Baiting Inside Attackers using Decoy Documents

Brian M. Bowen, Shlomo Hershkop, Angelos D. Keromytis, Salvatore J. Stolfo
Department of Computer Science
Columbia University
New York, NY 10027

Abstract— **The goal of this work is to design insider attack mitigation strategies to thwart and deter insider attacks, and challenge security professionals to design and develop more sophisticated security architectures than is presently the norm.Insider attack detection is unlikely to be solved by a single technology solution, but rather by a set of detection techniques that are hard to avoid by a clever adversary. At the core of any approach to detect malfeasance is a monitoring technology that audits and analyses system events, whether these events are initiated by a user directly or indirectly by a proxy such as planted malware. The goal of such a monitoring infrastructure is to detect suspicious acts that are clear violations of policy. We posit that no one monitoring infrastructure alone will do the job effectively. Sophisticated insiders may easily avoid one or more detectors. Instead, one must design a "web of detectors" that may be stealthy and hard to avoid. In our work, we have developed tools and techniques to monitor user-initiated events using host-level sensors. We have also developed new trap-based decoy techniques and technologies that are designed with multiple detectable elements, including watermarked documents with embedded beacons and decoy credentials recognized by external monitoring infrastructures (eg., available at remote banking sites). The combination of host-level sensors, trap-based defenses and (remote) network detectors provides a formidable defense against insider attack.**

I. Introduction

Much research in computer security has focused on the means of preventing unauthorized and illegitimate access to systems and information. Unfortunately, the most damaging malicious activity is the result of internal misuse within an organization, perhaps since far less attention has been focused inward. Despite classic internal operating system security mechanisms and the body of work on formal specification of security and access control policies, including Bell-LaPadula [1] and the Clark-Wilson models [4], we still have an extensive insider attack problem. Indeed in many cases, formal security policies are incomplete and implicit or they are purposely ignored in order to get business goals accomplished. There seems to be little technology available to address the insider threat problem.

Insider attack has overtaken viruses and worm attacks as the most reported security incident according to a report from the US Computer Security Institute (CSI) [20]. The annual Computer Crime and Security Survey for 2007 surveyed 494 security personnel members from US corporations and government agencies, finding that insider incidents were cited by 59 percent of respondents, while only 52 percent said they had encountered a conventional virus in the previous year. The state-of-the-art seems to be still driven by forensics analysis after an attack, rather than technologies that prevent, detect, and deter insider attack.

We define insider threats by differentiating between Masqueraders (attackers who impersonate another inside user) and Traitors (an inside attacker using their own legitimate credentials). One possible solution for masquerade detection involves anomaly detection [19]. In this approach, users actions are profiled to form a baseline of normal behavior. Subsequent monitoring for abnormal behaviors that exhibit large deviations from this baseline [17] signal a potential insider attack. The common strategy to prevent inside attacks involves policy-based access control techniques to limit the scope of systems and information an insider is authorized to use, and hence, limit the damage the organization may incur when an insider goes awry. Prevention techniques may not always succeed, and thus, monitoring and detection techniques are needed when prevention fails. In this paper, we are focused on different techniques aimed at detecting masqueraders and traitors.

We note that some external attackers can become insiders when an outsider attains internal network access. Many attacks use spyware and rootkits [3], which give outsiders internal access. Such software can easily be installed on systems from physical or digital media (e.g., email, downloads, etc.) and allow an attacker administrator or "root" access on a machine along with a means to gather sensitive data. Rootkits have the ability to conceal themselves and elude detection, especially when the rootkit is previously unknown, as is true in zero-day attacks [8]. An external attacker that manages to install rootkits internally in effect becomes an insider, thereby multiplying the ability to inflict harm. Although the techniques described in this paper may have utility for these cases, in this paper our primary focus is on human insiders attempting to exfiltrate sensitive information. By exfiltration we mean unauthorized copying and transmission of information by any means including human memory.

The insider attack defense system described in this paper is of an offensive nature, intended to confuse and deceive a traitor by leveraging uncertainty, to reduce the knowledge they ordinarily have of the systems and data they might be authorized to use. This work considers methods to detect insider actions against enterprise systems as well as individual hosts and laptops. We introduce a deception system to distribute potentially large amounts of decoy information with the aim to detect nefarious acts as well as to increase the workload of an attacker to identify real information from bogus information, rather than providing unfettered access as broadly exists today. We developed a system to generate and place *decoy documents* within a file system. Our system generates decoy documents containing decoy credentials that are monitored (e.g., Gmail credential monitoring) for misuse and stealthily embedded beacons that signal an alert when the document is opened. Beacons are embedded in documents using methods of deception and obfuscation gleaned from studying malcode embedded inside documents as seen in the wild [16]; we thus turn the tables on attackers.

To achieve the goal of wide spread deception we must consider methods to trap a wide variety of potential insiders with varying levels of sophistication. Toward this goal, we developed a proof-of-concept system we call D^3, the Decoy Document Distributor system. Samples of D^3 generated documents are presented in the Appendix. The contributions of this paper include:

- A novel set of generally applicable properties are proposed to guide the design and deployment of decoys and maximize the deception they induce for different classes of insiders who vary by their level of knowledge and sophistication.
- A large-scale automated creation and management system for deploying decoys that can detect the presence (and, in some cases, "identity") of malicious insiders, or at least indicate malicious insider activity. This provides a means for ordinary users to deploy honey documents without having to setup sophisticated honeypot systems and sensors.
- An offensive trap-based defense system is proposed to detect masqueraders and traitors, and to flood attackers with bogus exfiltrated information that they must analyze in order to find real information of value. Hence, our long term goal is to flood the miscreant marketplace with bogus information devaluing their quarry.
- A design of decoy information that combines a number of methods and monitors, both internal and external, to detect insider exploitation using a common and ubiquitous set of baited targets, ordinary looking documents.
    1) A watermark is embedded in the binary format of the document file to detect when the decoy is loaded in memory, or egressed in the open over a network.
    2) A "beacon" is embedded in the decoy document that signals a remote website upon opening of the document indicating the malfeasance of an insider illicitly reading bait information.
    3) If these methods fail to detect an insider attack or an exfiltration of baited documents, the content of the documents contain bait and decoy information that is monitored as well. Bogus logins at multiple organizations as well as bogus and realistic bank information is monitored by external means.
- An easy to use system to broadly deploy decoys to ordinary users who are alerted by email when a decoy has been touched on their laptops and personal computers; no such system presently exists.

The reader is encouraged to visit the Decoy Document Distribution (D^3) website to evaluate our technology developed to date at: http://www.cs.columbia.edu/ids/RUU/Dcubed[1].

II. Related Work

The use of deception, or decoys, plays a valuable role in the protection of systems, networks, and information. The first use of decoys (i.e., in the cyber domain) has been credited to Cliff Stoll [27], [24] and detailed in his novel "The Cuckoos Egg" [25], where he provides a thorough account of his crusade to catch German hackers breaking into Lawrence Berkley Laboratory computer systems. Stoll's methods included the use of bogus networks, systems, and documents to gather intelligence on the German attackers who were apparently seeking state secrets. Among the many techniques waged, he crafted "bait" files, or in his case, bogus classified documents that really contained non-sensitive government information and attached "alarms" to them so that he would know if anyone accessed at them. To Stoll's credit, a German hacker was eventually caught and it was found that he had been selling secrets to the KGB.

Deception-based information resources that have no production value other than to attract and detect adversaries (like those used by Stoll) are commonly known as Honeypots [11]. Honeypots serve as effective tools for profiling attacker behavior and to gather intelligence to understand how attackers operate. Honeypots are considered to have low false positive rates since they are designed to capture only malicious attackers, except for perhaps an occasional mistake by innocent users. Spitzner described how honeypots can be useful for detecting insider attack[23], in addition to the common external threats for which they are traditionally known. He discusses the use of honeytokens, which he defines as "a honeypot that is not a computer" [24], citing examples that include bogus medical records, credit card numbers, and credentials, with descriptions of how they can be used to detect malicious insiders [23], [24]. In current systems, the decoy/honeytoken creation is a laborious and manual process requiring large amounts of administrator intervention. In contrast, we propose the seeding of decoy information (of various different types) throughout an operational system. Our work extends these basic ideas to an automated system of managing the creation and deployment of these honeytokens.

Yuill et al. [27] extend the notion of honeytokens with a "honeyfile system" to support the creation of bait files, or as they define them, "honeyfiles." The honeyfile system is implemented as an enhancement to the Network File Server. The system allows for any file within user file space to become a honeyfile through the creation of a record associating a filename to userid. The honeyfile system monitors all file access on the server and alerts users when honeyfiles have been accessed. Their work does not focus on the content or automatic creation of files, but they do elicit some of the

[1]Some features are restricted for internal use only.

challenges of creating deceptive files (with respect to names) that we address in section 4.

In this paper, we introduce a set of properties of decoys to guide their design and maximize the deception they induce for different classes of insiders who vary by their level of knowledge and sophistication. To the best of our knowledge, the synthesis of these properties is indeed novel a contribution. Bell and Whaley [2] have described the structure of deception as a process of hiding the real and showing showing the false. They introduce several methods of hiding that include masking, repackaging, and dazzling, along with three methods of showing that include mimicking, inventing, and decoying. Yuill et al. [28] expand upon this work and characterize deceptive hiding in terms of how it defeats an adversary's discovery process. They describe an adversary's discovery process as taking three forms: direct observation, investigation based on evidence, and learning from other people or agents. Their work offers a process model for creating deceptive hiding techniques based on how they defeat an adversary's discovery process.

The decoy documents introduced in this paper utilize similar deception mechanisms as well as beacons to signal a remote detect and alert in real-time time when a decoy has been opened. Web bugs are a form of silent embedded beacons which have been used to track user habits of web or email. Web bugs are a class of silent embedded tokens which have been used to track usage habits of web or email users [18]. Unfortunately, they have been most closely associated with unscrupulous operators, such as spammers, virus writers, and spyware authors who have used them to violate users privacy. Typically they will be embedded in the HTML portion of an email message as a non-visible white on white image, but they have also been demonstrated in other forms such as Microsoft Word, Excel, and PowerPoint documents [22]. When rendered as HTML, a web bug triggers a server update which allows the sender to note when and where the web bug was viewed. Animated images allow the senders to monitor how long the message was displayed. The web bugs operate without alerting the user of the tracking mechanisms. The advantage for legitimate advertisers is that this allows them to monitor advertisement effectiveness, while privacy advocates worry that this technology can be misused to spy on users' habits. Our work leverages the same ideas, but extends them to other document classes and is more sophisticated in the methods used to draw attention. In addition, our targets are insiders who should have no expectation of privacy on a system they violate.

III. Threat Model - Level of Sophistication of the Attacker

The insider seeks to identify and avoid the decoys and abscond with "real" information. We broadly define four monotonically increasing levels of insider sophistication and capability. Some will have tools available to assist in deciding what is a decoy and what is real. Others will only have their own observations and thoughts.

Low: Direct observation is the only tool available. The adversary largely depends on what can be gleaned from a first glance. We strive to defeat this level of adversary with our beacon documents, even though decoys with embedded beacons may be distinguished with more advanced tools.

Medium: A more thorough investigation can be performed by the insider; decisions based on other, possibly outside evidence, can be made. For example, if a decoy document contains a decoy account credential for a particular identity, an adversary may verify that the particular identity is real or not by querying an external system (such as www.whitepages.com). Such adversaries will require stronger decoy information possibly corroborated by other sources of evidence.

High: Access to the most sophisticated tools are available to the attacker (e.g., super computers, other informed people who have organizational information). The notion of the "Perfect Decoy" described in the next section may be the only indiscernible decoy by an adversary of such caliber.

Highly Privileged: Probably the most dangerous of all is the privileged and highly sophisticated user. Such attackers might even be aware that the system is baited and will employ sophisticated tools to try to analyze, disable, and avoid decoys entirely. As an example of how defeating this level of threat might be possible, consider the analogy with someone who knows encryption is used (and which encryption algorithm is used), but still cannot break the system because they do not have knowledge of an easy-to-change operational parameter (the key). Likewise, just because someone knows that decoys are used in the system does not mean they should be able to identify them. This is the principal– coming up with a scheme to satisfy it remains an open problem.

IV. Generating and Distributing Bait

In order to create decoys to bait various levels of insiders, one must understand the core properties of a decoy that will successfully bait an insider.

A. Decoy Properties

We enumerate various properties and means of measuring these properties that are associated with decoy documents to ensure their use will be likely to snare an inside attacker.

Believable[2]: Capable of eliciting belief or trust; capable of being believed; appearing true; seeming to be true or authentic.

A good decoy should make it difficult for an adversary to discern whether they are looking at an authentic document from a legitimate source or if they are indeed looking at

[2]For clarity, each property is provided with its definition gleaned from online dictionary sources.

a decoy. We conjecture that believability of any particular decoy can be measured through experiment. We define a decoy believability experiment as follows:

- Choose two documents such that one is the decoy we wish to measure the believability of and the second is chosen at random from a pool of authentic documents.
- Select a volunteer at random to participate in a user study.
- The volunteer is given access to the documents chosen in step one and tasked to decide which of the two is authentic.

For concreteness, we build upon the definition of "Perfect Secrecy" proposed in the cryptography community [13] and define a "perfect decoy" to be a decoy that is chosen in a believability experiment with a probability of 1/2 (the outcome that would be achieved if the volunteer decided completely at random). That is, a perfect decoy is one that is completely indistinguishable from one that is not. A benefit of this definition is that the challenge of showing a decoy to be believable, or not, reduces to the problem of creating a "distinguisher" that can decide with probability better than 1/2.

In practice, the construction of a "perfect decoy" might be unachievable, especially through automatic means, but the notion remains important as it provides a goal to strive for in our design and implementation of systems. For many threat models, it might suffice to have less than perfect believable decoys. For our proof-of-concept system described below, we generate receipts and tax documents, and other common form-based documents with decoy credentials, realistic names, addresses and logins, all information that is familiar to all users.

We note that the believable property of a decoy may be less important than other properties defined below since the attacker may have to open the decoy in order to decide whether the document is real or not. The act of opening the document may be all that we need to trap the insider, irrespective of the believability of its content. Hence, enticing an attacker to open a document may be a more effective defense strategy.

Enticing: highly attractive and able to arouse hope or desire; "an alluring prospect"; lure.

Herein lies the issue of how does one measure the extent to which a decoy arouses desires, how well is it a lure? One obvious way is to create decoys containing information with monetary value, such as passwords or credit card numbers that have black market value [15], [26].

However, enticement depends upon the attacker's intent. Hence, we posit that by defining several general categories of "things" that are of "attacker interest", one may compose decoys using terms or words that correspond to desires of the attacker that are overwhelmingly enticing. For example, if the attacker desires money, any document that mentions or describes information that provides access to money should be highly enticing. We believe we can measure frequently occurring (search) terms associated with major categories of interest and use these as the constituent words in decoy documents. To measure the effectiveness of this generative strategy, it should be possible to execute content searches and count the number of times decoys appear in the top 10 list of displayed documents. This is a reasonable approach also, to measuring how conspicuous, defined below, the decoys become based upon the attacker's searches associated with their interest and intent.

Conspicuous: easily visible; easily or clearly visible; obvious to the eye or mind; Attracting attention.

Here, a *conspicuous* decoy should be easily found or observed. When a user first logs in, a conspicuous decoy should either be in full view on the desktop, or viewable after one (targeted) search action. One simple user action is optimal for a highly conspicuous decoy. Thus, a measure of conspicuousness may be a count of the number of search actions needed, on average, for a decoy to appear in full view. The decoy may be stored in the file system anywhere if a simple content-based search locates it in one step. But, this search act depends upon the query executed by the user. The query can either be a location (eg., search for a directory named "TAX" in which the decoy appears) or a content query (eg., using Google Desktop Search for documents containing the word "TAX.") In either case, if a decoy document appears after one such search, it is conspicuous. But, this depends upon what search terms the attacker uses to query! If the decoy never appears because the attacker used the wrong search terms, the decoy is not conspicuous. We posit that the property of *enticing* is likely the most important property, and a formal measure to evaluate enticement will generate better decoys. In summary, an enticing decoy should be conspicuous to be an effective decoy trap.

Detectable; to discover or catch (a person) in the performance of some act: to detect someone cheating.

We designed the decoy documents with several techniques to provide a good chance of detecting the malfeasance of an inside attack in real-time.

- At time of application start-up, the decoy document emits a beacon alert to a remote server.
- At the time of memory load, a host-sensor, such as an AV scanner, may detect embedded tokens placed in a clandestine location of the document file format.
- At the time of exfiltration, a NIDS such as Snort may be used to detect these embedded tokens during the egress of the decoy document in network traffic where possible.
- At time of information exploitation and/or credential misuse, monitoring of decoy logins and other credentials embedded in the document content by external systems will generate an alert that is correlated with the decoy document in which the credential was placed.

This extensive set of monitors forces the attacker to expend considerable effort to avoid detection, and hopefully will serve as a deterrent to reduce internal malfeasance within organizations that deploy such a trap-based defense. In the proof-of-concept implementation reported in this paper, we focus our evaluation on the fourth item. We utilize monitors at our local IT systems, at Gmail and at an external bank.

Variability: The range of possible outcomes of a given situation; the quality of being subject to variation.

Attackers are humans with insider knowledge, even possibly with the knowledge that decoys are liberally spread throughout an enterprise. Their task is to identify the real documents from the potentially large cache of decoys. One important property of the set of decoys is that they are not easily identifiable due to some common invariant information they all share. A single search or test function would thus easily distinguish the real from the fake. The decoys thus must be highly varied.

Clearly, a good decoy generator should produce an unbounded collection of enticing, conspicuous but distinct and variable documents. They are distinct with respect to string content. If the same sentence appears in 100 decoys, one wouldn't consider such decoys with repetitive information as highly variable; the common invariant sentence(s) can be used as a "signature" to find the decoys, rendering them distinguishable (and clearly, less enticing).

Non-interference: Something that does not hinder, obstructs, or impede.

How might a decoy interfere with regular operations of the *legitimate user*? One would expect that the more conspicuous a decoy is, the more it would interfere (since it could be found more easily). Conspicuous may help catch a thief, but the unwitting user may be ensnared as a by-product.

Although we seek to create decoys to ensnare an inside attacker, a legitimate user whose data is the subject of an attacker must still be able to identify their own real documents from the planted decoys. The more enticing or believable a decoy document may be, the more likely it would be to lead the user to confuse it with a legitimate document they were looking for. Our goal is to increase believability, conspicuousness and enticingness while keeping interference low; ideally a decoy should be completely non-interfering. The challenge is to devise a simple and easy to use scheme for the user to easily differentiate their own documents, and thus a measure of interference is then possible as a by-product.

As an outsider, we presume the attacker lacks some specific knowledge known to the creator of the real document, or the attacker lacks access to some "physical key" owned by the user who created the document. This crucial property therefore requires that the legitimate owner of the document be able to easily differentiate the real document they created from the bogus generated to thwart the attacker. Hence, another important property is as follows.

Differentiable: to mark or show a difference in; constitute a difference that distinguishes; to develop differential characteristics in; to cause differentiation of in the course of development.

It is important that decoys be "obvious" to the *legitimate user* to avoid interference, but "unobvious" to the insider stealing information. How might we easily differentiate a decoy for the legitimate user so that we maintain "non-interference" with the user's own actions and legitimate work?

The remote thief who exfiltrates all of a user's files onto a remote hard drive may be perplexed by having hundreds of decoys amidst a few real documents; the thief should not be able to easily differentiate between the two cases. If we store a hundred decoys for each real document, the thief's task is daunting; they would need to test embedded information in the documents to decide what is real and what isn't, which should complicate their end goals. For clarity, decoys should be easily *differentiable* to the legitimate user, but not to the attacker without significant effort.

B. The Decoy Document Distributor (D^3) System

The D^3 web-based service generates and distributes decoy documents to registered users. The decoy properties guide the design of decoy templates in D^3 that are used to generate specific documents for download. The content of each decoy document includes several types of "bait" information such as online banking logins provided by a collaborating financial institution[3], login accounts for online servers, and web based email accounts. In our deployment we used Columbia University student accounts and Gmail email accounts as bait, but these can be customized to any set of monitored credentials. These decoy credentials are "bait" and are enticing targets for different types of adversaries [15], [14]. These particular examples of bait credentials are monitored internally and externally.

C. Decoy Document Design

The primary goal of the trap based defense is to detect malfeasance. Since no system is foolproof, we propose that multiple overlapping signals be embedded in the decoy documents to ensure *detectability*. Any alert generated by the multiple decoys is an indicator that some insider activity has occurred. Since the attacker may have varying levels of sophistication, a combination of traps are used in decoy documents to increase the likelihood one will succeed in generating an alert. A sophisticated attacker may, for example, disable the internal beacon, or cut off network connections avoiding communication, disable or kill local host monitoring processes, or they may exfiltrate documents via a web-browser without opening them locally. The documents are designed with several means of detecting their misuse:

- embedded honeytokens, computer login accounts created that provide no access to valuable resources, and that are monitored when (mis)used;
- embedded honeytoken banking login accounts specifically created and monitored for this trap-based technology demonstration specifically to entice financially motivated attackers;
- a network-level egress monitor that alerts whenever a marker, specially planted in the decoy document, is detected (we are collaborating with Cornell to use Cayuga

[3]By agreement, the institution request that its name be withheld.

[5] for this purpose. Presently Snort may be used as simple signature detector as a proof-of-concept);

- a host-based monitor that alerts whenever a decoy document is "touched" in the file system such as a copy operation;
- an embedded "beacon" alerts a remote server at a site at Columbia, that we call SONAR. The website emits an email to the registered user who created and downloaded the decoy document. The implementation of document beacons is described in the next section.

1) Beacon Implementation: The highly sophisticated attacker will likely attempt to differentiate between a real document and a decoy by analyzing the binary file format prior to opening a file. This necessitates a design where beacon code and watermarks in decoy documents are hidden to avoid their easy identification. The attacker would surely avoid the decoys if they could easily identify them by a simple static test for an embedded beacon. The beacon code can be embedded in documents in a number of ways and made to appear statistically equivalent to its surrounding data using a blending technique called "spectrum shaping" (see [21], [6]). Such obfuscation techniques are very hard to defeat [16].

Using common techniques developed for malware, beacons attempt to silently contact a centralized server with a unique token embedded within the document at creation time. The token is used to identify the decoy and document, IP address of the host accessing the decoy document. In addition to passing the token and IP address, some addition data is collected. This is dependent on the particular document type, and the rendering environment used during viewing of the beacon document.

The first proof-of-concept beacons have been implemented in Word and PDF and deployed through the D^3 website.

2) Word: Microsoft Word is a rich editing platform which allows users to automate tasks by recording a set of common actions that can be triggered on demand. These "Word Macros" and tasks are encoded and interpreted in Microsoft's Visual Basic scripting language.

Due to security concerns, firewalls strictly limit the ability of Word to access the Internet. To bypass this issue and allow beacons to be passed to the server, the local browser can be invoked from within a Word macro. Information such as local machine directories, user's credentials, and the machine's IP address can all be encoded and passed through the firewall by the local browser agent. As long as the document is digitally signed, Word will allow some level of macro activity on the host. The macros are automatically triggered upon opening the document.

Due to their misuse in the past, macros are sometimes disabled on the local system. We have embedded an alternative method suggested by [22] to allow a beacon to be triggered. A remote image is embedded in the decoy document and rendered by Word's document browser when the user views the document. The D^3 website supports this feature by intercepting image requests and parsing out stealthy tokens embedded in the request.

3) Adobe PDF: PDF is an open standard published by the ISO and is supported on most platforms and configurations. In the latest version, Adobe has embedded a Javascript interpreter in the application to be able to verify form data as the user enters them in. We leverage this feature to issue a data request upon the initial opening of the document through some Javascript code. The beacon contains the token to identify the document so that the system can track individual documents as they are read across different systems. Due to security concerns, the latest releases of Adobe Reader now prompt the user for permission to contact a remote server. On the users own host, this action can be "memorized" so that subsequent requests do not issue warnings. Earlier versions of the Adobe Reader do not show an alert, allowing them to silently contact the SONAR server also on remote systems. Not all readers support the Javascript PDF so this particular beacon is limited on those systems where the default reader is not Adobe.

The D^3 site includes a tutorial guiding the user on how to generate, download, and open a newly generated decoy document to "memorize" beacon triggers to allow silent communication on the host.

4) Embedded Marker implementation: Beacon documents contain embedded markers that a host or network sensor may detect either when documents are loaded in memory or egressed in the open. The markers are constructed as a unique pattern of word tokens uniqly tied to the document creator. The sequence of word tokens are stealthy embedded within the beacon document's meta-data area or reformated as comments within the document format structure. Both locations are ideal for embedding stealthy markers since most rendering programs ignore these parts of the document. The embedded markers can be used in Snort signatures for detecting exfiltration. The technique is partly based on "Chaffing and winnowing" which is an attempt to maintain network confidentiality in the absence of encryption [29].

V. Evaluation

A. Masquerade detection using Decoy Documents as Bait

We have defined the general properties that decoys should have and discussed how we may measure these properties, but here we focus on the most important property: *detectability*. Under ideal testing conditions, decoy efficacy could be shown through deployment on true operational systems either within an enterprise environment, or on personal computers, by the number of attacks they are able to detect or thwart (they have a deterrence effect). However, given reasonable time limits, the infrequency of attacks within the insider threat model makes this approach impractical within a university environment. As we mentioned we are now seeking a larger user population to study and measure decoy generation over time.

Another approach to evaluation is a user study in which users are organized and asked to evaluate decoys based on each of the key decoy properties mentioned earlier. We take human evaluation to be the gold standard of evaluation since the human mind is the ultimate target of our decoys. That is, we wish to show how well our decoys can induce deception on human test subjects. One of the challenges of conducting a traditional user study lies in the logistics of obtaining volunteers. In our methodology, we attempt to reduce this challenge by leveraging external attackers to serve as participants in our study on masquerade detection. To do so, we "invite" attackers (or more accurately, bamboozle them) into our study by attracting them with a set of vulnerable systems on the university network, which also serve as our testing platform.

Our test platform is embedded within a honeynet [9]. It consists of several virtual machines running Linux and configured with Sebek [10] to capture attacker activities including commands and file references. In order to limit potential damage from system compromise and still allow for testing, we configured the honeynet to allow all incoming connections while restricting the number of outgoing connections.

The virtual machine hosts within the honeynet were configured with accounts and home directories for three decoy usernames. To make the environment as real as possible, genuine data from personal accounts on other systems were loaded into each of the home directories. We changed name references within the data to reflect those of the appropriate decoy users. In total, our phony user accounts contained 15 or more directories and 50-100 files. The hosts were then seeded with several of D^3's decoy files using the decoy distributor utility. The decoy files were generated to have conspicuous names such as "stolen passwords", "credit card", "private data", and "Gmail AccountInfo", but were distributed within the polluted home directories of the decoy accounts, making the environment as real as possible.

To lure test subjects into the study, our initial approach was to use attackers that attempt to gain internal access via password scanning. Password scanning attacks are common on the university network, where attempts on a typical machine are in the range of thousands per day. To enable attacker access, we conducted a short study to first determine the most common usernames and passwords (excluding those for root and actual users) used in these attempts. We created accounts with several of these usernames and passwords, to quickly learn that this breed of attacker was not going to suffice for our user study; their sole purpose seemed confined to creating zombies for botnets. While this may be a valid threat to study while evaluating decoys [7], allowing bots to operate on the university network poses too much risk.

In our second and more aggressive approach, we narrowed our recruitment effort to web forums and IRC channels with the expectation and hope that we would get fewer attacks involving botnets. In this approach, we selected several high volume forums to solicit volunteers and posted variations of invitations with messages that included hostnames, usernames, and passwords. The idea was to provide just enough innocent-looking information from a novice to lure people into our machines without providing direct evidence that we were conducting a deception-based experiment. Note that we deliberately omit the names of the forums used and the exact details of the messages, as this is an ongoing study.

While our methodology could, in theory, provide anyone with access to our test platform, by selectively choosing the location of postings and contents postings, we expected to recruit two primary classes of individuals:

- Legitimate and generally curious computer-savvy individuals. These users have no interest in extending privileges in an unauthorized way, but participate in the study out of curiosity, as there is no other incentive.
- Unscrupulous opportunistic hackers who attempt to extend their network access by whatever means afforded to them. These individuals are enticed by our posting as they see our machines as low "hanging fruit" in their targeting campaign.

In either case, we believe these individuals to be suitable candidates for our study (with one caveat mentioned later). Both classes of individuals can be used in measuring the enticement property of decoys. We measure this by examining the behavior exhibited in file access, both with respect to the particular files a user attempts to read and in the order in which the files are read. For example, if all users consistently read the same file first, we know the file must indeed be enticing.

In regards to indistinguishability of the decoys, we note that the content of these decoys contains bait information in the form of monitored credentials on real systems. Certainly, if our attackers take the time to use the decoy credentials, there is an implication that they must also be believable. More importantly though, if they use the credentials and we detect their use, we have also answered the most important question of - can we *detect* the attacker? Note that the first class of the individuals is by definition, not useful for this part of the study. That is, attempting to use credentials found on our machines is clearly an illicit activity, which they would not partake in. Unfortunately, we do not have a good way of distinguishing the two classes, so we get slightly skewed results.

Over the span of the first week [4], our hosts netted 20 unique users, determined anecdotally by source IP address and reinforced by the unique behaviors exhibited by each. The length of user sessions ranged from minutes to hours. The users that spent hours were focused on activities that included writing code and attempting to install other software, presumably with malicious intent. The most common activities exhibited for users upon login included activities such as: determining who else was on the machine, checking and deleting the command history, and exploring the system (i.e., reading the most *conspicuous* files). We note that in almost all cases, our decoys were among the first few files viewed by users upon initial login (as noted, viewing the history file was also popular). The most significant observations were made for 6 of the 20 unique users:

1) There were three attempts to use Gmail credentials that were contained in a decoy document, which triggered an alert on SONAR.
2) One person succeeded in changing the password on a bogus Gmail account, which also triggered a SONAR

[4]We note that most attacks occurred within the first 8 hour period after posting.

alert.

3) There were at least two attempts to exfiltrate decoy files with Secure Copy and Secure FTP, one named "stolen-passwords" that contained credentials to the university systems.
4) There was one attempt to use the university credentials contained in the "stolenpassword" file, which we were alerted to by the monitor that signaled an alert to SONAR.

We take these results as evidence that D^3/SONAR indeed has value as a defense against masqueraders. While only 5 of the 20 users sounded an alarm on SONAR, we emphasize that our methodology did include an unknown proportion of benign users. Furthermore, the focus of study was on masquerade detection; admittedly, we do not yet have a good way of evaluating our system on traitors, but this will be the focus of future work.

One flaw in our evaluation methodology that was revealed during testing was that we allowed users to make changes to the file system. We did this deliberately to increase the realism of the environment in the experiments. The problem this created was that it made decoy defense vulnerable to deletion (e.g., several of our visitors executed wholesale deletion of files with "rm -rf *") . This poses a problem in our testing methodology, but not necessarily in practice. That is, the act of deleting files is in itself a detectable behavior that would alert monitors of suspicious behavior.

In this study, we omitted testing decoy documents with embedded beacons. The honeypots set up to attract remote attackers were stripped down Linux machines that had no installed applications necessary to open and render the decoy documents. We believe the value of beacon documents to be self-evident. We encourage the reader to visit and test the D^3 site, and participate in our planned longitudinal study. In the next section we describe tests of the beacon implementation on multiple hosts.

B. Beacon Implementation Tests

To test the robustness of the beacon implementations we wanted to them it on the most common configurations of operating systems and document viewers. To this end, we contacted a random group of users across the Internet and sent them each two types of beacon documents along with a request that they open them as part of a benign experiment. The results of tests conducted on PDF and Word beacons are presented in Table 1 and 2 below. These results are a representative sample of real users across multiple hosts accessing the beacon documents. For the most part the beacon technology works well on the windows platform while not as well on Mac and Linux operating systems. The reason is that the default PDF reader is not Adobe's and does not execute Javascript embedded within the documents. Similarly, Word document beacons do not work when applications other than Microsoft Word (e.g. OpenOffice or Google Docs) are used to open them. We are currently researching ways to address these limitations and will focus on them in future work.

TABLE I
PDF BEACON TEST RESULTS

OS	Application	#Tests	#Pings
Windows XP	Adobe	6	6
Windows Vista	Adobe	4	4
Mac OS	Preview	1	0
Mac OS	Adobe	1	1
Ubuntu	Evince	1	0

TABLE II
WORD BEACON TEST RESULTS

OS	Application	#Tests	#Pings
Windows XP	Word	5	4
Windows XP	GoogleDocs	1	0
Windows Vista	Adobe	4	4
Mac OS	Word	2	2
Linux	OpenOffice	1	0

VI. CONCLUSION

Our work focuses on the study and creation of bait information with the aim of exposing or thwarting the exploitation of exfiltrated information. Although the use of bait information and similar trap-based defenses is well known, most of those efforts have focused either on artifacts that are logically separate from the operational systems (e.g., honeypots [23]) or on low-level snippets of information created manually (e.g., fake database records [24]). The D^3 system is a scalable and automated trap-based defensive system that forces attackers to expend considerable effort to identify realistic useful information from purposely planted bogus information intended to deceive. Naturally, the probability of exposing a malicious insider with trap-based defense tactics increases with the amount of decoy information that is generated and disseminated. D^3 offers the novel service of automatically creating and managing decoy documents, enabling the throttling of bait based on the desired protection level or cost (e.g., *interference*) one is willing to pay.

REFERENCES

[1] Bell D. E. and LaPadula L. J., "Secure Computer Systems: Mathematical Foundations". MITRE Corporation, 1973.
[2] Bell, J. and Whaley, B. Cheating and Deception, Transaction Publishers, New Brunswick, NJ. 1982.
[3] Butler, J., Sherri S., "Security: Spyware and Rootkits", Login, Vol 29, No 6, December 2004.
[4] Clark, D. D. and Wilson, D. R., "A Comparison of Commercial and Military Computer Security Policies". IEEE Symposium on Security and Privacy, 1987.
[5] Demers, A., Gehrke, J., Hong, M., Panda, B., Riedewald, M., Sharma, V., White, W., "Cayuga: A General Purpose Event Monitoring System". CIDR 2007.
[6] Detristan, T., Ulenspiegel, T., Malcom Y., and Von Underduk, M. S. "Polymorphic Shellcode Engine Using Spectrum Analysis". Phrack 11, 61-9 (2003).
[7] Friess, N., and Aycock, J.,"Black Market Botnets", Department of Computer Science, University of Calgary, TR 2007-873-25, July, 2007.
[8] Hoang, M. "Handling Today's Tough Security Threats", Symantec Security Response, 2006.
[9] The Honeynet Project. `http://www.honeynet.org`
[10] The Honeynet Project, "Know Your Enemy: Sebek, A Kernel based data capture tool", November, 2003.

[11] Honeypots. http://www.honeypots.org/

[12] Honeypot Mailing List, Security Focus. http://www.securityfocus.com/archive/119

[13] Katz, John and Yehuda L., Introduction to Modern Cryptography, Chapman and Hall CRC Press, 2007.

[14] Kravets, D., ”From Riches to Prison: Hackers Rig Stock Prices”, Wired Blog Network, September, 2008. http://blog.wired.com/27bstroke6/2008/09/from-riches-to.html

[15] Krebs, B., “Web Fraud 2.0: Validating Your Stolen Goods”, The Washington Post, August 20, 2008.

[16] Li W., Stolfo S. J., Stavrou A., Androulaki E., and Keromytis A., ”A Study of Malcode-Bearing Documents”. DIMVA, 2007.

[17] Maloof, M. and Stephens, G. D., “ELICIT: A System for Detecting Insiders Who Violate Need-to-know”. Recent Advances in Intrusion Detection (RAID), 2007.

[18] McRae, Craig M. and Vaughn, Rayford B. “Phighting the Phisher: Using Web Bugs and Honeytokens to Investigate the Source of Phishing Attacks”, Proceedings of the 40th Hawaii International Conference on System Sciences, 2007.

[19] Nong Ye, “Markov Chain Model of Temporal Behavior for Anomaly Detection”, Proceedings of the 2000 IEEE Workshop on Information Assurance and Security, United States Military Academy, West Point, NY,6-7 June, 2000.

[20] Richardson R., “CSI/FBI Computer Crime and Security Survey”, 2007.

[21] Song Y., Locasto M. E., Stavrou A., Keromytis A. D., and Stolfo S. J.. “On the infeasibility of modeling polymorphic shellcode”. In Proceedings of the 14th ACM conference on Computer and communications security (CCS07), pages 541-551. ACM, 2007.

[22] Smith, R. M., “Microsoft Word Documents that Phone Home”, Privacy Foundation, August, 2000.

[23] Spitzner, L., “Honeypots: Catching the Inisder Threat” Proceedings of ACSAC. Las Vegas, December, 2003.

[24] Spitzner, L., “Honeytokens: The Other Honeypot”, Security Focus, 2003.

[25] Stoll, C. The Cuckoo’s Egg, Doubleday, 1989.

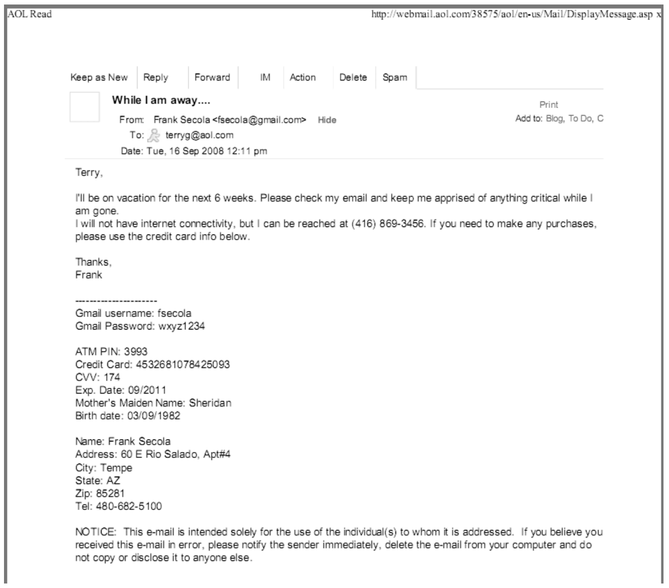
[26] Symantec. Global Internet Security Threat Report, April 2008. Trends for July –December 07.

[27] Yuill, J., Zappe M., Denning D., and Feer F.. “Honeyfiles: Deceptive Files for Intrusion Detection”, Proceedings of the 2004 IEEE Workshop on Information Assurance, United States Military Academy, West Point, NY, June 2004.

[28] Yuill, J., D. Denning, Feer, F., “Using Deception to Hide Things from Hackers : Processes, Principles, and Techniques”, Journal of Information Warfare, 5(3):26-40, November, 2006.

[29] Rivest L. Ronald. ”Chaffing and Winnowing: Confidentiality without Encryption”, MIT Lab for Computer Science, March 18, 1998

APPENDIX



While I am away....

From: Frank Secola <fsecola@gmail.com>
To: terryg@aol.com
Date: Tue, 16 Sep 2008 12:11 pm

Terry,

I'll be on vacation for the next 6 weeks. Please check my email and keep me apprised of anything critical while I am gone.
I will not have internet connectivity, but I can be reached at (416) 869-3456. If you need to make any purchases, please use the credit card info below.

Thanks,
Frank

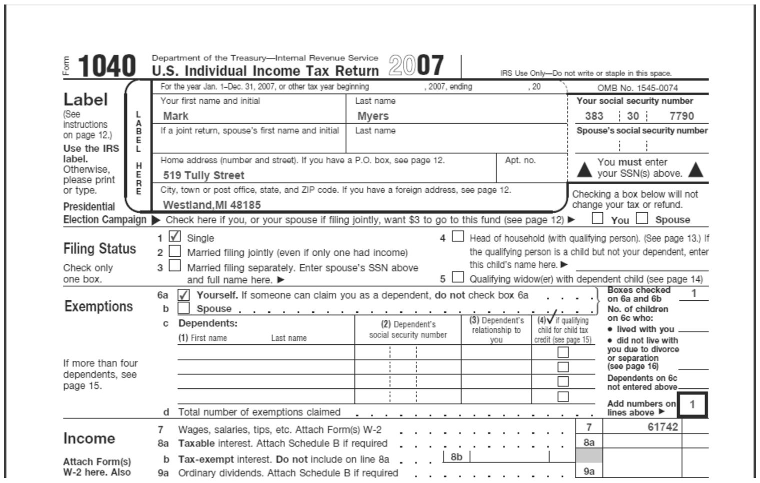
Gmail username: fsecola
Gmail Password: wxyz1234

ATM PIN: 3993
Credit Card: 4532681078425093
CVV: 174
Exp. Date: 09/2011
Mother's Maiden Name: Sheridan
Birth date: 03/09/1982

Name: Frank Secola
Address: 60 E Rio Salado, Apt#4
City: Tempe
State: AZ
Zip: 85281
Tel: 480-682-5100

NOTICE: This e-mail is intended solely for the use of the individual(s) to whom it is addressed. If you believe you received this e-mail in error, please notify the sender immediately, delete the e-mail from your computer and do not copy or disclose it to anyone else.

Fig. 1. Decoy sample email message with embedded gmail account information.



Form **1040** Department of the Treasury—Internal Revenue Service
U.S. Individual Income Tax Return 2007 | IRS Use Only—Do not write or staple in this space.

For the year Jan. 1–Dec. 31, 2007, or other tax year beginning , 2007, ending , 20 | OMB No. 1545-0074

Label (See instructions on page 12.) **Use the IRS label.** Otherwise, please print or type. LABEL HERE

Your first name and initial: Mark	Last name: Myers		**Your social security number**: 383 30 7790
If a joint return, spouse's first name and initial	Last name		**Spouse's social security number**
Home address (number and street). If you have a P.O. box, see page 12.: 519 Tully Street		Apt. no.	You must enter your SSN(s) above.
City, town or post office, state, and ZIP code. If you have a foreign address, see page 12.: Westland,MI 48185			Checking a box below will not change your tax or refund.

Presidential Election Campaign ▶ Check here if you, or your spouse if filing jointly, want $3 to go to this fund (see page 12) ▶ ☐ You ☐ Spouse

Filing Status Check only one box.
1 ☑ Single
2 ☐ Married filing jointly (even if only one had income)
3 ☐ Married filing separately. Enter spouse's SSN above and full name here. ▶
4 ☐ Head of household (with qualifying person). (See page 13.) If the qualifying person is a child but not your dependent, enter this child's name here. ▶
5 ☐ Qualifying widow(er) with dependent child (see page 14)

Exemptions
6a ☑ **Yourself.** If someone can claim you as a dependent, **do not** check box 6a — Boxes checked on 6a and 6b: 1
b ☐ Spouse
c **Dependents:**

(1) First name Last name	(2) Dependent's social security number	(3) Dependent's relationship to you	(4) ✓ if qualifying child for child tax credit (see page 15)
			☐
			☐
			☐
			☐

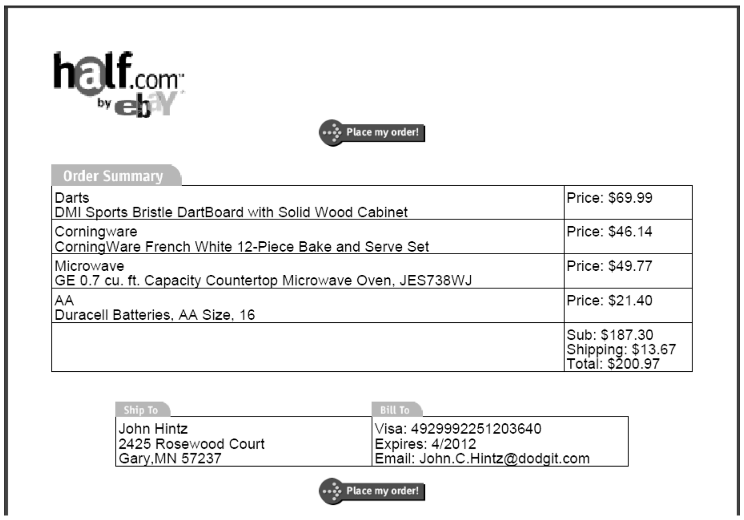
If more than four dependents, see page 15.

No. of children on 6c who:
● lived with you
● did not live with you due to divorce or separation (see page 16)
Dependents on 6c not entered above
d Total number of exemptions claimed — Add numbers on lines above ▶ 1

Income Attach Form(s) W-2 here. Also

7	Wages, salaries, tips, etc. Attach Form(s) W-2	7	61742
8a	**Taxable** interest. Attach Schedule B if required	8a	
b	**Tax-exempt** interest. **Do not** include on line 8a — 8b		
9a	Ordinary dividends. Attach Schedule B if required	9a	

Fig. 2. Decoy tax document with bogus user information.



Darts DMI Sports Bristle DartBoard with Solid Wood Cabinet	Price: $69.99
Corningware CorningWare French White 12-Piece Bake and Serve Set	Price: $46.14
Microwave GE 0.7 cu. ft. Capacity Countertop Microwave Oven, JES738WJ	Price: $49.77
AA Duracell Batteries, AA Size, 16	Price: $21.40
	Sub: $187.30 Shipping: $13.67 Total: $200.97

Ship To	Bill To
John Hintz 2425 Rosewood Court Gary,MN 57237	Visa: 4929992251203640 Expires: 4/2012 Email: John.C.Hintz@dodgit.com

Fig. 3. Decoy eBay receipt.