Bait and Snitch: Defending Computer Systems with Decoys

Jonathan Voris, Jill Jermyn, Angelos D. Keromytis, and Salvatore J. Stolfo

Department of Computer Science

Columbia University, New York, NY 10027

{jvoris@cs., jj2600@, angelos@cs., sal@cs.}columbia.edu

Abstract

Threats against computer networks continue to multiply, but existing security solutions are persistently unable to keep pace with these challenges. In this paper we present a new paradigm for securing computational resources which we call decoy technology. This technique involves seeding a system with data that appears authentic but is in fact spurious. Attacks can be detected by monitoring this phony information for access events. Decoys are capable of detecting malicious activity, such as insider and masquerade attacks, that are beyond the scope of traditional security measures. They can be used to address confidentiality breaches either proactively or after they have taken place.

This work examines the challenges that must be overcome in order to successfully deploy decoys as part of a comprehensive security solution. It discusses situations where decoys are particularly useful as well as characteristics that effective decoy material should share. Furthermore, we describe the tools that we have developed to efficiently craft and distribute decoys in order to form a network of sensors that is capable of detecting adversarial action that occurs anywhere in an organization's system

of computers.

1 Introduction

Organizations across the globe are becoming increasingly aware of the importance of securing their computer systems. As a consequence, worldwide sales of security software rose by 7.5% in 2011 [5]. Government agencies are particularly conscious of the need to defend their computing infrastructure. This is exemplified by the fact that the United States government increased funding for cybersecurity research by 35% from 2011 to 2012 [15]. Attentiveness to security practices has also risen at the individual level, as 90% of American adults now believe that a safe Internet is critical to the U.S. economy [16].

Yet in spite of the heightened scrutiny that security practices have been under, computer crimes continue to flourish. A recent study by the Ponemon Institute found that the number of cyberattacks has more than doubled since 2010 [14]. Since these attacks are also becoming more complex and difficult to anticipate, an average company can currently expect to be the victim of 1.8 successful offensive efforts per month [14].

The vast majority of existing computer security measures focus on controlling access to keep malicious actors out. Other approaches attempt to eliminate system vulnerabilities or at least prevent their exploitation. The aforementioned trends in security statistics leave little doubt that these techniques are not capable of sufficiently securing today's computer networks. Over time such security solutions will invariably fail, allowing adversaries to illicitly access system credentials, data, and financial resources. Furthermore, traditional security techniques offer no defense against "insiders" who initially hold legitimate credentials but later choose to go rogue.

This paper proposes the use of decoys as a new paradigm for addressing computer security issues that existing defenses are not capable of detecting. Decoys are constructs which contain data that appears valuable but is in fact spurious. Since authentic users will have a natural

familiarity with their working environment, they are capable of remembering which resources are real and which are fabricated. They will have no need, therefore, to access inauthentic decoys that contain no truly useful data.

Adversaries without a thorough knowledge of a target system, on the other hand, will have difficulty differentiating decoys from desirable data. After the number of decoy access events pass a certain threshold, an organization can respond by enacting more restrictive security measures and launching an investigation into the account which caused the alerts to occur. Monitoring access to decoy files and content can thus provide protection against cyberattacks in a practical and cost effective fashion.

Decoy technology also addresses the asymmetry that currently exists with respect to responding to different types of security violations after they have already occurred. In the case of data integrity breaches, recovery mechanisms have been developed that allow administrators to "roll back" systems to a checkpointed state that existed prior to when the malicious event took place. Similarly, attacks against the availability of computer networks can be thwarted by increasing the amount of redundant resources that are deployed. Previously, there was no such solution for reacting to attacks against system confidentiality after the fact, however. Decoys can serve this role by providing a mechanism through which data can be tracked after an adversary has already absconded with it.

The rest of this paper is organized as follows. Section 2 reviews previous research related to decoy technology. Next, Section 3 discusses desirable decoy properties in detail. Several scenarios in which decoys can be deployed as a beneficial security measure are proposed in Section 4. Section 5 contains information on how decoy material can be created in an automated fashion, while Section 6 talks about how decoys can be efficiently placed throughout an organization. Finally, we draw conclusions in Section 7.

2 Related Work

The use of deceptive techniques, such as disinformative propaganda, to thwart one's enemies has played a part in military conflict since antiquity. No one has summarized the importance of disinformation in the context of combat more concisely than Sun Tzu, who wrote that "all warfare is based on deception" in the Art of War [17]. A well known example of deception in a military context is Operation Bodyguard, which was an Allied plan used during World War II to distract German forces from the invasion of Normandy [8]. Although deception is an ancient concept, it has only recently been applied to the process of securing computer systems. Cliff Stoll was the first person known to utilize misdirection in order to secure a network of computers. Stoll established a spurious set of computing resources in order to catch hackers who were attempting to exfiltrate information from Lawrence Berkeley National Laboratory [6].

Computers whose primary function is to attract the attention of malicious actors are often called "honeypots." Entire networks of such spurious machines are known as "honeynets." These systems are usually constructed in a way such that they appear as though they are an unassuming component of a larger network architecture. In reality, however, they fail to contain any useful data and are cordoned off from network resources which are actually valued.

Honeypots and honeynets can be quite effective when used to detect external threats. Their applicability towards defending against attacks originating from within an organization is limited, though. This is due to the fact that this class of adversaries typically already have the knowledge that is required to access the portion of a network where legitimate data resides. Furthermore, honeypots offer no utility after a successful attack has already occurred.

Spitzner extended the concept of honeypots to the domain of insider threat detection by inventing the concept of "honeytokens" [12]. Honeytokens are deceptive security constructs that work at a much finer granularity than honeypots or honeynetworks. They are individual

pieces of information that are intended to attract adversarial attention but lack meaningful data. Illegitimate access credentials or forged personally identifiable information can be considered examples of honeytokens. Yuill et al. coined the term "honeyfiles" to describe a document that contains such enticing information [10].

Since the inception of honeyfiles, research has focused on how they can be effectively designed and efficiently deployed. In [1], Bowen et al. developed the Decoy Document Distributor (D^3) System, a tool for automatically generating and monitoring decoys. D^3 is summarized in Section 5. The authors of [1] also established a set of properties that can be used to assess the effectiveness of decoys. A detailed description of these properties can be found in Section 3.

Next, Ben Salem and Stolfo conducted a user study to test the efficacy of decoys that were deployed with these characteristics in mind [13]. This experiment confirmed the ability of decoys to detect attacks by masqueraders. It also identified several trade offs between decoy attributes that can be optimized to defend against specific types of attackers. Furthermore, these authors suggested techniques that can be used to increase the attractiveness of decoys to insiders without interfering with the expected workflow of legitimate users [13].

Most recently, the authors of [9] discussed how language manipulation can be used to craft decoy content that adversaries may find more appealing but normal users would be capable of immediately recognizing as fake. Researchers have also begun investigating how the decoy concept can be applied to other domains. For example, in [19], Park and Stolfo develop a system for protecting software repositories by using decoy Java programs to confuse potential thieves.

3 Properties of Decoys

It is easy to see that some decoy material is more applicable to certain scenarios than others. For example, if an adversary is motivated by financial gain, a decoy document containing

bank credentials is more likely to capture their malicious activity than one whose content concerns medical information. Similarly, certain genres of decoys may be more applicable to specific corporate environments. We have included a detailed discussion of decoy usage scenarios in Section 4.

In order to design decoys that are as effective as possible, it is also beneficial to analyze them in a more general sense by considering characteristics that are independent of a particular context. As initially explored by Bowen et. al in [1], several abstract properties exist that define how a decoy should operate under ideal circumstances. Some of these attributes concern the relationship between adversaries and decoy data, while others pertain to the interactions between legitimate users and deceptive material. A "perfectly believable decoy" would precisely conform to all of these guidelines, though practical restrictions prevent this from occurring in most situations. Although there exists some overlap between these traits, it is also worth noting that they are not completely orthogonal. For example, believability and differentiability are in contention to some extent.

3.1 Believability

One of a decoy's primary functions is to be believable. Upon inspection, a decoy should appear authentic and trustworthy. In the absence of any additional information, it should be impossible to discern a spurious decoy from authentic data. For example, a decoy tax document should contain all of the same fields as one that is actually in use, and each of its fields should be populated with realistic values.

Believability can be formalized via the following thought experiment. Consider a pool of files, some of which contain real data and some of which are fabricated decoys. Select a decoy file and real piece of data from this pool, and present it to an adversary. The selected decoy can be considered perfectly believable if this attacker has an equal probability of selecting the decoy and the legitimate document.

This characteristic is of critical importance to externally observable features of decoys.

For example, in the case of decoy documents, it is imperative that these files have realistic file names and modification dates lest even casual observation reveal their phony nature. In comparison, the believability of document content is of a lower priority. This is because an attacker would have already triggered an alert when opening the document by the time this information came in to play.

3.2 Enticingness

This property takes our idealized decoy material one step further. Decoys should not only appear valid, but also attract an adversary's attention. This, of course, will be heavily influenced by an adversary's objectives. Some malicious actors will be motivated by financial gain, and thus would be interested in documents containing monetary information. Others may seek more specific data such as medical records or a competitor's secrets.

A document's level of enticingness can be thought of as the probability that an adversary would be interested in its exfiltration. A collection of interesting documents is the subset of documents for which this probability is above a certain threshold. In these terms, it is desirable that the probability of accessing any fake document which a decoy distribution system generates is at least equal to the real documents that are in the adversary's pool of interest.

3.3 Conspicuousness

Conspicuousness is closely related to enticingness, as both influence the odds of an attacker accessing a document. Enticingness models how curious an adversary is about a decoy, while conspicuousness concerns how easy a decoy is to access. A conspicuous document is one that is easy to find and access. Conspicuousness can be thought of as the amount of effort an adversary must put in to discovering a decoy, or more formally, the number of actions that are required to access it.

This characteristic captures the fact that decoy documents should be placed in obvious

locations such as a user's desktop. It also demonstrates that it is helpful to place documents in high traffic file system locations, including working folders where files that are accessed on a day-to-day basis are stored. File system searches are also user actions that may result in the presence of decoys. Conspicuous decoys should therefore be easily located by search queries. This relates conspicuousness back to enticingness, however, as the search terms that an adversary employees will be heavily dependent on their underlying motivation.

3.4 Detectability

The aforementioned decoy properties all concern the relationship between decoy documents and a potential attacker. Detectability, on the other hand, describes the ability of decoys to notify their owner when they have been accessed. An ideal decoy system would issue an alert each and every time a decoy is accessed, but technical challenges, including network availability and variability between software platforms, mean that this may not always be possible in practice.

Deploying multiple overlapping decoy monitors that operate at different system levels can help mitigate the possibility of an attacker accessing a decoy while remaining undetected. Features of the decoy documents themselves can be leveraged to equip them with embedded alert code. Monitoring software can be placed in the operating system to detect predetermined tokens placed within decoys when they are opened. Further, operating system auditing can be enabled to record decoy interactions. In order to check for document exfiltration, software can be placed on network equipment to check for such tokens as well. Finally, the content of decoy documents can also serve as an alert system. For example, credentials for spurious accounts can be placed within a decoy. Since there is no reason that a legitimate user would ever access these accounts, any activity they exhibit would send a strong signal of malicious intent.

It is particularly critical that decoy access events are detectable while an attack is taking place. Continuing to monitor this information allows for confidentially violations to be

handled after adversarial action has been carried out. Decoy material usage should thus continue to raise alerts after such data has been exfiltrated. Although it may be possible to evade detection in a particular practical decoy deployment, utilizing an extensive monitoring network will at the very least increase the time and effort that is required to execute an attack. This will make exfiltration more difficult and slow down or discourage adversaries as a net effect.

3.5 Variability

Although a decoy distribution system should strive to make its fake documents seem as authentic as possible, it would certainly be undesirable if precisely the same well-crafted decoy file were placed repeatedly throughout a given system or network. This would greatly simplify the task of distinguishing between legitimate data and the planted decoys that serve as monitors. In general, there should be as much variability between decoy documents as there exists in the pool of documents that they are intended to detect. That is, the task of identifying a decoy should not be reducible to identifying a particular invariant that exists between all generated decoys.

A different way to conceptualize variability is to consider the task of an adversary who wishes to extract information from a system while remaining undetected. Assume that the attacker has been able to discern which documents that have been accessed thus far are authentic and which are traps. With a collection of "perfectly variable" decoys, this adversary would still be unable to discern future decoy material from real data with a probability greater than one half. Previous decoy knowledge, therefore, should not impact the task of identifying future decoys. Note the relationship that exists between the trait of variability and the believability characteristic. Variability among decoys essentially means that decoys should remain believable even after the presence of other decoys has been revealed.

3.6 Stealth

While it is clearly desirable that every decoy access event be perceptible to the owners of a system, care must be taken lest the alarms that accomplish this arouse suspicion. An overt mechanism for issuing alert beacons would provide adversaries with an obvious signal that an element contains a trap, which completely violates the property of decoy variability. The messages that are transmitted by decoys must therefore be as subtle and covert as possible.

Raising an alert that decoy content has been accessed necessarily involves taking some action, however. Even if precautions are taken, there is always the possibility that this act will be perceptible to a malicious actor. It is therefore also desirable to trigger beacon events as early as possible to prevent their interception. For example, alerts for file based decoys should be raised as soon as they are accessed and prior to any content being displayed, if feasible. This would eliminate the possibility of a decoy being recognized and discarded before the decoy system has an opportunity to detect that is has been accessed.

3.7 Non-interference

This property is the first to describe how decoys should coexist with legitimate users who are not masquerading with assumed credentials. An optimal masquerader detection network would not affect the habits of typical users in any way. By inserting decoy material into an operating environment, however, we introduce the possibility that this data will confuse users or otherwise hinder their ability to complete their everyday tasks. It is therefore desirable for decoys to demonstrate the property of non-interference by not obstructing the behavior of normal users.

If a file system is populated with decoy documents that serve as intrusion sensors, for example, the probability that the file system's primary owner is able to access a particular standard document should remain the same as it was prior to the introduction of the decoy content. Similarly, introducing decoy applications to a mobile device's operating system should not impact a user's ability to access real applications as they normally would.

3.8 Differentiability

In effect, the property of decoy non-interference means that true users must be able to easily differentiate between spurious decoy content and authentic data. This can be thought of as the opposite of the believability property. Although decoys should seem as realistic as possible to adversaries, they should appear to be obviously fake for users who should actually be accessing a system. A decoy can be considered fully differentiable if a real user will always succeed at this task.

Balancing the differentiability for authentic users against believability for adversaries is one of the most critical aspects of any practical decoy deployment system. Though this may seem quite challenging, in practice, there are many properties that may be utilized to assist decoy designers in this regard. Legitimate users should be very familiar with detailed aspects of their data. They will also utilize their system in fairly predictable ways. Masqueraders, on the other hand, will have a limited knowledge of the files they are trying to exfiltrate. This gap in knowledge can be leveraged to increase decoy differentiability without affecting believability in the process.

3.9 Shelf Life

The data that is relevant to a normal user's tasks gradually changes as new events occur. The timeliness of data is perhaps even more relevant to attackers, who frequently wish to abscond with the most recent data that they can possibly access. The freshness of material that a decoy contains therefore plays a large part in determining how it will be perceived and how closely it will reflect the aforementioned desirable characteristics.

In order to make decoys appear conspicuous as well as enticing, they can be marked with a very recent date. This creates a very appealing target by leading adversaries to believe that the decoy content has been added even more recently than the authentic data that a system contains. Of course, as time moves on and data is updated while new files are created, these decoys will lose effectiveness. This can be seen as a shelf life during which decoys maintain

an optimal level of functionality and after which their efficacy begins to diminish. It is thus desirable for decoy deployment systems to include a mechanism by which decoys can be updated, potentially extending their shelf life indefinitely.

4 Decoy Usage Scenarios

This section introduces several broad situations in which security can be bolstered by deploying a decoy defense network. It also attempts to discuss some of the challenges that must be met in order for decoys to be used effectively in each environment.

4.1 Host Decoys

The most common usage scenario for decoys is to place them on a terminal that is within a local computer network. This is the operating environment for which decoy documents were originally designed, and as such they require little modification to be utilized in this manner. There are still some deployment questions that must be considered, however. For example, system administrators must determine whether they will push decoy documents out to client systems or require users to pull decoys from a distribution source themselves. The former places less of a burden on individual users, but may temporarily lead to an increase in false positives as users become acquainted with the new decoys that have been placed in their workspace. Experiments from [13] show how decoy access is affected by a document's file system location and the number of deployed decoys.

4.2 Network and Behavioral Decoys

In an effort to detect silent attackers as they eavesdrop on transmissions between computer systems, we have also developed decoys that operate on a network level. Rather than host based data files, these decoys consist of bogus data flows that are injected into a network. This traffic must appear legitimate in terms of protocol specifications but also contains bait

information, such as a username and password that can be monitored in order to detect adversarial activity. A study performed in [4] caught an adversary in the act of accessing a trap account on a web based email service after such credentials were leaked through decoy network traffic.

Similar techniques can be used to generate fake but believable user activity on end systems as well. BotSwindler, which is described in [3], is a system designed to accomplish this in order to detect crimeware. BotSwindler creates realistic user-like actions within a virtual machine in order to convince malicious applications that they are monitoring a legitimate system and are therefore learning valuable information, such as credentials that are used to access other systems. As with network based decoys, accounts corresponding to phony injected user tokens can be monitored to catch criminals after their crimeware has been deceived.

The ability to simulate believable user activity of interest to crimeware is pertinent to the success of BotSwindler. In anticipation of malware that attempts to distinguish simulated actions from human behavior, the system is designed to be difficult to detect and uses a formal language that provides a means for generating variable, realistic simulation actions. A study in [4] asked users to discern generated decoy traffic from authentic information flows; participants were not able to do so with an accuracy greater than guessing randomly.

4.3 Cloud-Based Decoys

Insider attacks are a major concern in the cloud, since these systems require that trust be placed in third party cloud administrators. These individuals have access to all information that their cloud service stores; therefore they can deceitfully, by stealing data, or unintentionally, by making configuration errors, threaten the security of sensitive data. Additionally, in traditional systems, management functionality is usually available only to a few administrators, but in the cloud everyone with access to the infrastructure typically has access to every resource. Thus, there is an increased risk of malicious employees stealing or manipulating

sensitive corporate or organizational data. Furthermore, gaining access to an organization's infrastructure on the cloud is achieved through providing a username and password. Such a lightweight authentication mechanism presents an increase in the problem of masquerader insider threats.

According to [18], observing deviations from typical user behavior can be used in tandem with decoy documents in a cloud environment. If abnormal data access is noticed, the cloud can return decoy information that looks legitimate to the attacker who triggered the unusual usage pattern. In the event of a false positive, an authorized user would be able to recognize any false information returned by the cloud and then correctly respond to a series of authentication challenges to prove his or her legitimacy. The cloud would prevent any unauthorized disclosure of information by continuously returning false data to adversaries.

Also useful in a distributed environment such as the cloud is the concept of computational decoys. Often a single component of a cloud will be responsible for a certain task. If this component is compromised, the distributed system allows migration of the uncorrupted components elsewhere. The compromised components can then actually use computational deception to return false data, thus diverting an attacker's attention [11].

4.4 Software Decoys

Companies are constantly faced with the challenge of preventing employees from stealing proprietary software. In May of 2012, for example, a computer programmer was found guilty of illegally copying software from the Federal Reserve Bank of New York [2]. Software is perhaps the most valuable asset for an organization and is consequently a highly profitable target for insiders. To guard against the unauthorized exfiltration of proprietary software, we have devised software decoys that look like legitimate source code but have beacons that trigger when the code is compiled or executed [19].

In addition to the properties of decoys described in Section 3, software decoys should adhere to several additional requirements. The code must be compilable and executable in

order to appear believable and authentic. Adversaries should also not be able to distinguish software decoys from legitimate source code. They must therefore exhibit a similar style to that of a company's true source code and not look as if it has been drawn from an outside source, such as open source software repositories.

4.5 Voicemail Decoys

Typically people who shy away from sending sensitive data via email opt to select voice as their next method of choice. Yet, there is no guarantee that voicemails are safe from attackers. Voicemail decoys, messages that sound legitimate to those who are uninformed but contain false sensitive information, can be used to detect malicious activity. For example, a voicemail decoy with a human's voice spelling out authentication credentials to a web site or credit card information can attract an attacker to steal such information. Usage of the credentials can then be tracked in order to catch attackers in a similar fashion to that discussed in Section 3.

4.6 Mobile Decoys

With the increased prevalence of mobile computing, it is becoming crucial to protect corporate use of mobile devices and sensitive data stored on them. In an effort to increase productivity and instant contact to employees out of the office, companies offer employees access to company data on mobile devices. In addition, such small and portable machines are easily lost and stolen. As a result, these devices must be secured against unauthorized access.

There are two main methods for using decoy technology to serve this purpose. Deceptive applications can be created that look and feel authentic, but are in fact useless replicas of a true program. Companies that ban non-corporate software usage on mobile devices can monitor access to these programs in order to catch employees who violate corporate policy by installing them.

A second use of decoys in a mobile setting, which is similar to their usage in a cloud environment, is for remote access to sensitive documents. People are increasingly using mobile phones to access private information. We are currently working on a mobile application that interfaces with a user's documents remotely and detects unauthorized access. When uploading documents to the remote server, a user creates a authentication gesture that is later used to retrieve a document's content. If a user provides an incorrect gesture when attempting to access a document, illegitimate data is returned.

5 Decoy Generation

Having established what qualities decoys should possess and where they are intended to be used, we now turn our attention to how they should be created. Users could certainly craft decoy material by hand. For instance, upon completing an invoice, a user could create a secondary fake invoice that mirrors the formatting or the authentic version but contains bogus information. Such decoys would be high quality in terms of their believability because they would closely mirror real data. They would also be very differentiable; since users would have created the documents themselves, they would easily be able to recognize their phony content.

The process of manually introducing decoy content to a system is very tedious, however, since each time new information is saved on the system an equal amount of spurious material would need to be created as well. Users would also be responsible for checking access events for these files. Needless to say, manual decoy creation would scale very poorly to a large organization with many computers and users.

Making, managing, and monitoring decoys is thus a nontrivial problem. As an alternative to performing these steps manually, we suggest using a system that does so with minimal user involvement. This is precisely the purpose of the Decoy Document Distributor (D^3) System [1]. D^3 is a tool for generating and monitoring decoys which can be accessed by

registered users in order to generate decoys for download. It can also be used as a source of data for decoy use in host and network components.

D^3, also known as FOG, is a web site that offers several decoy related services to users [7]. After creating an account, users can request different types of files that contain material appealing to malicious actors, such as tax documents and banking statements. D^3 will then automatically craft a document of the type specified by populating a corresponding template with personal and financial information that is taken from a database of fake identities.

These documents can be downloaded and deployed immediately or reserved for future use. Furthermore, they will also be equipped with code that serves as an alert "beacon." Whenever a D^3 decoy document is accessed, this code will issue an alert by establishing a silent connection with the FOG server. The D^3 system can then take appropriate action, such as logging the alert for future analysis and issuing an alert to the document's owner via email.

Alternatively, D^3 also provides users with the option to upload their own documents to be "beaconized" so they will issue alerts when opened. Since these decoy variants contain actual content, documents that are modified in this way have the advantage of not interfering with the workflow of legitimate users in any way. The FOG site also allows users to manage their collection of decoys as well as calculate statistics on decoy usage and access events.

6 Decoy Distribution

Another issue that must be overcome in order to make use of decoys is deployment. In order to meet this need we have developed the Decoy Distributor Tool (DDT) which can be used to disseminate decoys throughout a file system with minimal manual involvement. Our solution does not require any prior knowledge of the organization or content of the file system in which decoys are to be placed.

Consider the task of a system administrator who wishes to deploy decoy documents

throughout the computers on his or her network in order to defend them from insider threats. To manually place decoys in a file system, the system administrator would first have to collect feedback from users regarding which locations in their file system they would like to place decoys. After aggregating this data, he or she would then have to request a batch of decoys and then go through the painstaking process of copying them to their destination directories one by one.

Such a process does not scale well to an environment with a large number of computers and users. In contrast, the DDT requires neither knowledge regarding file system specifics nor individual file placement. It reduces the task of decoy document management to the simple steps of specifying how many decoys are desired and in which portion of a file system to place them. Our distribution application is therefore capable of reducing the time required to establish a system of insider threat sensors by at least an order of magnitude while retaining all of the security benefits of manual decoy usage.

The DDT has two main objectives. The first is to automatically determine locations in a computer's file system that are most likely to be accessed by a malicious insider. The second is to place decoy documents in these selected locations, either directly along with existing documents or in a separate folder. The DDT allows a user to select a source directory of decoy documents that should be distributed on the target machine. The user can choose an existing folder containing decoys or create a new set by accessing the FOG web site [7] through the DDT and specifying a number of decoys to be generated. This approach enables flexibility in the types of documents that are deployed as decoys.

Once the user chooses a source directory of decoy documents, he or she then specifies a destination directory. This destination directory is used as a root from which target locations are selected. Enabling the user to specify the root from which target directories are identified allows more freedom in decoy placement. For example, if a particular owner of a machine only uses directories within "C:\Sal," he or she may want to consider placing decoys solely in directories branching from this root.

6.1 Creating Decoy Documents

As mentioned in Section 5, Bowen et al. describe properties of effective decoys in [1]. The DDT tackles conspicuousness and non-interference via the tool's scheme for selecting locations to place decoys, while the FOG system deals with detectability by alerting the FOG account owner when a decoy document is accessed. When creating decoys, the DDT has two primary considerations for each document, namely the filename of the document and its modification and creation dates. The DDT's naming convention and date assignment techniques attempt to solve the problems of believability, enticingness, variability, non-interference, and differentiability. Shelf-life is also handled by the DDT's date assignment method, with possible future work described in Section 6.4 as an alternative solution.

6.2 Determining locations to place the decoy documents

A study performed in [13] demonstrated that the placement of decoy documents greatly affects the probability of a user accessing these files. Locations in which decoys are placed in a file system should be selected so that the decoys remain conspicuous to malicious insiders but do not impede a legitimate user's normal actions. The DDT scans the target machine's file system starting at the specified root and identifies ten folders with the most recently accessed documents as well as ten folders containing the greatest number of files with common document extensions .pdf, .doc, .docx, .ppt, .xls, .txt, .html, and .htm. Selecting the most populated and most recently accessed folders increases the conspicuousness of decoys, since these are directories that would be the most probable targets for malicious insiders.

Once the DDT identifies the directories that are in the top ten on the file system for both recent activity and volume, it then proceeds to populate the remainder of the ten most crowded directories, followed by any other of the top ten directories which showed the most activity. In all three cases, decoys are placed evenly among the resultant destination directory list by iterating through them in descending rank order. Within these destination folders, one can select the option to deploy the decoy documents directly so that they blend

in with currently existing documents, or in separate folders that contain decoys only.

6.3 Naming of Decoy Documents and Folders

As with location, the names of documents directly influence how enticing they look to adversaries. Folder names can also impact a user's decision to access a location in the file system. When creating new directories in which decoys will be placed, the DDT creates four enticingly-named new folders in the specified destination directory and evenly disperses the decoys over these folders.

In an effort to increase the variability of decoys, the DDT uses three methods for naming decoy documents when attempting to blend them with existing documents in a folder, randomly selecting among these approaches. The prime objective of the naming scheme is to create filenames that blend in with existing legitimate documents so that they do not look overtly suspicious. At the same time, the decoy names should lure malevolent users into opening the documents.

The first naming method selects an existing file in the target directory and appends either "-final" or "-updated" to the end of the filename. The logic behind this scheme is that the most recently modified versions of files may seem more official than older versions. Note that if either of the terms "final" or "updated" are already used in the filename, the DDT selects another naming method for this decoy.

The second naming method employed by the DDT appends a date string to the end of a randomly selected existing filename in the target directory. Appending a date string makes a document appear as if it has been marked as a more authentic, official version. The DDT uses the format *mmddyy* for the date strings. For example, a decoy document may be named *company_employees-010412.pdf* and placed in the target folder alongside a legitimate document named *company_employees.pdf*.

Blending filenames by use of delimiters is the final naming approach. The DDT calculates the delimiter used most often in the target directory and modifies a decoy's filename

to use this delimiter. For example, if the target directory contains files *phone_bill_1011.pdf*, *shopping_list_april.doc*, *tempDirectory.txt*, and *friends_birthdays.docx*, the DDT would change decoy document *alice-taxes-33.pdf* to *alice_taxes_33.pdf*, since "_" is the most common delimiter in the destination folder. The above directory example demonstrates the importance of this naming technique. The file *tempDirectory.txt* clearly stands out as a file that does not belong; perhaps the file was copied from another machine or automatically generated by an application. Since this filename doesn't use the same delimiters as the other filenames in the same directory, the astute user may assume the file was not created by the regular user of the computer, who seems to typically name files using the "_" delimiter.

When inserting decoy documents into a separate folder as described above, the DDT does not change the filename generated by the FOG system. Since the DDT creates a brand new folder for the decoy documents, there are no existing documents in this folder and therefore no need to blend filenames. Placing the decoys documents in a separate directory is therefore the option that enables the most differentiability of decoys from legitimate documents.

6.4 Modification and Creation Dates of Decoy Documents

Although users do not immediately see file modification and creation dates when first entering a directory, prudent or suspicious adversaries may check these attributes when carefully searching for sensitive information. For this reason, it is imperative that the modify and creation dates blend in with existing documents as well as possible so that they do not appear statically assigned without regard to their destination directory. Dates can also be used to increase the allure of certain decoys, such as updated versions of existing documents.

When blending decoy documents into a folder with existing documents, the DDT chooses a file's date depending on the naming convention selected. If following a method that appends either "-final" or "-updated" to a filename, the DDT will set the document's creation date to at most 48 hours after the most recently created document in the destination directory. The document's last modified date will then be set to up to two days after the resultant creation

date. If an adversary decided to sort a folder's contents by date, the decoy documents would then appear at the top of the sorted list, making them conspicuous and likely to be more attractive targets.

A naming convention that appends a date string to the decoy's filename should intuitively set the file's creation date to that used in the selected date string. The DDT first determines an appropriate date for a decoy document by finding the median creation date of existing files in the decoy's target directory. The decoy's creation date is then set within a 48-hour window of the median. The date string appended to the filename is obtained from the proposed creation date.

When placing decoy documents in a separate new folder, the DDT has no existing document dates on which to base a file's target date. Therefore, the modify dates of the decoys are determined by the date that the decoys are generated by the FOG system. A creation date is then set to up to two days before the existing modify date. The DDT subsequently alters the modify and creation dates of the new folder in which the decoys are placed. The creation date of the folder is set to the creation date of the oldest decoy file placed inside; the modify date is set to that of the most recently modified document. We considered this dating approach the most practical for the case in which new decoy folders are created, since users have direct control over the dates applied to the decoy documents. In our future work, the DDT will include a feature to "refresh" the dates of decoys so that they remain among the most recent documents in the file system.

7 Conclusions

To summarize, this paper introduced a novel security paradigm which we refer to as decoy technology. Decoys represent a drastic departure from existing security solutions in several important ways. By placing content that is spurious yet believable and enticing in the path of potential adversaries, decoys can serve as a potent last line of defense against attacks

that traditional security mechanisms fail to adequately defend against. Decoy content can be proactively seeded throughout a system to defend against potential attacks, or fed to an adversary once malicious activity has been detected. Furthermore, by tracking decoy material, violations of confidentiality can be addressed after they have occurred. This is a capability that alternative security measures are not capable of offering.

Although the deceptive techniques that form the basis of decoys have existed for ages, they have only recently been leveraged to protect computing resources. This paper discussed several dimensions along which this process can be refined and extended. It included attributes that all high quality decoys should share as well as contexts in which decoys are particularly applicable. Furthermore, we shared techniques for efficiently generating decoy material and disseminating it throughout an organization to create an insider detection network. Decoys can be integrated as useful components of any full featured security solution and will only increase in prominence as threats against computer systems continue to grow.

Acknowledgment

This material is based on work supported by the Defense Advanced Research Projects Agency (DARPA) under the ADAMS (Anomaly Detection at Multiple Scales) Program with grant award number W911NF-11-1-0140 and through the Mission-Resilient Clouds (MRC) program under Contract FA8650-11-C-7190. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA. Professors Stolfo and Keromytis are founders of Allure Security Technology, Inc.

References

[1] B. Bowen and S. Hershkop and A. Keromytis and S. Stolfo. Baiting Inside Attackers Using Decoy Documents. In *Conference on Security and Privacy in Communication*

Networks, 2009.

[2] B. Katz. Chinese Man Pleads Guilty to NY Fed Cyber Theft. Available at `http://www.reuters.com/article/2012/05/29/usa-crime-fed-idUSL1E8GTBG120120529`, 2012.

[3] B. Bowen, P. Prabhu, V. Kemerlis, S. Sidiroglou, A. Keromytis, and S. Stolfo. Botswindler: Tamper resistant injection of believable decoys in vm-based hosts for crimeware detection. In *Recent Advances in Intrusion Detection*, page 118137, 2010.

[4] B. M. Bowen, V. P. Kemerlis, P. Prabhu, A. D. Keromytis, and S. J. Stolfo. Automating the injection of believable decoys to detect snooping. In *Proceedings of the third ACM conference on Wireless network security*, page 8186, 2010.

[5] C. Pettey and R. van der Meulen. Gartner Says Security Software Market Grew 7.5 Percent in 2011. Available at `http://www.gartner.com/it/page.jsp?id=1996415`, 2012.

[6] C. Stoll. The Cuckoo's Egg, 1989.

[7] Columbia University Intrusion Detection Systems Lab. FOG Computing. Available at `http://ids.cs.columbia.edu/FOG/`, 2012.

[8] J. Rubin. Deception: The other 'D' in D-Day. Available at `http://www.msnbc.msn.com/id/5139053/ns/msnbc_tv-the_abrams_report/t/deception-other-d-d-day`, 2004.

[9] J. Voris and N. Boggs and S. Stolfo. Lost in Translation: Improving Decoy Documents via Automated Translation. In *Workshop on Research for Insider Threat*, 2012.

[10] J. Yuill and M. Zappe and D. Denning and F. Feer. Honeyfiles: Deceptive Files for Intrusion Detection. In *Workshop on Information Assurance*, 2004.

[11] A. D. Keromytis, R. Geambasu, S. Sethumadhavan, S. J. Stolfo, J. Yang, A. Benameur, M. Dacier, M. Elder, D. Kienzle, and A. Stavrou. The MEERKATS cloud security

architecture. In *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*, page 446450, 2012.

[12] L. Spitzner. Honeytokens: The Other Honeypot. Available at `http://www.symantec.com/connect/articles/honeytokens-other-honeypot`, 2003.

[13] M. Ben Salem and S. Stolfo. Decoy Document Deployment for Effective Masquerade Attack Detection. In *Conference on Detection of Intrusions and Malware and Vulnerability Assessment*, 2011.

[14] M. Schwartz. Cybercrime Attacks, Costs Escalating. Available at `http://www.informationweek.com/security/attacks/cybercrime-attacks-costs-escalating/240008658`, 2012.

[15] P. Thibodeau. Obama Seeks Big Boost in CyberSecurity Spending. Available at `http://www.computerworld.com/s/article/9209461/Obama_seeks_big_boost_in_cybersecurity_spending`, 2011.

[16] S. Kilcarr. Cyber Security Concerns Keep Mounting. Available at `http://fleetowner.com/blog/cyber-security-concerns-keep-mounting`, 2012.

[17] S. Tzu. The Art of War. Available at `http://classics.mit.edu/Tzu/artwar.html`, 2009.

[18] S. J. Stolfo, M. B. Salem, and A. D. Keromytis. Fog computing: Mitigating insider data theft attacks in the cloud. In *Security and Privacy Workshops (SPW), 2012 IEEE Symposium on*, page 125128, 2012.

[19] Y. Park and S. Stolfo. Software Decoys for Insider Threat. In *ACM Symposium on Information, Computer and Communications Security*, 2012.