Decoy Document Deployment for Effective Masquerade Attack Detection

Malek Ben Salem and Salvatore J. Stolfo

Computer Science Department
Columbia University
New York, New York 10027, USA
{malek,sal}@cs.columbia.edu

Abstract. Masquerade attacks pose a grave security problem that is a consequence of identity theft. Detecting masqueraders is very hard. Prior work has focused on profiling legitimate user behavior and detecting deviations from that normal behavior that could potentially signal an ongoing masquerade attack. Such approaches suffer from high false positive rates. Other work investigated the use of trap-based mechanisms as a means for detecting insider attacks in general. In this paper, we investigate the use of such trap-based mechanisms for the detection of masquerade attacks. We evaluate the desirable properties of decoys deployed within a user's file space for detection. We investigate the trade-offs between these properties through two user studies, and propose recommendations for effective masquerade detection using decoy documents based on findings from our user studies.

1 Introduction

The *masquerade attack* is a class of attacks, in which a user of a system illegitimately poses as, or assumes the identity of another legitimate user. Identity theft in financial transaction systems is perhaps the best known example of this type of attack.

Masquerade attacks can occur in different ways. A masquerader may get access to a legitimate user's account either by stealing a victim's credentials, or through a break-in and installation of a rootkit or key logger. A masquerade attack may also be caused by the laziness and misplaced trust of a user, such as the case when a user leaves his or her terminal or client open and logged in, allowing any nearby co-worker to pose as a masquerader. In most cases, the attacker's objective is to steal data that could be used for financial gains from home users or enterprise users, such as financial credentials or intellectual property. Masquerade attacks have been widespread. A Forbes report estimated that fraud incidents affected 11.2 million consumers in the United States in 2009, causing $54 billion in ID fraud costs [6]. In 2009, researchers who have monitored the Torpig botnet affirmed that, within the span of 10 days only, Torpig obtained the credentials of 8,310 accounts at 410 different institutions [15]. Enterprise users have also been targeted by different malware such as Confickr/Downadup [7].

The main approach for detecting masquerade attacks relies on computer user profiling and the modeling of statistical features, such as the frequency of certain events, such as issued user commands, the duration of certain events, the co-occurrence of multiple events combined through logical operators, and the sequence or transition of events. The focus of this line of work is primarily on accurately detecting change or unusual user command sequences [12, 10, 16]. These approaches suffered, like most anomaly detection-based approaches, from high false positive rates ranging between 1.3% and 7.7% [2].

Another approach for detecting masqueraders is the use of baits such as honeynets and honeypots. Honeypots are information system resources designed to attract malicious users. They have been widely deployed in De-Militarized Zones (DMZ) to trap attempts by external attackers to penetrate an organization's network. Some researchers proposed the use of honeyfiles, a type of honeypot, to detect malicious insider activity [4]. They introduced the concept of *perfectly believable decoys* and proposed several properties to guide the design and deployment of decoys, namely:

1. Believability: The attacker would not use the bait information if it did not appear authentic to the attacker.
2. Enticingness: No attack detection is possible if the attacker does not access the bait information because it does not look attractive enough.
3. Conspicuousness: Decoys should be easily located or retrieved in order to maximize the likelihood that an attacker takes the bait.
4. Detectability: If the access to the bait asset is not detectable than the deployment of the decoys is useless.
5. Variability: Decoys should not be easily identifiable to an attacker due to some shared invariant.
6. Non-interference: Decoys should not interfere with the legitimate user's normal activity. Non-interference has been defined as the likelihood that legitimate users access the real documents after decoys are introduced [4].
7. Differentiability: Legitimate users should be able to easily distinguish decoy documents from authentic documents, which has a direct affect on non-interference.
8. Shelf-life: Decoys may have a limited time period during which they are effective.

While all of the above are important decoy properties, it can be difficult to design and deploy decoy documents that would perfectly maximize the properties, which in turn would assure effective detection of a masquerade attack. One has to find the right trade-off between these properties in order to use them effectively. Such trade-offs may vary depending on the type of attack.

For example, while believability is a very important property of decoys when used for detecting insider attacks that aim to ex-filtrate sensitive information, it becomes of a lesser importance when the decoys are aimed at detecting masquerade attacks. In the case of an insider attack, the attacker already has legitimate access to the system where the sensitive information assets are located. Access to such assets does not necessarily constitute evidence of malicious intent or activity. However, subsequent ex-filtration and use of such information does. If the attacker identifies the decoy document as bogus, then they would not use the information contained in that document, which is why the *believability* of the decoy document is important. In the case of a masquerade attack, the mere access to the decoy document does constitute evidence of an attack as the masquerader is not a legitimate user of the system, and therefore should not be accessing any assets residing on that system. Whether the masquerader finds the decoy document believable or not, after having accessed it, is irrelevant to the detection of the attack, as evidence of the malicious activity has been already established.

In this paper, we attempt to investigate these trade-offs between decoy properties when applied to the masquerade detection problem through two user studies. The contributions of this work include:

- A host-sensor that detects access to decoy documents when loaded in memory using stealthy HMACs embedded in the decoy documents
- An investigation of the trade-offs between deployment properties of decoy documents when applied to the masquerade attack detection problem through two user studies
- A set of recommendations for the effective use of decoy documents for masquerade attack detection

The rest of this paper is organized as follows. In section 2, we briefly present the results of prior research work on masquerade detection. Section 3 expands on the threat model and presents the baiting technique used to detect masquerade attacks. In Sections 4 and 5, we describe two user studies to evaluate the different properties of the decoy documents and present the findings of these studies. Section 6 presents some recommendations for the effective use of decoy documents in masquerade attack detection in light of our findings. Finally, Section 7 concludes the paper by discussing directions for future work.

2 Related Work

Honeypots have been widely used in De-Militarized Zones (DMZ) to detect external attackers when attempting to penetrate an organization's network. Spitzner presented several ways to adapt the use of honeypots to the detection of insider attacks [13]. He introduced the notion of honeytokens, defined as 'information that the user is not authorized to have or information that is inappropriate' [13]. This information could then direct the attacker to the more advanced honeypot that could be used to discern whether the attacker's intention was malicious or not, a decision that may be determined by inspecting the attacker's interaction with the honeypot.

'Honeyfiles', or decoy files, were introduced by Yuill et al. [17].The authors proposed a system that allows users to turn files within the user space on a network file server into decoy files. A record that associates the filename with the userid is used to identify the honeyfiles.

Bowen et al. extended the notion of a decoy document system, and developed an automated system for generating decoy files [4, 3]. They also proposed several decoy properties as general guidelines for the design and deployment of decoys.

Honeyfiles suffer from some shortcomings. First, the attacker may not ever use or interact with the decoy file, especially if their identity is known to, or discovered by the attacker. Moreover, if an attacker discovers a honeyfile, they can potentially inject bogus or false information to complicate detection. In this paper, we investigate decoy deployment properties that should increase the likelihood of detecting a masquerade attacker.

3 Trap-based Masquerader Detection Approach

3.1 Threat Model

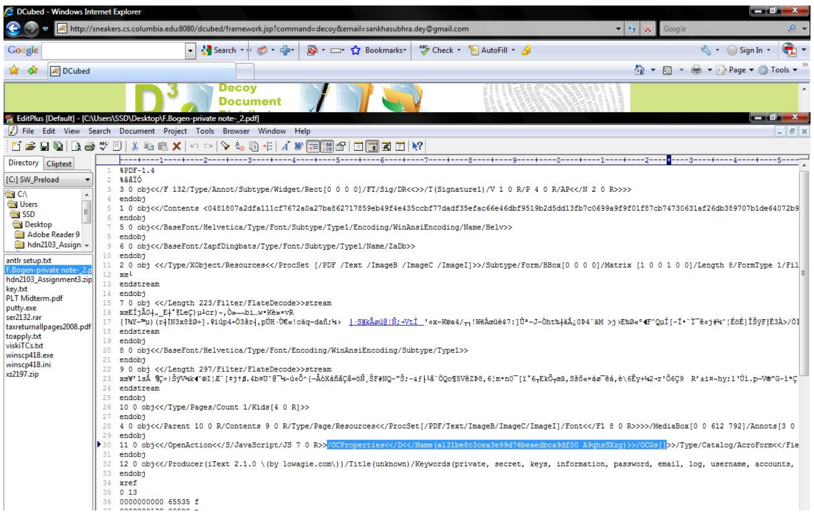
Masqueraders impersonate legitimate users after stealing their credentials when they access a system. When presenting the stolen credentials, the masquerader is then a legitimate user with the same access rights as the victim user. To that extent, masquerade attacks represent one type of insider attacks. However, masquerade attacks can be characterized by the low amount of knowledge the attacker has about the system and policies in place. In this work, we focus on masqueraders and assume that the attacker has little knowledge about the system under attack. In particular, we assume that the attacker does not know whether the system is baited or not.

We have architected a Decoy Documents Access (DDA) sensor and designed decoy documents in such a way that a sophisticated attacker with more knowledge and higher capabilities, in particular an inside attacker, would not be able to escape detection if they touched a decoy document. Decoy documents have a HMAC tag computed over the document's contents as described in subsection 3.2. A sophisticated attacker with wide resources would not be able to distinguish the HMAC tags of decoy documents from random functions. Moreover, a highly privileged attacker would not be able to turn off the DDA sensor without getting detected. This is ensured through a self-monitoring mechanism as described in subsection 3.3. Both types of attackers would have to know that the system under attack is baited, and the detection of this class of attack is beyond the scope of this paper. Here, we devise user studies for attackers, who have no knowledge that the system is baited, with the objective of investigating the decoy deployment properties. The study of attacker behavior and their perception of risk and expected gain based on their knowledge of the existence of decoys on the system is beyond the scope of this paper, and will be the subject of a future user study.

3.2 Trap-based Decoys

The trap-based technique used by our sensor relies on trap-based decoys [4] that contain 'bait information' such as online banking logins, social security numbers, and web-based email account credentials. Users of the DDA sensor can download such decoy files from the Decoy Document Distributor (D^3) [3], an automated service that offers several types of decoy documents such as tax return forms, medical records, credit card statements, e-bay receipts, etc..

The decoy documents carry a keyed-Hash Message Authentication Code (HMAC) [9] embedded in the header section of the document, and visible only if the document is opened using a hex editor. The HMAC is computed over a file's contents using a key unique to the user, and is hidden in the header section of the file. For instance, the use of the full version of the SHA1 cryptographic function in combination with a secret key to tag the decoy documents with an HMAC tag prevents the attacker from distinguishing the embedded HMAC from a random function [8]. An example of a decoy document with an embedded HMAC is shown in Figure 1. It is this marker or HMAC tag that our sensor uses to detect access to a decoy document. In the next section, we describe how our sensor makes use of this marker.


DCubed - Windows Internet Explorer
http://sneakers.cs.columbia.edu:8080/dcubed/framework.jsp?command=decoy&email=sankhasubhra.dey@gmail.com
DCubed
Decoy Document
EditPlus [Default] - [C:\Users\SSD\Desktop\F.Bogen-private note-_2.pdf]
File Edit View Search Document Project Tools Browser Window Help
Directory Cliptext
[C:] SW_Preload
C:\
Users
SSD
Desktop
Adobe Reader 9
hdn2103_Assign
antlr setup.txt
F.Bogen-private note-_2.p
hdn2103_Assignment3.zip
key.txt
PLT Midterm.pdf
putty.exe
ser2132.rar
taxreturnallpages2008.pdf
toapply.txt
viskiTCs.txt
winscp418.exe
winscp418.ini
xz2197.zip

```
%PDF-1.4
%âãÏÓ
3 0 obj<</F 132/Type/Annot/Subtype/Widget/Rect[0 0 0 0]/FT/Sig/DR<<>>/T(Signature1)/V 1 0 R/P 4 0 R/AP<</N 2 0 R>>>>
endobj
1 0 obj<</Contents <0481807a2dfa111cf7672a0a27ba862717859eb49f4e435ccbf77dadf35efac66e46dbf9519b2d5dd13fb7c0699a9f9f01f87cb74730631af26db389707b1de64072b9
endobj
5 0 obj<</BaseFont/Helvetica/Type/Font/Subtype/Type1/Encoding/WinAnsiEncoding/Name/Helv>>
endobj
6 0 obj<</BaseFont/ZapfDingbats/Type/Font/Subtype/Type1/Name/ZaDb>>
endobj
2 0 obj <</Type/XObject/Resources<</ProcSet [/PDF /Text /ImageB /ImageC /ImageI]>>/Subtype/Form/BBox[0 0 0 0]/Matrix [1 0 0 1 0 0]/Length 8/FormType 1/Fil
xœ└
endstream
endobj
7 0 obj <</Length 223/Filter/FlateDecode>>stream
xœEÍjÃ0┼„_E┼°‼LeÇ)µ┴cr)~,Ò»¬¬bi…w•Ké»×vR
{↑¾Y-™µ)(r┼↑N3x8žØ÷].♀iùp4←Ö3šr┤,pÚH·Ù€«!cäq-dañ;¼› |·S‰kÃøù@¦Ñ;→VtÌ '◦x-K®a4/┬┐!WëÀœüè$7:]Ûª-J-Òht%┼äÂ¿0Þ4¨äM >j›E%Ø«°◄F^QµÍ[-Ï•¨Ï¯ê◦j#¼^¦ÉöÈ}ÎŠÿF|Ë3À>/Ö
endstream
endobj
8 0 obj<</BaseFont/Helvetica/Type/Font/Encoding/WinAnsiEncoding/Subtype/Type1>>
endobj
9 0 obj <</Length 297/Filter/FlateDecode>>stream
xœ¥'1sÂ ¶Ç÷|ŠÿV¼k◄^®I¦Æ¨[‡j†♫.4b¤U^@¬¼~ú◦Õ^{—ÅóXàñãÇã=öÑ,ŠF#NQ-"Š;-&f├└ã¨ÖQo¶5VêZÞ8,6¦m•n0¯[ï°6┬EkÕ┬œß,Sšõ«×àø¯êá,è\6Êy+¼2→r'Ô6Ç9  R'±i¤~hy;l'Úì.p—V®"G-ì*Ç
endstream
endobj
10 0 obj<</Type/Pages/Count 1/Kids[4 0 R]>>
endobj
4 0 obj<</Parent 10 0 R/Contents 9 0 R/Type/Page/Resources<</ProcSet[/PDF/Text/ImageB/ImageC/ImageI]/Font<</F1 8 0 R>>>>/MediaBox[0 0 612 792]/Annots[3 0
endobj
11 0 obj<</OpenAction<</S/JavaScript/JS 7 0 R>>/OCProperties<</D<</Name(a131be8c3cea3e99d76beaedbca9df00 A9qhsSXzg)>>/OCGs[]>>/Type/Catalog/AcroForm<</Fie
endobj
12 0 obj<</Producer(iText 2.1.0 \(by lowagie.com\))/Title(unknown)/Keywords(private, secret, keys, information, password, email, log, username, accounts,
endobj
xref
0 13
0000000000 65535 f
```

Fig. 1. HMAC in the OCP Properties Section of a PDF Document

3.3 Decoy Documents Access Sensor

The DDA sensor detects malicious activity by monitoring user actions directed at HMAC-embedded decoy documents, as any action directed toward a decoy document is suggestive of malicious activity [1]. When a decoy document is accessed by any application or process, the host sensor initiates a verification function. The verification function is responsible for distinguishing between decoys and normal documents by computing a HMAC as described in Section 3.2 for that document and comparing it to the one embedded within the document. If the two HMACs match, the document is deemed a decoy and an alert is triggered; otherwise, the document is deemed normal and no action is taken. The

DDA sensor detects when decoy documents are being read, copied, or zipped. The sensor was built for the Windows XP platform and relies on hooks placed in the Windows Service Table. The hooking is performed by injecting code into the address space of the processes, and by replacing the address of the *file open* system call which is present in the kernel (.dll) library of windows. This code injection guarantees that our code will be executed first, and post processing it will call the actual system call. This approach also enables the configuration of the list of processes that should be hooked into or should be excluded from hooking into.

In order to prevent the sensor from being shut down by the adversary, we used a random directed cycle of n monitors to protect the DDA sensor as proposed by Chinchani et al. [5] and Stolfo et al. [14]. One of the n monitors, call it m, monitors the critical processes of the sensor. If an attacker attempts to shut down any of the sensor processes, monitor m will issue an alert. Similarly, if m is shutdown, another monitor, as defined by the directed cycle of monitors, issues an alert. The directed cycle of monitors is created based on a seed known only to the owner of the system. It defines a unique shutdown sequence that must be followed in order to shut down the sensor without any alerts.

4 User Study 1

4.1 Experiment Design

Our first user study aims to measure decoy document accesses performed by the legitimate users of the system, which can be considered as false positives. We seek to answer two questions through this user study:

1. Does the number of decoy files planted in a file system have an impact on their non-interference with the legitimate user's normal activities?
2. What are the best locations for planting decoys on a file system, so as to minimize their non-interference?

To answer these questions, we designed an experiment where we controlled the number n of decoy documents planted in a file system. We postulate that non-interference is a variable that is dependent on the number of decoy documents n. We do not measure non-interference as a probability. However, we measure the average number of decoy accesses per one week of computer usage. To that extent, we asked four user groups of thirteen computer science students each, to plant ten, twenty, thirty, or forty decoy documents generated by D^3 on their own file systems. The 52 students downloaded a total of 1300 decoy documents from D^3. We encouraged the participants in the user study to carefully consider where to place the decoy files and how to name them by taking into account the desired properties of such documents, particularly enticingness, conspicuousness and non-interference [4]. The objective is to maximize the likelihood that a potential masquerader will get detected when they illegitimately access the victim's computer, while minimizing the likelihood that they (the legitimate user) accidentally accesses these documents due to confusion or interference with their normal activity. For instance, the user can choose file names that are easily recognizable as decoy by them, while remaining enticing to the adversary. The file name could, for example, include the name of a person who is outside the social network of the user. For instance, one participant renamed a decoy file to *TaxReturnSylvia.pdf*, while he did not file any tax returns jointly with *Sylvia*, nor did he know anyone with that name. Carefully selecting the file names would make the file easily recognizable as a decoy file by the legitimate user, but could make it intriguing for the attacker.

The participants in the user study, who installed our DDA sensor before downloading the decoy documents, agreed to share their data. The experiment lasted for about seven days on average, during which access to decoy files was monitored. The data collected by the DDA sensor was uploaded to a central server for analysis.

4.2 Experiment Findings

At the end of the user study, the participants reported the directories under which they placed the downloaded decoy files. We summarized the results of these reports and ranked the directories based on decreasing numbers of placed decoys. The top 42 directories are shown in Table 1. Subdirectories under the *My Documents* and *Desktop* directories seemed to be the most popular choices by the participants. In the following, we summarize the main findings of this user study.

Interference Increases with More Decoy Files: Recall that non-interference is defined as the likelihood of the legitimate user accessing the authentic files after installing the decoy files. Decoy files planted on a file system for masquerade detection are not supposed to be accessed by the legitimate user. They are placed there in order to entice attackers to open and use them. Any accidental accesses to decoy files by the legitimate users of the system, i.e. accesses that are not caused by an attacker gaining access to the file system, are considered as false positives. We have ignored all alerts issued within the first hour of the students installing the decoy documents on their systems, giving them an opportunity to decide where to place the decoy documents and how to rename them, based on the recommendations given to them in our user study description. Table 2 presents the number of false positives and shows that it grows super-linearly with the number of decoy files planted in the file system. The higher the number of decoy files placed in the file system, the higher the likelihood of a legitimate user accidentally accessing one of these decoy files, thus, the lower the non-interference of these decoy files with the normal activities of the legitimate user. While a more longitudinal study is needed to investigate the relationship between the number of decoys planted and their impact on non-interference, our preliminary results show that non-interference decreases with the number of decoys planted.

Table 1. Decoy Document Placement

Decoy File Number	**Directory where Decoy was Placed**
1	C:\Documents and Settings\username \My Documents\Personal\Shopping\
2	C:\Documents and Settings\username\My Documents\Taxes\
3	C:\
4	F:\
5	C:\Documents and Settings\username\My Documents\Receipts\
6	C:\Documents and Settings\username\Desktop\
7	C:\Documents and Settings\username\My Documents\Financial\Bank Statements\
8	C:\Documents and Settings\Administrator\
9	C:\Documents and Settings\username\My Documents\
10	C:\Documents and Settings\username\My Documents\Financial\
11	C:\Documents and Settings\username\My Documents\Private\
12	C:\Documents and Settings\username\My Documents\Personal\
13	C:\Documents and Settings\username\My Documents\Private\Medical
14	C:\Documents and Settings\username\My Documents\Downloads\
15	C:\Documents and Settings\username\My Documents\Financial\Lost Card\
16	C:\Documents and Settings\username\My Documents\Financial\Disputes\
17	C:\Documents and Settings\username\My Documents\onn\bills and all\eBay\
18	C:\Documents and Settings\username\Desktop\Important\
19	C:\Program Files\License \
20	C:\Windows\Temp\
21	C:\Documents and Settings\username\My Documents\Personal\Visa Applications\
22	C:\Documents and Settings\username\My Documents\Private Vacation \
23	C:\Windows\
24	C:\Documents and Settings\username\My Documents\Confidential\
25	C:\Documents and Settings\username\Cookies\
26	C:\Documents and Settings\username\Favorites\
27	C:\Documents and Settings\username\workspace\
28	C:\Documents and Settings\username\My Documents\Investments\
29	C:\Documents and Settings\username\My Documents\Resume\
30	C:\Documents and Settings\username\Desktop\My Journal\
31	C:\Backup\
32	C:\Documents and Settings\username\My Pictures\
33	C:\Documents and Settings\username\Desktop\Notes\
34	C:\Documents and Settings\username\My Documents\Confidential\Employee Evaluations\
35	C:\Documents and Settings\username\Recent\
36	C:\Documents and Settings\username\Start Menu\
37	C:\Documents and Settings\username\Desktop\Insurance\
38	C:\Documents and Settings\username\Local Settings\
39	C:\Documents and Settings\username\My Documents\401K\
40	C:\Documents and Settings\username\My Documents\Mortgage\
41	C:\Documents and Settings\username\My Music\
42	C:\Documents and Settings\username\My Documents\Miscellaneous\

Table 2. Number of Decoys and Decoy Touches

Number of Placed Decoys	Number of Participants in Experiment	Number of Decoy Accesses
10	13	2
20	13	6
30	13	9
40	13	24

Distribution of False Positives: Figure 2 is a box-and-whisker plot of the decoy file accesses by the legitimate users for the four different values of decoys planted in the file system. The horizontal line in the middle of each of the boxes in these plots corresponds to the median value of decoy file accesses. Fifty per cent of the data falls within this box, while the top and bottom quartiles (25% of the data) of the data are represented by the whisker lines above and below this box. Data points whose value is above 1.5 times the upper quartile or lower than 1.5 times the lower quartiles are considered outliers, and are represented as small crosses. The short horizontal lines above and below the box represent the maximum and minimum data values excluding outliers.

The figure shows that for the case of ten decoys, only one false positive was recorded for any single user, whereas that number reaches up to nine false positives when 40 decoys are placed in the file system. Although the nine false positives is considered an outlier in this figure, more than 50% of the users who placed 40 decoy documents in their file systems did accidentally access at least one decoy file and experienced some level of interference with their normal activities. As the figure shows, not only does the likelihood of interference for each user grow with the number of decoy documents placed in the file system, but the amount of interference for each affected user increases non-linearly as well.

Placement of Decoy Files: Figure 3 shows the number of false positives by decoy location across the top 42 most popular locations as reported by the user study participants. The specific directory locations are listed in Table 1. The number of false positives varies widely by decoy document path or location. It is noteworthy that only fifteen of the top 40 decoy file locations were accidentally accessed by the legitimate users. Many decoy files were never accessed by these users demonstrating that non-interference of the decoy documents varies by the chosen decoy placement in the file system. While the ideal decoy placement that minimizes interference should be customized by the user based on their file system access habits, it seems that certain locations should be avoided such as the *high traffic* location or those locations that get automatically scanned by applications installed on the system.

The highest number of false positives are due to accesses to decoy files placed in location number 14, i.e. under the *Downloads* directory. While eight of the nine false positives in this location were triggered by a single user, the results show that decoy files in this location can introduce a high level of interference.

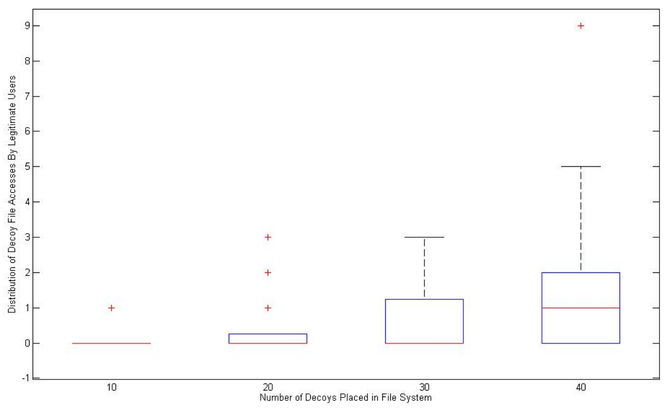


Fig. 2. Distribution of the Number of Decoy Document Accesses by Legitimate Users

This is not surprising knowing that most browsers save downloaded files in the *Downloads* directory by default, thus forcing a lot of user activity and *traffic* on files in this directory.

Differentiability to the User is not Enough: The second decoy file location that exhibited a high number of false positives according to Figure 3 is the *My Music* directory. These false positives could be accidentally triggered by the legitimate users when manually browsing the directory, but they are more likely to be triggered by media player that are scanning the directory in order to discover recently added music files. Even though this scanning for media files is initiated by the user who knows exactly which files are decoy files, the media player or application performing a thorough scan cannot identify these decoy files, and therefore will access them in an attempt to identify whether these files are indeed music or video files.

We will further investigate the decoy placement strategies in the next section, where we will show that decoy placement, and therefore decoy conspicuousness, is also tightly linked with the ability to detect masqueraders.

5 User Study 2

5.1 Experiment Design

In this experiment, we investigate two decoy deployment-related properties, namely enticingness and conspicuousness. Evaluating the design-related properties such as believability, particularly as it pertains to the contents of the decoy document, is not very relevant to the masquerade attack problem. Recall that we detect access to the decoy files before the attacker sees the contents of

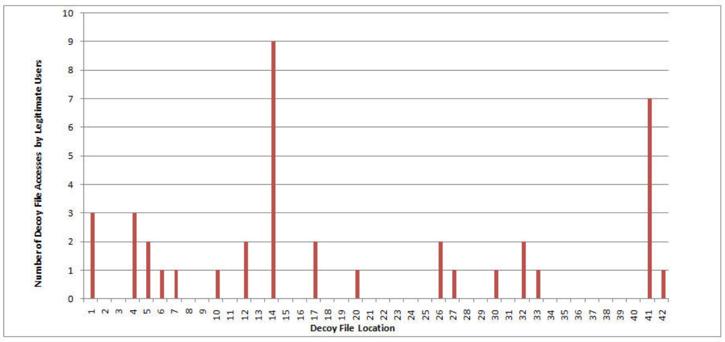


Fig. 3. Accidental False Positive Decoy Document Accesses by Legitimate Users

the file. We ensure variability by tagging all files on the system with a pseudo-random HMAC tag. Detectability can be ensured through the use of the DDA sensor and protecting the sensor, as well as protecting the key used to compute the HMAC tags of the decoy files. Note that any attempt to modify the HMAC tag by the attacker requires that the decoy file gets first accessed and loaded into memory, which would trigger an alert by the DDA sensor.

We seek to answer the following questions through this experiment:

1. How many decoy documents are needed to detect with a high probability masqueraders looking to steal information?
2. Where are decoy documents most likely to trap masquerade attackers, *i.e.* in which directories should one place the decoys in order to maximize the likelihood of catching a masquerader, while not interfering with the legitimate user's normal activities?
3. What is the number of decoy documents that leads to the best trade-off between masquerade detection while not interfering with the legitimate user's activities?
4. To what extent do decoy file accesses reveal a masquerader's *malicious* intent?

5.2 Experimental Set-Up

We conducted a set of experiments where we simulated masquerader attacks. While simulating masquerader attacks in the lab is not ideal, it was the best available option. We randomly selected 40 computer science students to participate in our user study and gave all participants a specific scenario that described the difficult financial situation that they were going through, and their need for additional financial resources. The participants were asked to perform a specific task, which consisted of stealing, from a coworker's desktop computer, any information that could be used for financial gain. They had access to the coworker's computer for 15 minutes while the coworker left for lunch. Besides their financial difficulties, the scenario also described the personal problems they were having with the coworker. In particular, they believed that their coworker had been undeservingly promoted instead of them, making any PII information theft or the cause of financial damage to their colleague, that they might engage in, justifiable to them. The task was goal-specific in order to capture the intent of the users. Although one may argue that simulating a masquerade attack is not appropriate, and that the participants may be constrained because of their ethical concerns. We claim that, with a well-written scenario and very clear instructions, the participants will act as masqueraders. To this extent, we refer the reader to the very well-known Milgram experiment [11]. Milgram showed that subjects would blindly follow clear instructions presented by an authority figure, even when they contradict their own values and ethics [11].

The participants in the user study knew that they were being monitored. They had unlimited access to a computer in our lab for 15 minutes using an administrator's account. So they had access to all the file systems mounted to that computer. None of the participants had access to this file system before, which was designed to look very realistic and to include potentially interesting patent applications, personally identifiable information, as well as account credentials. We installed the DDA sensor on this computer, and planted n decoy files in the file system. The goal was to monitor when and where the masqueraders accessed these decoy files. Table 3 shows how we varied the number of decoy documents n for each experiment, which involved ten participants acting as masqueraders.

In order to reduce experimenter bias when placing these decoy documents, we have surveyed 52 computer science students as to where they would place decoy documents on their file systems. The directory list shown in Table 1 indicate where we placed the decoy documents for each experiment (top 40, top 30, top 20, and top 10 directories respectively). Recall that these decoy locations have been selected based on the reports collected from the fifty-two student participants in our first user study.

The number of decoy documents n is the independent variable that we control in this user study. The number of decoy file touches is the dependent variable on the number of decoys n planted in the file system.

The file system used in our lab had more than 100,000 files indexed by desktop search tools installed on the system, and totaling a size of more than 70 GBytes. The file system included at least eighty files containing personal information that could be used for identity theft or financial gain. The contents of certain personal files were sanitized, so that no personal information was leaked.

Table 3. Total Number of Decoy Alerts by Experiment

Experiment	**Number of Placed Decoys**	**Number of Decoy Alerts**
1	40	96
2	30	90
3	20	71
4	10	56

Before conducting the experiments, we ensured that all non-decoy documents on the file system were tagged with a random tag, that is indistinguishable from the HMAC tag of the decoy documents. This was done so that a more knowledgeable attacker cannot distinguish the decoy files from the normal files by editing them through a hex editor and looking for the HMAC tag within the document. This ensures the fifth desirable property of decoy documents, namely that they are not easily distinguishable as decoys to the attacker.

Besides tagging the real *non-decoy* files, we have indexed all the decoy files using the available desktop search tools on the system. Recall that the DDA sensor issues an alert if the contents of the decoy file are read or it gets loaded into memory. Thus, indexing the decoy files during the deployment phase can reduce potential false positive alerts that could be issued by the DDA sensor while conducting the experiments. The goal is to have more reliable and accurate results, where the sensor alerts are truly caused by the masquerader's activity, and not by desktop search tools suddenly scanning the decoy files.

5.3 Experiment Findings

In the following section, we list the major findings of this user study.

The Use of Decoys is Effective: The main purpose of our use of decoys is to detect masquerader attacks. While non-interference with legitimate users' activities is desirable, decoys would be useless if they fail to attract masqueraders by being enticing and conspicuous. The results displayed in Figure 4 suggest that at least one access to a decoy document was detected by the DDA sensor for every masquerader, regardless of the number of decoys planted in the file system. This finding shows that well-placed decoys can be very effective for masquerade detection. When combined with other intrusion detection techniques, they could potentially provide even more effective and accurate detectors.

Recommended Number of Decoys: Figure 4 shows the distribution of the number of decoy file accesses by attackers in the four experiments by the number of decoy files placed in the file system. One could clearly see that the average number of touched decoys (horizontal line within each box) is comparable in the case of 20, 30, and 40 decoys. Whereas it is slightly lower (5.6 touches) when 10 decoys are planted in the file system. Considering Figures 2 and 4 then, one can conclude that the use of twenty decoy documents provides the best trade-off between detection capability and non-interference. Although the minimum number of touched decoys jumps from 1 or 2 to 5 in the case of 40 planted decoys, this high number could be attributed to the location of the planted decoys and not merely to the higher number of decoys. This can be clearly seen in Figures 8 and 9 which show the conspicuousness of decoy files 33 and 37 for instance. We will elaborate on this point in the following discussion.

High Number of Decoy Accesses is Indicative of Attacker's Malicious Intent: The question is how many of these decoy file accesses are caused by

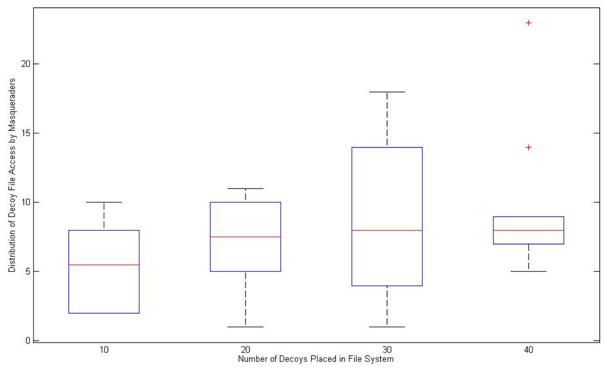


Fig. 4. Distribution of the Number of Decoy Document Accesses

the masquerader's malicious activity. To answer this question, we conducted an experiment with three groups of ten participants each. Each group was given a different scenario to execute: a 'malicious attacker' scenario which we described in subsection 5.2, a 'benign masquerader' scenario, or a 'neutral' scenario. In the *benign* scenario, the participants in the experiment were allowed to access a coworker's computer for 15 minutes in order to accomplish a specific work-related task, after their own computer experienced a hard drive failure. In the *neutral* scenario, the participants were left to freely choose whether they wanted to access their coworker's desktop while the coworker left for 15 minutes. They had no specific motive to do so. In all three scenarios, we placed 30 decoys in the file system of the lab computer that served as the participant's coworker's computer. The benign and malicious scenarios can provide a baseline to compare the decoy touches of a malicious masquerader to. Figure 5 shows the distribution of decoy touches by scenario. One can clearly see that the number of decoy touches is very low in the benign and neutral scenarios when compared to the malicious scenario.

The Use of Decoys is Efficient: According to the results shown in Figure 6, 17 attackers were detected during the first minute of their masquerade activity, while another ten were detected during the second minute after accessing the victim's computer. All masqueraders were detected within ten minutes of their accessing the system under attack. Figure 6 also shows that for attackers 31-40, who participated in experiment 4 with only 10 planted decoy documents, the average period of undetected masquerader activity (3.6 minutes) is much longer, than for attackers 1-10 (1.3 minutes), 11-20 (1.6 minutes), and 21-30 (1.2 minutes).

Attackers 1 and 14 recorded the longest times of undetected masquerader activity. These attackers had a different strategy for finding personal information. One attacker was mostly after information related to the victim's online banking

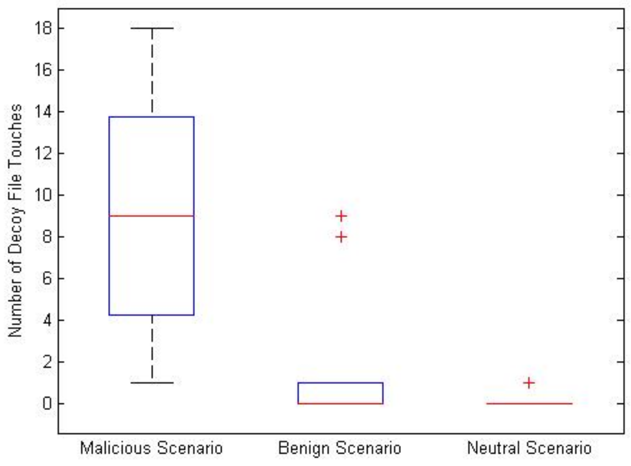


Fig. 5. Distribution of the Number of Decoy Document Accesses by Scenario

activity, rather than after information that existed locally on the file system. The attacker, therefore did not navigate or search the *My Documents* directory, where most decoys were located. Not surprisingly, the first decoy that they hit was decoy number 16. located under Internet Explorer's *Favorites* directory. Attacker 14, on the other hand, suspected that no important information would be stored under the user's file system, but rather under the administrator's file system, which is why he only hit a single decoy document, namely decoy number 8 (see Table 1 for the exact decoy location). Note that attacker 1 took part in the first experiment where 40 decoys were planted, and attacker 14 participated in experiment 2, where 30 decoys were used. So even with a larger number of decoys, the attackers managed to escape detection for 10 minutes, which suggests we need a well thought-out placement of decoys, that covers all potential attacker strategies for locating and stealing information.

Consider the null hypothesis that a masquerader activity within a 10-minute time window gets missed with at least 10% probability in the presence of at least 40 decoy files. Based on the observed results, we can reject this hypothesis at the 2% significance level with a p-value=0.0148. In other words, we can claim that with a 98% probability, the probability of detecting a masquerader within 10 minutes of their activity on the victim's system is at least 90%.

Decoy Placement is Important: The objective is to identify the decoy document locations that would be less interfering with the normal activity of the legitimate user, while being conspicuous to potential attackers. While the experiments have not been conducted on the same system, and the decoy file locations vary by normal user (customized for their own non-interfering use of the system),

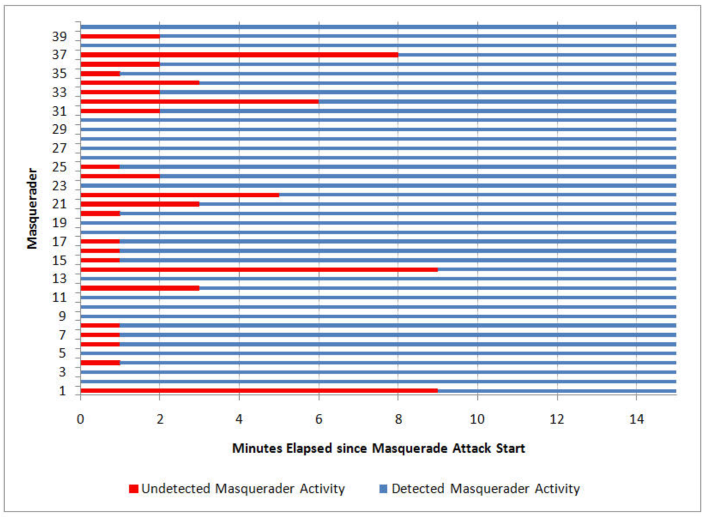


Fig. 6. Detection Time by User

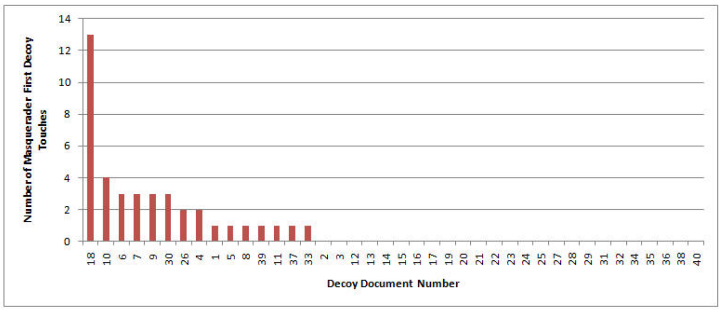


Fig. 7. Decoy File First Touches

we argue that the cumulative results do provide ground for comparison, as we have monitored the use for about 7 days on average for each user, for a total of 52 users.

Figure 8 shows the number of accesses to each decoy file by experiment. Note that the first ten decoy documents were present during all four experiments, whereas the decoys 31-40, for instance were in place only during the last experiment. This explains the falling line in Figure 8, which depicts the maximum number of potential attackers accessing these decoys, knowing that there were ten masqueraders in each experiment.

The number of decoy file accesses varies widely by file location. The locations are available in Table 1. Some files were accessed more than 25 times, while others, namely nine out of the forty decoy files, have never been touched by any attacker. This underlines the importance of decoy placement in highly conspicuous locations.

Considering figures 8 and 2 for instance, one can notice that the decoy placed in location 18 was touched by 28 masqueraders. Decoys placed in the same location by the participants in the first user study were not touched by the legitimate users and did not trigger any false positives. On the other hand, decoys placed in locations 19, 23, 25, and 31 for instance caused several false positives without catching any masquerader. The two figures clearly show that the location of the decoy document plays an important role in its detection capability and non-interference with normal user activities. While the locations that yield the least interference depend on the legitimate user's file system access patterns, there is clearly room for placing decoys under directories, so that high detection rates with very low interference rates.

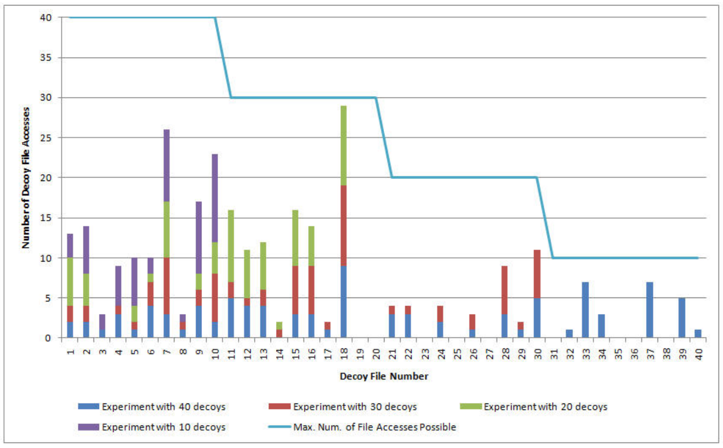


Fig. 8. Number of Decoy Document Accesses

Conspicuousness Comes First: In order to understand how effective each decoy file was in catching masqueraders, we have ranked the decoy files based on the probability of an attacker accessing them in Figure 9. Recall that an access in this case refers to viewing, editing, copying, zipping, or moving the decoy file. Considering Figure 9 and the table in Table 1, the top three decoy files that are most likely to be accessed by an attacker reside on the *Desktop*. These three files reside respectively in the *Important*, *Notes* and *Insurance* directories. The first file was a payment receipt. The second file included a budget plan, and the third file was a copy of test results from a medical lab. Other decoys with contents similar to the contents of decoy 18, such as decoys 4 and 5 which constituted payment receipts also, did not achieve the same detection capability as decoy 37. This leads us to the conclusion, that conspicuousness is more important than enticingness. Although the attackers were mostly after information with financial value, they were more likely to access files with health-related information that were highly conspicuous (located on the desktop), than other decoys with more relevant information to their objective, i.e. finance-related information.

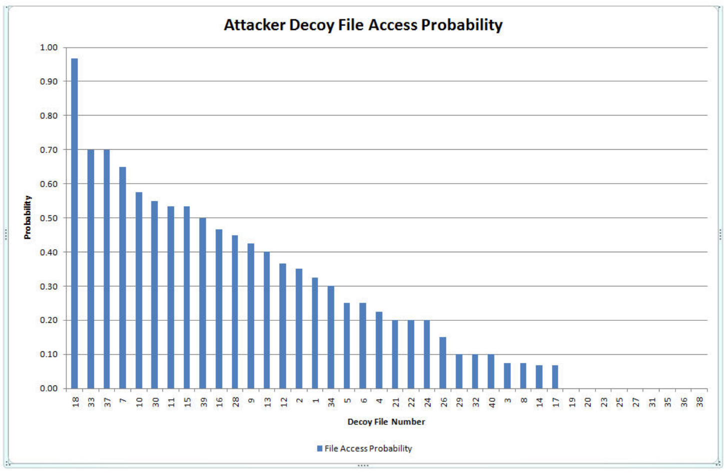


Fig. 9. Probability of an Attacker Opening the Decoy File

6 Discussion and Recommendations

We list here a few recommendations related to the deployment of decoy documents for effective masquerade attack detection based on the findings of our user studies.

6.1 Recommendation 1

While, the detectability property of the decoy documents is the most important property, our second user study has shown how important the *conspicuousness* property is. The latter is even more important than the *enticingness* property, as revealed by the user study.

We recommend considering the following ranking of decoy properties when designing and deploying decoy documents for masquerade attack detection. The ranking is based on decreasing importance of the decoy properties.

1. Detectability
2. Conspicuousness
3. Enticingness
4. Non-Interference and Differentiability

5. Believability

The importance of the *variability* property varies by attacker sophistication. Similarly, the importance of the *decoy shelf-life* depends on the deployment environment.

6.2 Recommendation 2

While the number of false positives varies widely by user and by decoy document location, overall we averaged less than 1 false positive per user per week. This is a very encouraging number, but it could be even further reduced with more intelligent placement of the decoy documents. For example, the decoy files placed under the *My Pictures* and *My Music* directories could be accessed by applications scanning the file system for picture or music files respectively. Such accesses are not deliberate accesses by the legitimate user and could be avoided, if the decoy files are placed under directories that are not by default scanned by such applications. The user may choose to ignore decoy alerts triggered by these applications by configuring the sensor accordingly. Recall that, as described in section 3.3, the hooking mechanism used by the sensor enables the filtering of decoy alerts issued by user-defined processes.

7 Conclusion

In this paper, we presented an experimental evaluation of the different deployment-related properties of decoy documents. We also made a few recommendations based on the findings from our experiments. These recommendations should guide the deployment of decoy documents for effective masquerade detection.

In our future work, we will repeat the experiments in different environments, other than universities, to ascertain whether the results are broadly applicable. We will also evaluate other decoy document properties, including the *believability* of documents. Furthermore, we will investigate the decoy document properties for masquerade attacks perpetrated through the installation of rootkits and malware such as Torpig. Finally, we will study how attacker behavior changes based on their knowledge about the monitoring mechanisms running on the victim's system and their perception of risk and expected financial gain.

Acknowledgment

We thank Shlomo Hershkop and Brian Bowen for their work on D^3. We further thank Shlomo Hershkop for his support with the DDA sensor.

References

1. Ben-Salem, M. DDA Sensor: http://www1.cs.columbia.edu/ids/ruu/data/.

2. Ben-Salem, M., Hershkop, S., and Stolfo, S. J. A survey of insider attack detection research. In *Insider Attack and Cyber Security: Beyond the Hacker* (2008), Springer.
3. Bowen, B., and Hershkop, S. Decoy Document Distributor: http://sneakers.cs.columbia.edu/ids/ruu/dcubed/.
4. Bowen, B. M., Hershkop, S., Keromytis, A. D., and Stolfo, S. J. Baiting inside attackers using decoy documents. In *SecureComm'09: Proceedings of the 5th International ICST Conference on Security and Privacy in Communication Networks* (2009).
5. Chinchani, R., Upadhyaya, S., and Kwiat, K. A tamper-resistant framework for unambiguous detection of attacks in user space using process monitors. In *Information Assurance, 2003. IWIAS 2003. Proceedings. First IEEE International Workshop on* (march 2003), pp. 25 – 34.
6. Greenberg, A. ID Theft: Don't Take it Personally. http://www.forbes.com/2010/02/09/banks-consumers-fraud-technology-security-id-theft.html, February 2010.
7. Higgins, K. J. Widespread Conficker/Downadup Worm Hard To Kill. http://www.darkreading.com/security/attacks-breaches/212901489/index.html, January 2009.
8. Kim, J., Biryukov, A., Preneel, B., and Hong, S. On the security of hmac and nmac based on haval, md4, md5, sha-0 and sha-1 (extended abstract). In *SCN'06: Proceedings of the 5th International Conference on Security and Cryptography for Networks* (Berlin, Heidelberg, January 2006), OUP Oxford, pp. 242–256.
9. Krawczyk, H., Bellare, M., and Canetti, R. RFC2104, HMAC: Keyed-Hashing for Message Authentication. The Internet Engineering Task Force (IETF).
10. Maxion, R. A., and Townsend, T. N. Masquerade detection using truncated command lines. In *DSN '02: Proceedings of the 2002 International Conference on Dependable Systems and Networks* (2002), IEEE Computer Society, pp. 219–228.
11. Milgram, S. *Obedience to Authority: An Experimental View.* Harpercollins, January 1974.
12. Schonlau, M., Dumouchel, W., Ju, W., Karr, A. F., Theus, M., and Vardi, Y. Computer intrusion: Detecting masquerades. *Statistical Science 16* (2001), 58–74.
13. Spitzner, L. Honeypots: Catching the insider threat. *Annual Computer Security Applications Conference* (2003).
14. Stolfo, S. J., Greenbaum, I., , and Sethumadhavan, S. Self-monitoring monitors. In *Columbia University Computer Science Department, Technical Report # cucs-026-09* (2009).
15. Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, R., Kruegel, C., and Vigna, G. Your botnet is my botnet: analysis of a botnet takeover. In *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security* (New York, NY, USA, 2009), ACM, pp. 635–647.
16. Wang, K., and Stolfo, S. J. One-class training for masquerade detection. In *Proceedings of the 3rd IEEE Workshop on Data Mining for Computer Security* (2003).
17. Yuill, J., Zappe, M., Denning, D., and Feer, F. Honeyfiles: deceptive files for intrusion detection. In *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC* (June 2004), pp. 116 – 122.