Fox in the Trap: Thwarting Masqueraders via Automated Decoy Document Deployment

Jonathan Voris
New York Institute of Technology
Computer Science Department
jvoris@nyit.edu

Jill Jermyn
Columbia University
Computer Science Department
jill@cs.columbia.edu

Nathaniel Boggs
Columbia University
Computer Science Department
boggs@cs.columbia.edu

Salvatore Stolfo
Columbia University
Computer Science Department
sal@cs.columbia.edu

ABSTRACT

Organizations face a persistent challenge detecting malicious insiders as well as outside attackers who compromise legitimate credentials and then masquerade as insiders. No matter how good an organization's perimeter defenses are, eventually they will be compromised or betrayed from the inside. Monitored decoy documents (honey files with enticing names and content) are a promising approach to aid in the detection of malicious masqueraders and insiders. In this paper, we present a new technique for decoy document distribution that can be used to improve the scalability of insider detection. We develop a placement application that automates the deployment of decoy documents and we report on two user studies to evaluate its effectiveness. The first study indicates that our automated decoy distribution tool is capable of strategically placing decoy files in a way that offers comparable security to optimal manual deployment. In the second user study, we measure the frequency that normal users access decoy documents on their own systems and show that decoy files do not significantly interfere with normal user tasks.

1. INTRODUCTION

Malicious insiders and stealthy attacks that allow adversaries to masquerade as insiders using authorized credentials pose serious threats to information security. Most security measures attempt to prevent unauthorized access and exploitation of system vulnerabilities. These measures will eventually fail, allowing an attacker to steal authorized credentials and access data in a manner that is indistinguishable from a real user. Organizations have the difficult task of determining whether authorized credentials are in fact being used for legitimate or malicious purposes. Many approaches to solving this problem are either prohibitively expensive or inadequate. Provably tracking all sensitive data and preventing its exfiltration requires it to be tagged, requiring a substantial overhead and the modification of all programs which use that information. Commercial Data Leak Prevention (DLP) systems offer a partial but easily evaded solution.

Decoy documents, which are realistic looking files with fake data that issue alert beacons when accessed provide an inexpensive means for detecting malicious masqueraders and insiders. Remotely triggered "beaconized" decoy documents enhance DLP, offering Data Loss Alerting (DLA) services. Since decoy documents are fake, legitimate users have no need to open them. Legitimate users are familiar with the surrounding file system, so they are likely to remember which documents are decoys and access these files less often than attackers. On the other hand, masqueraders or insiders who are unfamiliar with the surrounding documents will be far more likely to open decoys. Previous large-scale user studies support this claim [9]. The alert threshold can be set such that rare accesses are ignored while attackers that access more decoy documents than normal users are caught. Once a sufficient number of decoys have been touched, an organization can take action and investigate the credentials that were used to access the spurious documents.

While handcrafted decoy documents could be painstakingly created by each user to entice attackers, large organizations with managed security services require a scalable solution. In this paper, we focus on the automated distribution of decoys. The distribution problem includes automated creation of decoy files, file naming conventions and their arrangement in the target file system to be protected, and the precise number of placed decoys. Our goal is to increase the chance of detecting insider malfeasance while minimizing the number of accidental touches by legitimate users using scalable methods.

The central contribution of this work is a Decoy Distributor Tool (DDT) which can be used to disseminate decoys throughout a file system with minimal manual involvement. This reduces the time required to deploy a system of decoy document insider threat sensors making the use of decoy documents practical for large organizations. Our solution does not require any prior knowledge of the structure or

EUROSEC April 21, 2015, Bordeaux, France

content of the file system in which decoys are to be placed. The DDT gathers this information automatically and selects target directories, either manually or automatically, and applies file naming conventions from the environment. To implement and test our solution, we report on two user studies to evaluate various decoy document placement strategies using the DDT.

In the first study, we compare two placement strategies with a baseline case of common user-selected placements. We present additional results with a slightly modified masquerader scenario in which we specifically inform users that decoy documents are present. The results of these studies provide evidence that our automated decoy distribution application is capable of placing decoys in a manner that offers comparable security to manual deployment. Based on our observations, we can assert with 95% confidence that the difference between the number of decoys accessed in our two automated deployment scenarios was statistically significant.

Our second user study measures the extent to which automatically placed decoy documents interfere with the habits of legitimate users. We accomplish this by having users deploy decoys on their own systems. We then monitor the frequency at which the decoys are encountered using host sensor software. A large population user study shows that decoys are touched frequently when first introduced, but false positive accesses decay rapidly thereafter. This provides strong evidence that decoys do not interfere with normal user work. The false positive results from this user study were mainly collected for a future paper that explores the use of machine learning methods to detect abnormal behavior. This paper is focused entirely on the method of distributing decoys and the impact that may have on overall performance.

2. RELATED WORK

Using deception to gain an advantage over enemy forces 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 [10]. The first individual to apply this principle to the security of computer systems is Cliff Stoll. Stoll utilized deception by setting up fake computing systems to detect hackers who were attempting to steal secrets from Lawrence Berkeley National Laboratory [3].

Computers that are mainly meant to attract the attention of malicious actors are often called "honeypots." Entire networks of spurious machines are known as "honeynets." These systems are usually set up as though they are a standard part of a broader network architecture, yet in reality they lack meaningful data and are cordoned off from valuable network resources. Honeypots and honeynets can be very effective tools for detecting and monitoring external threats, but have a limited potential to defend against insiders since this class of attackers typically already has the ability to connect to the legitimate portions of a network.

The concept of a "honeytoken" was introduced by Spitzner as a way of adapting honeypots to the context of insider threat detection [8]. Rather than spurious machines or networks, honeytokens are pieces of information that are designed to garner adversarial attention but actually contain no data of value. Examples of honeytokens include illegitimate access credentials or personally identifiable information. Yuill et al. expanded this concept by creating the term "honeyfiles" to refer to documents which contain such enticing information [6].

Detecting insider threats is an arduous task that many traditional defenses have not mastered. However, decoy files have proven successful in this challenge since they possess the ability to detect when attackers begin accessing files and are effective even after all other defenses have been evaded. The nontrivial problem of creating, distributing, and managing these files has been investigated in prior research. As described in [2], Bowen et al. developed the Decoy Document Distributor (D^3) System, a tool for generating and monitoring decoys.

D^3, also known as FOG, is a web site that allows users to download files, such as tax documents and receipts, that appear authentic but actually contain spurious information [4]. When opened, these documents establish a "beacon" connection with the D^3 server, which can then issue an appropriate alert to the document's owner. Additionally, users can upload their own legitimate sensitive documents to the distributor web site to be "beaconized" so they issue alerts through the D^3 system when opened. Since they contain actual content, documents that are modified in this way have the advantage of not interfering with the workflow of legitimate users.

To aid in the process of creating decoys, the authors of [2] also established a set of properties that can be used to assess the effectiveness of decoys. These documents should be *believable* and appear to be legitimate. That is, adversaries should have a difficult time discerning decoy files from authentic ones. In addition to being realistic, decoy content should also appear *enticing* to malicious actors. Documents that are attractive targets, such as those containing financial or personally identifiable information, are much more likely to be accessed, increasing the odds that they will detect an attack. For similar reasons, decoys should be *conspicuously* placed in easy-to-find locations.

Since the main idea behind decoys is to monitor adversarial behavior, a critical characteristic that all decoy documents must possess is *detectability*. [2] established a variety of ways in which decoy access can be observed, including the aforementioned beacons when documents are initially accessed, using host sensors, and by embedded credentials. Decoys must also lack any shared attributes that an attacker could test for to discern true documents from fake ones. In other words, there should be a high degree of *variability* between decoy documents.

The attributes listed thus far all pertain to the relationship between adversaries and decoy files, but it is also important to consider how typical users will interact with them. Decoy material should be *non-interfereing* in that it should not hamper the usual workflow of legitimate users of the system in which it is placed. As a result, it is just as important that normal users are able to *differentiate* between decoy data and real data as it is for the decoy content to appear believable to adversaries.

With these properties in mind, Ben Salem and Stolfo conducted a user study to test the success of various decoy deployment techniques [9]. This process allowed them to identify tradeoffs between these properties that are dependent on the type of attack that is being defended against. In addition, they proposed methods for increasing the allure of

decoys to insider attackers without fooling authentic users and interfering with their expected workflow [9].

An additional body of research has recently proposed using deception in approaches other than honeyfiles to improve system defenses. Jules and Rivest suggest using "honeywords", or false passwords, for improving the security of hashed passwords [7]. Attackers who attempt to invert hashed passwords will not be able to tell whether their uncovered passwords are legitimate or honeywords. A password-checking server can differentiate between honeywords and true passwords and set off an alarm for each submitted honeyword. [1] proposes a methodology for reformulating security patches into "honeypatches", which are software patches that confuse attackers by making it difficult to determine if an attack has succeeded or failed. When their system detects an exploit attempt, the honeypatch redirects the attacker to an unpatched decoy where the attack is allowed to succeed.

In brief, there has been much previous work in the area of using deceptive data to lure out potential adversaries. Yet prior research has not addressed the placement of decoy honeyfiles in a systematic fashion, which is the primary contribution of this work. Existing research has also not attempted to systematize the automatic generation and management of decoys, which is a key desiderata in order to scale to the sizable file systems that are employed by large organizations.

3. THREAT MODEL: USER CREDENTIAL THEFT AND MASQUERADER ATTACK

The security measures discussed in this paper are primarily intended to detect attacks that are launched by masqueraders. This class of adversaries uses illicit means, such as credential theft, to act as authorized individuals in order to gain access to computer systems. Because they pose as authentic users, masqueraders can be viewed as a special case of inside attackers. Unlike insiders, though, malicious masqueraders typically have limited knowledge of the systems they are attacking. For our first user study, we assume that a masquerader has no information regarding the internal file systems and security measures. Once decoy documents become more widely deployed, masqueraders will be more likely to be aware that decoys are present. To test the effect of additional knowledge, we follow up with another user study, presented in Section 6.3, where we inform participants that decoy documents are present, but not how they are configured.

Though we focus on masqueraders, our techniques may slow down insiders as well. An insider who is aware of the precise decoy generation and deployment algorithms that are in use will be able to circumvent these traps, but may have to be more careful and slower in exfiltration in order to avoid accidentally tripping over decoys. In this way, the use of decoy technology changes the paradigm from one where defenders have to be perfect to one where attackers have to be perfect and not trip over more decoys than normal users. Blocking such difficult insider threats entirely is beyond the scope of this work.

We model adversaries by means of user scenarios that we describe in Section 5.2. The aim of masqueraders and malicious insiders is to exfiltrate sensitive information from within an organization. For the purposes of our IRB approved study, the scenarios that we provided to our subject users indicate that their end goal is to steal financial, personal, or corporate information. This observation motivated our folder naming conventions used while deploying decoys, which we discuss in Section 4.

4. DECOY DOCUMENT DISTRIBUTION APPLICATION DESIGN

We created the DDT system to automatically place decoy documents within a target computer's file system. This tool 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 a decoy document generation web site [4] through the DDT. 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 that serves as the root from which target locations are selected.

A study performed in [9] demonstrates that the placement of decoy documents greatly affects the probability a user will access them. Locations in which decoys are placed in the file system should be selected so that they 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 a variable number of folders (we selected ten as a reasonable value for experimentation) with the most recently accessed documents as well as a variable number (we again used ten for our tests) of folders containing the greatest number of files with the following 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. Studies of exploitation incidents suggest that these directories are the most probable targets of attack [5].

As with location, the names of documents and folders directly influence how enticing they look to adversaries. The DDT creates new folders with appealing names and disperses decoys across them. Our tool also creates filenames which are realistic and enticing according to a two-step process. First, the decoy generation system [4] assigns filenames using a standard template of the form *[personal name]-[file type]-[random number].pdf*. The DDT then uses one of three methods for naming decoy documents to blend them with existing documents in a folder. The first naming method selects an existing file in the target directory and appends either "-final" or "-updated" to the end of the filename. Another naming method appends a date string to the end of a randomly-selected existing filename in the target directory. The final naming approach blends filenames by observing the most commonly used delimiter in the target directory and applying it to the filename created by the decoy generation system. The DDT calculates the delimiter used most often in the target directory and modifies a decoy's filename to use this delimiter. Although these were the three naming conventions we employed during our user study, the DDT

allows customizable schemes that can be tailored depending on the target system.

5. EVALUATION

We conducted an IRB-approved user study of the DDT system to determine how well it operates in comparison to the manual placement of decoys by users. The key performance objective of a decoy deployment is to detect insider attacks without substantially increasing the rate at which legitimate users trigger decoy beacon alerts. We thus set out to design experiments that would measure both the masquerader and normal user decoy access rates that resulted from manual deployment and automated deployment via our distributor prototype.

5.1 Experimental Framework

The first step in establishing the experimental framework was to set up a target computer system containing decoys with which users could interact. Rather than establishing a new environment, we utilized an existing system from our laboratory. This was done because the desktop machine we selected had already been in use for several years. As a result, it had accrued an authentic accumulation of data and applications that would have been difficult to replicate in as realistic of a fashion.

Next, we used the DDT prototype to download 40 decoys from our decoy generation web site [4]; this decoy quantity was selected to facilitate comparison with previous experiments [9]. The same 40 decoy files were utilized for all of the tested deployment techniques in order to minimize confounding variables between the different setups. We then created three distinct deployment arrangements using these documents: an "integrated" deployment pattern in which decoy documents are placed in existing folders alongside real files, a "separated" deployment situation where decoy files are placed on their own in new subfolders with enticing files created by the deployment tool, and a "manual" deployment method where the distributor tool is actually not employed at all. Rather, decoy files are manually placed in the directories that are most likely to be accessed by an adversary as determined in [9].

It was necessary to monitor our participants' decoy access patterns in order to determine the decoy access rate that resulted from each of these deployment strategies. To achieve this in our masquerader study, we linked our decoy documents to a study administrator's email address using the decoy generator web site beacon management function [4]. Whenever a decoy in our testing environment was accessed, we received an email notification containing the name of the decoy file, the location of the document within the test file system, and the time at which the access event occurred. In order to measure the decoy access rate in our normal user studies, a longer term behavior monitoring solution was necessary. We therefore requested that volunteers install host sensor software which we developed in order to gather usage data for this study.

5.2 Experimental Design

We solicited volunteers for our experiment under an IRB-approved protocol from a population of Columbia University students. Participants were recruited by posting flyers around our campus and sending out email announcements advertising the study.

The participants in the human user study were asked to perform tasks on the target computer as though they were masqueraders. Though it would have been preferable to observe real masqueraders in action, this was not possible in a laboratory setting. In order to encourage our volunteers to act adversarially, we gave each participant a formal detailed scenario describing how they were undergoing serious financial hardship and desperately needed a new source of money. The instructions noted that the test subject was experiencing personal issues with a coworker in the same office space, and that this coworker recently received an unfair promotion. The scenario concluded by stating that the participant had decided that their only possible recourse was to resort to stealing any financially lucrative information that they could find on their coworker's machine. They were provided with a 15 minute window of opportunity to do so while their colleague was out of the office during a lunch break.

We prepared two variants of this scenario that differed in precisely one way, namely the level of detail that we provided regarding the computer security measures that were in place on the coworker's system. One scenario did not mention anything about decoy documents, but instead only vaguely stated that the system was being monitored. The second version of the scenario explicitly mentioned that fake documents had been placed on the targeted coworker's machine, and that these documents should be avoided lest they raise an alert.

We designed our experiment as a "between subjects study," meaning that each participant tested a single combination of one of the aforementioned deployment methods and user scenarios. Our study proceeded as follows for each test subject. After "pseudorandomly" determining which experimental configuration the volunteer would receive, the virtualized test machine was restored to a snapshot of the corresponding decoy deployment setup. The participant was then given the scenario script containing the level of security detail that went along with his or her test setup. Each volunteer then proceeded to use the test virtual machine for a 15 minute interval as though they were searching through a coworker's unattended machine.

We also sought to experimentally determine the normal user decoy access rates for our decoy deployment methods. That is, we wanted to determine how often a computer's owner would open decoys that were placed within their own file system by the DDT. Since the error rates of manual decoy placement were already analyzed in a prior study [9], we chose to focus on the normal user access rate using our automated deployment methods. We recruited a separate group of volunteers to measure the frequency of these occurrences.

Each of our test subjects was provided with detailed instructions which asked them to deploy 40 decoy documents, which was the largest quantity tested in [9], across their file system using the DDT in accordance with one of our two automated scenarios. Each user was then asked to install a sensor application on their computer which monitored interactions with their file system, including the planted decoys. The total duration of the study was for two months, which amounted to approximately 800,000 log entries of user activity.

6. STUDY RESULTS

6.1 Masquerader Decoy Access Rate Experiment

For this study, we had two automated deployment configurations and one manual deployment setup. Recall from Section 5.1 that the automated decoy deployments were performed using the DDT. The manual decoy distributions, on the other hand, were achieved by having the authors copy decoys one-by-one into the directories that received the most adversarial attention as determined in [9]. For the deployments that utilized the DDT, we also performed tests with two user instruction variations in order to measure the impact of decoy awareness; this is covered in Section 6.3.

6.1.1 Masquerader Decoy Access Rate Data

Figure 1 shows a box plot of the number of decoy access events that occurred while the simulated masqueraders searched through our test system. The volunteers received the scenario variant that did not discuss the presence of decoys on the system in all three of these cases. Each column displays the five-number summary for a particular deployment configuration. The horizontal line in the center of each box represents the median amount of decoy access occurrences, while the top and bottom of the boxes shows the upper quartile and lower quartile of the data, respectively. The "whisker" lines above and below each box show the maximum and minimum number of decoys that were encountered by a single participant.

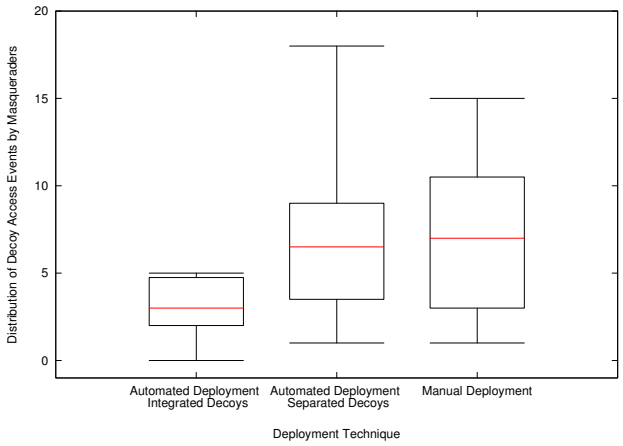

Figure 1: Comparison of Decoy Document Access Event Distributions for Different Deployment Techniques

As Figure 1 illustrates, the least amount of decoy traffic occurred as a result of using the DDT to integrate decoys within existing directories. Using the distributor prototype to place decoys in enticingly named subdirectories, on the other hand, resulted in a distribution of decoy activity that closely resembles that observed when decoy files were deployed manually. Their first quartile, median, and third quartile values are quite similar. However, a higher maximum was observed when the DDT was used. In the case of both DDT deployment with separate directories and manual deployment, all participants touched at least one decoy file. In contrast, some volunteers managed to avoid touching any decoy documents when deployment was automated within existing directories.

A similar plot is shown in Figure 4, but this chart depicts the differences between the set of users who were informed that the system contained decoys and those whose scenarios made no mention of decoys. The DDT was employed for decoy dissemination in all four groups. Columns 1 and 3 of this figure correspond to columns 1 and 2 of Figure 1.

We performed a statistical analysis of the decoy access data that we collected in order to gain further insight into our results. The outcomes of our assessment are presented in Table 1. We first utilized an F-test to determine which data sets shared equal variances. This data was used to determine whether homoscedastic or heteroscedastic Student's t-tests should be used for each pair of data sets. Using this information, we ran unpaired t-tests with one-tailed distributions to determine if the differences between our data sets were statistically significant.

As the data in Table 1 indicates, we can assert with 95% confidence that the difference between the number of decoys accessed in the integrated DDT deployment and the separated DDT deployment was statistically significant. This is due to the resulting p-value of 0.00431 from running the t-test on these sets of data being less than 0.05. By the same logic, the difference between the amount of decoy access events that occurred for users of the automated integrated deployment and the manual decoy setup was also statistically relevant, having earned a p-value of 0.01432. The variations existing between the three remaining pairs of configurations were found to lack statistical significance.

The main advantage of the DDT is that it allows both users and administrators to utilize decoys without going through a lengthy process of determining ideal positions for decoys within a file system and then placing them one at a time. Instead, the only interactive steps that this tool requires in order to efficiently disseminate decoys is for users to specify how many decoy documents to use and a root directory in which they should be placed. Using the DDT will therefore allow users to save time and effort while harnessing the ability of decoys to defend against insider threats.

6.2 Normal User Decoy Access Rate Experiment

Since our proposed solution involves planting decoy documents on users' file systems, one may wonder about the frequency with which these decoys would be encountered by legitimate users. Does the presence of decoy documents interfere with everyday user activity? The results that we present in this section suggest definitively that *decoys do not interfere with normal user operations.*

For our normal user experiment, users were randomly assigned instructions corresponding to the integrated or separated deployment scenario based on the client identifier of their sensor software. Twenty users were placed in the integrated deployment group and seven received documentation for separated distribution. More than 318 hours of data were collected across all volunteers. To determine the decoy access rate for each user, we divided their total number of decoy access occurrences by the duration that their sensor recorded. In order to provide an even comparison to our masquerader results, we expressed this as the average number of decoy hits per fifteen minutes of observation. Figure 2 contains a box plot of this data. Operationally, this means

Comparison	F-Test P-Value	Variance	T-Test P-Value	Significance
Integrated vs Separated	0.00431	Unequal	0.01695	Significant at 95% Confidence
Integrated vs Manual	0.00582	Unequal	0.01432	Significant at 95% Confidence
Separated vs Manual	0.90575	Equal	0.50000	Not Significant

Table 1: Statistical Comparison of Different Test Groups

that legitimate users can be expected to accidentally touch decoys less than 7 times per 8 hour work day. This range of accidental touches, therefore, is what attackers have to stay under to appear normal. If modeled per user, it may be much less for most users.

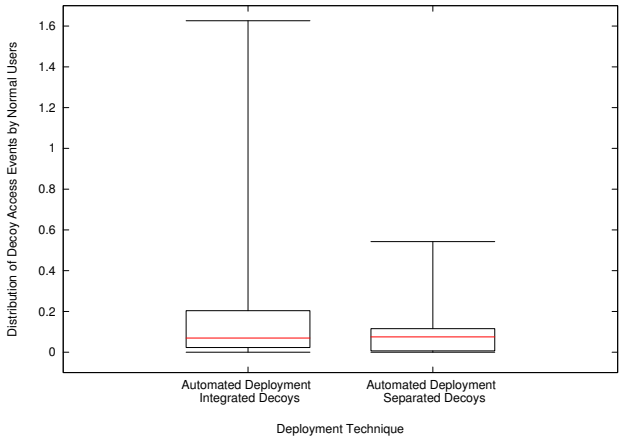

Figure 2: Normal User Decoy Document Access Event Distributions for Automated Deployment Scenarios

Figure 3 shows the average decoy access behavior for users in our long term false positive study. The x-axis represents the study's progression, starting at 0% at the start of the experiment. As the study commences, there is a large uptick in decoy access events. This result conformed to our expectations, as our monitoring software records interactions with decoys as they are distributed to users' file systems. Further, many users typically display an initial curiosity regarding the new files which have appeared on their computer. However, as the figure indicates, user curiosity decays extremely rapidly after the initial placement process, and decoy access events by typical users are extremely rare thereafter.

Based on Figure 2, we can conclude that normal users will occasionally stumble into their own decoy material. In this experiment, 40 decoy files were placed in each users' system, which represents a sizable quantity. Previous research has found that reducing the volume of decoys that are present in a system will reduce the likelihood of these documents interfering with a typical user's workflow [9]. Even at this amount, however, the majority of users accessed their system's decoys at an average rate of far less than one per fifteen minutes. This measurement was based on an initial snapshot of data collected from normal users and as such may be biased by the curiosity exhibited towards these files at

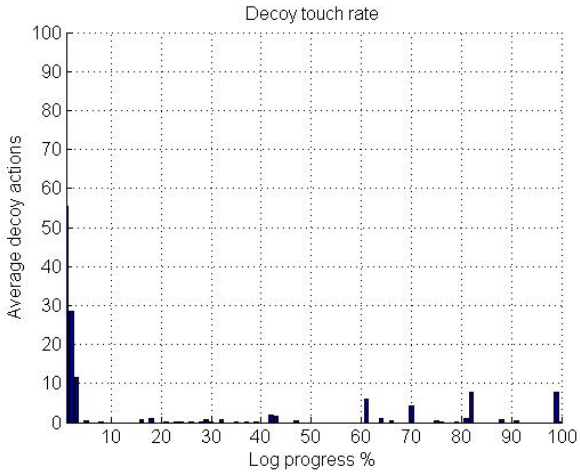

Figure 3: Average Decoy Touch Rate for Users During Long Term False Positive Study

their initial appearance. We anticipate that this touch rate will decay quickly as users become acclimated to the decoy files and hope to corroborate this with a longer term measurement window in the future. In contrast, Figure 1 shows higher access rates by masqueraders across all distribution and awareness scenarios. A simple threshold scheme would therefore be sufficient to differentiate between the bulk of masqueraders and legitimate users.

6.3 Effect of Decoy Awareness

Section 5.2 mentions that we performed an additional experiment to measure the effect of masquerader knowledge on the effectiveness of decoys as a security mechanism. To study this variable, we crafted a variant of our study scenario that differed in precisely one way, namely the level of detail that we provided regarding the computer security measures that were in place on the coworker's system. The original scenario did not mention anything about decoy documents, but the second version of the scenario explicitly mentioned the existence of fake documents and that they should be avoided. This resulted in two additional test cases: automated integrated deployment with users aware of decoys and automated separated deployment with users aware of decoys. We recruited 10 volunteers to run our trial for each of these 2 combinations for a total of 20 additional participants.

Figure 4 depicts the differences between the set of users who were informed that the system contained decoys and

those whose scenarios made no mention of decoys. The DDT was employed for decoy dissemination in all four groups. Columns 1 and 3 of this figure correspond to columns 1 and 2 of Figure 1.

The added scenario information about the presence of decoys did not have a substantial impact on our volunteers' decoy access patterns. As might have been anticipated, notifying users of the presence of decoys slightly decreased the median number of decoy accesses for the separated deployment strategy. Surprisingly, though, it seems to also have shifted the rest of the five-number summary values up for the separated decoy distribution technique.

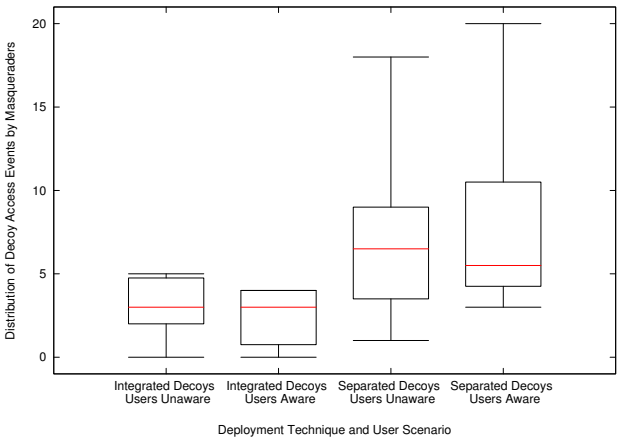

Figure 4: Comparison of Decoy Document Access Event Distributions for Different User Scenarios

6.3.1 Decoy Awareness Discussion

We initially hypothesized that alerting users to the presence of decoys on the test system would cause them to act with more caution and thus come into contact with fewer decoys. Our observations during this study did not fully support this claim, however. As the box-and-whisker plot in Figure 4 shows, the distribution of decoy touches increased slightly when users were aware that decoys were present. These increases were slight, however, and the differences between the data for the scenarios where users were informed and uninformed about decoy documents were subtle and found to lack statistical significance. Thus the more surprising conclusion is that informing our masquerader study participants did not have an impact on their behavior with respect to distinguishing decoys from authentic documents.

Yet for some individuals, receiving a notification about the presence of decoy documents seems to have actually increased the amount of observed decoy activity. The most likely explanation for this counterintuitive result is that while a subset of our participants were cognizant of the fact that the system they were using contained decoys, none of these volunteers had ever encountered a decoy document before. Without any experience with decoys, they had no way of differentiating between authentic and spurious files. A simulation that would more closely resemble an adversarial insider would thus be to expose users to our decoy creation and distribution system prior to asking them to pose as masqueraders; such a study is a goal of our future work.

7. CONCLUSION

To summarize, this paper introduces a prototype decoy distribution application which we have named the Decoy Distributor Tool (DDT). This software is capable of automatically fetching and deploying decoy document files on arbitrary file systems, allowing the use of decoys to scale from single user machines to networks with a great deal of users and computers. The DDT places decoys where they will have the greatest impact by searching for directories that are the most active and therefore the biggest targets for adversaries.

We performed a user study of our tool in which volunteers acted as simulated attackers and decoys were placed on legitimate users' systems. The results of our experiments indicate that the decoys placed by our automated prototype are capable of detecting malicious activity comparable to files that have been manually deployed in predetermined ideal locations. The simulated masqueraders we observed in our study touched precisely the same number of decoys on average, 7.1, in both the automated separate deployment folder and manual deployment scenarios. The DDT can therefore provide system administrators with the full security against insider threats that decoys are capable of providing without going through the time consuming process of manually seeding every computer in their system with decoys.

8. REFERENCES

[1] F. Araujo, K. W. Hamlen, S. Biedermann, and S. Katzenbeisser. From patches to honey-patches: Lightweight attacker misdirection, deception, and disinformation. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 942–953, New York, NY, USA, 2014. ACM.

[2] 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.

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

[4] Columbia University Intrusion Detection Systems Lab. FOG Computing. Available at `http://ids.cs.columbia.edu/FOG/`, 2014.

[5] D. Kostadinov. The Cyber Exploitation Life Cycle. Available at `http://resources.infosecinstitute.com/the-cyber-exploitation-life-cycle/`, 2013.

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

[7] A. Juels and R. L. Rivest. Honeywords: Making password-cracking detectable. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 145–160, New York, NY, USA, 2013. ACM.

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

[9] 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.

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