Active Directory Kerberoasting Attack: Detection using Machine Learning Techniques

Lukáš Kotlaba, Simona Buchovecká and Róbert Lórencz
Department of Information Security, Faculty of Information Technology, Czech Technical University in Prague, Czech Republic

Keywords: MS Active Directory, Machine Learning, Kerberoasting, Attack Detection, Cybersecurity.

Abstract: Active Directory is a prevalent technology used for managing identities in modern enterprises. As a variety of attacks exist against Active Directory environment, its security monitoring is crucial. This paper focuses on detection of one particular attack - Kerberoasting. The purpose of this attack is to gain access to service accounts' credentials without the need for elevated access rights. The attack is nowadays typically detected using traditional "signature-based" detection approaches. Those, however, often result in a high number of false alerts. In this paper, we adopt machine learning techniques, particularly several anomaly detection algorithms, for detection of Kerberoasting. The algorithms are evaluated on data from a real Active Directory environment and compared to the traditional detection approach, with a focus on reducing the number of false alerts.

1 INTRODUCTION

Active Directory (AD) is a directory service based on Lightweight Directory Access Protocol (LDAP). It stores information about objects on a domain network, such as users, groups, computers, and many others, together with their attributes in a hierarchical structure. It is typically used for a broad range of identity-related services in a domain; basic examples are authentication, authorization, and accounting (AAA) services for users and computers. (Desmond et al., 2013)

Although AD is a proprietary service developed by Microsoft, it represents an essential part of enterprise networks, integrating both Windows and non-Windows devices. Nowadays, the usage of AD technology spans even further, to hybrid and public cloud environments with products such as Azure AD or AWS Directory Service.

Understanding what data AD stores, and considering its significance, it is not surprising that it represents an interesting target for cyber attackers. Once adversaries get access to AD, the consequences can be fatal, often resulting in full domain compromise.

Our research focuses on Kerberoasting attack. In this attack, adversaries target service accounts with the aim to crack their passwords. If successful, the obtained credentials may enable privilege escalation and further lateral movement to other domain accounts.

In response, organizations put a lot of effort into securing their AD environments and developing countermeasures. Besides preventive actions in the domain administration, real-time security monitoring and attack detection are the essential defense mechanisms against AD threats.

Traditional detection of AD attacks, including Kerberoasting, is based on a rule-based analysis of relevant data. Detection rules contain specific conditions that are checked against Windows logs collected from machines over the network. If the log data matches the defined conditions, the rule is triggered, and a security alert is generated.

However, in practice, it turns out that rule-based detection has its limitations and is not always sufficient. Firstly, static rules may have high false alarm ratio (FAR). The rules should have the ability to catch malicious activity even if the attackers use legitimate computers and user accounts to perform their actions. On the other hand, such rules may trigger on regularly occurring events with these accounts, e.g., login failures due to wrong passwords or actions carried by administrators. A high number of false alerts may result in a real alert being overlooked or ignored.

Rule-based detection may also have other downsides. Since the rules are defined statically, they often contain various manually defined thresholds, con-

Kotlaba, L., Buchovecká, S. and Lórencz, R.
Active Directory Kerberoasting Attack: Detection using Machine Learning Techniques.
DOI: 10.5220/0010202803760383
In *Proceedings of the 7th International Conference on Information Systems Security and Privacy (ICISSP 2021)*, pages 376-383
ISBN: 978-989-758-491-6

stants, or signatures. These are difficult to determine in the creation process, but on the other hand, relatively easy for an attacker to overcome.

Adversaries may modify their attack procedure to evade detection, such as limit the number of attempts to fit under a numeric threshold used in the detection rule, change various values in the attack tools to avoid signature detection, or spread the attack over a more extended time period to limit the rate of attempts per time unit.

In our previous research (Kotlaba et al., 2020), we developed a set of signature-based rules that can be used to detect Kerberoasting. We found out that the FAR of the designed rules may vary, especially in diverse environments with many users. Non-standard approaches that used honeypots or PowerShell monitoring for detection, performed better compared to the rules based only on the number of service ticket requests. However, those carry on implementation overhead and therefore may not be suitable for usage in every environment.

In this paper, we propose applying machine learning (ML) techniques for detecting Kerberoasting. ML may improve existing signature-based methods, help to overcome their limitations, improve detection accuracy by reducing FAR, and further enhance detection capabilities.

Multiple anomaly detection algorithms are discussed, especially in terms of their suitability and efficiency for detection of Kerberoasting. Several configurations of ML models are compared. The evaluation is done using log data originating from an AD environment of a real organization, which is diverse in its structure and contains hundreds of daily users.

The paper is structured as follows: Section 2 contains background information on Kerberoasting attack, followed by a discussion about the possibilities of its detection. Further, details of the used ML algorithms are provided. Section 3 is dedicated to the existing research related to the topic of AD attack detection, as well as applications of ML techniques in cybersecurity. Our proposed detection approach is presented in Section 4, together with a summary of achieved results. Section 5 concludes the paper.

2 BACKGROUND

2.1 Kerberoasting Attack

Kerberoasting attack was first presented by Tim Medin to attack the credentials of a remote service without sending any traffic to the service itself. (Medin, 2014)

In Mitre ATT&CK framework, Kerberoasting is listed as a sub-technique of Steal or Forge Kerberos Tickets technique, which falls under Credential Access tactics. The attack differs from the other two sub-techniques (Golden and Silver Ticket) in the level of permissions required. Kerberoasting can be executed without the need for a local administrator account or an account having higher permissions in the domain. A valid domain account or the ability to sniff traffic within a domain is enough for an attacker to carry on Kerberoasting. (The Mitre Corporation, 2020)

Kerberoasting takes advantage of how service accounts leverage Kerberos authentication. Kerberos is the default protocol used within an AD domain. It allows users to access services on the network by merely using tickets (instead of passwords), generated for every session and used over a short time period.

Users can access remote services by simply requesting a service ticket from a domain controller (DC), which serves as the key distribution center (KDC) in the AD implementation of Kerberos. When clients request service tickets for given services from a DC, they use unique identifiers called service principal names (SPNs). To enable Kerberos authentication, it is required that SPNs are registered in AD with at least one service logon account (an account specifically dedicated to run the service). (The Mitre Corporation, 2020)

The course of Kerberoasting attack can be summarized in the following steps:

1. Services are identified by their SPNs registered in AD. An attacker that controls a valid user account may obtain these identifiers by an enumeration technique called SPN Scanning.
2. The attacker requests a service ticket by specifying the SPN value to a DC. In response, the DC presents the service ticket, part of which is encrypted with the target service account's password hash. In case a weak cipher suite is negotiated, such as RC4-HMAC-MD5, the service account's NT password hash is used to encrypt the service ticket. There is no communication with the system hosting the target service.
3. NT password hash is prone to brute-force password cracking. The cracking can be done offline, without any communication to the target service or the DC, meaning no events indicating failed logins are recorded. If the ticket is exported from memory, the cracking can be done on a different machine, outside of the target domain.
4. If the service account's password was not complex enough, the attacker would recover it in plaintext.

In order for Kerberoasting attack to be successful, it is enough that service accounts are set-up with a weak password (typically not long enough or dictionary-based). The attack also requires older encryption types to be enabled. That is, however, the default behavior, and often a necessity for backward compatibility. Therefore, the Kerberoasting attack requirements may be, unfortunately, easily fulfilled by underrated administration of service accounts in organizations. Furthermore, if these accounts are over-permissioned, an attacker controlling such an account may be able to access sensitive data or simply pivot through the domain.

There are plenty of tools available to perform Kerberoasting beyond proof of concept. Furthermore, SPN Scanning and subsequent requesting of service tickets can be also performed using PowerShell commands only. (Metcalf, 2017b)

Kerberoasting technique uses Kerberos authentication as designed and intended, without exploiting any part of the process. This makes the attack challenging to detect because service tickets are requested regularly as users access resources in the domain. To detect Kerberoasting, it is necessary to distinguish between legitimate and malicious requests.

Kerberos service ticket requests are logged in Windows Event ID 4769 *A Kerberos service ticket was requested* (both for success and failure), which is generated every time DC receives a service ticket request. Valuable information contained in this event includes account name, target service, source IP address, encryption type used, and failure code. (Microsoft Corporation, 2017)

There is no default way of detecting Kerberoasting, and thus building custom detection rules is necessary. As proposed by (Metcalf, 2017b), the detection logic can be built on top of the event 4769, using the following indicators:

- excessive requests for different services with a small-time difference from the same user,
- Kerberos service ticket requests with a weak encryption (such as RC4).

We utilized these indicators in our previous research (Kotlaba et al., 2020), where we developed several static rules applicable for detection of Kerberoasting. The rules were set to trigger an alert if there was higher number of service ticket requests with weak encryption types from a particular source than a specific predefined threshold.

However, these rules require that the threshold would be set to a reasonable value, which may be challenging to determine. The rules do not contain any built-in capability of reflecting normal vs. abnormal number of requests in the domain.

To overcome these limitations, we propose utilizing machine learning in addition to the used detection logic with the aim to gain the self-adopting capability and improve detection accuracy.

2.2 Machine Learning Algorithms

Detection of Kerberoasting technique represents a problem of differentiating between normal and unusual behavior in terms of service ticket requests. In our research, we have focused on the possibility of solving this problem by applying machine learning. Given the nature of our problem, we utilized ML algorithms suitable for anomaly detection.

Anomaly detection is intended for detecting abnormal or unusual observations in the data. As described in (Pedregosa et al., 2020), we may further distinguish between two slightly different types of anomaly detection:

- outlier detection - unsupervised anomaly detection, where the training data contains outliers which are defined as observations that are far from the others,
- novelty detection - semi-supervised anomaly detection, where the training data is not polluted by outliers, and we want to decide whether a new observation is an outlier.

In our approach, we have chosen one representative algorithm of each kind, in particular One-Class Support Vector Machine (SVM) for novelty detection, and Local Outlier Factor (LOF) for outlier detection.

2.2.1 One-Class SVM

One-Class SVM is an algorithm introduced by (Schölkopf et al., 1999), developed as an extension of SVM for the purpose of novelty detection. SVM is a method that performs classification by constructing hyperplanes in a multidimensional space that separate points of different classes. One-class classification differs from standard classification in the sense that it learns data from one class only - the normal class, while there are no or few samples from the other class - the abnormal class.

To identify suspicious observations, the algorithm estimates a distribution that encompasses the majority of the data and then labels the data points that lie far from it as suspicious, with respect to a suitable metric. The solution is built by estimating a probability distribution function which makes most of the observed data more likely than the rest, and a decision rule that separates these observations by the largest possible margin. If the new data points lay outside

of the determined margin, we can say that they are abnormal with given confidence in our assessment.

One-Class SVM can be used with multiple parameters, out of which the most important is the choice of a kernel and a scalar parameter ν. Usually, the radial basis function (RBF) kernel is chosen due to its non-linear properties. The ν parameter, also known as the margin of the One-Class SVM, corresponds to the probability of finding a new observation outside the frontier, which can also be interpreted as the proportion of outliers we expect to find in our data. (Pedregosa et al., 2020)

2.2.2 Local Outlier Factor

Local Outlier Factor (LOF) is an unsupervised algorithm proposed by (Breunig et al., 2000), which can be used to find outliers in a given dataset. The separation is made based on a score that represents the likelihood of a particular data point being an outlier.

LOF is based on the concept of local density. The density is obtained for every data point by estimating its distance from *k*-nearest neighbors, where the distance is computed according to the chosen metric. A normal instance is expected to have a local density similar to that of its neighbors, while an abnormal is expected to have much lower density. Thus, the main idea is to detect the samples that have a significantly lower density than their neighbors. (Pedregosa et al., 2020)

The advantage of LOF is that due to the local approach, it is able to identify outliers that would not be identified in a global perspective. The focus on locality can be directly influenced by setting the number of neighbors for the LOF calculation. By setting a small value, LOF considers only nearby points, which may be prone to errors when having much noise in the data. On the other hand, setting a large number may cause local outliers to be missed.

3 RELATED WORK

There are multiple publications and existing research that deals with detecting attacks against AD or Windows environment. The majority focuses on elements such as signatures and characteristics of attacks that can be used to build detection rules.

(Metcalf, 2017a) mentions various artifacts and signatures that can be found in Windows Security events and are usable for detection of multiple AD attack scenarios, especially attacks related to Kerberos protocol, including Kerberoasting.

CERT-EU published whitepaper Kerberos Golden Ticket Protection (Soria-Machado et al., 2016), where detection of Golden Ticket is proposed based on monitoring specific string signatures in the events and the lifetime of Kerberos tickets.

However, a common disadvantage of the mentioned approaches is that the attacks would remain undetected if attackers were able to evade the used signatures, e.g., by changing the tools' filenames or timing of the attack. Furthermore, false positives can occur if legitimate administrators use commands that match the defined signatures during their daily operations.

Apart from detection methods based on static rules and signatures, there is research available attempting to utilize machine learning to discover anomalous behavior in Windows environments and detect attack techniques related to lateral movement in AD infrastructure.

(Hsieh et al., 2015) propose a threat detection method built on accounts' behavior sequences. For each account, the probability model based on Markov chains is built and used to detect abnormal behavior. By testing this approach, it was able to give the best performance of about 66.6% recall and 99.0% precision rates when combined with prior knowledge.

(Goldstein et al., 2013) suggest monitoring abnormal user behavior by utilizing unsupervised learning, in particular with the k-NN algorithm. The algorithm was used to find anomalies in Windows authentication logs. Besides misconfigurations, it was able to find some interesting insights about the infrastructure.

(Matsuda et al., 2018) propose a method for outlier detection using unsupervised learning with the One-Class SVM algorithm. The detection is based on Windows Events 4674 *An operation was attempted on a privileged object* and 4688 *A new process has been created*, and focused on detecting attacks that require Domain Administrator privilege. The chosen approach showed high recall and precision ratios.

(Uppströmer and Råberg, 2019) utilized a supervised machine learning approach for detection of lateral movement. Attack activities related to lateral movement included AD enumeration, Pass the Hash and Pass the Ticket attacks. Several classifiers were compared on a semi-synthetic dataset. In the mentioned experiments, high accuracy was observed with all tested classifiers, but they performed differently in terms of recall and precision.

Anomaly-based methods based on clustering and principal component analysis were applied and compared by (Meijerink, 2019) for detection of lateral movement in Windows environments. The results showed that clustering generally performed better, but

with a relatively high false-positive ratio, which further reduction is desired.

To our best knowledge, there is currently no related research published that would study the possibilities of adopting machine learning techniques for detection of Kerberoasting attack.

4 PROPOSED APPROACH

Our research aims to utilize machine learning for detecting Kerberoasting attack. Our idea is to apply algorithms suitable for anomaly detection to identify unusual patterns, especially a higher number of ticket requests for non-typical services from sources that usually do not perform this kind of activity. If identified, such activity is worth investigating for the possibility of Kerberoasting attack execution. To validate our approach, we compared Kerberoasting detection based on static rules to several models based on ML techniques on real log data.

4.1 Implementation

4.1.1 Technology

For practical implementation, we have chosen Splunk platform, as it offers capabilities for both developing static detection rules and using ML techniques. Also, Splunk is commonly used by organizations for security monitoring purposes, which might help to apply our results in real environments easily.

Splunk is a platform for analyzing machine data. From a technical perspective, Splunk ensures collecting, parsing, and indexing Windows Events and other log sources. Ingested data can be queried by using its proprietary Search Processing Language (SPL) syntax. (Splunk Inc., 2020)

ML support is provided by Splunk Machine Learning Toolkit add-on, which provides an interface for using ML algorithms from the Python library Scikit-learn (Pedregosa et al., 2011), directly within the Splunk platform. Together with the underlying Scikit-learn library, the add-on supports both One-Class SVM and LOF algorithms that we have chosen to implement our approach.

4.1.2 Dataset Preparation

The dataset used for the experiments is non-synthetic, originating from an AD environment of a real organization with hundreds of daily users. It consists of Windows Events 4769 collected over the time span of six weeks.

As the event 4769 belongs to one of the most numerous events logged in a Windows domain, the dataset was further filtered. Only ticket requests with weak encryption types were preserved. Further, requests made from a domain controller itself, or requests for service names ending with a dollar sign (that identify computer accounts) were filtered out.

Filtered data were then split into bins, where a single bin represents a time frame of one day. Then statistics were calculated across the events by day, IP address, and source username. As a result, one data point in the dataset represents cumulative statistics of ticket requests per unique combination of username and IP address, per one day. Having aggregated statistics per day is more suitable, as the amount of activity differs significantly over the day (business vs. out-of-business hours).

For the purpose of applying ML techniques, the dataset was split into training, validation, and testing dataset based on time as follows:

- training – 4 weeks, 6265 events, 0 malicious,
- validation – 1 week, 1601 events, 1 malicious,
- testing – 1 week, 1489 events, 3 malicious.

Training data contains only regular traffic from the environment. A few crafted malicious events, which are to represent Kerberoasting activity, were injected into validation and testing datasets. Those events were designed to look very similar to regular events, each with a different number of services requested - both higher and lower values were used, compared to the numbers observed in the regular events.

4.1.3 Feature Engineering

Since detection of Kerberoasting is based solely on Event 4679, the number of possible features usable for ML models is limited by the fields contained in this event. The most important fields for detection of Kerberoasting identify the source (Account Name and Client Address) and the service for which a ticket was requested (Service Name/ID).

Extracting features from Windows events is problematic due to the small number of available fields, and the categorical nature of their values. That means the majority of fields cannot be used directly as features in most ML models, as those usually require numeric features.

Account Name values cannot be directly converted to a numeric equivalent. Furthermore, this field may theoretically contain an unlimited number of different values, thus encoding is not the option either. In environments where users choose their own usernames, we may attempt to obtain numeric or statistical values, such as username length. On the contrary,

in environments where standardized naming convention is in place, the account's properties may be extracted, such as the type of the account, i.e., personal, non-personal, or system account.

Service Name field is quite similar to Account Name. Depending on the naming convention used for service identifiers, features identifying service properties or its purpose may be extracted.

Client Address field typically contains an IPv4 address. Although IP addresses seem to be numeric, they cannot be treated as such. An IP address represented as a 32-bit binary or decimal number does not represent an ordinal value.

Common approach of representing IP addresses is obtaining location information and representing it as GPS coordinates. However, this is not applicable in our scenario, as we are dealing with private IP addresses. In environments with strict network segmentation, where different IP subnets are assigned to different network segments, IP addresses can be labeled based on their segment, which may reveal valuable information about the type of source computer.

Apart from using existing fields, several numeric fields can be calculated. An example would be the number of services requested for a specific source, which is the most important indicator for Kerberoasting detection. Moreover, the number may be split into several numeric features, representing a count of service tickets per service type, if we were able to distinguish service types based on their identifiers.

Based on the properties of our dataset and the discussion above, we selected the following features for use with ML models in our experiments:

- number of requests for distinct services from a particular source,
- number of requests for distinct services from a particular source split by service type (application, database, etc.),
- type of source account requesting the ticket (personal, non-personal, system account, etc.).

4.2 Experiments and Results

Experiments were conducted with the goal to determine the best combination of features and values of hyperparameters for the ML models that can be used to solve the given problem. To achieve that, the following methodology was used:

1. ML algorithms were fitted on training data with different values of hyperparameters. Their influence on the output was measured on the validation dataset. Appropriate values of hyperparameters were selected for every algorithm.
2. Models with the selected configuration of hyperparameters were applied to the dataset using different combinations of features. The best combination of features was chosen for every algorithm.
3. The algorithms were evaluated and compared on the testing dataset, using the best-determined configuration for every algorithm.

4.2.1 Static Threshold Rule

Kerberoasting can be detected by a static rule containing a numeric threshold for the number of requested service tickets, as proposed in (Kotlaba et al., 2020). The key is to define the right threshold, as this value greatly influences the number of false alerts, as well as the detection sensitivity of the rule.

Based on the distribution of the number of requested services in the training dataset, the threshold value for service count was set to 15. This means that an alert would be generated for every data point with a higher number of service requests. Table 1 summarizes the results of the rule applied to the datasets.

Table 1: Detection results for the static rule.

Dataset	Training	Test
TP	0.0%	0.1%
TN	98.2%	97.7%
FP	1.8%	2.1%
FN	0.0%	0.1%

4.2.2 One-Class SVM

As described in Section 2, One-Class SVM requires choice of a kernel and the ν parameter. Our measurements with different kernels unambiguously confirmed the advantages of the RBF kernel. The γ parameter, which is a coefficient for the RBF kernel, seems to be greatly influencing the number of FP results.

For the experiment with different feature combinations, the ν parameter was set to the value of 0.001, as we expect approximately 0.1% of malicious events in the validation dataset. The γ parameter of the RBF kernel was adjusted experimentally for every model due to its sensitivity. Other parameters were kept at their default values.

The use of different feature combinations greatly influences the results of One-Class SVM. The primary indicator - the number of requested services was present in all the combinations. Adding one more feature, such as service count by type, or the type of user, helped to reduce the number of false positives compared to the model fitted only on aggregated service count.

Table 2: Comparison of the approaches for Kerberoasting detection.

	Static rule		One-Class SVM		Local Outlier Factor	
TP	2	0.1%	3	0.2%	1	0.1%
TN	1455	97.7%	1452	97.5%	1457	97.8%
FP	31	2.1%	34	2.3%	29	2.0%
FN	1	0.1%	0	0.0%	2	0.1%

On the other hand, when the service count was used as scaled, its importance among other features was not captured, and the model performed worse. The best results achieved the model using features representing both source user and target service types.

4.2.3 Local Outlier Factor

LOF is an unsupervised algorithm, and as implemented in Splunk, it cannot be fitted on training data and saved as a model. It is meant to be applied directly with a particular configuration of hyperparameters to a dataset that is to be scanned for outliers.

The output of LOF varies significantly for different counts of neighbors considered. Too big focus on locality was notable for values lower than 20, but on the other hand, there was almost no difference in FP detections on a scale from 20 neighbors to all points in the dataset. For the experiments with different feature combinations, the number of considered neighbors was set to all data points.

Another parameter influencing the output is the expected contamination of the dataset. This parameter seems to have linear impact on the number of FPs. However, when set to a value too low, the malicious event was not evaluated as an outlier.

In further experiments, we were changing the way local density is computed. However, the parameters as metric, algorithm, or even leaf size had little impact on the output of LOF models.

Surprisingly, results of LOF showed almost no sensitivity to different combinations of features used. We measured consistent numbers of false positives among the output of different configurations. Due to this property, the same features as with One-Class SVM were chosen for the comparison.

4.2.4 Comparison

The models with the best-determined configurations and the static threshold rule were all applied to the testing dataset. The comparison of the results is presented in Table 2.

In the static rule, the numeric threshold for the number of requested services was set to a relatively high value (15) to reduce FP detections. However, this resulted in one malicious event not being detected, as the number of requested services in the event was lower than the threshold set in the rule.

One-Class SVM model was able to detect all the three malicious events, but with a slightly higher number of false-positive detections. However, if we were to use the static rule with a lower threshold set - to be able to detect all three malicious events - the number of false-positive detections would be significantly higher than with the One-Class SVM model. This is illustrated in the Table 3, which demonstrates the comparison of results between the static rule with the threshold value set to 5 and One-Class SVM.

Table 3: Use of the static rule with a lower threshold.

	Static rule		One-Class SVM	
TP	3	0.2%	3	0.2%
TN	1342	90.1%	1452	97.5%
FP	144	9.7%	34	2.3%
FN	0	0%	0	0%

Local Outlier Factor in the used configuration yielded the least number of false positives, but on the other hand, it also missed two malicious events. This could be avoided by increasing the value of the contamination parameter, but in that case an increase in the number of FP events is also expected.

5 CONCLUSIONS

In this paper, we proposed adopting machine learning techniques to improve detection of Kerberoasting attack. Adversaries use this attack to target Active Directory environments, with the goal of extracting credentials of service accounts. Detection of Kerberoasting relies traditionally on custom signature-based rules that may not be effective in diverse environments.

We utilized anomaly detection algorithms to identify unusual patterns in authentication to remote services that may indicate the possibility of Kerberoasting attack execution. The algorithms were compared and evaluated on log data originating from a real organization. The solutions were implemented in widely-used Splunk technology, making them applicable in practice easily.

We conducted extensive experiments with One-Class SVM and Local Outlier Factor algorithms, using different configurations of their hyperparameters and features extracted from Windows Events. As the results have shown, ML approach significantly reduced the number of false-positive detections compared to the signature-based approach. At the same time, we did not observe an increase in the number of false negatives.

Furthermore, at the same percentage level of false positives, the One-Class SVM algorithm helped improve detection capabilities, as it detected one attack that was missed by the static rule. LOF algorithm has not proven to be equally effective but offers the advantage of no need for training data.

ACKNOWLEDGEMENTS

This work was supported by the Student Summer Research Program 2020 of FIT CTU in Prague and the grant no. SGS20/212/OHK3/3T/18. The authors acknowledge the support of the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics".

REFERENCES

Breunig, M., Kriegel, H.-P., Ng, R., and Sander, J. (2000). LOF: Identifying density-based local outliers. In *ACM Sigmod Record*, volume 29, pages 93–104.

Desmond, B., Richards, J., Allen, R., and Lowe-Norris, A. G. (2013). *Active Directory: Designing, Deploying, and Running Active Directory*, chapter 1-2, 9-10. O'Reilly Media, 5 edition.

Goldstein, M., Asanger, S., Reif, M., and Hutchison, A. (2013). Enhancing Security Event Management Systems with Unsupervised Anomaly Detection. In *ICPRAM*, pages 530–538.

Hsieh, C., Lai, C., Mao, C., Kao, T., and Lee, K. (2015). AD2: Anomaly detection on active directory log data for insider threat monitoring. In *2015 International Carnahan Conference on Security Technology (ICCST)*, pages 287–292.

Kotlaba, L., Buchovecká, S., and Lórencz, R. (2020). Active Directory Kerberoasting Attack: Monitoring and Detection Techniques. In *Proceedings of the 6th International Conference on Information Systems Security and Privacy, ICISSP 2020, Valletta, Malta, February 25-27, 2020*, pages 432–439. SCITEPRESS.

Matsuda, W., Fujimoto, M., and Mitsunaga, T. (2018). Detecting APT attacks against Active Directory using Machine Leaning. In *2018 IEEE Conference on Application, Information and Network Security (AINS)*, page 60–65.

Medin, T. (2014). Attacking Microsoft Kerberos Kicking the Guard Dog of Hades. In DerbyCon 4.0, Louisville, USA.

Meijerink, M. (2019). Anomaly-based Detection of Lateral Movement in a Microsoft Windows Environment. Master's thesis, Faculty of Electrical Engineering, Mathematics & Computer Science, University of Twente, 7500 AE Enschede, The Netherlands.

Metcalf, S. (2017a). Active Directory Security. https://adsecurity.org/. [Online; accessed on 14-September-2020].

Metcalf, S. (2017b). Active Directory Security: Detecting Kerberoasting Activity. https://adsecurity.org/?p=3458. [Online; accessed on 14-September-2020].

Microsoft Corporation (2017). Microsoft Docs: Security auditing: 4769(S, F): A Kerberos service ticket was requested. https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4769. [Online; accessed on 14-September-2020].

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2020). Scikit-learn: Novelty and outlier detection. https://scikit-learn.org/stable/modules/outlier_detection.html. [Online; accessed on 14-September-2020].

Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., and Platt, J. (1999). Support vector method for novelty detection. In *NIPS*, volume 12, pages 582–588.

Soria-Machado, M., Abolins, D., Boldea, C., and Socha, K. (2016). Kerberos Golden Ticket Protection. Technical report, CERT-EU Security Whitepaper 2014–007. Also available at http://cert.europa.eu/static/WhitePapers/UPDATED%20-%20CERT-EU_Security_Whitepaper_2014-007_Kerberos_Golden_Ticket_Protection_v1_4.pdf [Online; accessed on 14-September-2020].

Splunk Inc. (2020). Splunk Documentation. https://docs.splunk.com/Documentation. [Online; accessed on 14-September-2020].

The Mitre Corporation (2020). Steal or Forge Kerberos Tickets: Kerberoasting. https://attack.mitre.org/techniques/T1558/003/. [Online; accessed on 14-September-2020].

Uppströmer, V. and Råberg, H. (2019). Detecting Lateral Movement in Microsoft Active Directory Log Files. Master's thesis, Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden.