from employee compromise; (b) Probabilities that an asset is compromised; (c) Matrix of (direct) social influence; (d) Attack probabilities; (e) Cost of HP (f) Cost of HF.

	E-mail	PC	DB 1	DB 2	DB 3
CEO	10	5	0	0	0
Manager	2	4	0	0	0
Sys Admin	0.5	0.2	30	15	10
DA1	0.1	0.2	15	7. 5	5
DA 2	0.1	0.2	30	0	0
DA 3	0.1	0.2	30	0	0
DA 4	0.1	0.2	0	15	0
DA 5	0.1	0.2	0	15	0
DA 6	0.1	2.2	0	0	10
DA 7	0.1	0.2	0	0	10

(a)

	E-mail	PC	DB 1	DB 2	DB 3
CEO	1	0.6	0	0	0
Manager	1	0.6	0	0	0
Sys Admin	1	1	0.1	0.1	0.1
DA 1	1	0.8	0.3	0.3	0.3
DA 2	1	0.8	0.4	0	0
DA 3	1	0.8	0.4	0	0
DA 4	1	0.8	0	0.5	0
DA 5	1	0.8	0	0.5	0
DA 6	1	0.8	0	0	0.7
DA 7	1	0.8	0	0	0.7

(b)

	CEO	Manager	Sys Admin	DA 1	DA 2	DA 3	DA 4	DA 5	DA 6	DA 7
CEO	0	1	0	0	0	0	0	0	0	0
Manager	1	0	0.6	0.88	0.88	0.88	0.88	0.88	0.88	0.88
Sys Admin	0	0.85	0	0.91	0.9	0.9	0.9	0.9	0.9	0.9
DA 1	0	0.7	0.4	0	0.8	0.75	0	0.8	0.8	0
DA 2	0	0.3	0.4	0.6	0	0.9	0	0	0	0
DA 3	0	0.3	0.4	0.7	0.9	0	0	0	0	0
DA 4	0	0.3	0.4	0	0	0	0	0.9	0	0
DA 5	0	0.3	0.4	0.55	0	0	0.9	0	0	0
DA 6	0	0.3	0.4	0.6	0	0	0	0	0	0.9
DA 7	0	0.3	0.4	0	0	0	0	0	0.9	0

(c)

	CEO	Manager	Sys Admin	DA 1	DA 2	DA 3	DA 4	DA 5	DA 6	DA 7
P_i	0.25	0.35	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1

(d)

	CEO	Manager	Sys Admin	DA 1	DA 2	DA 3	DA 4	DA 5	DA 6	DA 7
C_i^u	2 500	2 500	750	750	750	750	750	750	750	750

(e)

	E-mail	PC	DB 1	DB 2	DB 3
C_j^a	1 000	2 000	10 000	10 000	10 000

(f)

have high probabilities of being attacked. That is why they are highly protected by HP. The CEO and Manager's e-mail account and PC are highly protected, since they have very a high value for the company and could be attacked. All the databases are protected by HFs.

2. If the company has more budget to implement more entities, the expected losses will decrease as seen in Figure 8(a). But if the company aims at minimizing the total expected expenses, defined as the cost of implementation of the honey system as well as the expected losses in case of attacks, the more protection does not mean less loss. If the budget of the company on honey entities is $B = \$1{,}800{,}000$, the expected loss drops to

Table 2 Optimal solution for (a) $1,500,000 budget and (b) $1,800,000 budget.

	HP	HF_{Email}	HF_{PC}
CEO	2	5	9
Manager	4	5	8
Sys Admin	1	4	3
DA 1	5	2	1
DA 2	3	2	3
DA 3	4	2	3
DA 4	5	1	3
DA 5	5	0	4
DA 6	2	1	1
DA 7	5	1	3

HF_{DB1}	HF_{DB2}	HF_{DB3}
14	12	8

(a)

	HP	HF_{Email}	HF_{PC}
CEO	5	5	21
Manager	4	5	23
Sys Admin	3	5	12
DA 1	3	5	24
DA 2	5	5	4
DA 3	5	5	7
DA 4	5	5	2
DA 5	5	5	10
DA 6	5	3	5
DA7	5	4	9

HF_{DB1}	HF_{DB2}	HF_{DB3}
56	38	44

(b)

$4.5 million. In fact, this budget minimizes the total expected expenses. This optimal solution is shown in Table2(b). This budget protects local assets and global databases even more, with highest level of protection on email accounts for all employees.

3. For the budget exceeding the optimal number $1,800,000, the total expected expenses increase which means over deploying honey entities is not necessary, since improvements in the expected loss will not cover the additional budget spent as shown by Figure 8(b).

6 Future Work

As enterprises move their assets into cloud infrastructure, the ability to deploy and manage a multi-layer deception approach becomes much more realistic. We are currently looking at implementing a multi-layer deception approach within a cloud-based enterprise network. The use of Virtual Desktop Infrastructure (VDI) to access company assets, hosted on some managed shared storage device, allows the possibility of implementing honey files from outside of the potentially infected desktop machine. In this case, honey files can be created and monitored from the shared storage device itself or using virtual machine introspection techniques to ensure that an attacker does not subvert this layer of deception. This also leads to a single location for honey file management instead of managing every individual employee machine located within the enterprise.

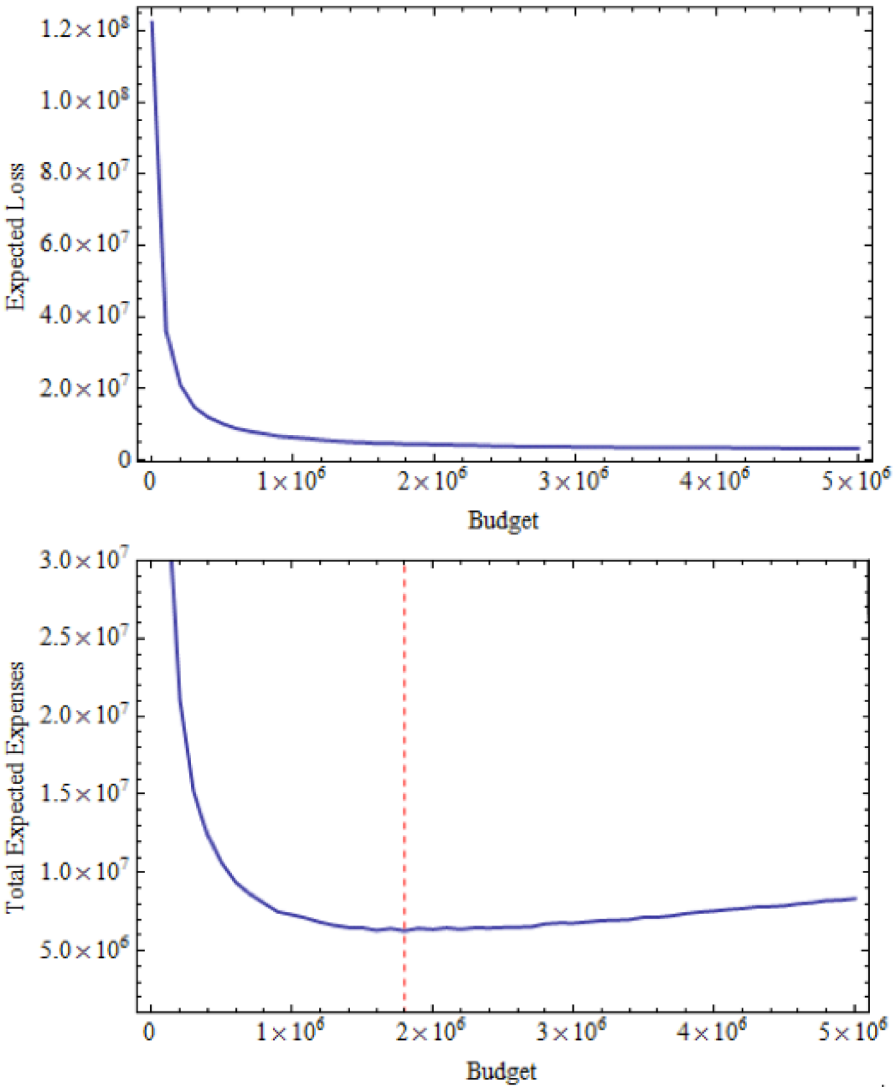

Figure 8 (a) Expected loss as a function of budget (in dollar units), (b) Expected total expenses as a function of budget (in dollar unit). The vertical line represents the optimal budget.

Within an enterprise cloud, honey servers can be easily implemented using cloned virtual machines of servers hosting ligitimate services. In this case, if an attacker scans the internal network for vulnerable servers, the honey server would automatically have the same services and fingerprint of the legitimate server. To the employees of the enterprise, these cloned servers may have slightly different domain names such that the user never travels to them.

In this type of implementation, the ease of spawning honey entities results in a tunable meter for honey entity generation. For example, during normal operation there may be a one to one mapping of honey servers throughout the important server assets within the enterprise network. If at some point in time one of the layers of deception is triggered, for a window of time after a potential compromise, an increased number of honey entities could be easily spawned throughout the other layers of deception in order to increase the probability of catching the attacker. The cost of this approach and its impact on employee productivity must be studied further during our future work.

Another piece of the future work is to better estimate the cost of deploying honey entities and values of each asset within the enterprise network. A survey will be planed to estimate these values, and we will run the optimization algorithm on a real world case to help validate our optimization result in this paper.

7 Conclusion

In this paper, we propose a multilayer deception system that provides in depth defense against sophisticated attacks. We propose defenses at each layer that an attacker may target, via deception based detection. The fact that multiple layers of deception are applied, the probability of detecting the presence of an attacker early is greatly enhanced. Furthermore, a mathematical optimization model is utilized to decide what deception entities should be deployed on which assets to minimize the total expected loss if being attacked, given a limited budget.

As future work, we plan to focus on system implementation in the cloud environment, in particular, we will study interactions between honey activities and honey servers. Also, we will implement a secondary channel to identify real file names and notify user to confirm alerts. In the optimization work, we will study more scenarios with different assumptions on the targeted assets.

References

[1] www.wired.com/threatlevel/2011/06/citi-credit-card-breach, 2011.

[2] Night dragon. blog.industrialdefender.com/?p=725, 2011.

[3] Edward G. Amoroso. Cyber Attacks: Protecting National Infrastructure. Elsevier Science, 2010.

[4] Ulrich Bayer, Imam Habibi, Davide Balzarotti, Engin Kirda, and Christopher Kruegel. A view on current malware behaviors. In Proceedings of the 2nd USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More, LEET'09, pages 8–8, Berkeley, CA, USA, USENIX Association, 2009.

[5] Malek ben Salem and Salvatore J. Stolfo. Modeling user search behavior for masquerade detection. In Proceedings of the Fourteenth Symposium on Recent Advances in Intrusion Detection, 2011.

[6] Andre Bergholz, Jan De Beer, Sebastian Glahn, Marie-Francine Moens, Gerhard Paass, and Siehyun Strobel. New filtering approaches for phishing email. Journal of Computer Security, 18:7–35, 2010.

[7] Jagjit S. Bhatia, Rakesh Sehgal, Bharat Bhushan, and Harneet Kaur. Multi layer cyber attack detection through honeynet. In New Technologies, Mobility and Security, (NTMS'08), pages 1–5, 2008.

[8] Brian M. Bowen, Shlomo Hershkop, Angelos D. Keromytis, and Salvatore J. Stolfo. Baiting inside attackers using decoy documents. 2009.

[9] Madhusudhanan Chandrasekaran, Krishnan Narayanan, and Shambhu Upadhyaya. Phishing e-mail detection based on structural properties. In Proceedings NYS Cyber Security Conference, 2006.

[10] William R. Cheswick. An evening with Berferd, in which a cracker is lured, endured, and studied. In Proceedings of the USENIX, January 1992.

[11] Damballa. The command structure of the aurora botnet. http://www.damballa.com/research/aurora/, March 2010.

[12] Manuel Egele, Theodoor Scholte, Engin Kirda, and Christopher Kruegel. A survey on automated dynamic malware-analysis techniques and tools. ACM Computing Surveys (CSUR), 44(2):6, 2012.

[13] C. Fiedler. secure your database by building honeypot architecture using a sql database firewall. http://archive.is/olTW.

[14] Yuelin Gao, Zaimin Ren, and Yang Gao. Modified differential evolution algorithm of constrained nonlinear mixed integer programming problems. Information Technology Journal, pages 2068–2075, 2011.

[15] Project HoneyPot. A web based honeypot network. projecthoneypot.org.

[16] Collin Mulliner, Steffen Liebergeld, and Matthias Lange. Honeydroid – Creating a smartphone honeypot. Technical report, Technische Universität Berlin, 2011.

[17] Younghee Park and Salvatore J. Stolfo. Software decoys for insider threat. In 7th ACM Symposium on Information, Computer and Communications Security, 2012.

[18] Honeynet Project. Know your enemy: Defining virtual honeynets. old.honeynet.org/papers/virtual, 2003.

[19] RSA. RSA security brief: Mobilizing intelligent security operations for advanced persistent threats, 2011.

[20] Bruce Schneier. `www.schneier.com/blog/archives/2011/11/fake_documents.html`.
[21] Lance Spitzner. Honeypots: Tracking Hackers. Addison-Wesley Longman Publishing, Boston, MA, 2002.
[22] Clifford Stoll. The Cuckoo's Egg. Doubleday, New York, 1989.
[23] Don Torrieri. An efficient algorithm for the calculation of node-pair reliability. In Proceedings IEEE Military Communication Conference, 1991.
[24] Wei Wang, Ilona Murynets, Jeffrey Bickford, Christopher Van Wart, and Gang Xu. What you see predicts what you get – Lightweight agent based malware detection. Wiley Journal, Security and Communication Networks, 2012.
[25] J. Yuill, M. Zappe, D. Denning, and F. Feer. Honeyfiles: Deceptive files for intrusion detection. In Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, pages 116–122, 2004.
[26] Jim Yuill, Dorothy Denning, and Fred Feer. Using deception to hide things from hackers: Processes, principles, and techniques, 2006.

Biography

Wei Wang finished her Ph.D. degree from Stevens Institute of Technology in 2010. Now she is a Member of Technical Staff in AT&T Security Research Center. Her research interests are mainly in data analysis, intrusion detection and prevention, and applying machine learning techniques in network security, especially in mobile networks.

Jeffrey Bickford is a researcher with the Chief Security Office at AT&T. He is currently completing his M.S. at the Rutgers University Department of Computer Science. He is interested in mobile device security with a focus on using virtualization techniques to create a secure and robust mobile platform. Prior to joining the Security Research Lab he was a summer intern at AT&T Research in Florham Park.

Ilona Murynets is a scientist at the Chief Security Office at AT&T. She obtained her Ph.D. in Systems Engineering, Stevens Institute of Technology. Ilona holds B.Sc. degree in Mathematics and M.S. degree in Statistics, Financial & Actuarial Mathematics from Kiev National Taras Shevchenko University, Ukraine. Ilona's research is in the area of data mining, optimization and statistical analysis in application to fraud detection, malware propagation, mobile and network security.

Ramesh Subbaraman is a Member of Technical Staff at the AT&T Chief Security Office's Security Research Center. His research interests are in

communication network design and architecture, networking protocols design & analysis, network data mining & analytics, and network security. In addition to traditional approaches, he is very interested in using principles from mathematical optimization, machine learning and mechanism design in networking.

Andrea G. Forte is a Researcher within the Chief Security Office at AT&T. He earned both his Master's Degree and Bachelor's Degree in Telecommunications Engineering at the University of Rome "La Sapienza" in Italy. The paper based on his dissertation work on fast handoffs for real-time media in IEEE 802.11 wireless networks was commercialized by SIPquest Inc. His research interests include mobility, real-time media, location-based services, wireless networks and Internet of Things.

Gokul Singaraju is a developer in AT&T Chief Security Office. His previous experience includes Motorola, NEC Laboratories America, Eulix Networks Inc., Hughes Network Systems, Indotronix International Corp, and Texas Instruments India Ltd. He received Masters in Technology (Computer Science) Indian Statistical Institute, Kolkata 1994.