Automated Detection of Masquerade Attacks with AI and Decoy Documents

Matt Berdychowski
School of Computing
Wichita State University
Wichita, KS, USA
mpberdychowski@shockers.wichita.edu

Sergio A. Salinas Monroy
School of Computing
Wichita State University
Wichita, KS, USA
sergio.salinasmonroy@wichita.edu

2024 Cyber Awareness and Research Symposium (CARS) | 979-8-3503-8641-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/CARS61786.2024.10778670

***Abstract*—Adversaries can launch masquerade attacks by stealing user credentials and then logging into a system, pretending to be the compromised user. These attacks are particularly challenging to detect, because the adversaries have the same permissions as the users they impersonate. Previous works have shown that masquerade attacks can be detected by deploying decoy documents on the users' computers. Since decoy documents are, in general, only accessed by attackers, they can be used to detect masquerade attacks. However, previous works lack data generated by real-world adversaries and do not propose detection mechanisms. To address this challenge, we design effective machine learning masquerade attack detection models. The models are trained using data from our previous real-world experiment, which includes real cybercriminal interactions with a honeyfile server. We find that our machine learning models are able to detect masquerade attacks in a real-world scenario with an accuracy of at least 98%.**

***Index Terms*—artificial intelligence, masquerade attack, cybersecurity, decoy documents, phishing, hacker, honeyfiles**

I. Introduction

Masquerade attacks occur when an adversary logs into a system as a legitimate user, usually using stolen credentials. Masquerade attacks are particularly serious due to the large amount of stolen credentials available and phishing attacks conducted by cybercriminals. According to Leyden [1], there were 24 billion usernames and passwords available for purchase on the Dark Web in 2022. Many people use popular, easy to guess usernames and passwords [2]–[6], despite repeated attempts to improve password security [7]–[10]. Moreover, adversaries have been able to bypass multi-factor authentication in various ways, including social engineering [11]–[15]. According to Cisco [16], 90% of data breaches originate from a phishing attack. A promising approach to detect masquerade attacks is to deploy decoy documents, also known as honeyfiles, in the legitimate users' folders. Decoy documents are designed to appear to contain important user data, but in reality they only contain worthless data. Legitimate users are aware of the presence of these files and are instructed not to access them. Hence, system administrators can use the access logs of decoy files to detect masquerade attacks.

However, as users may accidentally access their own honeyfiles, a masquerade attack detection mechanism must be able to differentiate between malicious access from attackers and accidental access from legitimate users. Previous works have primarily focused on analyzing legitimate users' interactions with decoy documents. However, their experiments are unable to establish how often real-world attackers access honeyfiles. Therefore, it is not certain if decoy documents are a viable masquerade attack detection tool and how to decide whether observed behavior should be considered an attack.

To address this challenge, we propose a detection mechanism based on machine learning (ML). We design and implement three machine learning models: support vector machine, decision tree and random forest. The models take information about a user's logon duration and number of honeyfiles that they accessed as input and output a decision on whether the logon session was malicious or not.

We measure the performance of the proposed detection mechanisms using data collected from a real-world experiment in our previous work [17]. In particular, we collected malicious decoy file interactions over a period of one hundred thirty three days using a honeypot. The honeypot was accessible from the Internet and had a user account with weak credentials to allow cybercriminals to access the server.

We then apply our proposed detection mechanisms to the collected data. We see that all machine learning models can detect malicious decoy file access with at least 98% accuracy. Thus, the models can be used as a fast, lightweight tool for automated masquerade attack detection.

We summarize our contributions as follows:

1) We propose a machine learning approach to detect masquerade attacks using decoy files.
2) Compared to previous works, we use data from real-world cybercriminals to train the models.
3) We provide extensive experiment results which show that our machine learning models can efficiently and accurately detect masquerade attacks.

The rest of the paper is organized as follows: Section II presents the related works, Section III describes the real-world experiment, Section IV discusses the threat model, Section V presents the machine learning models, Section VI analyzes the results, Section VII discusses possible future research directions, while Section VIII concludes the paper.

II. Previous Works

Spitzner [18] first described the concept of honeytokens as an extension and alternative to honeypots [19]. The honeyfiles we consider in this paper are an example of a honeytoken. Yuill et al. [20] introduce the term honeyfiles and provide a thorough analysis of their advantages and limitations. The concept of decoy documents has been extended to other areas. For instance, Lee et al. [21] extend the concept of decoy files to a complete file system. Sun et al. [22] extend the concept to malicious email attachments used by attackers for phishing.

There are only a few works that propose masquerade attack detection mechanisms using decoy files. Bowen et al. [23] propose a masquerade detection mechanism that considers any interaction with a decoy document as an attack. Salem et al. [24], as well as Voris et al. [25], [26], propose detection mechanisms that consider accidental access to the decoy documents by legitimate users.

The works in [24]–[26] propose a threshold to detect masquerade attacks by measuring the number of times a user interacts with decoy files within a given time period. If a user accesses honeyfiles a number of times that is higher than the threshold, it is determined that it is an attacker. Otherwise, it is considered a legitimate user. To determine the appropriate interaction threshold, the researchers in [24]–[26] conducted experiments with human subjects. Since the human subjects recruited for the legitimate user experiments were typical users, we use the interaction rates found in these works to inform our proposed machine learning detection mechanisms.

However, the attacker data collected in [24]–[26] cannot be readily generalized. In particular, to determine how often attackers interact with decoy documents, the researchers in [24]–[26] conduct a controlled lab experiment whose participants volunteered to pretend to be cybercriminals. Since these volunteers had no previous experience conducting attacks against any computer system, their behavior may not accurately reflect the behavior of real-world attackers. Moreover, these works did not propose a systematic approach to detect masquerade attacks.

To address this challenge, we conducted a real-world experiment in our previous work [17]. We exposed a vulnerable server with decoy documents to the Internet, and collected the file interactions logs generated by attackers that compromised the server.

In this work, we propose machine learning models to detect masquerade attacks using data collected from real-world attackers [17]. We describe in detail our detection mechanisms in the following sections.

III. A Real-world Decoy File Interaction Dataset

To address the lack of real-world data on how attackers interact with decoy documents, we conducted a real-life experiment in our previous work [17]. The experiment exposed a server with decoy documents to real hackers from all over the world. Next, we explain this experiment in detail for completeness.

A. Experiment Setup

1) Honeypot server: A virtual server running the Windows Server 2019 Operating System in the Amazon Web Services cloud was configured to be accessible over the Internet using the Remote Desktop Protocol through a public IP address. One user account was protected with a weak password. The account was named `admin`, to create an impression that it belonged to the system administrator, when in fact it was a non-administrator account to prevent hijacking of the server [17]. The virtual server ran for one hundred and thirty three consecutive days. To accurately replicate a real server, nine user accounts were created. The user accounts and the `Administrator` account were secured with strong passwords. To make the exposed `admin` account appear active, nearly three hundred non-decoy documents were placed in its folders [17].

2) Decoy Documents: Twenty decoy documents were placed on the honeypot server to appear as real documents to the attackers. We prefixed decoy file names with an underscore and appended one word to draw the attacker's attention. The words that were appended are: ”Final”, ”Latest”, ”Update”, ”Updated”, ”Complete”, ”Detailed” or ”Summary” [17]. Decoy documents were deployed according to different scenarios, including placing them among regular files or in separate subfolders. A decoy-less scenario was also implemented. Details of all scenarios can be found in [17].

B. Collected Dataset

The dataset consists of information about 536 logons to the honeypot server. Based on the interaction rate with honeyfiles for legitimate users found in [25], we divided the logons into malicious and benign. 208 logons were classified as benign because their decoy document access rate was below the threshold. The remaining 328 were classified as malicious.

Each row in the dataset table represented a single logon. Columns corresponded to twenty-two features. The first twenty columns show how many times each of the decoy documents deployed in the honeypot was accessed during a logon. The 21st column contains the logon's duration in seconds. The 22nd column shows the decoy access rate for each logon. We calculate the decoy access rate using the data from the first twenty one columns. The last column in the dataset is the label for each logon: 1 for attack, 0 for benign logons. Table I shows a simplified header of the table containing the dataset.

Columns 1-20 number of interactions with each of 20 decoys	Logon duration in seconds	Decoy access rate per 15 minutes	Label

TABLE I
Simplified header of our dataset.

IV. Threat Model

In this section, we describe our assumed system architecture and the assumed capabilities of the adversary, as shown in Fig. 1.

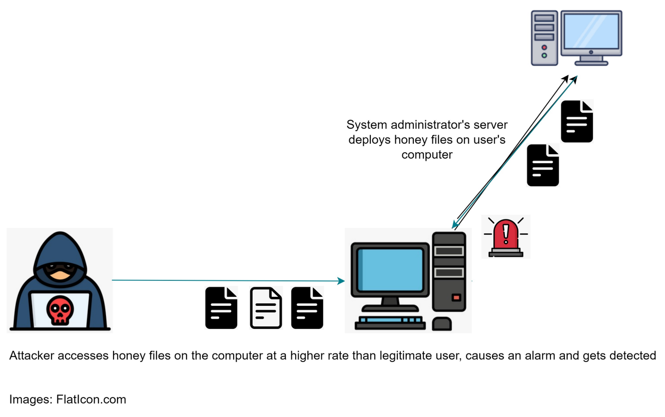
System administrator's server deploys honey files on user's computer
Attacker accesses honey files on the computer at a higher rate than legitimate user, causes an alarm and gets detected
Images: FlatIcon.com

Fig. 1. Threat model

We consider a legitimate user that has permissions to access the files in certain folders within a file system. The file system can be located on the local user's machine or on a remote server. A system administrator controls the permissions of the user to the file system. The system administrator adds decoy documents to the file server to facilitate the detection of masquerade attacks, as described in Section III-B.

We assume that the adversary has gained the ability to log on to the system as the legitimate user. The adversary can gain this ability in a number of ways, including phishing attacks, social engineering, etc. Once the adversary has gained access to the system, it inspects the files that the legitimate user can access.

We assume the adversary has no prior knowledge of the legitimate user's files. It also does not know if decoy files are being used. We also assume that the legitimate user is aware of the honeyfiles, and thus follows the decoy document interaction rate for legitimate users found in [25], [26].

Based on the results obtained in [25], [26], we assume that easy to find folders, such as the Desktop or Documents folder, are more likely to be accessed by the adversary. We confirmed these results in [17] by also placing decoys deeper in the folder tree and comparing their access rate to those in easier to access locations. We leave the case where a more powerful (or returning) adversary has prior knowledge of the files in the system for future work.

In this paper, we design our machine learning models to detect masquerade attacks launched by attackers with the capabilities and under the scenarios described above.

V. A Machine Learning Approach to Detect Masquerade Attacks

In this section, we present the machine learning models that we use to detect masquerade attacks. Unlike previous works, we use real-world data collected from our honeypot server experiment to train machine learning models. Moreover, we choose machine learning models that have low computational complexity during their inference phase. Thus, they can be deployed in real systems with limited resources.

A. Model training

We model the problem of detecting masquerade attacks as a binary classification problem. That is, the models take as input file interaction features from a logon session and output a prediction whether the interactions were malicious or not.

B. Models

To detect masquerade attacks we use three different types of machine learning models: support vector machine (SVM), decision tree (DT), and random forest (RF). SVMs are supervised learning models which can be used for classification. They are memory efficient and versatile. DTs are supervised learning models that can be used for classification. They use "if - else" decision rules, are easy to understand and visualize, as well as flexible in their complexity. RFs are ensemble methods that combine output from multiple decision trees into a single prediction, providing even more modeling flexibility and precision.

We choose these models because they have all been successfully used for binary classification in security problems, in particular email classification [27]–[30], a task very similar to logon classification.

VI. Performance Evaluation

In this section, we present the results of our performance evaluation of the proposed machine learning models to detect masquerade attacks.

The models described in Section V-B were trained using the dataset described in Section III-B. We implemented the models using the Python programming language and the Scikit-Learn library on a system with eight CPU cores with top frequency of 3.6 GHz and 16GBs of RAM. The models were trained using the default parameters set by the Scikit-Learn library. We created a training dataset with 80% of the data. The remaining 20% was used for testing.

A. Support Vector Machine

Fig. 2 shows the performance of the SVM model when it is trained using all features in the dataset. We see that it only misclassified one attack, but it misclassified 40 normal logons. The overall accuracy is 62%, which is low.

Next, we evaluate the performance of the SVM when we drop the decoy access rate feature. Fig. 3 shows that dropping this feature has no effect on the SVM performance compared to Fig. 2.

In Fig. 4, we show the performance of the SVM trained with a dataset without the decoy access rate and without the logon duration features, i.e. we only use the number of times each decoy document was accessed. We see that the accuracy of the SVM improved dramatically to 98.15%.

We summarize the performance metrics of the SVM model in Table II.

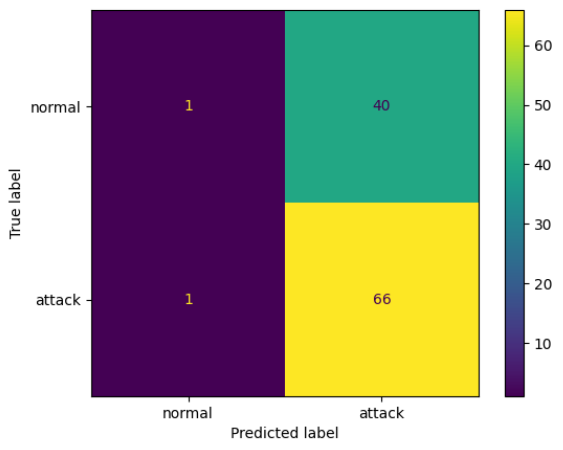

Fig. 2. Confusion matrix for SVM using all features

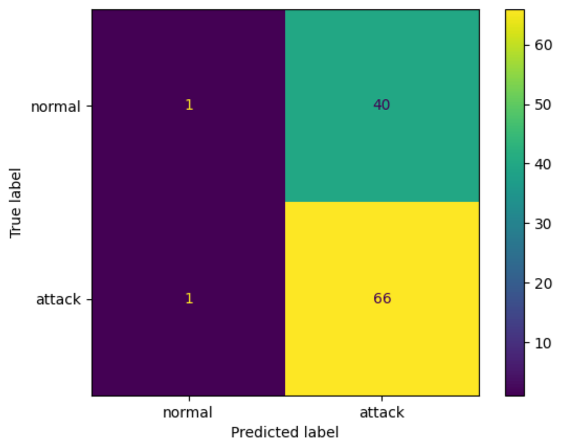

Fig. 3. Confusion matrix for SVM after the decoy access rate was dropped

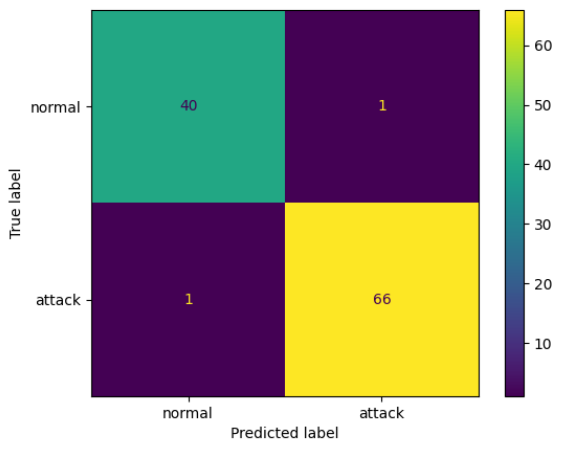

Fig. 4. Confusion matrix for SVM without decoy access rate and logon duration

Metric	All features	No Decoy Access Rate	No Rate and Logon Length
Accuracy	0.62	0.62	0.9815
Precision	0.6226	0.6226	0.985
Recall	0.985	0.985	0.985
F1 Score	0.763	0.763	0.985

TABLE II
SVM MODEL METRICS FOR DIFFERENT SIZES OF DATASET.

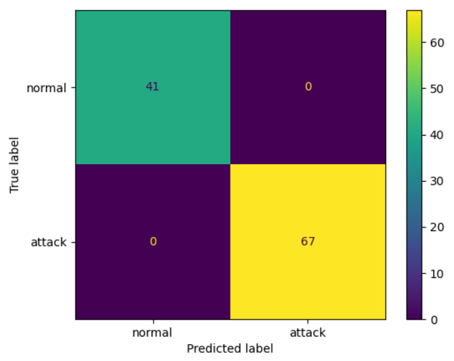

Fig. 5. Confusion matrix for DT and RF with all features present

B. Decision Tree

We also evaluate the performance of a decision tree model. Fig. 5 shows the performance of the DT model when trained on all features in the dataset. We see that it achieves an accuracy of 100%.

Fig. 6 demonstrates that after dropping the decoy access rate feature, accuracy of DT deteriorated only slightly to 98.15%. Two attacks were misclassified as legitimate logons, while no benign logons were misclassified.

Fig. 7 shows the DT's performance on the dataset without the decoy access rate and logon duration features. Accuracy remained at 98.15%, with only one normal logon and one

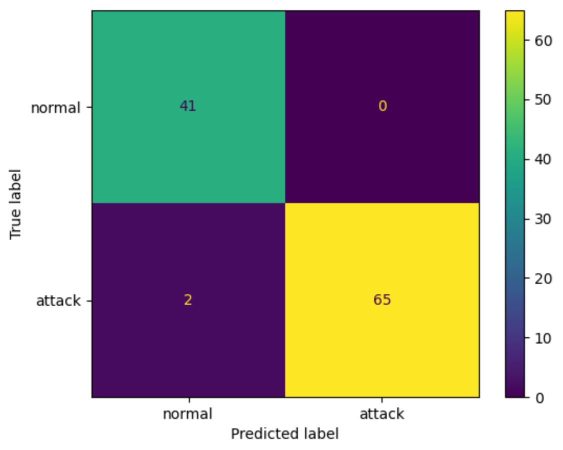

Fig. 6. Confusion matrix for DT and RF without decoy access rate

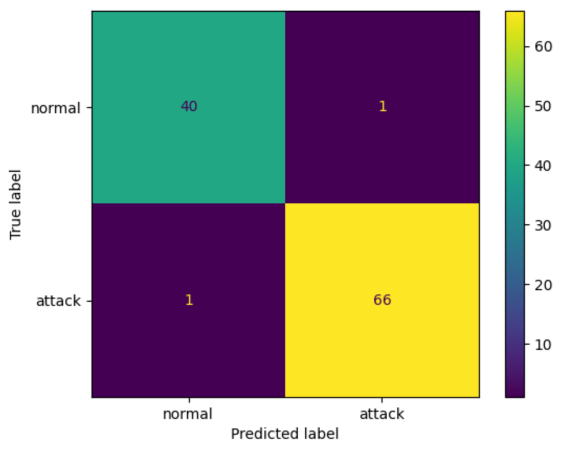

Fig. 7. Confusion matrix for DT and RF without decoy access rate and logon duration

attack being misclassified. It is worth pointing out that the confusion matrix for this set of features looks the same for DT and RF as it does for SVM (Fig. 4).

C. Random Forest

The performance of the random forest model followed exactly the performance of the decision tree. This is due to the fact that additional trees did not improve the accuracy. We summarize the results of the decision tree and random forest models in Table III.

Dataset Size	All features	No Decoy Access Rate	No Rate and Logon Length
Accuracy	1	0.9815	0.9815
Precision	1	1	0.985
Recall	1	0.97	0.985
F1 Score	1	0.9848	0.985

TABLE III
DT AND RF MODELS METRICS FOR DIFFERENT DATASET SIZES.

D. Comparison to Threshold-based Methods

Compared with existing threshold-based methods our proposed method showed a slightly lower accuracy. The method described in [17] can detect attacks when the decoy file access rate is above the threshold. Unfortunately, this approach requires the defender to calculate how many times the attacker interacted with decoy documents within a 15-minute period. Accessing logon duration data in real time is non-trivial in the Windows OS due to lack of built-in tools with appropriate capabilities.

In contrast, our proposed ML method only requires information about how many times the attacker accessed decoy documents during a logon. It does not need to count the time the attacker spent logged in, making it feasible for a defender to collect the data with common tools such as Process Monitor. Due to the additional time information available to threshold-based approaches, their performance is not fully comparable with the accuracy of our proposed method.

VII. DISCUSSION

Although our proposed method is highly-effective under the considered conditions, there are interesting research directions that can be pursued. Attackers may attempt to bypass detection by identifying decoy documents and ignoring them. If they know the threshold value, adversaries can also interact with documents at a rate lower than the threshold to avoid detection, even when all files they access turn out to be decoys. Investigating how the proposed method can be adjusted to address this attacker behavior is an interesting possible future research direction.

VIII. CONCLUSIONS

In this work, we have used data from real-world cybercriminals to implement a machine learning approach to detecting masquerade attacks with the use of decoy documents. Our experiment's results demonstrate that machine learning models can efficiently and accurately detect masquerade attacks. We observe that all three analyzed models deliver at least 98% accuracy with the smallest subset of features possible. Therefore, machine learning approach can be a viable alternative to measuring attacker honeyfile access rate and comparing it with a threshold.

REFERENCES

[1] J. Leyden, "Dark web awash with breached credentials, study finds," https://portswigger.net/daily-swig/dark-web-awash-with-breached-credentials-study-finds, 2022.

[2] J. Bonneau, "The science of guessing: analyzing an anonymized corpus of 70 million passwords," in *2012 IEEE Symposium on Security and Privacy*. IEEE, 2012, pp. 538–552.

[3] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. Lopez, "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms," in *2012 IEEE symposium on security and privacy*. IEEE, 2012, pp. 523–537.

[4] M. Weir, S. Aggarwal, B. De Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," in *2009 30th IEEE Symposium on Security and Privacy*. IEEE, 2009, pp. 391–405.

[5] D. Perito, C. Castelluccia, M. A. Kaafar, and P. Manils, "How unique and traceable are usernames?" in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2011, pp. 1–17.

[6] Y. Zhang, F. Monrose, and M. K. Reiter, "The security of modern password expiration: An algorithmic framework and empirical analysis," in *Proceedings of the 17th ACM conference on Computer and communications security*, 2010, pp. 176–186.

[7] S. Schechter, C. Herley, and M. Mitzenmacher, "Popularity is everything: A new approach to protecting passwords from statistical-guessing attacks," in *Proceedings of the 5th USENIX conference on Hot topics in security*, 2010, pp. 1–8.

[8] S. Houshmand and S. Aggarwal, "Building better passwords using probabilistic techniques," in *Proceedings of the 28th Annual Computer Security Applications Conference*, 2012, pp. 109–118.

[9] A. Forget, S. Chiasson, P. C. Van Oorschot, and R. Biddle, "Improving text passwords through persuasion," in *Proceedings of the 4th symposium on Usable privacy and security*, 2008, pp. 1–12.

[10] A. Juels and R. L. Rivest, "Honeywords: Making password-cracking detectable," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 145–160.

[11] S. Schechter, A. B. Brush, and S. Egelman, "It's no secret. measuring the security and reliability of authentication via "secret" questions," in *2009 30th IEEE Symposium on Security and Privacy*. IEEE, 2009, pp. 375–390.

[12] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda, "All your contacts are belong to us: automated identity theft attacks on social networks," in *Proceedings of the 18th international conference on World wide web*, 2009, pp. 551–560.

[13] A. Kak, "Mounting targeted attacks with trojans and social engineering," engineering.purdue.edu/kak/compsec/NewLectures/ Lecture30.pdf, 2020.

[14] A. K. Ghazi-Tehrani and H. N. Pontell, "Phishing evolves: Analyzing the enduring cybercrime," *Victims & Offenders*, vol. 16, no. 3, pp. 316–342, 2021.

[15] C. Chipeta, "6 ways hackers can bypass mfa + prevention strategies," https://www.upguard.com/blog/how-hackers-can-bypass-mfa, 2024.

[16] Cisco, "2021 cyber security threat trends: phishing, crypto top the list," learn-cloudsecurity.cisco.com/umbrella-resources/umbrella/2021-cyber-security-threat-trends-phishing-crypto-top-the-list, 2021.

[17] M. Berdychowski, "Using decoy documents to detect masquerade attacks," Master's thesis, Wichita State University, 2021.

[18] L. Spitzner, "Honeytokens: The other honeypot," community.broadcom.com/symantecenterprise/communities/community-home/librarydocuments/viewdocument?DocumentKey=74450cf5-2f11-48c5-8d92-4687f5978988&CommunityKey=1ecf5f55-9545-44d6-b0f4-4e4a7f5f5e68&tab =librarydocuments, 2003.

[19] ——, "Honeypots: Catching the insider threat," in *19th Annual Computer Security Applications Conference, 2003. Proceedings.* IEEE, 2003, pp. 170–179.

[20] J. Yuill, M. Zappe, D. Denning, and F. Feer, "Honeyfiles: deceptive files for intrusion detection," in *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.* IEEE, 2004, pp. 116–122.

[21] J. Lee, J. Choi, G. Lee, S.-W. Shim, and T. Kim, "PhantomFS: File-based deception technology for thwarting malicious users," *IEEE Access*, vol. 8, pp. 32 203–32 214, 2020.

[22] B. Sun, T. Ban, C. Han, T. Takahashi, K. Yoshioka, J. Takeuchi, A. Sarrafzadeh, M. Qiu, and D. Inoue, "Leveraging machine learning techniques to identify deceptive decoy documents associated with targeted email attacks," *IEEE Access*, vol. 9, pp. 87 962–87 971, 2021.

[23] B. M. Bowen, S. Hershkop, A. D. Keromytis, and S. J. Stolfo, "Baiting inside attackers using decoy documents," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2009, pp. 51–70.

[24] M. B. Salem and S. J. Stolfo, "Decoy document deployment for effective masquerade attack detection," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2011, pp. 35–54.

[25] J. Voris, J. Jermyn, N. Boggs, and S. Stolfo, "Fox in the trap: Thwarting masqueraders via automated decoy document deployment," in *Proceedings of the Eighth European Workshop on System Security*, 2015, pp. 1–7.

[26] J. Voris, Y. Song, M. B. Salem, S. Hershkop, and S. Stolfo, "Active authentication using file system decoys and user behavior modeling: results of a large scale study," *Computers & Security*, vol. 87, p. 101412, 2019.

[27] C. C. Anthoni and D. A. Christy, "Integration of feature sets with machine learning techniques for spam filtering," *International Journal of Computer Engineering & Technology (IJCET)*, vol. 2, no. 1, pp. 47–52, 2011.

[28] V. Christina, S. Karpagavalli, and G. Suganya, "Email spam filtering using supervised machine learning techniques," *International Journal on Computer Science and Engineering (IJCSE)*, vol. 2, no. 09, pp. 3126–3129, 2010.

[29] W. A. Awad and S. ELseuofi, "Machine learning methods for spam e-mail classification," *International Journal of Computer Science & Information Technology (IJCSIT)*, vol. 3, no. 1, pp. 173–184, 2011.

[30] S. K. Trivedi and S. Dey, "A comparative study of various supervised feature selection methods for spam classification," in *Proceedings of the second international conference on information and communication technology for competitive strategies*, 2016, pp. 1–6.