FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

ASSIGNMENT OF BACHELOR'S THESIS

Title:	Detection of Active Directory attacks
Student:	Lukáš Kotlaba
Supervisor:	Ing. Simona Buchovecká
Study Programme:	Informatics
Study Branch:	Computer Security and Information technology
Department:	Department of Computer Systems
Validity:	Until the end of summer semester 2019/20

Instructions

The goal of the thesis is to develop a set of detection rules for security monitoring of Microsoft Active Directory in the form of an extension of existing application Splunk ES Content Update. The rules will be used to detect possible security attacks from Windows Security and Active Directory logs.

1. Get familiar with the Windows Security auditing. Study known attacks targeting Microsoft Active Directory and analyze how they can be detected using Windows Security audit.
2. Get familiar with the Splunk tool and Search Processing Language (SPL).
3. Design and implement the designed rules in the form of an extension of existing application Splunk ES Content Update - design the rule in SPL and include all necessary documentation (attack description and relevant investigative or context searches).

References

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/security-auditing-overview
http://docs.splunk.com/Documentation
https://splunkbase.splunk.com/app/3449/#/overview

prof. Ing. Pavel Tvrdík, CSc.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague December 21, 2018

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Detection of Active Directory attacks

Lukáš Kotlaba

Department of Computer Systems
Supervisor: Ing. Simona Buchovecká

May 12, 2019

Acknowledgements

First and foremost, I would like to show gratitude to my supervisor Ing. Simona Buchovecká for her support and valuable guidance in every step throughout the process of creating this thesis. I would like to thank my colleagues at NN who helped to shape my focus in the security field, inspired me in the topic, and were willing to share their knowledge. My sincere thanks also go to Mgr. Zuzana Vladárová for her willingness to do language corrections. Most importantly, none of this could have happened without continuous support and encouragement from my family, closest friends, and last but not least, my dearest one.

Declaration

Citation of this thesis

Kotlaba, Lukáš. *Detection of Active Directory attacks.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

Organizace, které využívají Active Directory pro správu identit, musí chránit svá data před protivníky a bezpečnostními hrozbami. Tato práce analyzuje známé útoky na Active Directory a možnosti jejich detekce založené na Windows Security auditu. Implementační část je zaměřená na návrh detekčních pravidel pro analyzované scenáře útoků. Pravidla byla navrhnuta a implementována v technologii Splunk, následně otestována a vyhodnocena vykonáním útoků ve virtuálním prostředí. Navrhnutá pravidla, případně detekční principy v nich použité, mohou sloužit jako základ implementace bezpečnostního monitorování Active Directory prostředí v organizacích, a to nezávisle na vybrané technologii. Příloha práce obsahuje navrhnutá pravidla ve formě Analytic Stories, která rozširují obsah existující aplikace Splunk ES Content Update. Analytic Stories jsou navíc doplněna o relevantní vyhledávání, která poskytují kontext využitelný pro investigaci.

Klíčová slova detekce útoků, detekční pravidla, bezpečnostní audit, Lateral Movement, Microsoft Windows, Active Directory, Splunk, Windows Event Log

Abstract

Organizations that use Active Directory for managing identities have to protect their data from adversaries and security threats. This thesis analyses known attacks targeting Active Directory and the possibilities of detection based on Windows Security auditing. The implementation part focuses on designing detection rules covering the analyzed attack scenarios. The rules were designed and implemented in Splunk; tested and evaluated by performing the attacks in a virtual environment. The rules, or the detection principles used in them, can serve as a baseline for implementation of Active Directory security monitoring in organizations, regardless of the chosen technology. The appendix contains the designed rules set in the form of Analytic Stories, extending the content of an existing application Splunk ES Content Update. The Stories are supplemented by related searches providing context useful for investigation.

Keywords attack detection, detection rules, security auditing, Lateral Movement, Microsoft Windows, Active Directory, Splunk, Windows Event Log

Contents

Introduction **1**

Goals **3**

1 Active Directory **5**
1.1 Active Directory overview . . . 5
1.1.1 Introduction . . . 5
1.1.2 Logical structure . . . 6
1.1.3 Physical structure . . . 7
1.1.4 Identifiers . . . 8
1.2 Windows authentication overview . . . 9
1.2.1 Windows logon scenarios . . . 9
1.2.2 Authentication protocols . . . 10
1.2.3 Credential storage . . . 13

2 Attacks targeting Active Directory **15**
2.1 Active Directory as a target . . . 15
2.2 Credential Access . . . 17
2.2.1 Brute Force . . . 17
2.2.2 Kerberoasting . . . 18
2.2.3 Credential Dumping . . . 19
2.3 Lateral Movement . . . 20
2.3.1 Pass the Hash . . . 20
2.3.2 Pass the Ticket . . . 21
2.3.3 Overpass the Hash . . . 21
2.3.4 Golden/Silver Tickets . . . 21

3 Windows Security auditing **23**
3.1 Windows Event Log . . . 23

3.2 Active Directory . . . 26
3.3 Sysmon . . . 27
3.4 Typical auditing scenario . . . 28

4 Splunk and SPL **31**
4.1 Splunk Enterprise . . . 31
4.1.1 Data processing . . . 31
4.1.2 Splunk architecture . . . 32
4.1.3 Splunk licensing . . . 34
4.2 Search Processing Language . . . 34
4.3 Splunk ES Content Update . . . 36
4.4 Splunk in this thesis . . . 37

5 Implementation **39**
5.1 Lab environment . . . 39
5.1.1 Lab description . . . 39
5.1.2 Splunk configuration . . . 40
5.1.3 Windows configuration . . . 42
5.2 Workflow . . . 43
5.2.1 Analytic Stories . . . 43
5.2.2 Design of searches . . . 43
5.3 01 - Brute Force Attacks . . . 44
5.3.1 Information sources . . . 44
5.3.2 Detection searches . . . 46
5.3.3 Other searches . . . 51
5.3.4 Testing . . . 52
5.3.5 False positives . . . 53
5.4 02 - Kerberoasting . . . 53
5.4.1 Information sources . . . 53
5.4.2 Detection searches . . . 54
5.4.3 Other searches . . . 57
5.4.4 Testing . . . 58
5.4.5 False positives . . . 60
5.5 03 - Credential Dumping . . . 60
5.5.1 Information sources . . . 60
5.5.2 Detection searches . . . 62
5.5.3 Other searches . . . 66
5.5.4 Testing . . . 67
5.5.5 False positives . . . 70
5.6 04 - Credential Theft . . . 70
5.6.1 Information sources . . . 70
5.6.2 Detection searches . . . 71
5.6.3 Other searches . . . 77
5.6.4 Testing . . . 77

5.6.5 False positives 81

Conclusion **83**

Bibliography **85**

A Acronyms **95**

B Contents of enclosed CD **99**

List of Figures

1.1 An example of an AD forest . . . 7
1.2 AD object identifiers . . . 8
1.3 NTLM authentication . . . 11
1.4 Kerberos authentication . . . 13

2.1 Kill Chain phases . . . 15
2.2 Kill Chain for AD attacks . . . 17
2.3 Golden Ticket communication . . . 22
2.4 Silver Ticket communication . . . 22

3.1 An example of a Windows Event Log event . . . 26

4.1 Splunk search . . . 32
4.2 Splunk architecture . . . 33
4.3 Basic structure of SPL . . . 34
4.4 Detection search in the ESCU Add-on . . . 37

5.1 Assets in the lab environment . . . 40
5.2 Information contained in the event 4740 . . . 50
5.3 Detection of a brute force attack by S01D04 . . . 53
5.4 Kerberoasting detected by S02D01 . . . 59
5.5 Kerberoasting detected by S02D04 . . . 59
5.6 Kerberoasting detected by S02D05 . . . 59
5.7 Dumping of the *lsass.exe* process detected by S03D01 . . . 68
5.8 Dump of *lsass.exe* made by “calculator” . . . 68
5.9 Detection of saved dump files by S03D03 . . . 68
5.10 Installation of *mimidrv.sys* detected by S03D04 . . . 69
5.11 Use of *vssadmin.exe* detected by S03D06 . . . 69
5.12 Use of *DCSync* detected by S03D07 . . . 69
5.13 Events recorded during Pass the Hash . . . 72
5.14 Events recorded during Pass the Ticket . . . 73

5.15 Scenario for OPtH and PtT attacks 78
5.16 Detection of PtH by S04D01 79
5.17 Detection of PtT and OPtH attacks by S04D02 79
5.18 Detection of OPtH by S04D03 79
5.19 Investigation results returned by S04I03 80
5.20 Detection of forged tickets by S04D04 80
5.21 Detection of forged tickets by S04D05 80
5.22 Detection of MS14-068 attack by S04D06 81

List of Tables

2.1 Mapping of Techniques to Tactics and Kill Chain 16
2.2 Encryption types implemented in Windows 19

3.1 Default PowerShell versions in Windows OSs 26

4.1 Common SPL commands 35

5.1 Assets in the lab environment 40
5.2 Versions of Splunk components 41
5.3 Accounts used in the environment 42
5.4 Events for detection of brute force attacks 46
5.5 Differences in field values between the events 47
5.6 Content of fields in the recorded events 52
5.7 Events for detection of credential dumping 62
5.8 Sysmon events used for detection 62
5.9 GUID values for DS replication 65
5.10 Events for detection of credential theft 71

List of Listings

4.1 Grouping events into transactions 36
5.1 S01D01 - Logon attempts using a non-existing account (Kerberos) [snippet] 48
5.2 S01D02 - Logon attempts using a non-existing account (NTLM) [snippet] . 48
5.3 S01D05 - Excessive failed password attempts towards one account [snippet] 49
5.4 S01D06 - Multiple locked accounts from one source [snippet] 50
5.5 S01D07 - Repetitive lockouts of the same account [snippet] 51
5.6 S02D01 - Possible Kerberoasting activity [snippet] 55
5.7 S02D02 - Excessive service ticket requests from one source [snippet] 55
5.8 S02D03 - Suspicious external service ticket requests [snippet] 56
5.9 S02D04 - Detecting Kerberoasting with a honeypot [snippet] 56
5.10 S02D05 - Detecting Kerberoasting via Powershell [snippet] 57
5.11 Kerberoasting with *Impacket* 58
5.12 Kerberoasting with *Empire* 58
5.13 Kerberoasting with PowerShell 59
5.14 S03D01 - Possible dump of lsass.exe (Sysmon events) [snippet] 63
5.15 S03D03 - Creation of a dump file [snippet] 63
5.16 S03D04 - Installation of an unsigned driver [snippet] 64
5.17 S03D05 - Access to GPP honeypot in SYSVOL [snippet] 64
5.18 S03D07 - Possible dumping via DC synchronization [snippet] 66
5.19 S03C03 - Shadow copy creation [snippet] 66
5.20 S04D01 - Pass the Hash [snippet] 72
5.21 S04D02 - Pass the Ticket [snippet] 74
5.22 S04D03 - Overpass the Hash [snippet] 74
5.23 S04D04 - Golden/Silver Ticket based on PowerShell script [snippet] 76
5.24 S04D06 - Possible forged service ticket [snippet] 76

Introduction

Active Directory is the most commonly used technology for managing identities in modern enterprises. Its usage is ranging from small organizations, through mid-size businesses, to large corporations. The fact that whole domains are controlled by Active Directory, together with its prevalence, makes it an ideal target for attackers.

Attacks against computing infrastructures, whether simple or complex, have existed as long as computers have. However, within the past decade, increasing numbers of organizations of all sizes, in all parts of the world have been attacked and compromised.

Organizations implement network security controls, such as firewalls, or antivirus solutions fighting against malware, but often underestimate the concept of defense in depth. When the first line of security mechanisms fails and adversaries get access to Active Directory, the consequences are fatal, often resulting in full domain compromise.

The output of this thesis will help organizations monitor their Active Directory instances for attacks which they are exposed to. It aims to provide an example implementation of detection mechanisms by using a modern tool, which "sees" increasing popularity among companies, not only with security applications. Real-time detection of threats allows organizations to react and take actions in time. However, this thesis does not discuss the concepts of attack prevention and mitigation.

The thesis starts with an introduction to Active Directory, continues with an analysis of known security attacks targeting Active Directory, and subsequently, possibilities of detecting these attacks by using Windows Security auditing. This is followed by a part describing the technology of Splunk and capabilities of its Search Processing Language, which I have chosen as a tool for implementation of detection rules.

The implementation part involves the process of designing the detection rules by using various available approaches. It also incorporates the process of testing and evaluation of the designed rules. The rules are implemented in Splunk, in the form of Analytic Stories, extending the security content of an existing application Splunk ES Content Update. The Stories are supplemented by related searches providing context useful for investigation. Missing Analytic Stories related to Active Directory attacks are added to the application content, which as a whole offers a guideline for organizations addressing various security threats.

Goals

The goal of the theoretical part of this thesis is to analyze known attacks targeting Microsoft Active Directory and to determine characteristic aspects of the attacks that can be used for detection.

Its second goal is to evaluate possibilities of Windows Security auditing, Active Directory auditing, or eventually other auditing options that may help with attack detection. To support the implementation part, the goal is to review the capabilities of Splunk, its Search Processing Language and the structure of an existing application Splunk ES Content Update.

The goal of the implementation part is to design a set of detection rules which use the reviewed auditing options to detect the known attacks and implement the designed rules in the form of an extension of the application Splunk Splunk ES Content Update, including all necessary documentation (attack description and relevant investigative or context searches).

Chapter 1

Active Directory

Microsoft Active Directory (AD) is a common repository for information about objects that reside on a company network, such as users, groups, computers, printers, applications, and files. The objects have numerous attributes, specific permissions, and relations. AD stores all this data in a hierarchical organizational structure and provides access to it for users. This chapter serves as an introduction to AD key concepts, its structure, and its role in organizations.

1.1 Active Directory overview

1.1.1 Introduction

Active Directory is a proprietary implementation of a *directory service* for Microsoft's *Network Operating System (NOS)*. NOS is the term used to describe a networked environment in which various types of resources, such as user, group, and computer accounts, are stored in a central repository. This repository, called Active Directory, contains network, application, or NOS information that is controlled by administrators and accessible to end users. The directory service that provides access to this repository is called *Active Directory Domain Services (AD DS)*[1]. [1]

In general, there are many different systems that have characteristics of a directory service, for instance, Domain Name System (DNS) or email systems. However, there are standards for how a true directory service is implemented and accessed: X.500 Directory Access Protocol, and its evolution *Lightweight Directory Access Protocol (LDAP)*. It was the LDAP that served as a basis for implementing directory services in companies such as Netscape, Sun, Novell, IBM, and Microsoft. Microsoft Active Directory is based on the LDAPv3 protocol, which is an updated version of LDAP, introduced in 1997. The first version of Microsoft AD was released with Windows 2000 and has been a part of Windows Server operating systems (OSs) ever since. [1, 2]

[1]The terms Active Directory and Active Directory Domain Services are often used interchangeably.

1.1.2 Logical structure

Data stored in AD creates an illusion of a hierarchical structure, called *AD schema*. But in reality, the structure is stored in a flat database consisting of rows and columns. The AD database uses an Extensible Storage Engine (ESE) technology and it is stored in the file *NTDS.dit*[2], by default in `%SystemRoot%\NTDS` directory on *domain controllers*. [1, 3]

Every entry of the database is referred to as an object. The AD structure basically contains two types of objects: *containers* and *non-containers*, also called *leaf objects*. Leaf objects can be nested into containers, which can contain both containers and leaf objects. This is how the hierarchical structure is formed.

AD's logical structure is built around the concept of *domains*, referred to as *Windows domains* or *Active Directory domains*. A domain contains the logical components to achieve administrative goals in the organization and becomes the security boundary for the objects inside it. [2]

As defined in [1], an AD domain is made up of the following components:

hierarchical structure of containers and objects;

DNS domain name as a unique identifier;

security service which authenticates and authorizes any access to resources via accounts;

policies that dictate how functionality is restricted for users or machines.

Active Directory uses DNS domain names and *NetBIOS* domain names. The DNS domain name is also often referred to as *fully qualified domain name (FQDN)*. These two naming systems also apply to computer names and other objects in the AD. [4]

The domain is created as the root node of a hierarchical structure called a *domain tree*. A domain tree is a collection of domains and it reflects the structure of an organization. If the domain has subdomains, these create subtrees connected together in a hierarchical fashion.

While a domain tree is a collection of domains, a *forest* is a collection of one or more domain trees. A single domain composes a forest. By adding domains to the initial domain tree, or creating new domain trees, the result is still only a single forest. The forest is named after the first domain that is created, also known as the forest root domain.

Trees inside a forest, and also domains inside a tree, are implicitly connected through two-way transitive *trusts*. Two-way trust means the authentication requests can be processed in both directions; transitive means child domains are trusted too, even though there is no direct connection between them. Separate forests are not connected with trusts by default. However, to provide seamless resource access between them, a forest trust can be deployed. [1, 2]

The large-scale relationships between domains, trees, and forests described in the last few paragraphs are illustrated in Figure 1.1.

[2]NTDS stands for NT Directory Service, the .dit extension means Directory Infomration Tree.

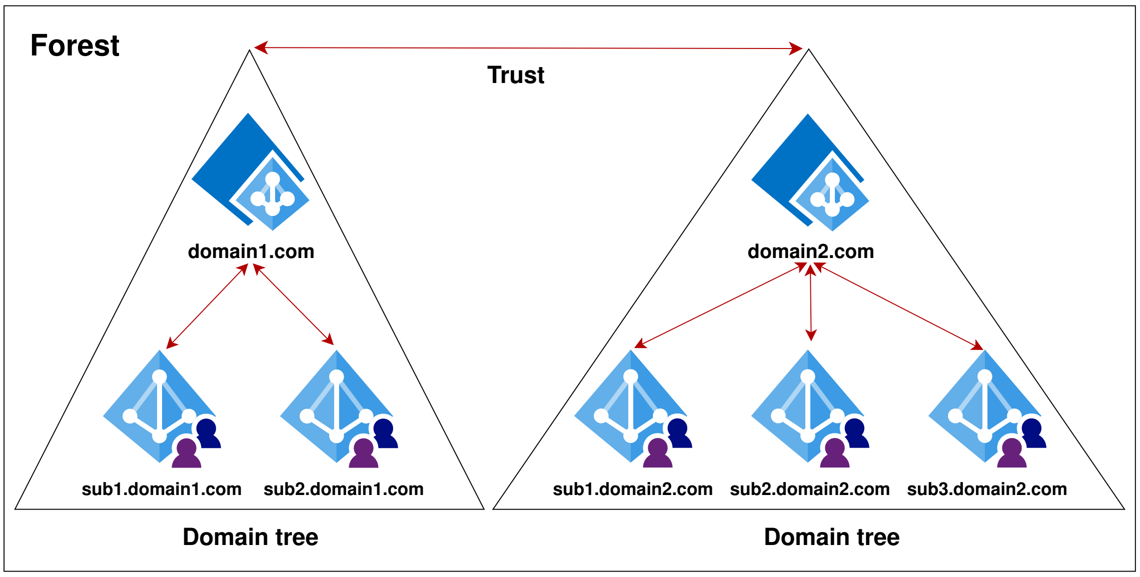


Figure 1.1: An example of an AD forest

In the small-scale, another hierarchical structure exists inside an AD domain. Objects that have similar security and administrative requirements are usually grouped in containers called *Organizational Units (OUs)*. These are used for building object hierarchies within a domain. An OU can have *group policies* applied to it. Once an object is assigned to an OU, it inherits its security settings and permissions. OUs also help delegate administrative control to individuals for specific tasks, according to *Least-Privilege Administrative Models* [5]. Typical objects contained in an OU are users, groups, contacts, computers, printers, or other OUs. [1, 2]

1.1.3 Physical structure

The basic physical component of AD is a *domain controller (DC)*. The DC is a computer that runs a Windows Server OS and holds the AD DS role.[3] The DC stores the *NTDS.dit* database file, which is replicated with other DCs in the domain by using *Directory Replication Service (DRS) Remote Protocol*. The domain can have any number of DCs, but one DC can be authoritative for one domain only. [1, 2]

Even though AD is a multi-master directory, there are some situations in which there should only be a single DC that can perform certain functions. The server that is the master for a particular function or role is known as the *Flexible Single Master Operator (FSMO)*. [1]

There are five FSMO roles defined:

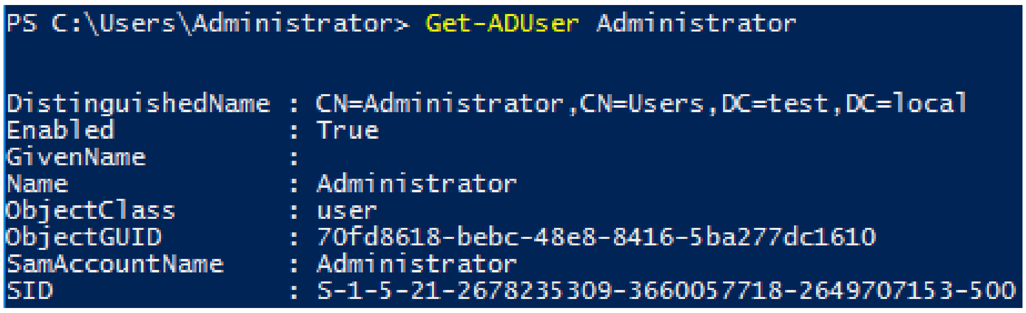
- Schema Master

[3]Other servers that are not DCs, are usually referred to as *member servers*.

- Domain Naming Master
- Relative Identifier (RID) Master
- Infrastructure Master
- Primary Domain Controller (PDC) Emulator

1.1.4 Identifiers

Active Directory introduces several ways how objects can be uniquely referenced and identified [1, 2]. Figure 1.2 shows an example of identifier attributes belonging to built-in *Administrator* account, retrieved from AD by `Get-ADUser` PowerShell cmdlet.



```
PS C:\Users\Administrator> Get-ADUser Administrator


DistinguishedName : CN=Administrator,CN=Users,DC=test,DC=local
Enabled           : True
GivenName         :
Name              : Administrator
ObjectClass       : user
ObjectGUID        : 70fd8618-bebc-48e8-8416-5ba277dc1610
SamAccountName    : Administrator
SID               : S-1-5-21-2678235309-3660057718-2649707153-500
```

Figure 1.2: AD object identifiers

Distinguished Name (DN) identifies an object based on its full path within the AD. There are three types of AD naming attributes that generate DNs:

organizationalUnitName (OU) refers to the OU in which the object is located;

domainComponent (DC) is the naming attribute for the domain and the DNS;

commonName (CN) refers to the objects and containers within the directory. [2]

The example in Figure 1.2 shows user object `Administrator` (CN), which is a member of OU `Users` (CN) and part of the domain `test.local` (DC).

Relative Distinguished Name (RDN) is the name used to reference an object within its parent container uniquely. It is possible to have the same RDN for multiple objects within the directory, but all of them need to be in separate containers. RDN of the example shown in Figure 1.2 would be `CN=Administrator`.

If the hierarchical path got changed for an object, its DN would be changed. Therefore, numeric identifiers that are resistant to such changes do exist.

Globally Unique Identifier (GUID) is a 128-bit number assigned to every AD object by the system at creation. Its value is not changed by modifying or moving objects. The value is stored in `objectGUID` attribute and is displayed as five groups containing 8-4-4-4-12 hexadecimal digits (Figure 1.2). GUID is not guaranteed to be unique but has a very low probability of being duplicated. [1, 2]

Security Identifier (SID) is a variable-length identifier used to identify a trustee or security principal. It is unique within the domain. If the object is migrated to another domain, the SID values associated with it will be changed. Also, the SID value assigned by one domain will not be accepted by another domain. Old SID values are stored in the `sidHistory` attribute. SID is composed of 2 fixed fields and up to 15 additional fields, all separated by dashes. There are several well-known SIDs; the example in Figure 1.2 shows the well-known SID ending with value `500`, used for the built-in *Administrator* account. [1, 2]

1.2 Windows authentication overview

The Windows OSs require all users to log on to the computer with a valid account to access local and network resources. *Authentication* is a process of verifying the claimed identity of an object; *authorization* is a process of verifying that the object has rights to access particular resources. AD is the default technology for storing identity information on domain-joined systems, and therefore it is tied closely to authentication and authorization processes. Following sections, based on the Microsoft documentation on this topic [6], provide an overview of some key concepts.

1.2.1 Windows logon scenarios

Users are authenticated to Windows-based computers by a logon process. Depending on how the logon process occurs, there are several scenarios defined:

- Interactive logon
  - Local logon
  - Remote logon
- Network logon
- Smart card logon
- Biometric logon

During an *interactive logon*, a user typically enters credentials in the credentials entry dialog box. Alternatives for presenting credentials in the form of username and password are smart card logon and biometric logon.

Users can perform an interactive logon by using a *local account* or a *domain account*. Depending on the account type, the logon process confirms the user's identification to the security database on the user's local computer or to the AD database. A *local logon* grants a user permission to access resources on the local computer or resources on networked computers. A *domain logon* grants a user permission to access local and domain resources. Domain logon requires that both the user and computer have their accounts in AD and the computer is physically connected to the network.

Users can perform an interactive logon to a computer either locally, when the user has direct physical access to the computer, or remotely, through Terminal Services or Remote Desktop Protocol (RDP). Such logon is qualified as remote interactive.

A *network logon* can only be used after user, service, or computer authentication has taken place. The network logon process does not use the credentials entry dialog boxes; the authentication is typically invisible to the user unless alternative credentials have to be provided. Previously established credentials are used to confirm identity to any network service that the user is attempting to access.

1.2.2 Authentication protocols

To provide network logon functionality, Windows systems include these authentication mechanisms:

- Public key certificates
- Secure Sockets Layer/Transport Layer Security (SSL/TLS)
- Digest
- NT LAN Manager (NTLM)
- Kerberos version 5 protocol

To authenticate users and computers within a domain, *Kerberos* and *NTLM* are primarily used protocols. Authentication based on public key certificates is utilized as an alternative within the smart card logon. Other protocols, such as Digest or SSL/TLS are mostly used for authentication within web services.

NTLM is a family of authentication protocols, which authenticate users and computers based on a challenge/response mechanism. When the NTLM protocol is used, a resource server must contact a domain authentication service on the DC to authenticate domain accounts, or in the case of a local account, look up the account in the local account database.

NTLM credentials are based on data obtained during the interactive logon process and consist of a domain name, a user name, and a one-way hash[4] of the user's password. Figure 1.3 illustrates following steps of NTLM authentication:

1. A user accesses a client computer and provides a domain name, username, and password. The client computes a cryptographic hash of the password and discards the actual password.[5]

[4]*NT hash* in modern Windows systems (also called NTLM hash), older systems used *LM hash*.

[5]This step applies for interactive authentication, only in the case that NTLM was used.

2. The client sends the username to the server (in plaintext).

3. The server generates a 16-byte random number, called a challenge or nonce, and sends it to the client.

4. The client encrypts this challenge with the hash of the user's password and returns the result to the server. This is called the response.

5. The server sends the following three items to the DC:

   - username;
   - challenge sent to the client;
   - response received from the client.

6. The DC uses the username to retrieve the hash of the user's password from the *Security Account Manager* database. It uses this password hash to encrypt the challenge.

7. The DC compares the encrypted challenge it computed (in step 6) to the response computed by the client (in step 4). If they are identical, authentication is successful.

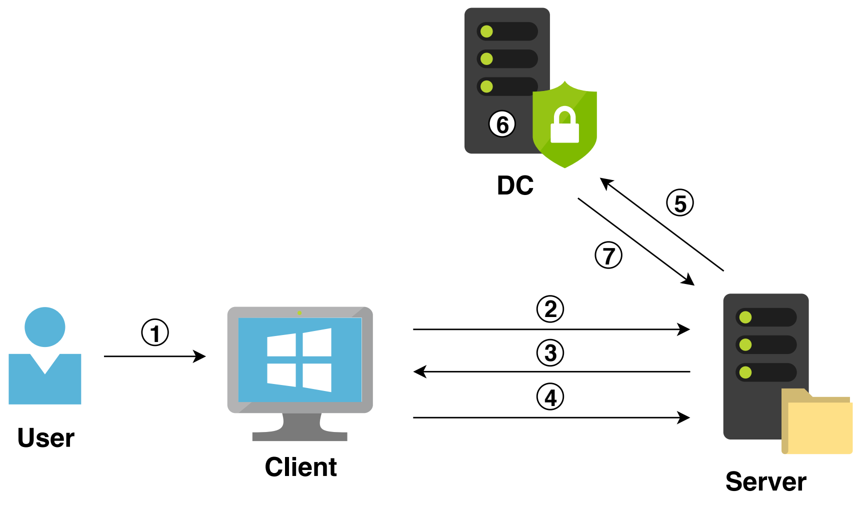


Figure 1.3: NTLM authentication

NTLM is used for local logon authentication on non-domain controllers, and also for authentication with systems configured as a member of a Workgroup. For AD domain environments, Kerberos authentication is the preferred authentication method. To select between Kerberos and NTLM authentication, Windows OSs implement *Negotiate* security package. Negotiate selects Kerberos unless it cannot be used by one of the systems involved in the authentication process.

Windows OSs implement Kerberos version 5 authentication protocol, which is specified in RFC 4120 [7]. Microsoft's proprietary implementation of this protocol adds some functionality beyond the RFC specification, such as authorization or optional *Privilege Account Certificate (PAC)* validation [8].

Kerberos is the default protocol used within an Active Directory domain. Kerberos authentication is more secure than NTLM authentication, as with Kerberos, passwords never traverse the network in plaintext or encrypted formats. Instead, session-specific keys are generated for use over a short period of time through the use of tickets. The tickets are issued by *Kerberos Key Distribution Center (KDC)*, which is integrated into a DC in the Microsoft's Kerberos implementation. The KDC uses AD as its security account database.

Figure 1.4 illustrates following Kerberos authentication steps (described in [1] and [9]), which occur when a user attempts to access a service:

1. To begin the authentication process, an *AS_REQ* message is sent from client to KDC. This message proves the user's identity and is partially encrypted with a hash of the user's password computed by the client computer.

2. The DC validates the request and produces a *Ticket Granting Ticket (TGT)*. The TGT is sent back to the client as *AS_REP* message. The TGT contains PAC with information about all the security groups in which the user is a member. It is encrypted and signed by the KDC service account (*krbtgt*). The client caches the TGT in memory.

3. The client sends a *TGS_REQ* message to the DC to request a *service ticket* for a specific service. Rather than providing credentials again, the message contains the cached TGT obtained in the previous step.

4. The DC validates the *TGS_REQ* and constructs a *Ticket Granting Service (TGS)* ticket[6] for the requested service. The TGS ticket, partially encrypted with a hash of the service's password, is sent back to the client in a *TGS_REP* message. The client caches this ticket in memory for subsequent use when authenticating directly to the service.

5. The client presents the TGS ticket to the service in an *AP_REQ* message. The service uses it to authenticate the user. The service might also use the user's access token (contained in the ticket) to perform authorization before allowing access.

6. Optionally, the service can respond with an *AP_REQ* message for mutual authentication of the service.

7. Optionally, the service may also send the TGS ticket to a KDC to validate the PAC to ensure the user's group membership presented in the ticket is accurate.

[6]TGS ticket and service ticket are equivalent terms.

8. If the PAC validation occurs, the KDC informs the server hosting the specific service about the validation result.

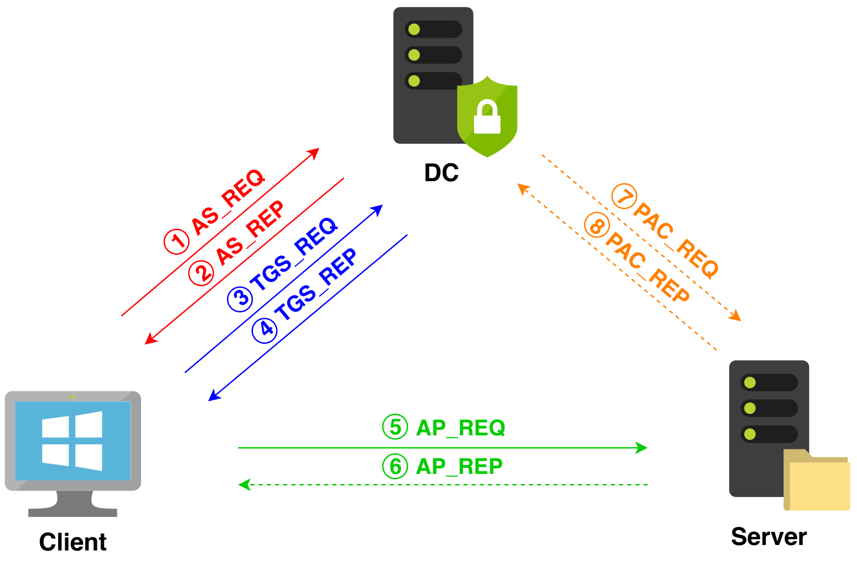


Figure 1.4: Kerberos authentication

Kerberos allows users to access services on the network transparently by simply requesting a service ticket. When clients request service tickets for given services from a DC, they use identifiers called *Service Principal Names (SPNs)*. SPN is stored in AD in the `servicePrincipalName` multivalued attribute. SPN is constructed in the form of a service identifier, followed by the hostname, and optionally, a port number. The service identifier is a predefined string that the client and server agree on. To enable authentication, Kerberos requires that SPN be associated with at least one service logon account. An example of a SPN for a database service could be `MSSQLSvc/DB01.test.local:1433`. [1]

1.2.3 Credential storage

Apart from *NTDS.dit*, the AD database file, there are other locations in Windows OSs where credentials can be found.

On a local computer, credentials are stored in *Security Account Manager (SAM) database*. This database contains local users and groups defined on the machine, along with their passwords and other attributes. On DCs, the SAM does not store the domain-defined users, but stores the system's administrator recovery account and password. The SAM database is stored in *Registry* under `HKEY_LOCAL_MACHINE\SAM` and its write-protected copy also under `HKEY_LOCAL_MACHINE\SECURITY`. [6, 10]

Local Security Authority (LSA) is a subsystem that authenticates and logs users onto the local system. Its functionality is supplied by a protected system process *Local Security Authority Subsystem Service (LSASS) - lsass.exe.* The major tasks of LSA service are:

- loading libraries for authentication;
- enforcing the security policy on the system;
- verifying users logging onto the system;
- handling password changes;
- creating access tokens;
- sending security audit messages to the Event Log.

The LSASS process stores credentials in its memory on behalf of users with active Windows sessions. The stored credentials let users seamlessly access network resources, without re-entering their credentials. LSASS can store credentials in multiple forms, including passwords in reversibly encrypted plaintext, Kerberos tickets (both TGT and TGS), NT hashes, or LM hashes. [6, 10]

LSASS policy database is stored under `HKEY_LOCAL_MACHINE\SECURITY`. Besides the local system security policy settings, it also stores *LSA Secrets* – secret pieces of data stored in encrypted form. The data includes, for instance, cached domain logons or service account logons. [10]

The components described so far directly support authentication processes, and hence store credentials. However, there are other not so obvious places where credentials can be located. Examples of such places are backups of DCs, scripts with hardcoded account credentials, or *Group Policy Preferences (GPP).*

GPP is a collection of *Group Policy* settings. Group Policy is a technology used for centralized management of settings on domain-joined computers. The settings are stored as a group of files and folders on the system volume (*SYSVOL*) of each DC, to which all *Authenticated Users* have read access. [11, 12]

One of the features of GPP is the ability to store and use credentials. Examples of its typical usage include managing passwords of local *Administrator* accounts or deploying scheduled tasks. [12]

Chapter 2

Attacks targeting Active Directory

By understanding what important role Active Directory plays in an enterprise domain, and what kind of data it stores, it is not surprising that AD is often a target of attacks. Indeed, AD does not even have to be the target itself, as it may only serve as a bare tool providing a path for compromising more interesting systems in the domain. This chapter analyzes known attacks and approaches used by adversaries to compromise AD.

2.1 Active Directory as a target

Active Directory resides inside an enterprise network, which is usually protected by multiple elements of perimeter security, such as firewalls or intrusion detection systems (IDSs). Adversaries that intend to compromise AD need to overcome these impediments by developing complex systematic attacks with the nature of an *intrusion*.

The essence of an intrusion is that the adversary must develop a payload to breach a security boundary, establish a presence inside a trusted environment, and from that presence, take actions towards objectives by moving laterally inside the environment. Such sophisticated attacks consist of several phases. The paper [13], published by Lockheed Martin [14], introduced a *Kill Chain* model (visualized in Figure 2.1) that describes the particular phases of an intrusion.

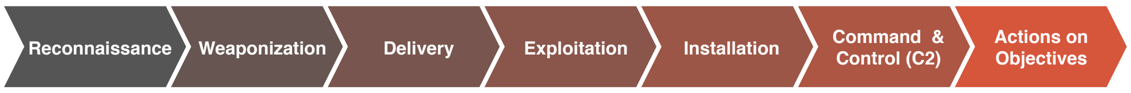


Figure 2.1: Kill Chain phases

To accomplish their goals, adversaries may use different approaches, especially different *Tactics, Techniques, and Procedures (TTPs)*. For every phase of the Kill Chain, an attacker chooses a strategy; in other words, *Tactics*. For the chosen Tactics, there may exist multiple *Techniques* that define courses of actions. And upon the particular Technique, the attacker follows a *Procedure*, which represents the modus operandi - a sequence of steps that need to be performed. The attacker then chooses tools which allow executing the desired actions.

Mitre ATT&CK framework is an extensive knowledge base of Tactics, Techniques, and Procedures widely used by both adversaries and defenders. It presents known TTPs in the form of matrices, where the columns represent Tactics and the elements within them represent related Techniques. The Tactics of ATT&CK framework cover the last five phases of the Kill Chain model. The Mitre Enterprise Matrix contains multiple Tactics and Techniques related to Active Directory. [15]

The attacks towards AD typically follow a very similar scenario. The goal of adversaries is to compromise an AD domain and maintain persistent access to it. Compromising an AD domain usually means gaining rights equivalent to *Domain* or *Enterprise Admins*. The article [16] describes steps required for gaining *Domain Admin* rights in AD:

1. initial access
2. reconnaissance (internal)
3. exploitation & privilege escalation
4. data access & exfiltration
5. persistence

The steps described above are somehow consistent with the Kill Chain presented in Figure 2.1. In general, the terminology within Kill Chain phases, Tactics, and Techniques is not so strictly followed. The Kill Chain model is generally applicable, and can be modified to reflect a particular group of attacks. Figure 2.2 shows Kill Chain model adapted to AD [17]. This model contains loops, which describe the course of intrusions targeting AD more accurately.

Following chapters focus on particular attack Techniques related to AD. Table 2.1 shows the mappings of these Techniques to Tactics and Kill Chain phases. [15]

Kill Chain	**Tactics**	**Techniques**
Exploitation	Credential Access	Brute Force
		Kerberoasting
		Credential Dumping
	Lateral Movement	Pass the Hash
		Overpass the Hash
		Pass the Ticket
		Golden/Silver Tickets

Table 2.1: Mapping of Techniques to Tactics and Kill Chain

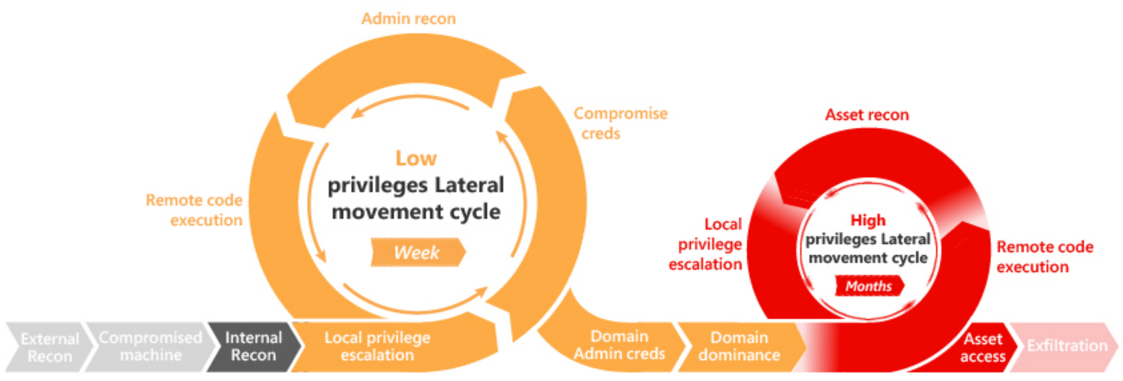


Figure 2.2: Kill Chain for AD attacks

2.2 Credential Access

Credential access is a tactic resulting in access to system, domain, or service credentials that are used within an enterprise environment. Adversaries aim to obtain legitimate account credentials to use within the internal network of an organization.

2.2.1 Brute Force

A *brute force attack* is a trial-and-error method used to get access to accounts. Adversaries may use brute force techniques to guess account credentials, normally a username and a password. In case that password hashes are obtained, attackers may attempt to crack them by trying to guess the original passwords used to compute the hashes. [15]

Adversaries with no knowledge about the user accounts in the domain may start by blindly trying to guess usernames. This method is not particularly efficient, as the attacker would subsequently need to guess the password for that particular username.

Assuming that adversaries were able to get information about usernames in the environment, or are aware of the naming convention used in the domain, they may attempt to guess passwords for those accounts. Another approach is to attempt passwords towards Windows built-in default accounts, such as *Administrator* or *Guest*.

The logon attempts can be carried manually, by typing the credentials into a logon credentials dialog box, or automatically, by using a script. Adversaries may supply the script by a list of known or commonly used passwords, or systematically generate various passwords. Usage of a script allows performing a high number of attempts in a short timeframe. The downside is that it may cause numerous authentication failures and account lockouts, depending on the *Account Lockout Policy* settings. However, adversaries with read access to the AD can query the attributes like `badPwdCount` or `lockoutThreshold` for the accounts, and adjust their script to include a sleep element, or to change target accounts prior to reaching the lockout threshold [18].

Account Lockout Policy settings allow accounts to be locked after several failed password attempts. After that, they may be unlocked automatically after some time, or they need to be manually unlocked by an administrator [19]. Various security considerations need to be taken into account while implementing this policy setting, as possible negative consequences may emerge. For example, a denial of service (DoS) attack could be performed on a domain that has an account lockout threshold configured [20].

To prevent account lockouts, there is another effective brute force technique, called *Password spraying*. This technique uses one password, or a small list of passwords, that may be commonly used, and attempts to logon with that password and many different accounts. Since the list of passwords is smaller than the account lockout threshold, the accounts do not get locked out. [21]

There are plenty of brute force scripts available on the Internet; some of them offer options to set advanced parameters or are able to perform automatic account enumeration. Examples of PowerShell brute force scripts are *Brute-LocAdmin* [22] or *Invoke-SMBAutoBrute* [23].

2.2.2 Kerberoasting

Example of Kerberos authentication in Section 1.2.2 described the process of requesting access to a service identified by Service Principal Name. Any authenticated user possessing a valid TGT may request one or more TGS tickets for any SPN from a domain controller.

This process can be abused by adversaries in a technique called *Kerberoasting*. An attacker that controls a user account can request a service ticket. The ticket may be encrypted with a weak cipher suite, such as *RC4-HMAC-MD5*, which means the service account's NT password hash is used to encrypt the service ticket. The attacker then exports the ticket from memory and attempts to crack it offline by trying different NT hashes. When the ticket is successfully opened, the correct service account password is discovered in plaintext. Cracking of hashes is usually done on adversary-controlled systems with high computational power, outside of the target network. [24, 25]

Table 2.2 shows implemented encryption types used by Kerberos in Windows OSs. Starting from Windows Server 2008 and Windows Vista, the suites containing AES cipher have been set as default, replacing previous default RC4 cipher suites. Also, cipher suites involving DES cipher have been disabled starting from Windows 7 and Windows Server 2008 R2. [26]

These updates comply with security issues arising from RC4 and DES ciphers, as these ciphers are considered obsolete nowadays. However, Windows allows enabling these suites via policy setting for backward compatibility [27].

The reason why Kerberoasting is successful is that many service account passwords are weak, and of the same length as the domain password minimum. Another problem is that service accounts often don't have passwords set to expire. Furthermore, most service

Type	Cipher suite name
0x1	DES-CBC-CRC
0x3	DES-CBC-MD5
0x11	AES128-CTS-HMAC-SHA1-96
0x12	AES256-CTS-HMAC-SHA1-96
0x17	RC4-HMAC-MD5
0x18	RC4-HMAC-EXP

Table 2.2: Encryption types implemented in Windows

accounts are over-permissioned; they contain rights to access certain objects or rights equivalent to *Administrator*. [25]

The first step of Kerberoasting attack is usually *SPN scanning*. Querying for registered SPNs enables an attacker to identify all service accounts supporting Kerberos authentication together with their role. Checking whether the service accounts have the attribute `AdminCount` equal to `"1"` identifies accounts which are members of highly privileged groups. Attackers use these methods to identify interesting service accounts to focus on. [25]

Kerberoasting and SPN scanning can be performed directly from PowerShell [25], or by using various tools. Such tools include PowerShell script *Invoke-Kerberoast*, which is also part of the offensive framework *Empire* [28], or *GetUserSPNs* module of *Impacket*, which is a collection of Python classes for working with network protocols [29].

2.2.3 Credential Dumping

Credential dumping is an unauthorized process of obtaining account credential information, usually executed as preparation for *Lateral movement* tactics. Credentials can be obtained in various forms, including plaintext passwords, hashes, or Kerberos tickets. [30]

Section 1.2.3 introduced different locations where credentials can be found on a Windows system. The tools used for credential dumping use various methods for gaining credentials from these places, both locally and remotely.

The primary techniques for dumping credentials from AD involve interacting with the LSASS process on a DC, grabbing a copy of the AD database file (*NTDS.dit*), or tricking a DC into replicating password data to the attacker. There are several different ways to execute commands remotely on a DC. The most reliable remote execution methods involve either the use of PowerShell or *Windows Management Instrumentation (WMI)*. [3]

Dumping credentials from a local machine involves interaction with the memory of the LSASS process, or access to the Registry hives storing SAM database and LSA secrets. For remote systems, the LSASS process memory can be dumped from the target host and analyzed on a local system. To harvest credentials from the memory of LSASS process *Administrator* or *SYSTEM* rights are required. To enumerate the SAM database or the Registry hives containing LSA secrets *SYSTEM* level access is required. [30]

The palette of tools that can be used for credential dumping is wide. For instance, the following tools can be used to dump the memory of the process *lsass.exe*:

Task Manager a Windows built-in tool for managing processes;

Mimikatz and particularly its module *sekurlsa* [31];

ProcDump a tool developed by *Sysinternals* [32].

The article [3] mentions several methods for dumping the AD database credentials:

- creating a *Volume Shadow Copy* of a DC by using *Volume Shadow Copy Service (VSS)*;
- running a utility *ntdsutil.exe* used with *DCPromo* to build a new DC;
- executing *DCSync* - a *Mimikatz* module to pull data from a DC by abusing the DRS protocol;
- using *reg.exe* to dump Registry hives directly;
- invoking *Mimikatz* directly on a DC.

Section 1.2.3 mentions an ability to distribute credentials by using GPP. However, this feature contained a serious vulnerability: the AES key that is used to encrypt the passwords was published online. Although Microsoft released a security bulletin and a patch for this vulnerability [33], the patch does not remove existing GPP files with passwords from *SYSVOL*. Thus, adversaries may still attempt to scan *SYSVOL* for the presence of credentials. [12]

2.3 Lateral Movement

Lateral movement is a tactic in which adversaries systematically move through a network, accessing one system for another. The process may involve the execution of tools or gathering information from remote systems.

2.3.1 Pass the Hash

Pass the Hash (PtH) is a technique that attacks use of NTLM authentication described in Section 1.2.2. This method bypasses the standard authentication steps that require a plaintext password. An attacker uses previously captured NT hash to impersonate the user and authenticate towards a remote system. A valid NT hash needs to be previously obtained by using a credential dumping technique. [34]

Although the PtH attack is known over twenty years, it is still functional in modern Windows OSs. By the time, numerous tools allowing to perform the attack were developed, and the PtH functionality was included in a variety of offensive frameworks, such as *Metasploit* [35, 36]. Also, Microsoft has provided advisory for mitigation and defense of PtH and other credential theft attacks [37].

2.3.2 Pass the Ticket

Pass the Ticket (PtT) is a technique that attacks Kerberos authentication. As explained in Section 1.2.2, during the authentication process TGT and TGS tickets of a particular user are cached in memory. Adversaries may obtain these tickets by using a credential dumping technique. An attacker can then use the tickets to access any resource that the user has privileges to access. TGTs can be used to request service tickets, whereas TGS tickets allow access to particular resources. [38]

Similarly, as with PtH, there are several tools and frameworks available that can pass Kerberos tickets. Within *Mimikatz*, the module `kerberos::ptt` allows performing of PtT and does not require any special privilege [31].

2.3.3 Overpass the Hash

Overpass the Hash (OPtH) can be thought of as a combination of PtH and PtT attacks described above. An attacker provides NT password hash, but instead of passing it to impersonate a user directly, it permits the Kerberos provider to request a TGT. The subsequent steps then follow the Kerberos authentication process, as explained in Section 1.2.2. [39]

OPtH functionality is implemented in newer versions of `sekurlsa::pth` module of *Mimikatz* [31].

2.3.4 Golden/Silver Tickets

Apart from passing valid Kerberos tickets obtained from memory, or using an NT hash to request them, some techniques allow the creation of forged tickets. There are two types of forged tickets: *Golden* and *Silver*.

Golden Tickets are forged authentication tickets allowing to generate TGTs for any user of the target domain. Since a Golden Ticket is a forged TGT, it is sent to DC as part of the *TGS_REQ* message requesting a service ticket. The communication skips the first two steps (*AS_REQ* and *AS_REP*), as shown in Figure 2.3. [40]

To generate Golden Ticket, an attacker has to know the AD *krbtgt* account's password hash. Golden Ticket can be generated and used on any machine, even not domain-joined. Also, when the ticket is provided, systems trust the ticket validity. Even if the ticket claims that it is valid longer than the setting applied in domain policy, it is accepted as such. [40]

Silver Tickets are forged service tickets (TGS). They are provided directly to the server hosting the specific service for which the ticket is generated. This means that there is no communication with the DC, as visible in Figure 2.4. The scope of a forged service ticket is limited to accessing a particular service on a specific server. [40]

Silver Tickets are encrypted by the service account password hash, which means that an attacker needs to know this hash to generate Silver Tickets. If no PAC validation occurs, the ticket can even include a PAC that is entirely spoofed. [40]

Golden and Silver tickets can be generated by using the respective *Mimikatz* modules `kerberos::golden` and `kerberos::silver`. [31]

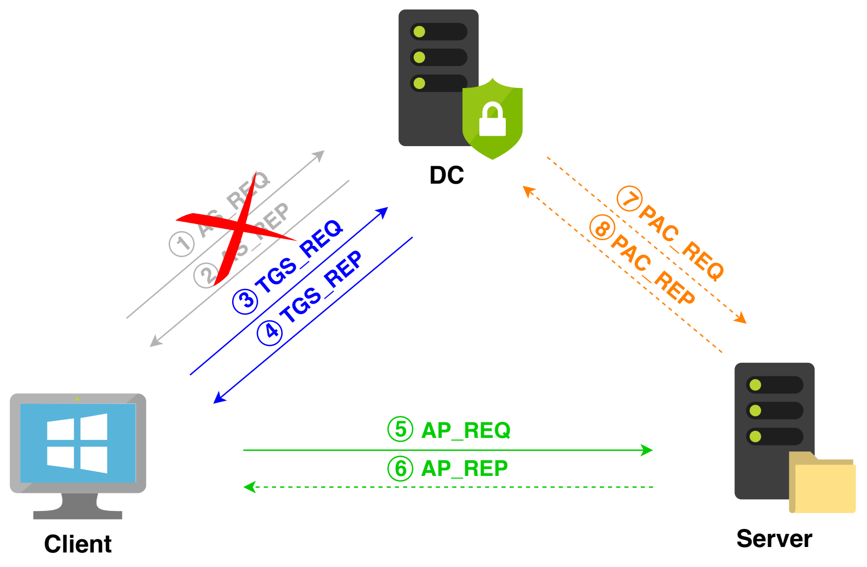


Figure 2.3: Golden Ticket communication

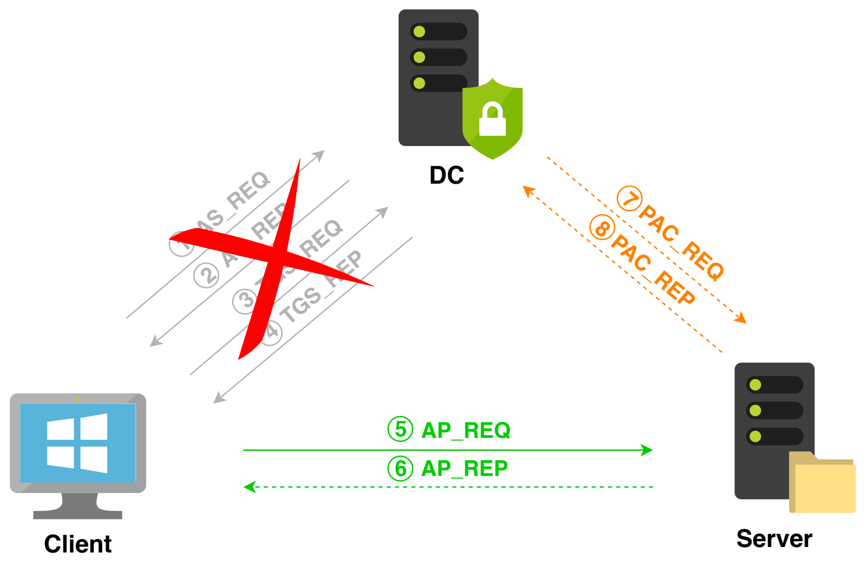


Figure 2.4: Silver Ticket communication

Vulnerability *MS14-068* that occurred in several versions of Windows Server OSs introduced yet another way how forged Kerberos tickets can be created. In this vulnerability, KDCs on domain controllers failed to properly validate signatures in PAC included in Kerberos ticket requests. This allowed a domain user to forge the information contained in the PAC to request higher user privileges than should be allowed. Fortunately, Microsoft released a patch for this issue. [9]

Chapter 3

Windows Security auditing

This chapter reviews options of Microsoft Windows and Active Directory auditing and possibilities of its usage for detection of attacks introduced in Chapter 2. Additionally, it summarizes a typical security monitoring scenario used by organizations, audit policy recommendations and use of other supporting tools for security auditing.

3.1 Windows Event Log

Microsoft Windows OS provides a standard, centralized solution to record important software and hardware events. The event logging service records events from various sources and stores them in a single collection called *Windows Event Log*. The Event Log serves for applications as well as the operating system itself. The Windows Event Log Application Programming Interface (API) provides functions that can be used by developers to write an event provider to record events or an event consumer to examine already logged events [41]. Alternatively, custom logs can be written to the Event Log using `Write-EventLog` PowerShell cmdlet [42].

Windows provides two standard utilities to view and manage Event Log: a command-line tool called *wevtutil* and a Microsoft Management Console (MMC) snap-in named *Event Viewer*. The Event Viewer offers a graphical user interface (GUI) which allows viewing, filtering, searching and managing logs easily. [43]

Microsoft has introduced various changes and improvements to the logging infrastructure since Windows Server 2008 and Windows Vista. Multiple categories of logs were added, and additionally, information about each event conforms to an XML schema. Processing of XML logs by third-party applications is much easier due to their predefined format. [44]

Based on Microsoft documentation [45], there are two main categories of event logs: *Windows Logs* and *Applications and Services Logs*. The Windows Logs category includes subcategories that are intended to store events from legacy applications and events that apply to the entire system. Applications and Services Logs category stores events from a single application or component rather than events that might have a systemwide impact.

Logs in this category are further divided into four subtypes. Windows Event Log is structured as follows:

- Windows Logs
  - Application log
  - Security log
  - Setup log
  - System log
  - ForwardedEvents log
- Application and Services Logs
  - Admin
  - Operational
  - Analytic
  - Debug

Certainly, the most important category for security monitoring is the *Security log*. Events under this category are controlled by security auditing settings. There are two approaches [46] available for setting security auditing:

Basic security audit policies under *Security Settings\Local Policies\Audit Policy*

Advanced security audit policies under *Security Settings\Advanced Audit Policy Configuration\System Audit Policies*

In general, these two audit policy settings are not compatible, and there are several differences between them [44]. To summarize, *Advanced security audit policies* allow more granular setting of the number and types of events to audit, and can be edited through Group Policy. *Basic security audit policies* offer nine basic settings compatible with older versions of Windows. Use of both policies at the same time may produce unexpected results. Following are the categories of *Advanced security audit policies*:

- Account Logon
- Account Management
- Detailed Tracking
- DS Access
- Logon/Logoff
- Object Access

- Policy Change
- Privilege Use
- System
- Global Object Access Auditing

The categories are divided into more subcategories. Each subcategory typically allows setting auditing separately for success and failure. Enabling the auditing of a subcategory results in generating one or more events in the Security log. Some categories log both success and failure into one event, while another generate separated events to signalize success or failure.

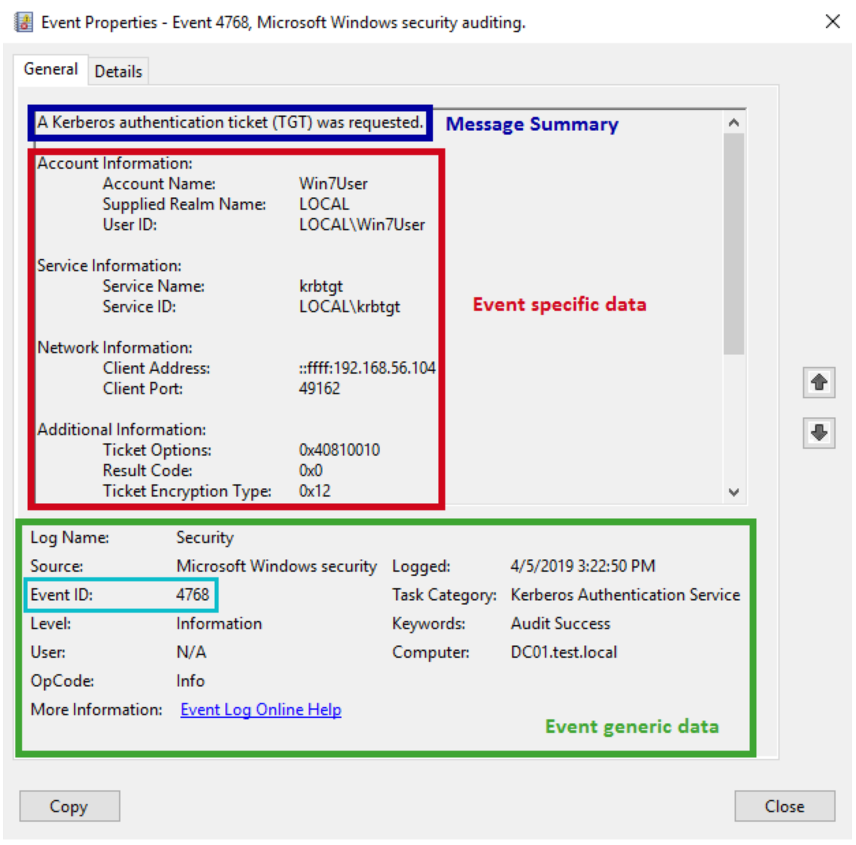
The events can be distinguished by *Event ID*, which is a unique number identifying the event, and *Message Summary*, which provides brief information about what happened. The event then contains other details about users, computers or processes involved, depending on its type. An example of an event viewed by Event Viewer is visible in Figure 3.1. The list of Event IDs generated by each subcategory of audit policies can be found in the summarizing document [47] and detailed event description in *Advanced security audit policy settings* documentation [48].

Several categories provided by the security audit policies represent an essential source of information for hunting attacks towards Active Directory. For instance, the categories *Account Logon* and *Logon/Logoff* track authentication and use of credentials, which is a core element of the attacks. Categories *Account Management* and *DS Access* record changes and replication of the AD schema. Other categories, such as *Detailed Tracking*, *Object Tracking*, and *Privilege Use* provide useful information that may be related to attack preparation, use of hacking tools, or resource access after the successful attack execution.

However, the *Security log* is not the only category of Windows Event Log that contains relevant data. The *System log* contains events logged by Windows system components. System components monitor their functions and may record information interesting for security monitoring. An example of such system component can be *Service Control Manager* [49] logging information about services.

The same principle applies to applications. An application can log its data into *Application log* subcategory of *Windows Logs*, or its own subcategory under *Application and Services Logs*. An application can also record relevant data for security monitoring, or it can even be misused by an adversary to perform a malicious activity. An excellent example is PowerShell, a Windows built-in tool widely used by attackers. Once configured, its logs can be found under *Application and Services Logs*, in *Microsoft-Windows-PowerShell/Operational* subcategory [50].

PowerShell logging features evolved together with PowerShell versions. *PowerShell Module Logging* records usage of all modules and is available since PowerShell 3.0. PowerShell 5.0 introduced S*cript Block Logging*, a feature that records compiled blocks of scripts into Event Log. If a script creates another script block, the resulting script block is logged as well [50]. Additionally, PowerShell can be configured to save full output into transcript



A Kerberos authentication ticket (TGT) was requested.

```
Account Information:
        Account Name:           Win7User
        Supplied Realm Name:    LOCAL
        User ID:                LOCAL\Win7User

Service Information:
        Service Name:           krbtgt
        Service ID:             LOCAL\krbtgt

Network Information:
        Client Address:         ::ffff:192.168.56.104
        Client Port:            49162

Additional Information:
        Ticket Options:         0x40810010
        Result Code:            0x0
        Ticket Encryption Type: 0x12
```

```
Log Name:          Security
Source:            Microsoft Windows security   Logged:         4/5/2019 3:22:50 PM
Event ID:          4768                         Task Category:  Kerberos Authentication Service
Level:             Information                  Keywords:       Audit Success
User:              N/A                          Computer:       DC01.test.local
OpCode:            Info
More Information:  Event Log Online Help
```

Figure 3.1: An example of a Windows Event Log event

files. Table 3.1 shows default PowerShell versions in Windows OSs. An important remark is that PowerShell 2.0 is usually present alongside with the newer versions. Its usage is preferred by adversaries, as it offers a way to bypass the advanced logging features of newer versions. [51]

PowerShell Version	Server OS	Desktop OS
2.0	Windows 2008 R2	Windows 7
3.0	Windows 2012	Windows 8
4.0	Windows 2012 R2	Windows 8.1
5.0/5.1	Windows 2016	Windows 10

Table 3.1: Default PowerShell versions in Windows OSs

3.2 Active Directory

Most system and service-related events relevant to AD infrastructure activities are covered by categories of *Advanced security audit policies*. For example, the category *DS Access* audits events related to AD objects access and modifications; *Account Management* tracks users and groups management activities; and *Logon/Logoff* together with *Account Logon* category record use of credentials.

Apart from the events under the *Windows Logs* category, AD DS related events can be found under the following logs [2] located under the *Applications and Services Logs* category:

- Active Directory Web Services
- DFS Replication
- Directory Service
- DNS Server
- File Replication Service

Another approach is to track changes in the entire AD schema. Active Directory offers several *Change Tracking Techniques.* A client application can register with a DC to receive change notifications, or it can poll for changes using a directory synchronization (*DirSync*) search or *USNChanged* attribute technique. [52]

3.3 Sysmon

Sysmon (System Monitor) is a part of Sysinternals Suite - a free set of tools and advanced system utilities, formerly known as Winternals. The company developing these tools was acquired by Microsoft in 2006, and its co-founder, Mark Russinovich, contributes to the project up to these days [32, 53].

Sysmon is a system service and device driver that, once installed on a system, remains resident across system reboots to monitor and log system activity to the Windows Event Log. It provides detailed information about process creations, network connections, or file creations [54]. Although Sysmon is not a native Windows tool, it offers strong logging capabilities and seamless integration to the Windows Event Log environment.

Sysmon logs its events into *Microsoft-Windows-Sysmon/Operational* subcategory of *Application and Services Logs.* It offers 21 events monitoring processes, drivers, files, Registry, or WMI. Some Sysmon events provide more detailed information than the Windows Security log, which is why Sysmon is used for log enrichment.

Sysmon allows enabling/disabling of particular events or filtering of generated events. These options can be defined by using command line switches or by an XML configuration file during installation. Event description and configuration entries are described in the official Sysmon documentation [54].

Sysmon could be thought of as a free extension of Windows Security log, whose deployment may enhance monitoring and detection capabilities of the environment. Especially, considering its advantages like event filtering and easy integration to Windows Event Log. However, Sysmon needs to be configured precisely to keep the volume of generated events in a manageable amount.

3.4 Typical auditing scenario

Together with other log sources, Windows logs are included in underlying log management infrastructure implemented in an organization. According to [55], a log management infrastructure typically comprises the following tiers:

- Log Generation
- Log Filtering and Normalization
- Log Collection
- Log Storage
- Log Analysis

Logs are normally decentralized, as network devices and systems that generate the log data stores it locally. Even security logs between domain controllers are not replicated. This implies the need for a centralized solution - servers that would receive and store log data from assets in the environment.

The next steps after storing logs include log normalization and event aggregation. Log data is converted to a particular consistent data representation, and similar entries are consolidated into a single entry.

Log analysis includes event correlation, log viewing, and reporting. Event correlation is finding relationships between two or more log entries, commonly done by rule-based correlation, using statistical methods or visualization tools. [56]

Organizations typically solve the problem of log management by using *Security Information and Event Management (SIEM)* software. SIEM products comprise multiple log management phases, including log collection, normalization, and correlation of events from various log sources. SIEM tools also contain features for security monitoring, such as alerting, incident management or a security knowledge base, all within a GUI environment. Alerts and incidents are then analyzed by security analysts in *Security Operations Centers (SOCs).* [56]

There are plenty of SIEM tools available on the market. They differ in features offered, complexity, and of course, pricing. According to multiple sources [57, 58, 59] some of the most popular SIEM tools are:

- Splunk Enterprise Security
- Micro Focus ArcSight ESM
- IBM QRadar
- SolarWinds Log & Event Manager
- McAfee Enterprise Security Manager

The pricing model of SIEM tools is typically based on data volume ingested per a time unit and/or the number of connected assets [60]. This fact introduces a problem for many organizations, as ingesting all logs from all assets would produce a large volume of events resulting in not affordable licensing costs. The large volume of logs poses also problem of infrastructure requirements, like performance, network bandwidth, storage capacity and others.

To imagine volume of generated data, the following approximate computation can be assumed. Taken that a DC can produce for instance 35 events per second [61], and each event has size of about 1kB[7], the result is nearly 3GB of data per day per a single DC. The estimation is not counting other servers, workstations and network devices, which can be even more "chatty".

The problem can be solved either by reducing the amount of logged data per host, excluding non-critical assets from security monitoring, or a reasonable compromise between these two. The question is, how to accomplish the desired balance, together with solid detection capabilities. As every environment is different, there cannot be a general solution applicable to all of them. Multiple factors, such as environment size, network bandwidth, types of assets and other infrastructure characteristics must be taken into account.

However, some recommendations for developing audit policy exist. Microsoft provides guidance for audit policy settings of its workstation and server products. Tables at [62] contain Windows default setting, the baseline recommendations, and the stronger recommendations for security auditing. As mentioned in Section 3.1, settings of *Advanced security audit policies* can be pushed through Group Policy to all Windows assets in the domain and allow creating a different set of settings for different groups of computers.

When it comes to limiting the number of monitored hosts, users' workstations are usually at the bottom of the list of asset monitoring priority. Workstations are typically the most numerous group of assets in organizations, which reflects significantly into licensing costs. Monitoring of workstations, especially laptops, introduces other challenges, as they are not always connected to the corporate network and there is a necessity to manage software for log collection and forwarding.

Considering all these aspects and the fact that sensitive data is stored mostly on domain controllers and member servers, workstation monitoring often plays a minor role. But still, even partial monitoring of workstations may significantly improve detection capabilities [63].

For the purposes of log management, and especially log analysis in this thesis, I have chosen to use Splunk tool. This decision was supported by Splunk licensing policy and manageable configuration, feasible enough to deploy Splunk in a lab environment. Use of Splunk in this thesis is described in the following chapter.

[7]The average size of an event can be computed directly by a Splunk search, using count of characters in raw events received from a domain controller.

Chapter 4

Splunk and SPL

Splunk is a platform for analyzing machine data. It is built on capabilities of *Search Processing Language (SPL)*, which allows to search, correlate, analyze and visualize machine-generated data. Logs of any kind, most importantly the security logs, are a typical example of data generated by machines in massive amounts. This data often has a complicated structure, different from one system to another. Considering these properties, the use of specialized tools for log management is a necessity. Splunk represents an example of such tools, providing great support for the security field.

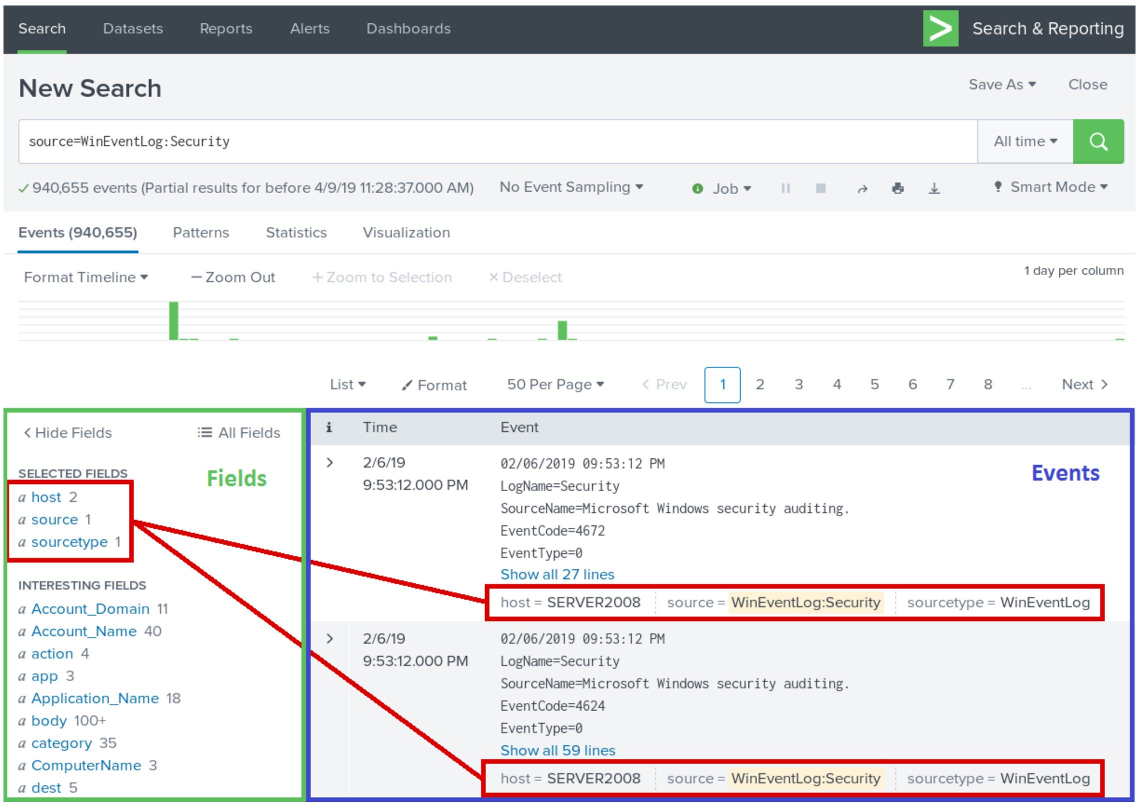
4.1 Splunk Enterprise

4.1.1 Data processing

Splunk can read machine data from any number of sources. The most common input sources are files, network, or scripted inputs. Splunk can process only textual data. Raw machine data is divided into discrete pieces of information known as *events*. Each event consists of separate parts of data called *fields*. The fields are represented as keyword/value pairs. There are four default fields always indexed along with the raw data: `source`, `sourcetype`, `host` and `_time`. The first three fields define the type of data and its origin. The fourth one, `_time` field is used to order events. [64]

Figure 4.1 shows the concept of Splunk search: events that represent result of a particular search and fields parsed from these events.

Machine data almost always contain some indication of when it was created or when an event described by the data occurred. Splunk's indexes are optimized to retrieve events in time-series order. If the raw data does not have an explicit timestamp, Splunk assigns the time at which the event was indexed. The time aspect is essential for security analysis where analysts need to investigate the actions of an attacker in the exact order they occurred. [64]



```
source=WinEventLog:Security
```

✓ 940,655 events (Partial results for before 4/9/19 11:28:37.000 AM)

SELECTED FIELDS
a host 2
a source 1
a sourcetype 1

INTERESTING FIELDS
a Account_Domain 11
a Account_Name 40
a action 4
a app 3
a Application_Name 18
a body 100+
a category 35
a ComputerName 3
a dest 5

```
2/6/19 9:53:12.000 PM
02/06/2019 09:53:12 PM
LogName=Security
SourceName=Microsoft Windows security auditing.
EventCode=4672
EventType=0
host = SERVER2008   source = WinEventLog:Security   sourcetype = WinEventLog

2/6/19 9:53:12.000 PM
02/06/2019 09:53:12 PM
LogName=Security
SourceName=Microsoft Windows security auditing.
EventCode=4624
EventType=0
host = SERVER2008   source = WinEventLog:Security   sourcetype = WinEventLog
```

Figure 4.1: Splunk search

4.1.2 Splunk architecture

Splunk can be used as a *single instance* or as a *distributed* deployment. The latter would be a typical scenario for security usage in organizations, where data needs to be collected from multiple assets and sent to a centralized solution. SOC analysts need to make correlations and search the data, typically in different time and a different physical location.

Splunk Enterprise consists of specialized instances, known as components, that focus on specific functions. Components that handle data are called *processing components* [65]. There are three main types of processing components:

forwarders ingest data, perform minimal processing and send it to an indexer;

indexers index, store data and search data in response to requests from the search head;

search heads interact with a user, handle requests and distribute them across the set of indexers.

The platform can be scaled by adding more components of each type, depending on the needs of the environment. In small environments, indexer and search head can be combined into one instance. Forwarders need to be deployed on every monitored asset to collect logs and send them to an indexer. There are three types of forwarders; for most use cases the *Splunk Universal Forwarder* is the best and utterly sufficient option [66]. The Figure 4.2 illustrates roles of the processing components in Splunk architecture. [67]

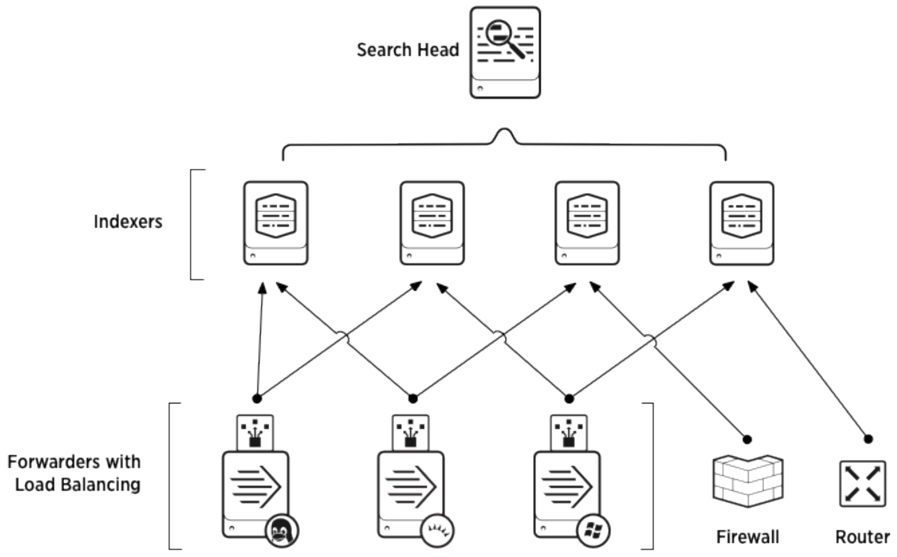


Figure 4.2: Splunk architecture

An indexer receives the data and transforms it into events, which it stores in indexes. The index is the repository of Splunk data. To which index particular data is saved, can be defined in the configuration. This can be decided on various criteria, commonly by the source of incoming data or log type. The index can be filtered during a search, narrowing the number of events that need to be searched and thus making the search faster. [64]

A user can access Splunk by logging on to a web client running on the search head. The client presents a GUI where the user can access Apps and manage settings. The most fundamental App is Search (shown in Figure 4.1), which allows to invoke searches.

Splunk functionality can be extended by *Apps* or *Add-ons*. Those can be either developed by Splunk team, third-parties, or anyone else, allowing companies to customize the platform for their particular needs. There is a slight difference between an App and an Add-on. Apps generally offer extensive user interfaces and features, while Add-ons generally enable the Splunk platform or another App to ingest or map a particular type of data. As the differences between App and Add-on are important mostly for development, I use these two terms interchangeably in this thesis. [68]

Examples of the above two focusing on security are:

Splunk Enterprise Security (ES) a premium App containing whole security ecosystem, with features for security monitoring, incident response and security intelligence, turning Splunk into a SIEM tool [69];

Splunk Add-on for Microsoft Windows a *Technology Add-on (TA)* containing predefined inputs configuration to collect data (logs) from Windows systems and ensure its correct parsing into fields [70].

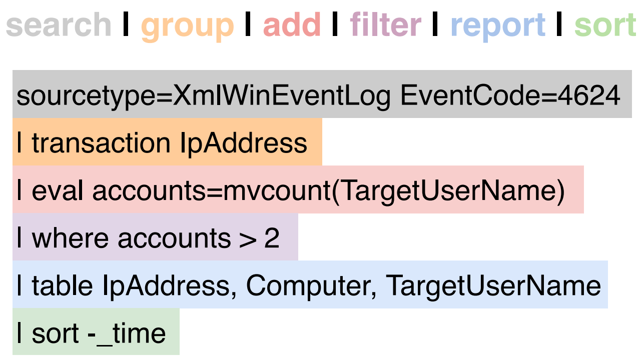
4.1.3 Splunk licensing

Splunk platform is offered in several editions. They differ in the amount of data that can be indexed, maximal number of users and several other parameters, not excluding pricing [71]. Among the editions, there are two interesting ones: *Splunk Free* and *Splunk Enterprise*. Splunk Free contains full SPL support, allows to index 500MB of data per day, the maximum of one user and can be used free of charge. Splunk Enterprise has more features and can be used by more users. However, it is billed depending on the amount of ingested data. A free developer license can be requested for six months, allowing to index 10GB of data per day with all enterprise features [72].

4.2 Search Processing Language

This section serves as a brief introduction to SPL. It does not aim to provide a comprehensive reference of SPL and its capabilities. The searches in later sections of this thesis are not explained command by command. For complete documentation, there is an official reference with examples available at Splunk Docs [73].

Search Processing Language is a language designed by Splunk for use with Splunk software. It encompasses all the search commands and their functions, arguments, and clauses. SPL contains more than 140 commands. Its syntax was originally based on the Unix pipeline and Standard Query Language (SQL). SPL uses pipes in a similar way like Bash: they separate commands, and results from each command are passed as input to the next command. The first keyword after a pipe is the name of a search command. The structure of a Splunk search with pipes is illustrated in Figure 4.3. [64]



```
sourcetype=XmlWinEventLog EventCode=4624
| transaction IpAddress
| eval accounts=mvcount(TargetUserName)
| where accounts > 2
| table IpAddress, Computer, TargetUserName
| sort -_time
```

Figure 4.3: Basic structure of SPL

The SPL commands can be divided into several categories based on their functionality. Table 4.1 shows examples of most common commands in each category.

Category	Description	Commands
Sorting results	Ordering results and (optionally) limiting the number of results.	`sort`

Filtering results	Taking a set of events or results and filtering them into a smaller set of results.	`search`
		`where`
		`dedup`
		`head`/`tail`

Grouping results	Grouping events to see patterns.	`transaction`

Reporting results	Taking search results and generating a summary for reporting.	`top`/`rare`
		`stats`
		`xyseries`
		`table`

Filtering, modifying and adding fields	Filtering out (removing) some fields, modifying, or adding fields to enrich results.	`fields`
		`replace`
		`eval`
		`rex`
		`lookup`

Table 4.1: Common SPL commands

The `search` command is one of the simplest and most powerful commands. It's such a basic command that it is not even necessary to type it anywhere before the first pipe, because it is invoked implicitly at the head of a search. The `search` command supports the use of keywords, phrases, fields, boolean expressions, and comparison expressions to retrieve events from an index. Important remarks are that there is always implied `AND` between terms and that field names are case-sensitive, while their values are not. The `search` command supports also use of wildcards. [64]

While working with logs, it is often necessary to correlate multiple entries containing related information. For this purpose, Splunk provides an useful command `transaction`, which groups subsequent events that meet various constraints. The time element for a transaction can be specified by using the `maxspan` parameter, defining that events must span less than the specified maximal length of time, or alternatively, with the `maxpause` parameter, that specifies the maximum duration of pause between events in the transaction. The command adds a new field `eventcount` to the raw events, allowing to further filter results on number of events in the transaction. [73]

Listing 4.1 shows an example search snippet that creates groups of subsequent events containing the same value in the `IpAddress` field, with the maximum of five minutes between them and with unlimited maximal number of events in each transaction. Results are filtered only to transactions containing more than five events for more than two different target user accounts.

```
...
| transaction IpAddress maxpause=5m maxevents=-1
| where eventcount > 5
| eval accounts=mvcount(TargetUserName)
| where accounts > 2
```

Listing 4.1: Grouping events into transactions

4.3 Splunk ES Content Update

Splunk ES Content Update (ESCU) is Add-on built by Splunk Security Research Team. It is released as a subscription service which delivers pre-packaged security content for use with Splunk ES App, mentioned in Section 4.1. Subscribers get regular updates which help security practitioners more quickly address ongoing and time-sensitive problems and threats. Unlike ES, the ESCU Add-on can be used free of charge. [74]

The security content in ESCU is delivered in the form of *Analytic Stories.* These Analytic Stories advise how to detect, investigate and take action on new threats. Each story also provides an explanation of what risk the threat represents and how to implement the story using the contained searches into existing Splunk ES environment.

Analytic Story in the ESCU starts with a brief *Description*, followed by a short *Narrative.* Purpose of these two sections is to introduce the security scenario, describe it shortly and explain the threats arising from it. Then the list of all searches belonging to the story follows. There are four types of searches that ESCU defines:

Detection detect malicious activities and threats in the environment,

Context reveal more context for Detection searches,

Investigative help security analysts to investigate the detected threat,

Support supply more information or add features to other searches.

After specifying a time range for the search, the search can be invoked by clicking a button. Results are displayed within the Add-on right in the context of the Analytic Story. Every search in the ESCU framework contains additional information encapsulated in these sections:

Description brief information about what the search finds;

Detailed Description detailed description of commands and logic used in the search;

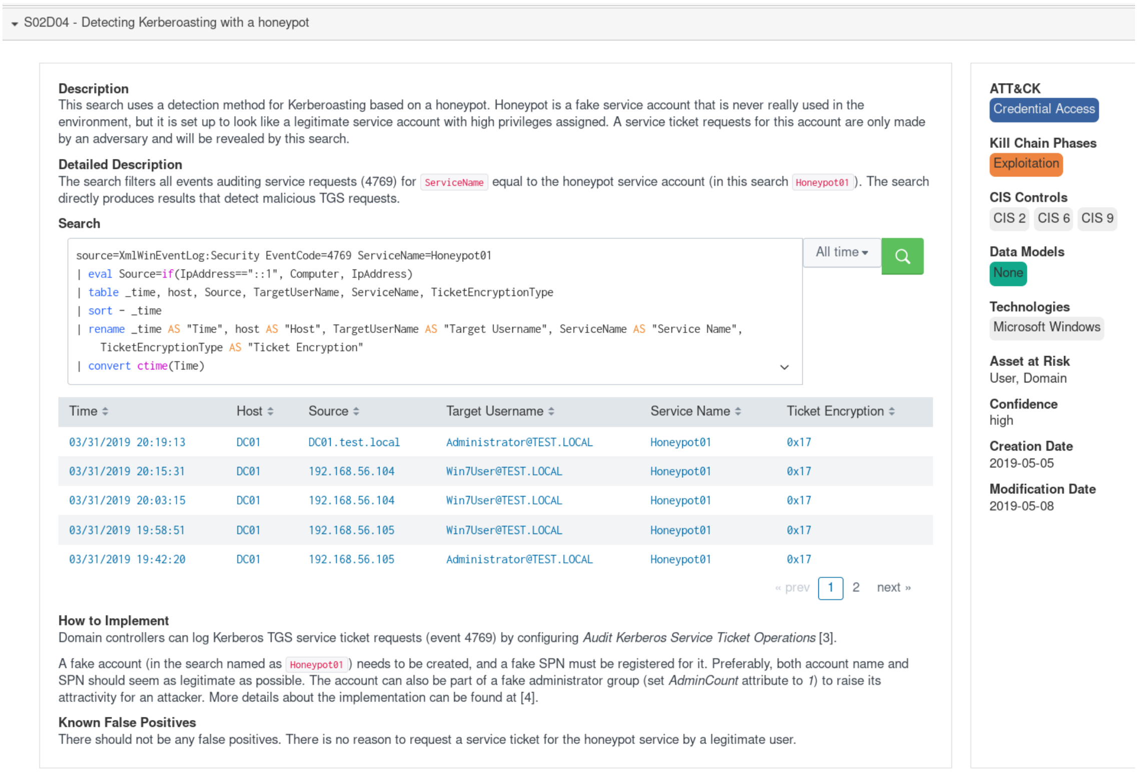
Search the search which can be invoked itself;

How to Implement advice on necessary configuration steps or parts of the search that needs to be tuned during its implementation;

Known False Positives information about possible false positive results that the search may yield, and eventually advice how to prevent them.

The ESCU framework focuses greatly on several security categorizations introduced in Section 2.1. Every Analytic Story and Detection search is classified with the respective *Tactics* of *Mitre ATT&CK* framework and phase of *Kill Chain*. Additionally, suitable actions of *CIS Controls* are assigned [75].

Every Detection search has marked related technologies that provide log inputs, assets which are at risk by the threat and a confidence level. The confidence level ranks *low*, *medium* or *high*, depending on the ratio of signal (valid/*true positive*) to noise (invalid/*false positive*) for every Detection search. Figure 4.4 illustrates the placement of a Detection search in the context of an Analytic Story in the ESCU Add-on.


▾ S02D04 - Detecting Kerberoasting with a honeypot

Description
This search uses a detection method for Kerberoasting based on a honeypot. Honeypot is a fake service account that is never really used in the environment, but it is set up to look like a legitimate service account with high privileges assigned. A service ticket requests for this account are only made by an adversary and will be revealed by this search.

Detailed Description
The search filters all events auditing service requests (4769) for `ServiceName` equal to the honeypot service account (in this search `Honeypot01`). The search directly produces results that detect malicious TGS requests.

Search

```
source=XmlWinEventLog:Security EventCode=4769 ServiceName=Honeypot01
| eval Source=if(IpAddress=="::1", Computer, IpAddress)
| table _time, host, Source, TargetUserName, ServiceName, TicketEncryptionType
| sort - _time
| rename _time AS "Time", host AS "Host", TargetUserName AS "Target Username", ServiceName AS "Service Name",
    TicketEncryptionType AS "Ticket Encryption"
| convert ctime(Time)
```

Time	Host	Source	Target Username	Service Name	Ticket Encryption
03/31/2019 20:19:13	DC01	DC01.test.local	Administrator@TEST.LOCAL	Honeypot01	0x17
03/31/2019 20:15:31	DC01	192.168.56.104	Win7User@TEST.LOCAL	Honeypot01	0x17
03/31/2019 20:03:15	DC01	192.168.56.104	Win7User@TEST.LOCAL	Honeypot01	0x17
03/31/2019 19:58:51	DC01	192.168.56.105	Win7User@TEST.LOCAL	Honeypot01	0x17
03/31/2019 19:42:20	DC01	192.168.56.105	Administrator@TEST.LOCAL	Honeypot01	0x17

How to Implement
Domain controllers can log Kerberos TGS service ticket requests (event 4769) by configuring *Audit Kerberos Service Ticket Operations* [3].

A fake account (in the search named as `Honeypot01`) needs to be created, and a fake SPN must be registered for it. Preferably, both account name and SPN should seem as legitimate as possible. The account can also be part of a fake administrator group (set *AdminCount* attribute to *1*) to raise its attractivity for an attacker. More details about the implementation can be found at [4].

Known False Positives
There should not be any false positives. There is no reason to request a service ticket for the honeypot service by a legitimate user.

ATT&CK
Credential Access

Kill Chain Phases
Exploitation

CIS Controls
CIS 2 CIS 6 CIS 9

Data Models
None

Technologies
Microsoft Windows

Asset at Risk
User, Domain

Confidence
high

Creation Date
2019-05-05

Modification Date
2019-05-08

Figure 4.4: Detection search in the ESCU Add-on

4.4 Splunk in this thesis

For purposes of design and implementation of detection rules in this thesis, I used the ESCU framework together with Splunk Enterprise, for which I have requested developer license (mentioned also in Section 4.1.3) [72]. However, all the searches in the thesis rely only on features of pure SPL and thus can also be used in Splunk Free. Splunk Enterprise Security is a premium licensed App for which any kind of developer license does not exist, so its features couldn't be used in this thesis.

In the documentation of ESCU, it is stated that it requires ES version 4.5 for operation. Still, the ESCU framework with a subset of features can also be used in Splunk Enterprise without the ES App installed. I have taken the ESCU Add-on as a framework defining the format of Analytic Stories, types of searches and security mappings. Advanced features, such as scheduling of searches, alerting, risk modifiers, binding to data models or Common Information Model (CIM) are not used in this thesis. Some of the searches, which are a part of the ESCU original content, cannot be invoked in Splunk Enterprise, since they rely on macros or data models defined by Splunk ES App.

The searches provided in this thesis would have to be designed a bit differently to be used with these advanced features. However, I do not see this fact as a disadvantage. Searches written using only basic SPL commands are easier to read for users with no, or little experience with Splunk. Also, it allows less effort to re-implement the detection principles in another search language, system, or SIEM tool.

Correlation rules are written as searches when using Splunk's SPL. Actually, everything in Splunk either is, or is somehow related to a search. I use terms *rule* and *detection search* interchangeably in this thesis, as these two mean effectively the same in the implementation.

Chapter **5**

Implementation

This chapter encloses the practical part of this thesis. It describes the actual process of designing the rules for detection of attacks introduced in Chapter 2, and their implementation as Splunk searches. The searches are set in the form of Analytic Stories and extend the existing content of the Splunk ES Content Update Add-on.

5.1 Lab environment

For implementation, I have built and configured a virtual lab environment to simulate an example of a small domain. The rules were created and tested on the data coming from this lab environment. This section describes the software versions and configuration used in the lab.

5.1.1 Lab description

The lab consists of one physical machine and five virtual machines (VMs). The host computer runs Linux OS, and is network-connected with the VMs to receive logs. The VMs include two servers, one domain controller (*DC01*) and one member server (*SERVER2008*), and two users' workstations (*WINDOWS7* and *WINDOWS10*), all of them running different versions of Microsoft Windows. This is to reflect the situation in real environments, as it is rare to have the same OS version on all computers in the domain. The last VM (*kali*) runs Kali Linux distribution and serves as a simulation of an external attacker having network connectivity to the domain. Table 5.1 shows basic information about all VMs in the environment and Figure 5.1 visualizes it graphically. Hostnames and IP addresses from the Table 5.1 are visible in detection examples in the following sections.

Logs from all monitored assets are sent to the physical machine where they are indexed by a Splunk Enterprise instance. As discussed in Section 3.4, it is essential to determine which assets to monitor. As very few companies can afford to monitor all computers in their network, I decided to follow the typical auditing scenario described in Section 3.4. It means that only logs from domain controllers (*DC01*) and critical servers (*SERVER2008*)

Hostname	IP address	OS	Version
DC01	192.168.56.102	Windows Server 2016 x64	1607 build 14393.2828
WINDOWS10	192.168.56.103	Windows 10 Enterprise x64	1809 build 17763.292
WINDOWS7	192.168.56.104	Windows 7 Enterprise x86	6.1 build 7601, SP1
SERVER2008	192.168.56.106	Windows Server 2008 R2 x64	6.1 build 7601, SP1
kali	192.168.56.105	Kali Linux x64	4.18.0-kali2-amd64

Table 5.1: Assets in the lab environment

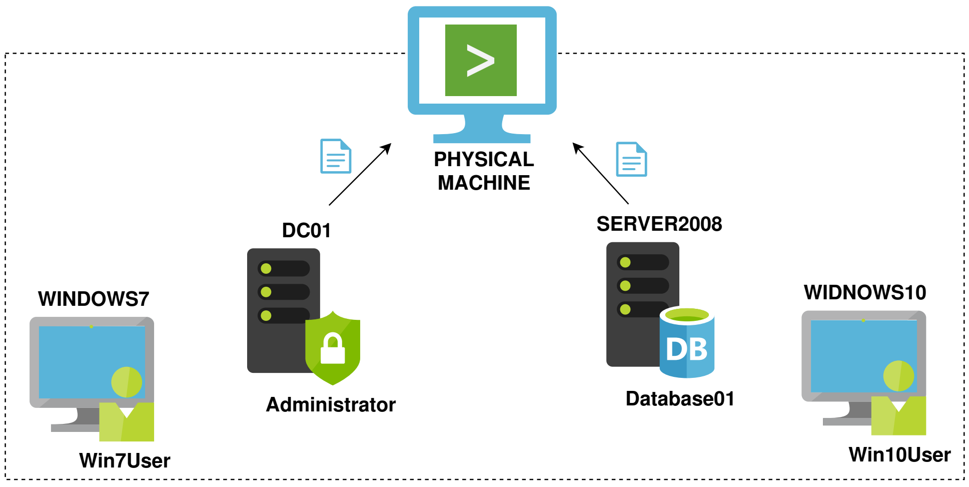


Figure 5.1: Assets in the lab environment

are sent to Splunk, and not all possible auditing settings are enabled. Configuration of auditing is described later in Section 5.1.3.

5.1.2 Splunk configuration

Splunk Enterprise instance is running on the host machine on a Linux OS. Splunk exists in the lab environment as a *single instance* deployment. As discussed in Section 4.1, this means that indexer and search head are merged, running on the host machine. On all monitored assets, there is *Splunk Universal Forwarder* [66] deployed. The forwarder ingests logs on a remote system according to its settings and forwards them to the indexer.

It is necessary to install and configure all additional Apps and Add-ons on the search head. This includes the ESCU Add-on, which must be installed, and configured to have permissions to all indexes that should be accessible for searching. But ESCU is not the only extension that is required. To parse logs coming from different systems, a so-called TAs need to be installed. These Add-ons provide the necessary field extractions, which would have to be specified manually otherwise. Since the designed rules rely on logs from Microsoft Windows, Active Directory and Sysmon, the following TAs are used:

- Splunk Add-on for Microsoft Windows [70]
- Splunk Add-on for Microsoft Active Directory [76]
- Add-on for Microsoft Sysmon [77]

Instances of Splunk Universal Forwarder on the monitored servers are configured to monitor these categories of events:

- WinEventLog://Security
- WinEventLog://Application
- WinEventLog://System
- WinEventLog://Microsoft-Windows-PowerShell/Operational
- WinEventLog://Microsoft-Windows-Sysmon/Operational

Additionally, monitoring of Active Directory is enabled on the domain controller. All the events are retrieved in XML format, from which the TA is able to parse fields more accurately. The purpose of monitoring all the mentioned event categories is described in the following sections.

Splunk Universal Forwarder instances are run under a domain account *Splunk*, created specifically for this purpose. The forwarder has all the permissions of this domain account and can collect all the data that the account has access to read. This setup is required for accessing the AD schema [78].

Table 5.2 summarizes all Splunk components used in the environment together with their versions.

Splunk component	**Version**
Splunk Enterprise	7.2.6
Splunk ES Content Update	1.0.29
Splunk Add-on for Microsoft Windows	5.0.1
Splunk Add-on for Microsoft Active Directory	1.0.0
Microsoft Sysmon Add-on	8.0.0
Splunk Universal Forwarder	7.1.3

Table 5.2: Versions of Splunk components

5.1.3 Windows configuration

All assets running Microsoft Windows are part of a Windows domain. The domain is the root domain of its own domain tree, within a single-forest AD environment. The FQDN is *test.local* and the NetBIOS domain name is set to *LOCAL*. AD DS are installed on *DC01*, which is the only DC in this lab environment. The *Domain Functional Level* is set to the level of Windows Server 2016. The DC is the global catalog server and holds all FSMO roles specified in Section 1.1.3. Domains with a single DC are very rare in reality, but I have chosen this setup due to its simplicity and performance limitations of the virtual environment.

The auditing settings are set up using the *Advanced security audit policies* and pushed to all monitored assets via Group Policy. Recommendations from Microsoft [62], especially the column *Stronger Recommendation*, is taken as a baseline for enabling or disabling the distinct audit options. In case the designed rules require logging configuration beyond the recommendations, it is mentioned in the search description. Export of the *Advanced Audit Policy* configuration in a CSV file can be found in the attachment on the enclosed CD.

Several designed Detection searches require PowerShell logging. *Module Logging* and PowerShell *Script Block Logging* are enabled via Group Policy settings as described in [50]. This requires monitoring *Microsoft-Windows-PowerShell/Operational* category of *Application and Services Logs*.

The servers are also monitored by Sysmon. This implies the need for monitoring yet another auditing category: *Microsoft-Windows-Sysmon/Operational*. Sysmon holds its configuration in an XML file that defines which events are generated. The listing of the used configuration file can be found in the attachment on the enclosed CD.

The default *Administrator* account is used as a single administrator account, being a member of *Administrators*, *Domain Admins*, *Enterprise Admins*, *Schema Admins*, and other built-in high-privileged groups. This setup does not reflect reality, as administrator rights should be divided into more accounts in accordance with the *Least-Privilege Administrative Models* [5]. However, these principles are often not followed precisely in real configurations [79].

Several other user accounts that represent different roles were created in the environment. All these accounts are members of the default *Domain Users* group. Table 5.3 lists all created domain accounts with its intended roles.

User account	Role in the environment
Administrator	Built-in account for administering the computers/domain
Database01	Service account used by a database service
Honeypot01	Account created as a honeypot with no real usage
Splunk	Account used by Splunk Universal Forwarder instances
Win7User	User account used to log on to WINDOWS7
Win10User	User account used to log on to WINDOWS10

Table 5.3: Accounts used in the environment

5.2 Workflow

5.2.1 Analytic Stories

As stated in Section 4.4, the ESCU was taken as a definition of the form in which to compose and design the searches. I have divided the designed searches into four Analytic Stories. Every Analytic Story focuses on a certain attack scenario, and the Stories are following the order of steps that would be potentially made by an attacker. All Analytic Stories belong to the category called *Active Directory*. The created Analytic Stories are:

- 01 - Brute Force Attacks
- 02 - Kerberoasting
- 03 - Credential Dumping
- 04 - Credential Theft

All the Stories and searches created are numbered to simplify orientation and references. Also, the custom numbering allows differentiating between the original security content of the ESCU Add-on and the content I developed for purposes of this thesis and added to the Add-on.

Analytic Stories are named starting with a story number *(01-04)*. The searches are named beginning with *SXXYZZ*, where *XX* is a number of the story, *Y* is one of *(D, C, I, S)* and determines the type of the search, and *ZZ* is a number of the search. For example, the name of the first Detection search in the third Analytic Story starts with *S03D01*.

5.2.2 Design of searches

The process of designing the detection rules starts with defining all related log sources that may contain relevant data. For events, it is crucial to identify what information they carry and under which circumstances they are logged, or whether they are generated at all. In many cases, also a trade-off between the added value and the volume of generated events has to be taken into consideration. This is discussed in the section *Information sources* under every Analytic Story.

Microsoft's documentation of *Advanced security audit policy* settings [48] and *Randy Franklin Smith's Log Encyclopedia* [80] are the ultimate reference sources of event descriptions, logging settings, event occurrences, and other information related to Windows Event Log. I used these sources to find information about events.

After a search is designed, it is necessary to test its detection capabilities and evaluate the relevancy of the returned results. Within every Analytic Story, a section named *Testing* describes the process of testing the rules by performing the attacks; the relevancy of the results is then discussed in a section called *False Positives*.

5.3 01 - Brute Force Attacks

The first Analytic Story focuses on brute force attacks described in Section 2.2.1. Although the majority of these attacks is based on an elementary method of trial and error, its reliable detection is not so trivial.

5.3.1 Information sources

Brute force attacks involve attempts to log on by using explicit credentials, that is, a username and a password. This implies that the source of information can be any logs auditing credential validation or account logon. Since there are several authentication protocols implemented in Windows OSs, especially NTLM and Kerberos, and there are several types of logon scenarios (described in Section 1.2), the particular information is logged under different event codes and the auditing is controlled by different settings.

Following categories of *Advanced security audit policies* create events related to credential use in Windows authentication:

- Account Logon
  - Audit Credential Validation
  - Audit Kerberos Authentication Service
  - Audit Other Account Logon Events
- Logon/Logoff
  - Audit Account Lockout
  - Audit Logoff
  - Audit Logon
  - Audit Other Logon/Logoff Events
  - Audit Special Logon

Credential Validation subcategory allows monitoring local accounts authentication attempts and, for domain accounts, NTLM authentication in the domain. The events occur on the computer that is authoritative for the credentials; for domain accounts, the DC is authoritative; for local accounts, the local computer is authoritative. The category is especially useful for monitoring unsuccessful attempts, to find brute force attacks, account enumeration, and potential account compromise. This category should generate events 4774(S, F), 4775(F), 4776(S, F), and 4777(F), however, only events 4776(S, F) appear.[8] [81]

Kerberos Authentication Service auditing setting determines whether to generate audit events for Kerberos authentication TGT requests. This subcategory contains events

[8]The information in brackets determines whether a particular event is logged for *success (S)*, *failure (F)*, or both *(S, F)*.

about issued TGTs and failed TGT requests. It also includes events about failed Pre-Authentications, due to wrong or expired passwords, which can be a sign of an attack attempt when generated in higher amounts. This category generates events 4768(S, F) and 4771(F); event 4772(F) is a defined event, but it is never invoked. This category generates events only on DCs. [82]

Other Account Logon Events subcategory does not contain any events and is intended for future use. There is no reason to enable it on any assets. [83]

Audit Account Lockout setting enables auditing security events that are generated by a failed attempt to log on to an account that is locked out. It logs only a single failure event 4625(F). For the lock-out action itself, the event 4740(S) belonging to the *Audit User Account Management* category is logged. Auditing of these events is vital especially if the *Account Lockout Policy* [19] is set in the domain. [84]

Audit Logoff determines whether the operating system generates audit events when logon sessions are terminated. These events occur on the computer that was accessed. In case of an interactive logon, these events are generated on the computer that the user logged on to. The subcategory logs events 4634(S) and 4647(S); it does not contain any failure event. The difference between these events is, that event 4647(S) is recorded when the logoff procedure was initiated by a specific account using the logoff function, whereas event 4634(S) shows that the session was terminated and no longer exists. Logoff events are not reliable for the detection of potential attacks, because they do not have to be generated every time. [85]

Logon audit events are related to the creation of logon sessions and occur on the computer that was accessed. For an interactive logon, events are generated on the computer that was logged on to; for network logon, such as accessing a share, events are recorded on the computer that hosts the resource that was accessed. Events are recorded for logon success: 4624(S), failure: 4625(F), use of explicit credentials: 4648(S) and SID filtering for specific AD trust: 4675(S). The first three events are of high significance and contain detailed information about logon actions. [86]

Other Logon/Logoff Events subcategory controls auditing of other events, including Remote Desktop session connections, locking and unlocking workstations, invoking or dismissing a screen saver, authentication towards networks, or an event 4649(S) for a possible Kerberos replay attack. These events bring a low additional value for detection but can be used during the investigation. The event 4649(S) does not seem to appear. [87]

Special Logon subcategory allows to audit events generated by special logons such as a logon that has administrator-equivalent privileges: 4672(S), or a logon by a member of a *Special Group*: 4964(S). Members of Special Groups can be defined in the Registry. While monitoring of these events is very important to track the suspicious activity of accounts with sensitive privileges assigned, they are not crucial for detection of brute force attacks. Nevertheless, they represent a fundamental element for the following investigation of attacker's actions. [88]

Table 5.4 summarizes existing events useful for detection of brute force attacks.

Event ID	Type	Message
4624	S	An account was successfully logged on.
4625	F	An account failed to log on.
4740	S	A user account was locked out.
4648	S	A logon was attempted using explicit credentials.
4776	S, F	The domain controller attempted to validate the credentials for an account.
4768	S, F	A Kerberos authentication ticket (TGT) was requested.
4771	F	Kerberos pre-authentication failed

Table 5.4: Events for detection of brute force attacks

5.3.2 Detection searches

5.3.2.1 Logon attempts using a non-existing account

Adversaries may attempt to guess usernames, or use non-existing accounts. Such attempts would generate events signalizing a logon failure with status code meaning that the username specified does not exist. If both username and password are wrong, the events will be logged for a wrong username, as evaluation of the password is not performed in that case. Usage of a script attempting to log on with different usernames will produce an unusually high amount of failed logon attempts in a short time, while all the events will contain the same information about the source computer.

The detection rule can be set to produce results if the number of attempts with wrong usernames from the same host within a short timeframe is higher than a specified threshold. This implies filtering failure events for status codes signalizing use of wrong username and monitoring its count for every source computer.

The question is, how to determine the right threshold values, like the number of events, accounts, or the time span for a reliable detection. A sneaky attacker can put a delay element into the script, lowering the frequency of attempts, and the probability of detection.[9] On the other hand, setting the values too low may bring numerous false positive detections, e.g. of users who mistyped their username accidentally. There is no ultimate answer applicable generally, although some guidelines on how to proceed do exist. This topic is further discussed in Section 5.3.2.2 together with failed password attempts.

The next step is to define source events for the detection. Based on logging options discussed in Section 5.3.1, there are three events available that record failed logons with a non-existing username:

4625 logged on the computer where the logon attempt was made, with value of the field `SubStatus=0xC0000064`;

4678 recorded on a domain controller for Kerberos authentication ticket requests, with `Status=0x6`;

[9]Of course, also lowering the efficiency of the attack, as it would take a very long time to guess successfully by making only 2 attempts per day.

4776 auditing NTLM authentication attempts, with `Status=0xC0000064`.

Since the event 4625 is always logged on the computer where the logon attempt was made, a reliable detection based on occurrence of this event would be possible only within logs from all workstations. Fortunately, the events 4768 and 4776 are logged alongside with event 4625 for a failed logon attempt.

The event 4768 is always generated on a DC. Event 4776 is a little problematic, as it occurs on the computer that is authoritative for the provided credentials. For domain accounts, it is logged on a DC. However, an account that does not exist, is not taken as a domain account. In such a case, NTLM authentication is performed on a local computer, as it was a local account, and the event 4776 is logged alongside with the event 4625 on that local computer.

To summarize, in case of a failed logon of a domain account, the event 4625 is logged on the computer where the logon attempt was made, and either event 4768, or event 4776 is recorded on a DC. With a detection rule built on occurrence of the events 4768 and 4776 it is possible to detect all failed login attempts of domain accounts and failed NTLM authentication attempts on all monitored assets (in this case DCs and critical servers).

Furthermore, there are differences in fields provided by each of these events. Table 5.5 contains the names of fields parsed from the original events, and shows the main incompatibilities between them. Events 4768 log only IP addresses, while events 4776 log only hostnames of source computers. Additionally, both events have different set of error codes.

Information	**4625**	**4776**	**4768**
Source IP	`IpAddress`	N/A	`IpAddress`
Source hostname	`WorkstationName`	`Workstation`	N/A
Target username	`TargetUserName`	`TargetUserName`	`TargetUserName`
Target domain	`TargetDomainName`	N/A	`TargetDomainName`
Error code	`Status` `SubStatus`	`Status`	`Status`

Table 5.5: Differences in field values between the events

Based on differences provided in the Table 5.5, it is not possible to easily group Kerberos and NTLM authentication attempts to a single detection rule. Therefore, I created separate detection rules for Kerberos and NTLM logon attempts. The logic of the rules is identical, differences are present only in evaluating source computer information and types of providing events with their respective error codes. The snippets of these rules are visible in Listing 5.1 and 5.2.

```
source=XmlWinEventLog:Security EventCode=4768 Status=0x6
| transaction IpAddress maxpause=5m maxevents=-1
| where eventcount > 5
| eval Source=if(IpAddress=="::1", Computer, IpAddress)
| eval accounts=mvcount(TargetUserName)
| where accounts > 2
| ...
```

Listing 5.1: S01D01 - Logon attempts using a non-existing account (Kerberos) [snippet]

```
source=XmlWinEventLog:Security EventCode=4776 Status=0xC0000064
| transaction Workstation maxpause=5m maxevents=-1
| where eventcount > 5
| eval accounts=mvcount(TargetUserName)
| where accounts > 2
| ...
```

Listing 5.2: S01D02 - Logon attempts using a non-existing account (NTLM) [snippet]

5.3.2.2 Excessive failed password attempts

Section 2.2.1 mentions several techniques which adversaries use to guess passwords. Such attacks can be executed in several ways:

1. attempts from *one* source targeting *one* account;
2. attempts from *one* source targeting *multiple* accounts;
3. attempts from *multiple* sources targeting *one* account;
4. attempts form *multiple* sources targeting *multiple* accounts.

This needs to be taken into account while grouping the events. For the scenarios 1 and 2, the grouping can be done on the source computer, very similarly as in the rules detecting use of non-existing account in Section 5.3.2.1. In the scenario 3, the grouping can be done on the target account, covering attempts from multiple sources. The scenario 4 is basically a combination of 2 and 3 and is covered by the rules created for these scenarios.

Almost the same source events can be used for detection of failed password attempts, as were used for detection of non-existing usernames in Section 5.3.2.1. The conditions differ in error code values. There is one important difference in Kerberos auditing, as the event 4768 is not generated for `Status=0x18` (meaning wrong password) and event 4771 generates instead. To summarize:

4625 logged on the computer where the logon attempt was made, with value of the field `SubStatus=0xC000006A`;

4771 recorded on a domain controller meaning that Kerberos pre-authentication failed, with `Status=0x18`;

4776 auditing NTLM authentication attempts, with `Status=0xC000006A` and the generating host depending on the credentials used.

The same principles apply also for the differences in the events mentioned in the Table 5.5. To cover failed attempts coming from one source, I created separate detection rules for Kerberos and NTLM logon attempts, to allow making transactions on the source host. The searches are *S01D03 - Excessive failed password attempts from one source (Kerberos)* and *S01D04 - Excessive failed password attempts from one source (NTLM)* and their full listings can be found in the attachment on the enclosed CD.

Since these rules count on any attempts coming from one source, they cover scenarios 1 and 2, and have an ability to detect Password spraying activity from one source (Password spraying technique is described in Section 2.2.1).

To cover scenario 3, transactions need to be made on the target account. From the Table 5.5 is visible, that the field `TargetUserName` is the only compatible field in all the events, and therefore, creation of separate rules is not necessary. I created the detection rule based on events 4771 and 4776[10], filtering the resulting transactions to only those containing attempts from more than one source. The rule is visible in Listing 5.3.

```
source=XmlWinEventLog:Security ((EventCode=4776
    Status=0xC000006A) OR (EventCode=4771 Status=0x18))
| eval src=if(src=="::1", Computer, src)
| transaction TargetUserName maxpause=5m maxevents=-1
| eval sources=mvcount(src)
| where eventcount > 5 AND sources > 1
| ...
```

Listing 5.3: S01D05 - Excessive failed password attempts towards one account [snippet]

Implementation of detection for failed password attempts needs to reflect *Account Lockout Policy* settings, if effective in the domain. It is also important to consider that adversaries may be aware of the policy settings (explained in Section 2.2.1). As discussed in [89], the baseline recommendations for these settings are not consistent even across various versions of Windows, nor external security community. Therefore, the optimal values vary from domain to domain. All these factors should be considered while setting the thresholds in the searches.

[10]Combining event 4625 into this rule is possible, but would produce duplicate results in some cases. This would complicate reliable filtering on the number of events.

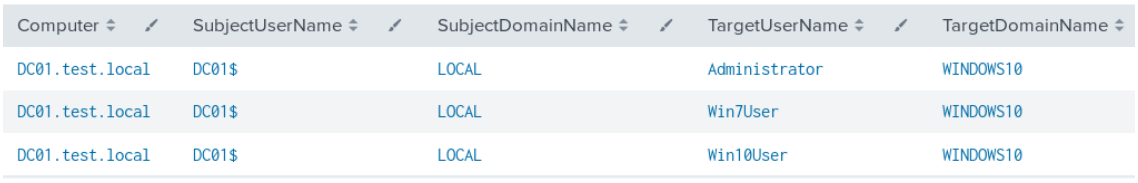
5.3.2.3 Account lockouts

If the *Account Lockout Policy* is effective in the domain, meaning that the user accounts can be locked, monitoring the lockout events may detect a possible brute force attack.

Multiple locked accounts caused by failed password attempts coming from a single source is suspicious, and so is a single account locked multiple times in a row within a short time period. This can happen especially if accounts are being unlocked automatically after some time.

As stated in Section 5.3.1, the event 4740 belonging to the *Audit User Account Management* category is generated for account lockouts. It contains information about two accounts: the name of the account that performed the lockout operation and the name of the account that was locked out.

The XML format of the event 4740 seems to be missing *Caller Computer Name* field. Based on examination of the events from the lab, the information about source was usually contained in the field `TargetDomainName`. As per Microsoft documentation [90], the content of this field vary and it can contain domain names, names of well-known security principals, or for local user accounts, the computer name that the account belongs to. However, during testing, this field contained information about source computer even for domain accounts, as visible in Figure 5.2. Therefore, I used this field in the detection rules.



Computer	SubjectUserName	SubjectDomainName	TargetUserName	TargetDomainName
DC01.test.local	DC01$	LOCAL	Administrator	WINDOWS10
DC01.test.local	DC01$	LOCAL	Win7User	WINDOWS10
DC01.test.local	DC01$	LOCAL	Win10User	WINDOWS10

Figure 5.2: Information contained in the event 4740

I created two detection rules for this scenario. The rule *S01D06 - Multiple locked accounts from one source* (Listing 5.4) makes transactions on the `TargetDomainName` field and returns results when more than one account is locked from the same source within an hour. *S01D07 - Repetitive lockouts of the same account* (Listing 5.5) produces results when the same account is locked more than two times within an hour.

```
source=XmlWinEventLog:Security EventCode=4740
| transaction TargetDomainName maxpause=1h maxevents=-1
| eval accounts=mvcount(TargetUserName)
| where accounts > 1
| ...
```

Listing 5.4: S01D06 - Multiple locked accounts from one source [snippet]

```
source=XmlWinEventLog:Security EventCode=4740
| transaction TargetUserName maxpause=1h maxevents=-1
| where eventcount > 2
| eval sources=mvcount(TargetDomainName)
| ...
```

Listing 5.5: S01D07 - Repetitive lockouts of the same account [snippet]

To monitor failed logon attempts towards locked accounts, the Detection searches in Section 5.3.2.2 can be used with a slight modification of the error codes in the `Status` or `SubStatus` fields. But the situation is not clear with Kerberos events. In the Microsoft documentation for event 4768 is stated: "The `Status=0x12` might be because of an explicit disabling or because of other restrictions in place on the account. For example: account disabled, expired, or locked out." [91] However, during my testing, the event 4768 with `Status=0x12` was generated only for logon attempts towards disabled accounts. For attempts towards locked accounts, the event 4771 with `Status=0x12` was generated instead.

5.3.2.4 Logon attempts towards disabled accounts

Monitoring failed logon attempts towards disabled accounts is yet another category of failure events that may help with detection of brute force attacks. Adversaries may try to use old accounts which are no longer in use in the domain (which may belong for instance to former employees). Furthermore, these detection rules will detect logon attempts towards Windows default accounts, such as *Administrator* or *Guest*, in the case they are disabled and not used. It can be assumed that some brute force attacks will include default accounts, as adversaries can be almost sure about their existence.

The detection rules created for this scenario follow the same principles as the other rules in this Analytic Story; differences are again in the status codes. Listings of the rules *S01D08 - Logon attempts towards disabled accounts (Kerberos)* and *S01D09 - Logon attempts towards disabled accounts (NTLM)* can be found in the attachment on the enclosed CD.

5.3.3 Other searches

The logic of several Detection searches in this Analytic Story can be applied also to events 4625. These events contain some additional information, e.g. the source and the target domain, and can be used for both hostnames and IP addresses. I used them to build Context searches that may be helpful during investigation. Following Context searches are based on this approach:

- S01C01 - Logon attempts using a non-existing account - additional info
- S01C02 - Excessive failed password attempts from one source - additional info

- S01C03 - Excessive failed password attempts towards one account - additional info
- S01C04 - Logon attempts towards disabled accounts - additional info

Investigation searches provided in this Story attempt to find out whether there were any successful logon attempts in the time of the detected brute force activity. Full listings of all the searches can be found in the attachment on the enclosed CD.

5.3.4 Testing

I tested the designed rules both manually, by typing wrong credentials several times in a row at the credential entry dialog box, and automatically, using scripts.

The testing for non-existing usernames revealed some interesting facts. Firstly, the interactive logon carried through the Windows login screen used Kerberos authentication even for the non-existing usernames, while the scripts used NTLM authentication.

I further investigated the point that users can type their usernames in multiple formats, and its implication on the recorded events. Possible formats are:

- <username>
- <username>@<FQDN>
- <NetBIOS>\<username>

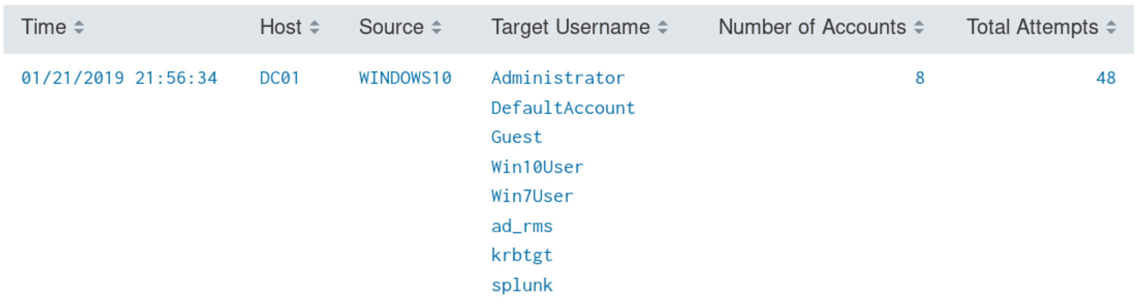
I used combinations of non-existing username with both existing and non-existing domain names and looked on how fields of the recorded events are populated. Table 5.6 summarizes my findings and shows that the formats vary depending on the provided format of the username. Differences in `TargetUserName` field can affect evaluation of the number of different accounts, as *Test01* and *Test01@test.local* are counted as two accounts, while they represent the same account in reality. The impact of this fact on the designed rules is minor. However, the field `TargetDomainNmae` proves that it is not always possible to rely on the consistency in formats or content of a specific field while building searches.

Typed username	Logged event	TargetUserName	TargetDomainName
Test01	4768 + 4625	Test01	LOCAL
LOCAL\Test01	4768 + 4625	Test01	LOCAL
Test01@test.local	4768	Test01@test.local	TEST.LOCAL
	4625	Test01@test.local	<empty>
TEST01\Test01	4625	Test01	TEST01
Test01@test.01	4768	Test01@test.01	**TEST.LOCAL**
	4625	Test01@test.01	<empty>

Table 5.6: Content of fields in the recorded events

For testing automatic logons, I used PowerShell script *Brute-LocAdmin* [22] and script *Invoke-SMBAutoBrute* [23], which provides options to set delay and lockout thresholds.

Figure 5.3 illustrates detection by the rule *S01D04*. The script was executed from the host *WINDOWS10*, attempting to log on to every available account in AD with a set of six different passwords provided.



Time	Host	Source	Target Username	Number of Accounts	Total Attempts
01/21/2019 21:56:34	DC01	WINDOWS10	Administrator DefaultAccount Guest Win10User Win7User ad_rms krbtgt splunk	8	48

Figure 5.3: Detection of a brute force attack by S01D04

5.3.5 False positives

Generally stated, the number of false positive (FP) detections provided by rules in this Analytic Story depends mostly on suitability of specific numeric constants and thresholds set in the Detection searches. There are conditions filtering results on the number of accounts, events, or sources in a specific time period. The optimal setting represents a trade-off between detection capabilities and an FP ratio and can be different for every environment. This can be fine-tuned during implementation, resulting in decrease of the probability of FP detections.

An obvious FP detection that may occur is a repetitive logon attempt of a specific user with a mistyped username or password. Also, sharing computers among multiple users can be problematic. Multiple users putting a wrong password in a short time may appear as a brute force attack coming from one source.

5.4 02 - Kerberoasting

The second Analytic Story involves the topic of service accounts and *Kerberoasting*, the attack technique introduced in Section 2.2.2. Since I recognized multiple approaches on how to detect such activities, I decided to dedicate an entire Analytic Story to this problem.

5.4.1 Information sources

The Kerberoasting technique is targeting Kerberos mechanism used to authenticate users who access protected network resources. The variety of events which contain useful information for this scenario narrows to a single subcategory of *Advanced security audit policies*: *Account Logon\Kerberos Service Ticket Operations*. This policy subcategory should generate three events:

4769(S, F) A Kerberos service ticket was requested;

4770(S) A Kerberos service ticket was renewed;

4773(F) A Kerberos service ticket request failed.

The Microsoft documentation narrows the choice of events even more. The event 4773 is defined but never invoked, and failure event 4769 is generated instead. Event 4770 logs every TGS ticket renewal. However, it has only informational character, and no security monitoring recommendations exist for it. [92]

The event 4769 generates every time KDC gets a Kerberos TGS ticket request. The event generates only on DCs, however, it is one of the most numerous events logged [25]. This event contains lots of valuable information, including account, service, or network information, encryption type used, and failure code. It is a key element for monitoring suspicious activities related to services.

Another type of logs that may be useful for this scenario, although not so directly, are PowerShell logs, which were described in Section 3.1. PowerShell *Script Block Logging* records compiled blocks of scripts into event 4104; PowerShell *Module Logging* records module usage into event 4103.

5.4.2 Detection searches

5.4.2.1 Detecting Kerberoasting via event 4769

Kerberoasting technique, as described in Section 2.2.2, involves the use of a valid domain user's authentication ticket (TGT) to request one or several service tickets using their SPNs. Since the goal of an attacker is to crack the service ticket offline, tickets encrypted with weak cipher suites are preferred.

Sean Metcalf[11] did some research and published several articles on this topic, which name elements suitable for detection of Kerberoasting. Most Detection searches used in this Analytic Story are based on or inspired by ideas published in these articles [25].

Unless there are incompatible or legacy systems used in the environment, all Kerberos authentication should use AES cipher suites, and therefore, any requests for TGS tickets with lower encryption types can be considered suspicious. The detection rule *S02D01 - Possible Kerberoasting activity* looks for any ticket requests with encryption type constants equal to the values of these cipher suites (visible from Table 2.2). The snippet of the search is in Listing 5.6.

In case some systems in the environment need to use older encryption types, these can be whitelisted (e.g. by their service name) directly in the search, or by using `lookup` command provided by a list of services to be whitelisted.

[11]Sean Metcalf is Founder of and Principal Consultant for Trimarc Security, LLC. He performs security research focused on the Microsoft platform and shares his guidance on *ADSecurity.org* and security conferences [93].

```
source=XmlWinEventLog:Security EventCode=4769
    (TicketEncryptionType=0x1 OR TicketEncryptionType=0x3 OR
    TicketEncryptionType=0x17 OR TicketEncryptionType=0x18)
| eval Source=if(IpAddress=="::1", Computer, IpAddress)
| table _time, host, Source, TargetUserName, ServiceName,
    TicketEncryptionType
| sort - _time
| ...
```

Listing 5.6: S02D01 - Possible Kerberoasting activity [snippet]

5.4.2.2 Suspicious service ticket requests

The next two searches focus on service ticket requests and aim to detect suspicious usage of services more generally. The rule *S02D02 - Excessive service ticket requests from one source* (Listing 5.7) triggers if there is a higher amount of different service requests observed in a short time from a single source. This kind of activity is even more suspicious if the service names are not related to each other, or if the type of requested services is unusual for that particular source.

The search uses events 4769. Service ticket requests for *krbtgt* service and computer account service names (those ending with `$` character) are filtered out from the results, as the search focuses mostly on service accounts that were intentionally created for specific resources. Subsequent events are grouped on `IpAddress` field by the `transaction` command. The number of services in each transaction is calculated and filtered to display only results where the number is higher than the one specified in the condition. Again, the number and time span used in the condition represents a variable and has to be adjusted to the needs of the particular environment.

```
source=XmlWinEventLog:Security EventCode=4769 ServiceName !=
    krbtgt
| regex ServiceName != "\$$"
| transaction IpAddress maxpause=5m maxevents=-1
| eval services=mvcount(ServiceName)
| where services > 5
| ...
```

Listing 5.7: S02D02 - Excessive service ticket requests from one source [snippet]

Another search, *S02D03 - Suspicious external service ticket requests*, follows a security recommendation described by Microsoft in the documentation for the event 4769 [26]. The search focuses on network information provided in the event. It monitors usage of well-known ports or any events where the IP address is not from the private IP ranges, which are signs of an outbound connection. The Listing 5.8 shows the detection logic used.

```
source=XmlWinEventLog:Security EventCode=4769 IpPort > 0
    (IpPort < 1024 OR (NOT (IpAddress=10.0.0.0/8 OR
    IpAddress=172.16.0.0/12 OR IpAddress=192.168.0.0/16 OR
    IpAddress=127.0.0.1 OR IpAddress=::1)))
| ...
```

Listing 5.8: S02D03 - Suspicious external service ticket requests [snippet]

The range of IP addresses can be narrowed to only those used in the environment. If there is a scenario where the monitored ports or IP addresses are used by legitimate services, the values can be whitelisted by modifying the detection condition.

5.4.2.3 Detecting Kerberoasting with a honeypot

In one of his articles, Sean Metcalf presents an effective method on how to detect Kerberoasting [94]. He suggests creating a *honeypot* - a fake account, with a fake SPN associated, having some attributes (e.g. `AdminCount`) set, making it attractive for potential attackers. This account has no effective role and privileges in the environment; it is created merely to attract attackers. Monitoring service ticket requests for this account gives clear results of malicious activities with a low false positive ratio, since there is no legitimate reason to request tickets for this service.

I named the account `Honeypot01` for illustration, but the account should look as legitimate as possible in reality. Apart from the `AdminCount` attribute set, it could be a member of seemingly privileged groups to lower potential suspicions of an attacker. Listing 5.9 shows the detection rule *S02D04 - Detecting Kerberoasting with a honeypot*.

```
source=XmlWinEventLog:Security EventCode=4769
    ServiceName=Honeypot01
| eval Source=if(IpAddress=="::1", Computer, IpAddress)
| table _time, host, Source, TargetUserName, ServiceName,
    TicketEncryptionType
| sort - _time
| ...
```

Listing 5.9: S02D04 - Detecting Kerberoasting with a honeypot [snippet]

5.4.2.4 Detecting Kerberoasting via PowerShell

Kerberoasting activity can be carried through PowerShell on a workstation controlled by an attacker. The search *S02D05* uses features of PowerShell logging and its goal is to catch SPN scanning activity or successful acquisition of the service ticket hash.

The search looks for PowerShell events 4103 and 4104 and performs a full-text search in them, looking for strings containing names of service accounts. Transactions are created for all subsequent PowerShell events coming from a single workstation. Results are produced if the number of events containing matching strings is higher than the specified threshold. The list of service accounts and SPNs must be prepared as CSV file and provided to the search by using `inputlookup` command. Listing 5.10 contains details of this search.

```
source="WinEventLog:Microsoft-Windows-PowerShell/
  Operational" (EventCode=4103 OR EventCode=4104)
| transaction Computer maxpause=15m maxevents=-1
| eval raw=_raw
| search
  [| inputlookup service_accounts.csv
  | eval raw="*" . account . "*"
  | fields raw]
| where eventcount > 2
| ...
```

Listing 5.10: S02D05 - Detecting Kerberoasting via Powershell [snippet]

5.4.3 Other searches

Two Context searches provided in this Analytic Story aim to provide valuable context for investigation. The search *S02C01 - List services used by hosts* calculates statistics of successful service requests and provides a matrix containing the number of requests by various sources. The statistics may be used to differentiate between normal traffic and outliers, which may confirm previously detected malicious activity.

The search *S02C02 - List service accounts and SPNs* uses information provided by Active Directory TA for schema changes. It lists a table of account names and their associated SPNs, gained from update events tracked by the Add-on. This table may become handy as a reference during investigations.

Investigation search *S02I01 - Ticket requests via Powershell* finds service ticket requests made via PowerShell. It goes through *PowerShell Operational log* and looks for the string `KerberosRequestorSecurityToken` in the payload. This string is a part of PowerShell command for requesting a TGS ticket.

Another investigation search, *S02I02 - Successful logons using service accounts*, looks for successful logons made by service accounts. Focusing on a time period after possible Kerberoasting activity, it can find the usage of these accounts by an attacker.

Listings of these and other designed searches are available in the attachment on the enclosed CD.

5.4.4 Testing

For testing the designed searches, I used three tools to request a service ticket:

- *GetUserSPNs* module of *Impacket* [29];
- *Invoke-Kerberoast* module of *Empire* [28];
- PowerShell commands based on [25].

GetUserSPNs module was executed on the *kali* machine (192.168.56.105) under the context of unprivileged user *Win7User*, as visible in Listing 5.11. Credentials of this user were provided during the execution.

```
./GetUserSPNs.py -request test.local/Win7User -dc-ip
    192.168.56.102
```

Listing 5.11: Kerberoasting with *Impacket*

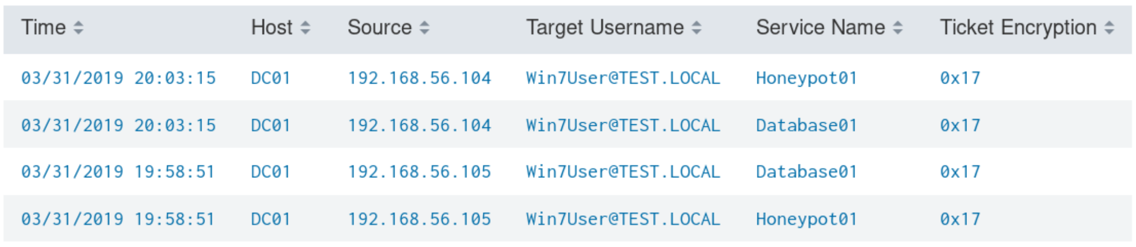
Empire was also executed on the *kali* host, but PowerShell was invoked on the *WINDOWS7* machine (192.168.56.104) via an *HTTP listener*. *Invoke-Kerberoast* module was then executed throughout this listener, as visible in Listing 5.12. The module was run under the context of *Win7User* account, which was signed in on the target machine.

```
usemodule credentials/invoke_kerberoast
set Domain test.local
execute
```

Listing 5.12: Kerberoasting with *Empire*

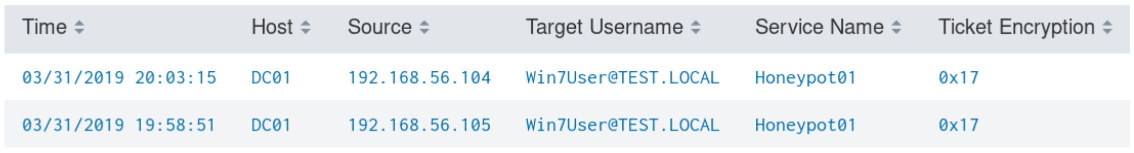
Both *Impacket* and *Empire* were used to perform SPN scanning and request service tickets for found services. Detection of the activities carried can be seen on two figures: Figure 5.4 displays detection results of the search *S02D01 - Possible Kerberoasting activity* and Figure 5.5 displays detection results caught by honeypot search *S02D04*.

I also tested requesting a service ticket for the honeypot account by PowerShell. This method is the simplest, as it does not require an execution of any third-party tools. Commands listed in Listing 5.13 were executed on the DC under the *Administrator* account.



Time	Host	Source	Target Username	Service Name	Ticket Encryption
03/31/2019 20:03:15	DC01	192.168.56.104	Win7User@TEST.LOCAL	Honeypot01	0x17
03/31/2019 20:03:15	DC01	192.168.56.104	Win7User@TEST.LOCAL	Database01	0x17
03/31/2019 19:58:51	DC01	192.168.56.105	Win7User@TEST.LOCAL	Database01	0x17
03/31/2019 19:58:51	DC01	192.168.56.105	Win7User@TEST.LOCAL	Honeypot01	0x17

Figure 5.4: Kerberoasting detected by S02D01



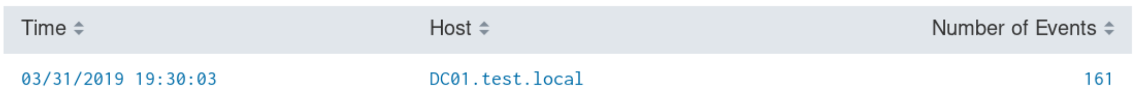
Time	Host	Source	Target Username	Service Name	Ticket Encryption
03/31/2019 20:03:15	DC01	192.168.56.104	Win7User@TEST.LOCAL	Honeypot01	0x17
03/31/2019 19:58:51	DC01	192.168.56.105	Win7User@TEST.LOCAL	Honeypot01	0x17

Figure 5.5: Kerberoasting detected by S02D04

I used the DC because of the availability of PowerShell logs from it. Figure 5.6 shows successful detection of this activity by the rule *S02D05*.

```
$SPNName = ‘MSSQLSvc/DC01.test.’local
Add-Type -AssemblyNAme System.IdentityModel
New-Object
    System.IdentityModel.Tokens.KerberosRequestorSecurityToken
    -ArgumentList $SPNName
```

Listing 5.13: Kerberoasting with PowerShell



Time	Host	Number of Events
03/31/2019 19:30:03	DC01.test.local	161

Figure 5.6: Kerberoasting detected by S02D05

The search *S02D05* turned out to be the least effective, and time-consuming even in the lab environment. This is caused by the fact that it performs a full-text search for a significant number of accounts in full raw PowerShell events. These can be quite numerous and may contain large script blocks. A possible solution to this problem is to narrow the time range as much as possible before the search invocation.

5.4.5 False positives

The number of false positive detections by rules in this story depends on several factors. Firstly, the usage of obsolete cipher suites in the environment. In case these suites are not disabled and whitelisting is not entirely implemented, false positive detections may appear in the search *S02D01*.

The second search, *S02D02*, contains numeric values that control thresholds for detection. These need to be adjusted, as the number of requests for different services in a small environment would not be on the same level as in large environments. Alternatively, the search *S02D02* can be combined with *S02D01* to see excessive service ticket requests with suspicious encryption types only.

Search *S02D03* should not trigger at all unless there actually is a configuration that allows the use of well-known ports or external IP addresses. The same applies to detection using honeypot in *S02D04*. There is no legitimate reason to request a service ticket for the honeypot account. Detected activities are very likely to be malicious.

If PowerShell is utilized for routine administration tasks for the specified service accounts, these activities will also be reported by the search *S02D05*. Reliable filtering is quite tricky due to the variety of commands that could be used by a potential attacker and nature of the PowerShell logs. They contain blocks of code, which limits parsing and also filtering options.

Since the detection rules in this Analytic Story use different approaches, it is likely that a potential attack would be detected by multiple of them, and may be consequently confirmed as true positive with higher probability.

5.5 03 - Credential Dumping

The third Analytic Story focuses on methods of credential dumping described in Section 2.2.3. Searches in this story detect usage of various tools and methods used to dump credentials from a Windows OS.

5.5.1 Information sources

Credential dumping technique involves execution of tools which create processes, creating dump files in the filesystem, performing operations under special privileges, accessing particular network resources, or using specific applications. The range of events that may contain valuable information is wide. The choice of monitored events depends significantly on the chosen detection method.

Some categories of *Advanced security audit policies* that generate interesting events are not included in the *Stronger auditing recommendations* [62], because they produce events in high volumes. Therefore, the choice of monitored events depends also on the capabilities of the log management solution deployed in the environment.

Following list summarizes subcategories which produce events used in the designed detection searches. The subcategories not included in the recommendations are marked with an asterisk (*). Some event IDs are overlapping, as they belong to multiple subcategories. Table 5.7 summarizes particular events used in the searches.

- Detailed Tracking
  - Audit Process Creation
  - Audit Process Termination*
  - Audit Token Right Adjusted*
- DS Access
  - Audit Detailed Directory Service Replication*
  - Audit Directory Service Access
  - Audit Directory Service Changes
  - Audit Directory Service Replication*
- Object Access
  - Audit Detailed File Share*
  - Audit File Share*
  - Audit File System*
  - Audit Handle Manipulation*
  - Audit Kernel Object*
  - Audit Registry*
  - Audit SAM*
- Policy Change
  - Audit Authorization Policy Change*
- Privilege Use
  - Audit Non Sensitive Privilege Use*
  - Audit Sensitive Privilege Use*

Security log is not the only source I used for the creation of searches in this story. The *System log* is used to monitor event 7045 created by Service Control Manager [49], which provides information about installation of services. It is used to detect a possible injection of a malicious service on the target machine.

A significant source of information for the detection rules is also Sysmon. Sysmon allows monitoring process creation and termination, loading of drivers, communication between processes and creation of files. Sysmon events are often richer in provided data, and thus offer a good alternative or supplement to Windows Security events. Table 5.8 summarizes Sysmon events used in this Analytic Story.

Event ID	Type	Message
4688	S	A new process has been created.
4689	S	A process has exited.
4662	S, F	An operation was performed on an object.
5145	S, F	A network share object was checked to see whether client can be granted desired access.
4656	S, F	A handle to an object was requested.
4663	S	An attempt was made to access an object.
4703	S	A token right was adjusted.
4673	S, F	A privileged service was called.

Table 5.7: Events for detection of credential dumping

Event ID	Message
1	Process creation
5	Process terminated
6	Driver loaded
8	CreateRemoteThread
10	ProcessAccess
11	FileCreate

Table 5.8: Sysmon events used for detection

5.5.2 Detection searches

5.5.2.1 Detecting dumping of lsass.exe

Memory of the process *lsass.exe* can be dumped by using several tools. Detection of these tools can be based on monitoring process creation events (4688). These events contain the name of the created process which can be checked to match names of known hacking tools. However, this approach is ineffective in the case when adversaries rename their executables to look like legitimate processes.

Sysmon event for process creation (1) contains additional information: hashes of the executed binaries. Renaming the executable file does not change the hash value. However, most attacker tools can be compiled from source, and a change of a single byte in the source code changes the hash of the resulting binary. Apparently, this approach also loses its advantages.

I was inspired by the approach presented by Teymur Kheirkhabarov[12] at Zero Nights 2017 talk. His presentation "Hunting for Credentials Dumping in Windows Environment" suggests monitoring of process access events and describes other events and artifacts that may be used for detection [95]. I based several searches in this Analytic Story on his work.

The first two searches, *S03D01* and *S03D02*, focus on possible access to the memory of the LSASS process. The rules look for objects targeting *lsass.exe* with granted access

[12]Teymur Kheirkhabarov is Head of SOC at Kaspersky Lab.

mask containing higher privileges required to obtain the credential data. The list of rights represented by particular access masks can be found at [96]. The searches use the method of whitelisting to exclude low-privileged and legitimate process accesses from detection. I based the whitelist mostly on [95], and the events that I saw during testing in the lab.

The rule *S03D01 - Possible dump of lsass.exe (Sysmon events)* is based on Sysmon event 8 - *CreateRemoteThread* and 10 - *ProcessAccess*, and its logic is visible in Listing 5.14. The rule *S03D02 - Possible dump of lsass.exe (Windows events)* contains the same detection logic but is based on Windows event 4656 (*A handle to an object was requested*), and its listing can be found in the attachment on the enclosed CD.

```
source=WinEventLog:Microsoft-Windows-Sysmon/Operational
    EventCode=8 OR EventCode=10 NOT GrantedAccess=0x1400 NOT
    GrantedAccess=0x1000 NOT GrantedAccess=0x100000
| where (TargetImage LIKE "%lsass.exe")
| search NOT SourceImage="C:\\Windows\\system32\\wininit.exe"
    NOT SourceImage="C:\\Windows\\system32\\csrss.exe"
| transaction host, SourceImage, SourceProcessId maxspan=15m
| ...
```

Listing 5.14: S03D01 - Possible dump of lsass.exe (Sysmon events) [snippet]

Dump files can be created by a variety of tools, including Windows built-in utility *Task Manager*. Some programs will create the dump file with `.dmp` extension by default or will not allow the user to change the filename at all. The search *S03D03 - Creation of a dump file* (Listing 5.15) detects the creation of files with the `.dmp` extension. This information, together with the process name, is provided by Sysmon event 11 - *FileCreate*.

```
source=WinEventLog:Microsoft-Windows-Sysmon/Operational
    EventCode=11 TargetFilename=*dmp
| ...
```

Listing 5.15: S03D03 - Creation of a dump file [snippet]

Some tools, such as *Mimikatz* [31], are able to install its own driver to the system. The driver operates in a high-privileged mode and allows bypassing security features. The search *S03D04 - Installation of an unsigned driver* (Listing 5.16) aims to detect such attempts by looking at the driver signature. Sysmon event 6 (*Driver loaded*) with the value `Signed=false` reveals the loading of unsigned drivers on the monitored systems. The filename, together with the file path and hashes can then be used during the investigation. Additional information to the detected events is provided by the Context search *S03C04*, which looks for events 7045. Its listing can be found in the attachment on the enclosed CD.

```
source=WinEventLog:Microsoft-Windows-Sysmon/Operational
    EventCode=6 Signed=false
| ...
```

Listing 5.16: S03D04 - Installation of an unsigned driver [snippet]

5.5.2.2 Group Policy Preferences honeypot

Section 2.2.3 mentioned motivation of attackers to find credentials in *SYSVOL* share. Detection of such attempts can be based on a Group Policy Preferences honeypot [12]. This strategy requires a fake GPP file to be put in *SYSVOL*, containing no effective settings and having *Deny* permissions set for *Everyone* on the file share level. A potential attacker would scan all files in *SYSVOL* for credentials, accessing also this fake preference file. Such attempt would generate a failure event 5145 - *A network share object was checked to see whether client can be granted desired access*, from which details about the source IP address and the account name are obtained. All of this is covered by the detection rule *S03D05 - Access to GPP honeypot in SYSVOL*, shown in Listing 5.17. I created an illustrative policy name `test.local\Policies\{12345}` for the detection rule. Of course, the policy name should not look so suspicious in reality.

```
source=XmlWinEventLog:Security EventCode=5145
    RelativeTargetName="*test.local\\Policies\\{12345}*"
| transaction IpAddress, SubjectUserSid maxspan=5m maxevents=-1
| ...
```

Listing 5.17: S03D05 - Access to GPP honeypot in SYSVOL [snippet]

5.5.2.3 Detecting dumping of credential databases

An obvious choice for attackers is *NTDS.dit*, the AD database file, as it contains lots of information they are after. Methods on how to gain access to this file were described in Section 2.2.3. One of them is to use VSS to create *Volume Shadow Copy* of a DC. It can be done either by a utility *vssadmin.exe* [97] or using *WMI* [98]. A WMI command can be invoked from PowerShell or by *wmic.exe*. Another approach involves using *ntdsutil.exe*, a utility used with *DCPromo* to build a new DC faster. Attackers may also use *reg.exe* to dump hives directly from Registry, where the *SAM database* and *LSA Secrets* are located. [3]

The rule *S03D06 - Possible credential database dumping* detects usage of these methods by looking at command line parameters of newly created processes. This information is provided by Sysmon event 1 - *Process creation* and Windows event 4688 - *A new process has been created*. The event 4688 does not record command line parameters by default. For it to do so, the policy setting *Administrative Templates\System\Audit Process Creation\Include command line in process creation events* must be enabled.

The search groups both Sysmon and Windows events into transactions. Invocation of several searches in parallel is ensured by the `multisearch` command. The events are filtered for the use of specific tools, containing particular strings in command line parameters. Transactions are made on `ProcessName` and `ProcessId` fields in events coming from the same host. In order to group the events correctly, some field names need to be unified, and values of `ProcessId` field need to be translated. Windows events contain this value in hexadecimal format, whereas Sysmon records it in decimal format. This is ensured by `tonumber(ProcessId, 16)` function of the `eval` command. The extensive listing of this rule can be found in the attachment on the enclosed CD.

5.5.2.4 Detecting DCSync

Modern hacking tools are able to abuse the DRS Remote Protocol used by DCs for synchronization and replication. *Mimikatz* includes a feature called *DCSync* [31], which effectively "impersonates" a DC and requests account password data from a real DC via DRS [3].

In case a DC replication occurs, event 4662 - *An operation was performed on an object* is recorded. In the second part of `Properties` field of this event, there is a tree of GUID values belonging to Access Control Entry (ACE) of the AD object, for which the operation was performed. Table 5.9 shows that DC replication can be determined from the GUID value contained in the `Properties` field [99]. The rule *S03D07 - Possible dumping via DC synchronization* uses these values for detection.

Access right	GUID
DS-Replication-Get-Changes	1131f6aa-9c07-11d1-f79f-00c04fc2dcd2
DS-Replication-Get-Changes-All	1131f6ad-9c07-11d1-f79f-00c04fc2dcd2
DS-Replication-Synchronize	1131f6ab-9c07-11d1-f79f-00c04fc2dcd2

Table 5.9: GUID values for DS replication

The event 4662 provides the name of an account, but no network information. However, to distinguish between legitimate and malicious activities, a further context of the related account is necessary. The rule extracts `TargetLogonId` field from the detected events and finds corresponding logon information provided by events 4624. Correlation of these events based on the identifier *Logon ID* is possible because this identifier remains unique between reboots. The detection based on event 4662 is implemented as a subsearch in the search context of events 4624. IP addresses of DCs are excluded from the search using `inputlookup` provided by file `domain_controllers.csv`. Details of the rule can be found in Listing 5.18.

```
source=XmlWinEventLog:Security EventCode=4624
  [ search source=XmlWinEventLog:Security EventCode=4662
   Properties="*1131f6aa-9c07-11d1-f79f-00c04fc2dcd2*" OR
   Properties="*1131f6ad-9c07-11d1-f79f-00c04fc2dcd2*" OR
   Properties="*1131f6ab-9c07-11d1-f79f-00c04fc2dcd2*"
  | fields SubjectLogonId
  | rename SubjectLogonId AS TargetLogonId] NOT
  [ inputlookup domain_controllers.csv
  | fields ip
  | rename ip AS IpAddress]
| ...
```

Listing 5.18: S03D07 - Possible dumping via DC synchronization [snippet]

5.5.3 Other searches

I designed multiple Context searches to support detection rules provided in this Analytic Story. Apart from already mentioned *S03C04*, Context searches *S03C01* and *S03C02* give additional information to events detected by the rules *S03D01* and *S03D02*. The whole detection rules are applied as subsearches, from which fields containing process names and process IDs are extracted. These are used to match other events that contain the same process information and thus provide more data for investigation. This applies to both Sysmon and Windows events.

The Context search *S03C03* helps in the case of *vssadmin.exe* utility usage. Besides looking into events logging creating of processes, it searches for event 8222. Its origin is *VSSAudit* - auditing of *Volume Shadow Copy Service* [97]. To get a process name from event 8222, the field `EventData_Xml` must be parsed further using `spath` command and processed by a regular expression to get the desired information. After that, the events can be correlated on the `ProcessName` field. Listing 5.19 shows the parsing process.

```
...
[ search source=XmlWinEventLog:Security EventCode=8222
| spath input=EventData_Xml
| rex field=Data "(?<ProcessName>.*vssadmin.exe)"]
| ...
```

Listing 5.19: S03C03 - Shadow copy creation [snippet]

The provided Investigative searches focus mostly on suspicious processes; Support searches help with resolving hostnames and translating process ID values. Full listings of all provided searches are available in the attachment on the enclosed CD.

5.5.4 Testing

To test the first three Detection searches, I used different tools to dump the memory of the process *lsass.exe*: Windows built-in *Task Manager*, the module `sekurlsa::logonpasswords` in *Mimikatz* [31] and also the *ProcDump* utility. Rules *S03D01* and *S03D02* produced almost identical output, visible in Figure 5.7. Additionally, the Figure 5.8 shows detection of a dump attempt made by the *ProcDump* executable renamed to "calculator.exe".

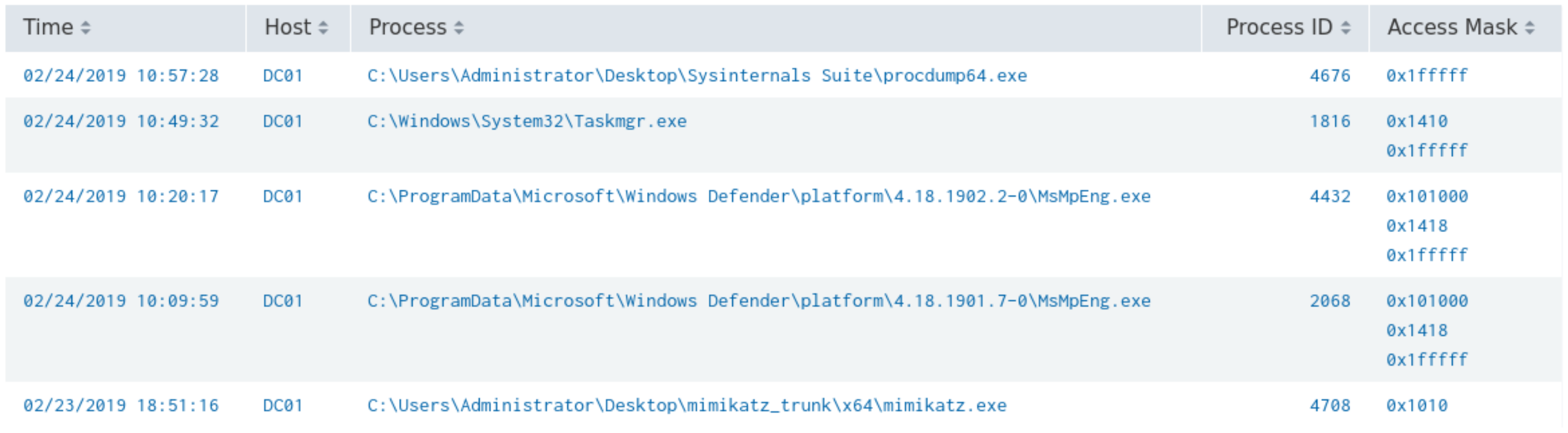
Dumps created by *Task Manager* and *ProcDump* were saved to a file. Both tools assigned the `.dmp` file extension by default, which enabled detection by the rule *S03D03*, as shown in Figure 5.9.

Installation of *Mimikatz* driver *mimidrv.sys* [31] was detected by the rule *S03D04*, evaluating the signature status as "Expired" (Figure 5.10).

Finding passwords in GPP can be as simple as invoking the line `findstr /S /I cpassword \\test.local\sysvol\test.local\policies\*.xml` in *cmd.exe* [12]. However, this accesses the honeypot preference file (even if there is no "cpassword" string in the file) and the activity gets detected by the rule *S03D05*.

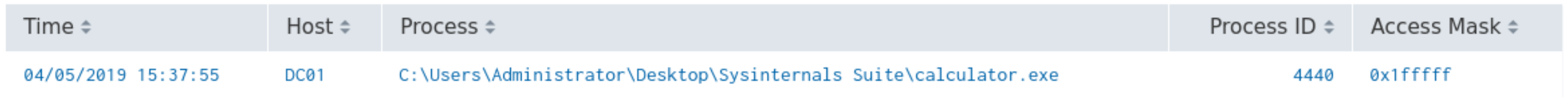
The search *S03D06* detected creation of a *Volume Shadow Copy* by `vssadmin create shadow /for=C:` based on match of the strings "vssadmin" and parameters "shadow" and "create". This is shown in Figure 5.11. However, if the attacker changed the name of the executable *vssadmin.exe* to something else, the activity would remain undetected. This is a weakness of the rules based on exact string matching in general.

Finally, Figure 5.12 shows detection of *DCSync*. I invoked *DCSync* (which is also part of Empire framework) from *kali* machine, throughout HTTP PowerShell listener on *WINDOWS7* host (which is definitely not a DC). *DCSync* was executed under the context of the *Administrator* account, as it requires account membership in particular groups to run successfully [100].



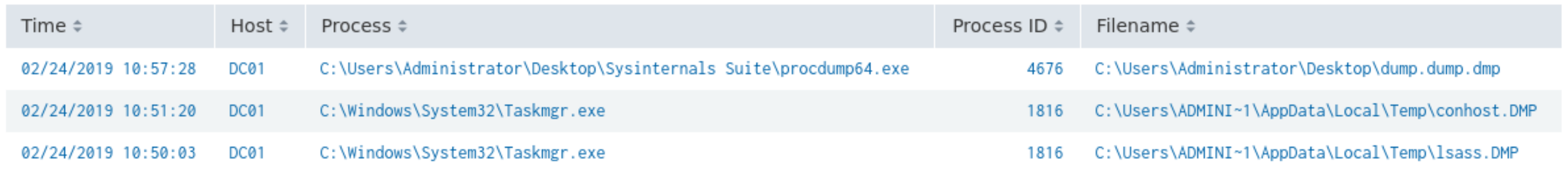
Time	Host	Process	Process ID	Access Mask
02/24/2019 10:57:28	DC01	C:\Users\Administrator\Desktop\Sysinternals Suite\procdump64.exe	4676	0x1fffff
02/24/2019 10:49:32	DC01	C:\Windows\System32\Taskmgr.exe	1816	0x1410 0x1fffff
02/24/2019 10:20:17	DC01	C:\ProgramData\Microsoft\Windows Defender\platform\4.18.1902.2-0\MsMpEng.exe	4432	0x101000 0x1418 0x1fffff
02/24/2019 10:09:59	DC01	C:\ProgramData\Microsoft\Windows Defender\platform\4.18.1901.7-0\MsMpEng.exe	2068	0x101000 0x1418 0x1fffff
02/23/2019 18:51:16	DC01	C:\Users\Administrator\Desktop\mimikatz_trunk\x64\mimikatz.exe	4708	0x1010

Figure 5.7: Dumping of the *lsass.exe* process detected by S03D01



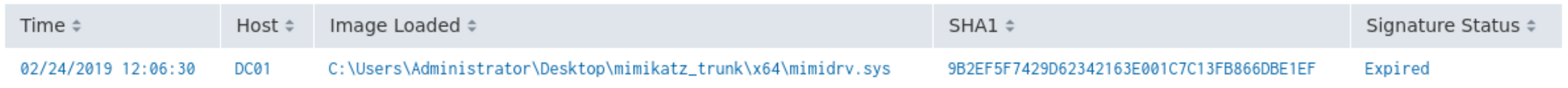
Time	Host	Process	Process ID	Access Mask
04/05/2019 15:37:55	DC01	C:\Users\Administrator\Desktop\Sysinternals Suite\calculator.exe	4440	0x1fffff

Figure 5.8: Dump of *lsass.exe* made by "calculator"



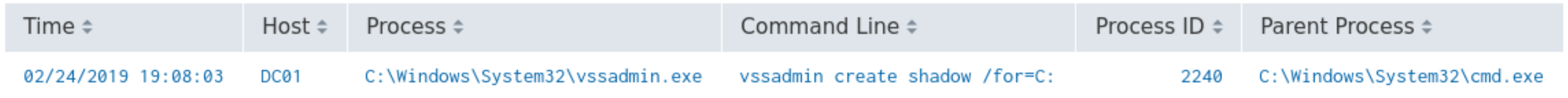
Time	Host	Process	Process ID	Filename
02/24/2019 10:57:28	DC01	C:\Users\Administrator\Desktop\Sysinternals Suite\procdump64.exe	4676	C:\Users\Administrator\Desktop\dump.dump.dmp
02/24/2019 10:51:20	DC01	C:\Windows\System32\Taskmgr.exe	1816	C:\Users\ADMINI~1\AppData\Local\Temp\conhost.DMP
02/24/2019 10:50:03	DC01	C:\Windows\System32\Taskmgr.exe	1816	C:\Users\ADMINI~1\AppData\Local\Temp\lsass.DMP

Figure 5.9: Detection of saved dump files by S03D03



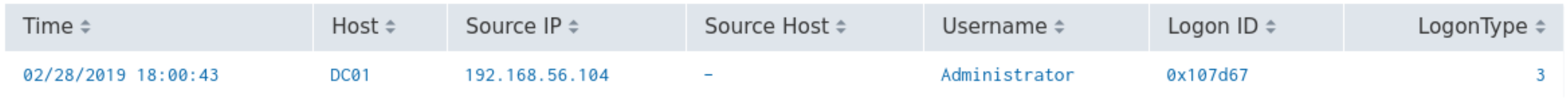
Time	Host	Image Loaded	SHA1	Signature Status
02/24/2019 12:06:30	DC01	C:\Users\Administrator\Desktop\mimikatz_trunk\x64\mimidrv.sys	9B2EF5F7429D62342163E001C7C13FB866DBE1EF	Expired

Figure 5.10: Installation of *mimidrv.sys* detected by S03D04



Time	Host	Process	Command Line	Process ID	Parent Process
02/24/2019 19:08:03	DC01	C:\Windows\System32\vssadmin.exe	vssadmin create shadow /for=C:	2240	C:\Windows\System32\cmd.exe

Figure 5.11: Use of *vssadmin.exe* detected by S03D06



Time	Host	Source IP	Source Host	Username	Logon ID	LogonType
02/28/2019 18:00:43	DC01	192.168.56.104	-	Administrator	0x107d67	3

Figure 5.12: Use of *DCSync* detected by S03D07

5.5.5 False positives

Several rules in this Analytic Story require whitelisting to prevent FP detections. Firstly, there are legitimate processes which need to access *lsass.exe* with higher privileges. For instance, Figure 5.7 shows false positive detection for process *MsMpEng.exe* executed by *Windows Defender*, which accesses *lsass.exe* for legitimate purposes. The current whitelist in the rules *S03D01* and *S03D02* needs to be extended to cover such processes.

The rule *S03D03* triggers for the creation of any dump file, which is visible also from Figure 5.9. The second result is a dump of *conhost.exe*, a process not related to the subject of this story. Certainly, there are reasons for creating dump files of processes, e.g. for debugging purposes. In case this happens regularly, narrowing the search or whitelisting particular files may help prevent false positives. An approach of blacklisting based on file name is not applicable, as the name can be spoofed by an attacker.

Installation of any unsigned service will be detected by the rule *S03D04*. Prevention of FP detections can be achieved by sensitive whitelisting of legitimate software that is normally used.

The rule *S03D05* should not trigger false positively, as there is no reason to access the honeypot file after its creation. An exception is vulnerability scanners that may access the share during their operation and trigger the rule.

Another rule that may need whitelisting is *S03D06*. VSS can be used regularly as a backup solution in the environment. Also, some of the tools included in the search can be used by administrators to perform their operations. In such a case, it is important to define roles and monitor if the actions were taken by legitimate accounts.

The search focusing on DC replication, *S03D07*, did not return any FP results during testing in the lab. However, I had no opportunity to test it in an environment containing more DCs, with the actual replication involved.

5.6 04 - Credential Theft

Adversaries who harvested domain credentials via credential dumping or other methods will likely use them to escalate privileges, access network resources, and move laterally through the domain. This Analytic Story covers techniques described in Section 2.3, in which attackers work directly with authentication protocols that use password hashes or tickets, instead of providing a plaintext password.

5.6.1 Information sources

The techniques in this story trick Windows authentication mechanisms to grant access to protected resources even for users who do not possess required privileges. However, the footprint of such activity looks like a legitimate authentication process. This implies that the spectrum of recorded events does not differ compared to the trail of standard authentication and resource access.

Table 5.10 summarizes the most important events containing valuable data for detection of credential theft. All these events were already mentioned in previous Stories, together with the respective *Advanced security audit policy* categories that generate them. To recapitulate, the events involved are auditing Kerberos and NTLM protocols, account logon and logoff, and the use of special privileges.

Event ID	Type	Message
4624	S	An account was successfully logged on.
4625	F	An account failed to log on.
4634	S	An account was logged off.
4648	S	A logon was attempted using explicit credentials.
4672	S	Special privileges assigned to new logon.
4776	S, F	The domain controller attempted to validate the credentials for an account.
4768	S, F	A Kerberos authentication ticket (TGT) was requested.
4769	S, F	A Kerberos service ticket was requested.

Table 5.10: Events for detection of credential theft

One Detection search in this Analytic Story uses custom events, numbered 3001 and 3002, written to the *Application log*. These events are generated by a PowerShell script using `Write-EventLog` cmdlet [42], which is described later.

5.6.2 Detection searches

5.6.2.1 Pass the Hash

Detection of PtH and similar attacks based on static rules is difficult. The attacks are not targeting the protocol itself; attackers rather create anomalies at the behavioral level. There is no predefined condition defining how to filter anomalies from the events to detect these attacks.

Multiple sources [37, 101, 102] emphasize the importance of role separation, high-value account usage definition, and privilege management, as detection in well-managed environments is more effective. In general, the detection is based on monitoring non-interactive logons of privileged accounts from unusual machines. Privileged accounts do not involve only members of administrator groups, but also service accounts, or any other accounts that may have access to sensitive data. To check whether there was a prior interactive logon with the related account on the source machine is then the subject of investigation.

Pass the Hash attack targets NTLM authentication. Figure 5.13 based on [101] demonstrates events logged during the attack scenario. The most important part is the event 4776 logged on a DC for the use of a privileged account from an unusual source, followed by an event 4624/4625 on the accessed machine. Logging of the event 4648 depends on the attacker's procedure and usually occurs on workstations, from which logs are not collected.

The Detection search *S04D01 - Pass the Hash* correlates events 4776 and logon events 4624/4625 for both success and failure. The search expects collection of events 4624 and 4625 at least from critical systems, which may represent potential targets. These events are

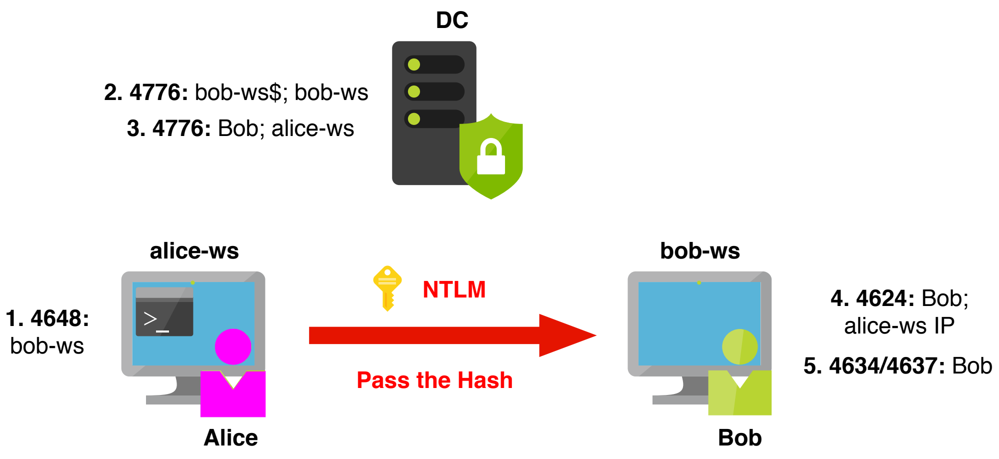


Figure 5.13: Events recorded during Pass the Hash

filtered for `LogonType=3`, meaning network logon and `AuthenticationPackageName="NTLM"` because events 4624/4625 are also logged for Kerberos authentication. The error value `Status=0xC000006A` (bad password) can detect attackers who fail to provide the correct password hash during the attack. The value `TargetUserName="ANONYMOUS LOGON"` is filtered out, as these events ordinarily occur during share enumeration. Only the transactions containing both event types are preserved in the results. A snippet of the search is visible in Listing 5.20.

```
| multisearch
  [ search source=XmlWinEventLog:Security EventCode=4624
   LogonType=3 AuthenticationPackageName="NTLM" NOT
   TargetUserName="ANONYMOUS LOGON"]
  [ search source=XmlWinEventLog:Security EventCode=4625
   SubStatus=0xC000006A LogonType=3
   AuthenticationPackageName="NTLM" NOT
   TargetUserName="ANONYMOUS LOGON"
  | rename SubStatus AS Status]
  [ search source=XmlWinEventLog:Security EventCode=4776
   Status=0xC000006A OR Status=0x0
  | rename Workstation AS WorkstationName]
| transaction TargetUserName, WorkstationName, Status
   maxspan=1h maxevents=-1
| search EventCode=4776 EventCode=4624 OR EventCode=4625
| ...
```

Listing 5.20: S04D01 - Pass the Hash [snippet]

5.6.2.2 Pass the Ticket

Opposing to PtH attack, Pass the Ticket targets Kerberos authentication, so the detection is based on Kerberos events. Figure 5.14 based on [101] shows events recorded during the attack. The most important element is the service ticket request logged in the event 4769, containing an unusual combination of privileged account and source IP address. Again, followed by an event 4624/4625 on the target machine. In PtT, also the first authentication event 4768 is available, as it is logged on a DC. However, the TGT ticket may be stolen prior to the time of the attack, and subsequently the event 4768 may not be a part of the resulting correlation.

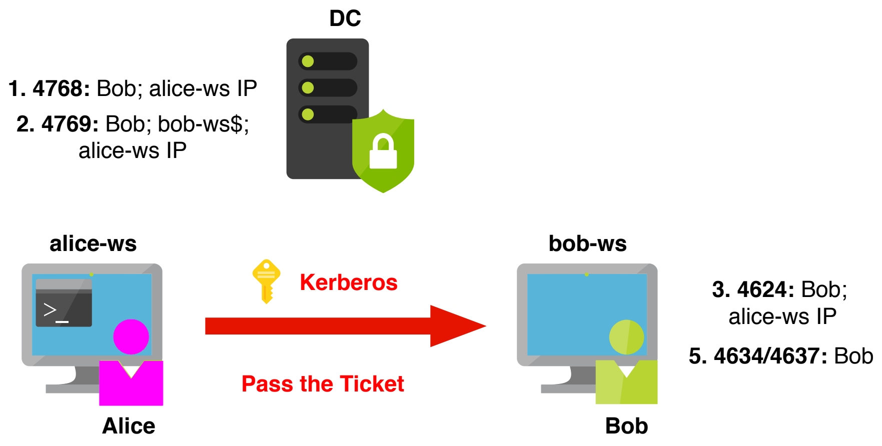

DC
1. 4768: Bob; alice-ws IP
2. 4769: Bob; bob-ws$; alice-ws IP
alice-ws
Kerberos
Pass the Ticket
bob-ws
3. 4624: Bob; alice-ws IP
5. 4634/4637: Bob
Alice
Bob

Figure 5.14: Events recorded during Pass the Ticket

The rule *S04D02 - Pass the Ticket* creates transactions containing event 4769 and one of the events 4624/4625. Logon events are filtered for Network logon type and Kerberos authentication package. Several fields must be further processed to allow correlating. For instance, `TargetUserName` is logged as a FQDN in the event 4769, so the domain suffix must be removed.

Additionally, a new field `SvcName` is created from `ServiceName` by trimming the ending `$` character. This new field is used to filter out service tickets for some computer accounts, especially domain controllers' computer accounts and computer account of the source workstation. List of DCs is provided by lookup file `domain_controllers.csv`. Another lookup file, `hostnames.csv`, is used to resolve IP addresses to domain names, as there are only IP addresses logged in Kerberos events. Resulting transactions are filtered on the number of services higher than one because requests for the *krbtgt* service are always present. Listing 5.21 shows the core of the search, containing field processing, lookups, and filtering of event 4769.

```
...
| eval TargetUserName=replace(TargetUserName, "(.*)@(.*)",
  "\1")
| eval SvcName=replace(ServiceName, "(.*)\$$", "\1")
| lookup hostnames.csv ip AS IpAddress OUTPUTNEW hostname AS
  Workstation
| where EventCode != 4769 OR (EventCode=4769 AND NOT
  [ inputlookup domain_controllers.csv
  | fields hostname
  | rename hostname AS SvcName ] AND Workstation != SvcName)
| ...
```

Listing 5.21: S04D02 - Pass the Ticket [snippet]

5.6.2.3 Overpass the Hash

Overpass the Hash is from the perspective of logged events effectively a PtT attack, except that it starts with TGT request with encryption type signalizing the use of NTLM hash. The rule *S04D03 - Overpass the Hash* is very similar to *S04D02*, having two differences: events 4768 are filtered to `TicketEncryptionType=0x17` and only transactions starting with this event are propagated to the results. This rule is effectively a subset of the search *S04D02*. However, the produced results may differ to the nature of the `transaction` command. The snippet in Listing 5.22 displays the changes applied to the rule *S04D02*.

```
| multisearch
  [ search source=XmlWinEventLog:Security (EventCode=4768
   Status=0x0 TicketEncryptionType=0x17) OR EventCode=4769 OR
   (EventCode=4624 LogonType=3
   AuthenticationPackageName="Kerberos")]
| ...
| transaction IpAddress, TargetUserName
   startswith=EventCode=4768 maxspan=1h maxevents=-1
| ...
```

Listing 5.22: S04D03 - Overpass the Hash [snippet]

5.6.2.4 Golden/Silver Tickets

Detection of Golden and Silver tickets from Windows security events is difficult. Some sources suggest detection based on anomalies in events [40, 103]. It is based on the fact that hacking tools often cannot create forged tickets as precisely as they would if they were created by the OS during authentication. Therefore, some anomalies in field values may be a sign of their usage.

For example, *Mimikatz* populated the account domain information differently in its several versions. Some versions filled it with constant strings containing "eo.oe", some left the field blank, and others filled the real domain name, but not in the correct format[13]. In latest versions of *Mimikatz*, forged Kerberos tickets no longer include a domain anomaly since the NetBIOS domain name is placed in the domain component of the Kerberos ticket. [40]

Nevertheless, I created the rule *S04D05 - Golden/Silver Ticket based on anomalies in events*, which looks for anomalies in the `TargetDomainName` field. Some events log domain information in `SubjectDomainName`. The rule renames the field where necessary, as the field names must be unified before further correlation.

Instead of matching exact strings, like "eo.oe", the search implements a method of whitelisting. Legitimate values are whitelisted, which includes the actual domain names (both NetBIOS domain name and FQDN), and also legitimate Windows service names that usually occur in this field (and were found during testing). It may also be a good idea to whitelist all the hostnames of workstations in the environment. In the search, these are provided by lookup file `hostnames.csv`. Events containing these values can be logged during authentication of local accounts. After correlation, only transactions including more than one event are preserved. This eliminates FP detections in case of misspelled domain name during an interactive logon. Listing of the rule is available in the attachment on the enclosed CD.

A much better approach for detection of forged Kerberos tickets is mentioned in [103] and described in [104], although it is more difficult to implement. This method is looking for TGTs which have a duration (lifetime) that is different from the value set per domain. The comparison is made by a PowerShell script that compares the `MaxTicketAge` from the domain policy to the difference in the `StartTime` and `EndTime` of the cached authentication tickets. Possible bad tickets are written to the *Application log.* It is required to run a scheduled task to invoke this script on monitored systems.

To implement this approach, I slightly modified the script from [104]. The main changes involved merging the original two scripts into a single one and writing the event data in JSON format to enable better parsing of the events in Splunk. This was done by using `ConvertTo-Json` cmdlet [105], which was introduced in Windows PowerShell 3.0[14]. The source code of the script is available in the attachment on the enclosed CD.

Following are the basic steps performed by the script:

1. Retrieve the Kerberos `MaxTicketAge` from the GPO / Domain Policy.

2. Use `klist.exe sessions` to view the cached sessions.

3. View the TGTs using `klist tgt -li <sessionid>`.

4. View the TGS tickets using `klist tickets -li <sessionid>`.

[13]The events contained FQDN, while they should contain NetBIOS domain name.
[14]This may require upgrade of PowerShell version on some systems, see Section 3.1.

5. Extract the `EndTime` and `StartTime` values.

6. Subtract the `EndTime` from `StartTime` and compare that value against the configured `MaxTicketAge`.

7. Write information about suspicious tickets to the *Application log* with Event IDs 3001 and 3002 (unless configured differently).

The search *S04D04 - Golden/Silver Ticket based on PowerShell script* (Listing 5.23) looks for the generated events and performs necessary parsing of JSON event data by using `spath` command. The same cached tickets may exist between two invocations of the scheduled task on the monitored systems, and thus duplicate information may be written to the *Application log*. The `dedup` command eliminates possible duplicate entries and keeps only the oldest result found.

```
source=XmlWinEventLog:Application EventCode=3001 OR
    EventCode=3002
| spath input=EventData_Xml output=EventData_JSON path=Data
| spath input=EventData_JSON
| dedup host, Client, Server, SessionID, LogonType,
    SessionKeyType, StartTime, EndTime, RenewTime sortby +_time
| ...
```

Listing 5.23: S04D04 - Golden/Silver Ticket based on PowerShell script [snippet]

Another, more generic approach for detecting forged Kerberos tickets is to look for unusual error codes in failure events 4769. This is implemented in the Detection search *S04D06 - Possible forged service ticket* (Listing 5.24), which looks for any service ticket requests ending with failure, apart from already whitelisted failure codes. The pre-defined whitelist includes failure codes `0x20`, `0x21`, and `0x25` that occurred during testing. It is possible to add more codes from the list available at [26].

This rule has one interesting detection capability: it can detect adversaries trying to exploit vulnerability *MS14-068* [106] (mentioned in Section 2.3) in a patched environment. As described in [9], events 4769 will be logged with a failure code in such a case.

```
source=XmlWinEventLog:Security EventCode=4769 NOT Status=0x0
    NOT Status=0x21 NOT Status=0x20 NOT Status=0x25
| transaction IpAddress, Status maxspan=1h maxevents=-1
| ...
```

Listing 5.24: S04D06 - Possible forged service ticket [snippet]

5.6.3 Other searches

Context searches in this Story focus further on behavioral anomalies that may appear in the events. Notably, the searches *S04C01* and *S04C02* find events in which the username used is not consistent with its SID value. The values from logs are compared to the real data contained in the AD schema. The information from AD can be supplied in the form of a lookup file, or by using direct subsearch, which will retrieve the required data. The Support search *S04S01* is prepared to be used for this purpose. The best solution is a combination of the previous two – use of the lookup file, which will be regularly updated by *S04S01*. This will eliminate unnecessary invocations of the subsearch, which would produce the same results most of the time.

Apart from the fundamental review of account activity and interactive logons prior to the possible credential theft detected, Investigative searches in this story aim to help by providing some statistical data. The search *S04I01* gives information about user accounts logging on from a particular workstation. It may help determine which accounts use the workstation regularly, and on the other hand, identify accounts used for suspicious activity. The search *S04I02* performs a similar evaluation, but for service requests.

The goal of an investigation is also to determine the potential impact of the detected activity. The search *S04I03* shows attempts to access network shares made from a specific source and lists resources that were accessed by the impersonated user account. Full listings of these searches are available in the attachment on the enclosed CD.

5.6.4 Testing

For testing the majority of the rules in this Analytic Story, I used the "credential multi-tool" *Mimikatz*. For the PtH attack, I used *psexec* module of the *Metasploit* framework [35]. This was because the newer versions of *Mimikatz* module `sekurlsa::pth` perform OPtH instead [31].

I attempted to execute PtH several times, by invoking *psexec* from the *kali* machine (192.168.56.105), towards *SERVER2008* and account *Database01*. The Figure 5.16 shows detection of these activities. Note that the source hostname *WORKSTATION* was automatically inserted by *Metasploit* and does not exist in the environment. Another important remark are the results containing `Status=0xc000006a`. These were attempts where I (as an attacker) failed to provide the password hash in the correct format for the target account.

PtT and OPtH attacks were executed by using the respective modules of *Mimikatz*. For these two attacks, I created the same scenario: an attacker has compromised local administrator account on the host *WINDOWS7* (*IEUser*) and uses *Mimikatz* to perform OPtH and PtT to impersonate account *Database01*, which has access to the protected network share on the host *SERVER2008*. Diagram in Figure 5.15 illustrates the attack scenario together with the recorded events. Figure 5.17 shows detection of both activities by *S04D02*, Figure 5.18 shows detection of OPtH by *S04D03* and results of the Investigation search *S04I03* can be seen in Figure 5.19. These confirm the access to the protected share *SharedFolder*. From the figures, it is visible that the two attack methods differentiate in the `Ticket Encrypiton` field.

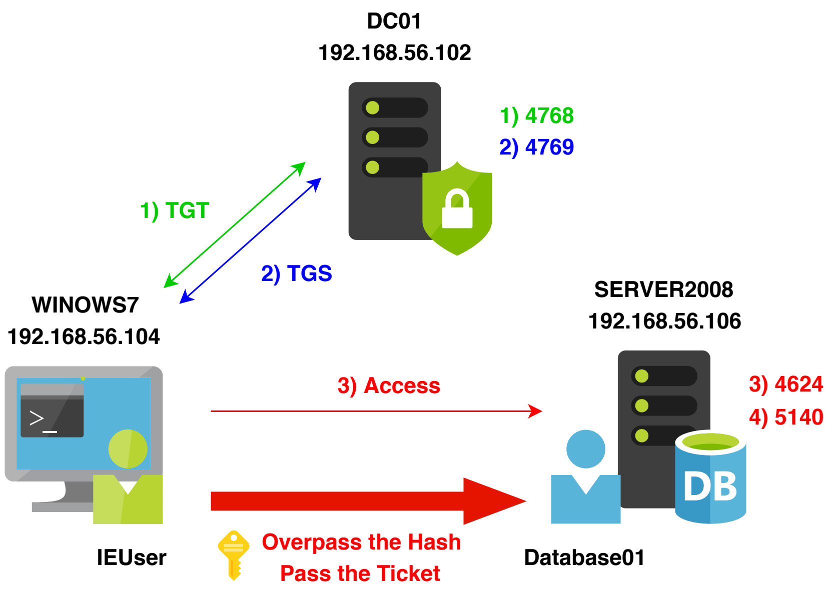
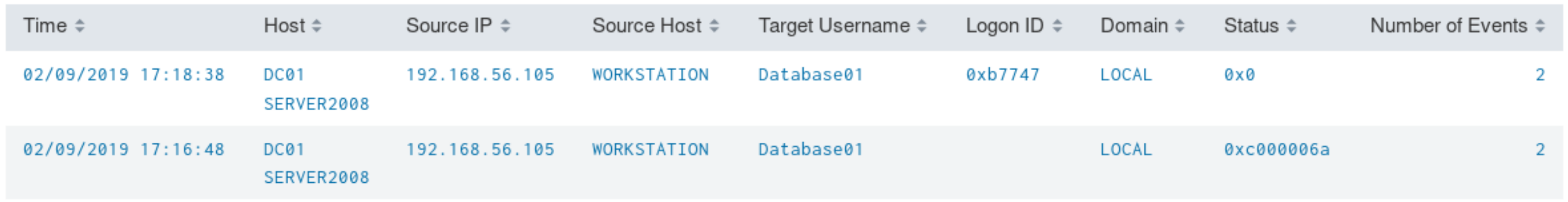


Figure 5.15: Scenario for OPtH and PtT attacks

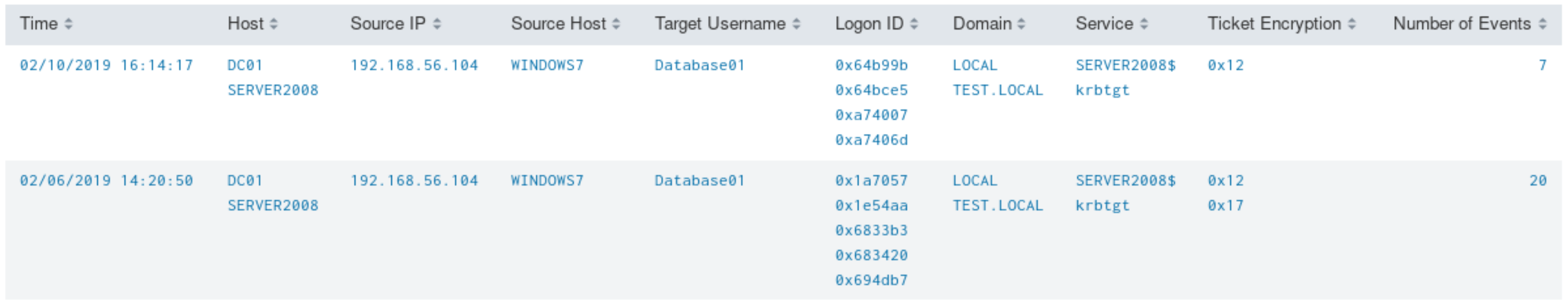
Creation of Golden and Silver Tickets was tested by executing *Mimikatz* modules `mimikatz::golden` and `mimikatz::silver` from the host *SERVER2008*[15] under account *Database01*, and used to impersonate *Administrator* to execute `dir \\DC01\C$` in *cmd.exe*. Figure 5.20 shows the detection of these invocations by the rule *S04D04*. The PowerShell script was scheduled to run every 15 minutes. Results starting with *krbtgt* in `Server` field detected creation of Golden Tickets.

[15]Execution on a monitored asset with a configured scheduled task invoking the PowerShell script was necessary to test detection by the rule *S04D04*.



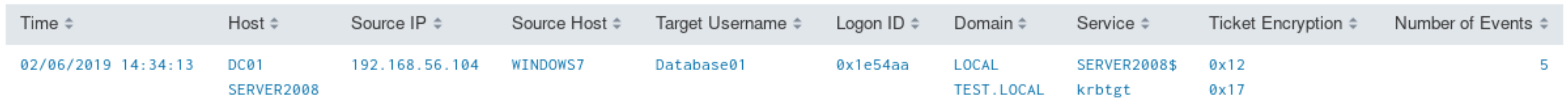
Time	Host	Source IP	Source Host	Target Username	Logon ID	Domain	Status	Number of Events
02/09/2019 17:18:38	DC01 SERVER2008	192.168.56.105	WORKSTATION	Database01	0xb7747	LOCAL	0x0	2
02/09/2019 17:16:48	DC01 SERVER2008	192.168.56.105	WORKSTATION	Database01		LOCAL	0xc000006a	2

Figure 5.16: Detection of PtH by S04D01



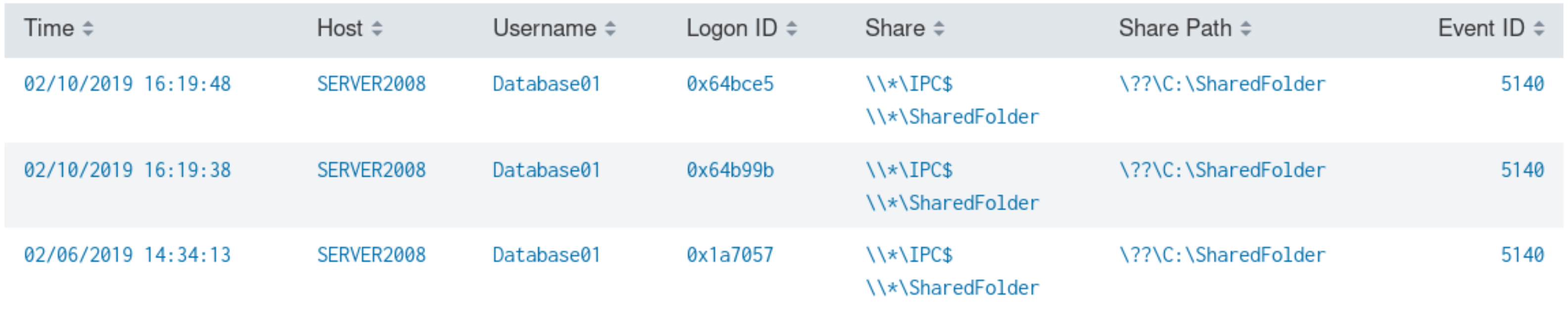
Time	Host	Source IP	Source Host	Target Username	Logon ID	Domain	Service	Ticket Encryption	Number of Events
02/10/2019 16:14:17	DC01 SERVER2008	192.168.56.104	WINDOWS7	Database01	0x64b99b 0x64bce5 0xa74007 0xa7406d	LOCAL TEST.LOCAL	SERVER2008$ krbtgt	0x12	7
02/06/2019 14:20:50	DC01 SERVER2008	192.168.56.104	WINDOWS7	Database01	0x1a7057 0x1e54aa 0x6833b3 0x683420 0x694db7	LOCAL TEST.LOCAL	SERVER2008$ krbtgt	0x12 0x17	20

Figure 5.17: Detection of PtT and OPtH attacks by S04D02



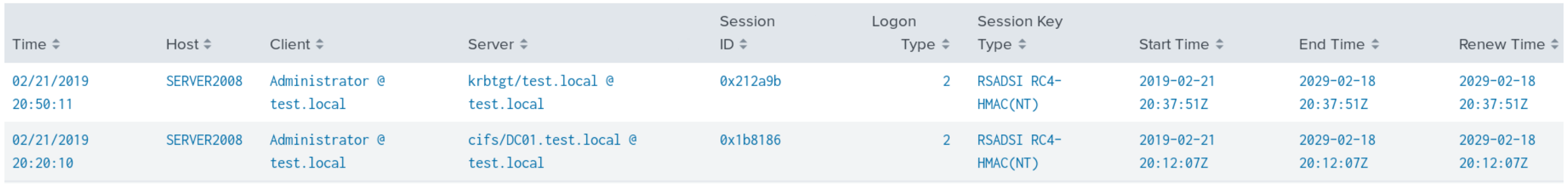
Time	Host	Source IP	Source Host	Target Username	Logon ID	Domain	Service	Ticket Encryption	Number of Events
02/06/2019 14:34:13	DC01 SERVER2008	192.168.56.104	WINDOWS7	Database01	0x1e54aa	LOCAL TEST.LOCAL	SERVER2008$ krbtgt	0x12 0x17	5

Figure 5.18: Detection of OPtH by S04D03



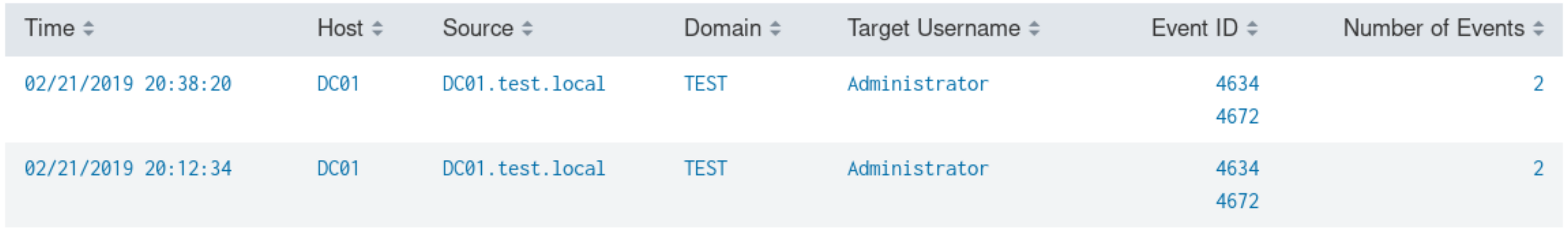
Time	Host	Username	Logon ID	Share	Share Path	Event ID
02/10/2019 16:19:48	SERVER2008	Database01	0x64bce5	*\IPC$ *\SharedFolder	\??\C:\SharedFolder	5140
02/10/2019 16:19:38	SERVER2008	Database01	0x64b99b	*\IPC$ *\SharedFolder	\??\C:\SharedFolder	5140
02/06/2019 14:34:13	SERVER2008	Database01	0x1a7057	*\IPC$ *\SharedFolder	\??\C:\SharedFolder	5140

Figure 5.19: Investigation results returned by S04I03



Time	Host	Client	Server	Session ID	Logon Type	Session Key Type	Start Time	End Time	Renew Time
02/21/2019 20:50:11	SERVER2008	Administrator @ test.local	krbtgt/test.local @ test.local	0x212a9b	2	RSADSI RC4-HMAC(NT)	2019-02-21 20:37:51Z	2029-02-18 20:37:51Z	2029-02-18 20:37:51Z
02/21/2019 20:20:10	SERVER2008	Administrator @ test.local	cifs/DC01.test.local @ test.local	0x1b8186	2	RSADSI RC4-HMAC(NT)	2019-02-21 20:12:07Z	2029-02-18 20:12:07Z	2029-02-18 20:12:07Z

Figure 5.20: Detection of forged tickets by S04D04



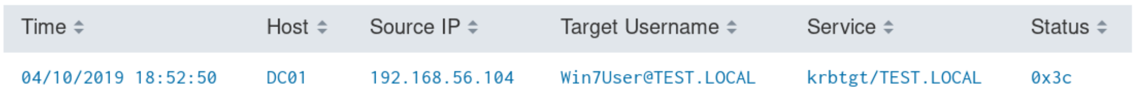
Time	Host	Source	Domain	Target Username	Event ID	Number of Events
02/21/2019 20:38:20	DC01	DC01.test.local	TEST	Administrator	4634 4672	2
02/21/2019 20:12:34	DC01	DC01.test.local	TEST	Administrator	4634 4672	2

Figure 5.21: Detection of forged tickets by S04D05

Interestingly, the activities were also caught by the rule *S04D05* (Figure 5.21, which was initially designed to catch forged tickets based on domain information anomalies in the field `TargetDomainName`. This occurred even though I used *Mimikatz* version 2.2.0 [31], which populates the field with the domain NetBIOS name and both domain NetBIOS name and FQDN were whitelisted in the rule.

It turned out that *Mimikatz* did not properly resolve NetBIOS domain name from FQDN, which was provided as a parameter while executing its modules `kerberos::golden` or `kerberos::silver`. As described in Section 5.1.3, I set the FQDN to *test.local* and the NetBIOS name to *LOCAL*. However, *Mimikatz* populated the field with the value *TEST*[16]. This simple mistake allowed the rule *S04D05* to detect the activities.

Although the only DC in the lab environment is based on Windows Server 2016 and therefore not vulnerable to *MS14-068*, I executed the available exploit to test detection capabilities of the rule *S04D06*. I executed *PyKEK* [9, 107] from *WINDOWS7* machine under the account *Win7User*, with the combination of `kerberos::ptc` module of *Mimikatz*, as described in example usage in [107]. An attempt to list `C$` share on the DC ended as *"Access is denied."*. This was expected since the environment is patched. But the activity was detected by the rule *S04D06*, as visible in Figure 5.22, with `Status=0x3c` signalizing *"Generic error"* [26].



Time	Host	Source IP	Target Username	Service	Status
04/10/2019 18:52:50	DC01	192.168.56.104	Win7User@TEST.LOCAL	krbtgt/TEST.LOCAL	0x3c

Figure 5.22: Detection of MS14-068 attack by S04D06

5.6.5 False positives

The rules *S04D01*, *S04D02*, and *S04D03* may have a lot of FP detections and trigger even in situations when legitimate resource access occurs. This is caused by the nature of the attacks and the fact that the rules do not contain any clear condition that would allow filtering only for malicious events. I observed the FPs even in the lab environment, which generated a significantly lower amount of logs from legitimate traffic comparing to the real environments.

Excluding events recording service tickets for DCs' computer accounts in rules *S04D02* and *S04D03* helped to reduce the number of false positives significantly (as such requests occur commonly), but also lowered the visibility. For example, if an attacker impersonates the *Administrator* account and does not request any TGS tickets other than for the DC account, such an attack will remain undetected. On the other hand, removing the exclusion makes the search return results for usual traffic, and thus unusable for detection.

The situation gets better with the rules detecting forged Kerberos tickets. I recognized possible FP detection for the rule *S04D05* looking for anomalies in events. It occurs when

[16]It actually makes more sense to set the NetBIOS name to *TEST*, as *local* is the domain suffix. But I configured the domain the other way…

a user mistypes the domain name, e.g. during interactive logon. However, this can be eliminated by adjusting the threshold condition in the rule. If there are other sources of anomalies in the environment identified, they can be reduced by configuration fixes or appropriate whitelisting. Also, a suitable whitelist for several error codes should eliminate FPs in the rule *S04D06*.

Although the relatively low FP ratio of these two rules is beneficial, their overall detection capabilities are weak, assuming the attacker would use modern hacking tools and would not make mistakes. On the other hand, the rule *S04D05* offers strong detection potential, with no FPs in theory, but also introduces high implementation overhead, as it requires the deployment and management of scripts on dozens of machines in the environment.

In conclusion, a reliable detection of advanced attacks such as PtH, OPtH, PtT, or forged Kerberos tickets based only on Windows Security events is difficult. However, there are other detection technologies available. *Microsoft ATA* [17, 108] inspects network traffic destining DCs and detects potential suspicious activities based on behavioral anomalies occurring in it.

Correlating data from multiple sources based on different detection mechanisms allows to detect attacks and eliminate FPs more effectively. But still, complete elimination is not possible, and the final decision is left to the process of manual analysis and investigation.

Conclusion

The goal of this thesis was to analyze known attacks targeting Microsoft Active Directory and possibilities of their detection from Windows Security logs. The main task was to develop a set of detection rules, which would be able to detect the analyzed attacks by using Windows Security auditing, together with related documentation. Splunk was chosen as the technology for the implementation. In particular, the rules were implemented as content extension of the application Splunk ES Content Update.

After preliminary acquaintance with the Active Directory technology, possibilities of Windows Security auditing and capabilities of Splunk's Search Processing Language, an analysis of adversary tactics and attack techniques related to Active Directory was performed, with focus on artifacts suitable for detection.

For every identified technique, several detection rules were designed by using various approaches. The designed rules were divided into four Analytic Stories, conforming with the format of Splunk ES Content Update application content. The Analytic Stories were supplied with related Context and Investigative searches and relevant descriptions. The developed rules were tested by performing attacks in the virtual lab environment.

Detection capabilities and the false-positive rate of the designed rules vary. Non-standard approaches that use honeypots or custom scripts for detecting forged Kerberos tickets offer strong detection capabilities with a low false-positive ratio but carry implementation overhead. Also, for many techniques, supplementing Windows Security events with PowerShell logs and Sysmon events, improved visibility and allowed building better detection rules. On the other hand, detection rules for advanced techniques, such as Pass the Hash or Pass the Ticket, based only on Windows Security events, produced a higher number of false positives. Reliable detection of these attacks would require a correlation between the rules and other detection mechanisms.

The content designed in this thesis may serve as a baseline for organizations implementing detection mechanisms for their Active Directory environments. The detection principles used in the searches are not limited to the use of Splunk technology. However, the rules require further adaptation to the requirements of particular environments. Extensive testing in real environments, containing different versions of Windows operating systems, with various configurations applied, is a way to further improve the quality of the designed rules.

Also, there are other places for future development. The designed content does not cover all related techniques, for example, those targeting multi-domain and multi-forest AD instances, as well as it does not involve other phases of the Kill Chain model. Security threats evolve all the time, new attack tools and techniques are being developed. Organizations have to react to it by constantly improving their detection mechanisms and implementing security countermeasures.

Bibliography

1. DESMOND, Brian; RICHARDS, Joe; ALLEN, Robbie; LOWE-NORRIS, Alistair G. Active Directory: Designing, Deploying, and Running Active Directory. In: [online]. 5th ed. O'Reilly Media, 2013, chap. 1-2, 9-10 [visited on 2019-05-12]. ISBN 978-1449320027. Available from: `https://learning.oreilly.com/library/view/active-directory-5th/9781449361211/`.
2. FRANCIS, Dishan. Mastering Active Directory. In: Birmingham: Packt Publishing, 2017, chap. 1-2. ISBN 978-1787289352.
3. METCALF, Sean. *Active Directory Security: How Attackers Dump Active Directory Database Credentials* [online]. © 2011-2017 [visited on 2019-05-12]. Available from: `https://adsecurity.org/?p=2398`.
4. MICROSOFT CORPORATION. Naming conventions in Active Directory for computers, domains, sites, and OUs. *Microsoft Support* [online]. 2017 [visited on 2019-05-12]. Available from: `https://support.microsoft.com/en-gb/help/909264/naming-conventions-in-active-directory-for-computers-domains-sites-and`.
5. MICROSOFT CORPORATION. *Microsoft Docs: Implementing Least-Privilege Administrative Models* [online]. 2018 [visited on 2019-05-12]. Available from: `https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/implementing-least-privilege-administrative-models`.
6. MICROSOFT CORPORATION. *Microsoft Docs: Windows Authentication* [online]. 2016 [visited on 2019-05-12]. Available from: `https://docs.microsoft.com/en-us/windows-server/security/windows-authentication/windows-authentication-overview`.
7. NEUMAN, Clifford; YU, Tom; HARTMAN, Sam; RAEBURN, Ken. *RFC 4120: The Kerberos network authentication service (V5)* [online]. 2005 [visited on 2019-05-12]. Available from: `https://tools.ietf.org/html/rfc4120`. RFC. MIT.

8. MICROSOFT CORPORATION. *[MS-KILE]: Kerberos Protocol Extensions* [online]. 2019 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-kile/2a32282e-dd48-4ad9-a542-609804b02cc9.
9. METCALF, Sean. *Active Directory Security: MS14-068: Vulnerability in (Active Directory) Kerberos Could Allow Elevation of Privilege* [online]. © 2011-2017 [visited on 2019-05-12]. Available from: https://adsecurity.org/?p=525.
10. SOLOMON, David A.; RUSSINOVICH, Mark E.; IONESCU, Alex; YOSIFOVICH, Pavel. Windows Internals, Part 1: System architecture, processes, threads, memory management, and more, Seventh Edition. In: [online]. 7th ed. Microsoft Press, 2017, chap. 7 [visited on 2019-05-12]. ISBN 978-0133986471. Available from: https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/.
11. MICROSOFT CORPORATION. *Microsoft Docs: Group Policy Preferences* [online]. 2016 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/dn581922(v%3Dws.11).
12. METCALF, Sean. *Active Directory Security: Finding Passwords in SYSVOL & Exploiting Group Policy Preferences* [online]. © 2011-2017 [visited on 2019-05-12]. Available from: https://adsecurity.org/?p=2288.
13. HUTCHINS, Eric M.; CLOPPERT, Michael J.; AMIN, Rohan M. *Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains* [online]. 2010 [visited on 2019-05-12]. Available from: https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf. Technical report. Lockheed Martin Corporation.
14. LOCKHEED MARTIN CORPORATION. *Cyber Kill Chain* [online]. © 2019 [visited on 2019-05-12]. Available from: https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html.
15. THE MITRE CORPORATION. *ATT&CK Matrix for Enterprise* [online]. © 2018 [visited on 2019-05-12]. Available from: https://attack.mitre.org/.
16. METCALF, Sean. *Active Directory Security: Attack Methods for Gaining Domain Admin Rights in Active Directory* [online]. © 2011-2017 [visited on 2019-05-12]. Available from: https://adsecurity.org/?p=2362.
17. MICROSOFT CORPORATION. *Microsoft Docs: What is Advanced Threat Analytics?* [online]. 2018 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/advanced-threat-analytics/what-is-ata.
18. METCALF, Sean. *Active Directory Security: Active Directory Recon Without Admin Rights* [online]. © 2011-2017 [visited on 2019-05-12]. Available from: https://adsecurity.org/?p=2535.

19. MICROSOFT CORPORATION. *Microsoft Docs: Account Lockout Policy* [online]. 2018 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/account-lockout-policy.
20. MICROSOFT CORPORATION. *Microsoft Docs: Account lockout threshold* [online]. 2018 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/account-lockout-threshold.
21. THE MITRE CORPORATION. *Techniques: Brute Force* [online]. © 2018 [visited on 2019-05-12]. Available from: https://attack.mitre.org/techniques/T1110/.
22. NETTITUDE. *Brute-LocAdmin.ps1* [software]. GitHub, © 2018 [visited on 2019-05-12]. Available from: https://github.com/nettitude/PoshC2/blob/master/Modules/Brute-LocAdmin.ps1.
23. SHELLNTEL. *Invoke-SMBAutoBrute.ps1* [software]. GitHub, 2016 [visited on 2019-05-12]. Available from: https://github.com/Shellntel/scripts/blob/master/Invoke-SMBAutoBrute.ps1.
24. THE MITRE CORPORATION. *Techniques: Kerberoasting* [online]. © 2018 [visited on 2019-05-12]. Available from: https://attack.mitre.org/techniques/T1208/.
25. METCALF, Sean. *Active Directory Security: Detecting Kerberoasting Activity* [online]. © 2011-2017 [visited on 2019-05-12]. Available from: https://adsecurity.org/?p=3458.
26. MICROSOFT CORPORATION. *Microsoft Docs: Security auditing: 4769(S, F): A Kerberos service ticket was requested.* [online]. 2017 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4769.
27. MICROSOFT CORPORATION. *Microsoft Docs: Network security: Configure encryption types allowed for Kerberos* [online]. 2017 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/network-security-configure-encryption-types-allowed-for-kerberos.
28. SCHROEDER, Will. *Invoke-Kerberoast.ps1* [software]. GitHub, 2016 [visited on 2019-05-12]. Available from: https://github.com/EmpireProject/Empire/blob/master/data/module_source/credentials/Invoke-Kerberoast.ps1.
29. SECUREAUTH CORPORATION. *Impacket: GetUserSPNs.py* [software]. GitHub, 2016 [visited on 2019-05-12]. Available from: https://github.com/SecureAuthCorp/impacket/blob/master/examples/GetUserSPNs.py.
30. THE MITRE CORPORATION. *Techniques: Credential Dumping* [online]. © 2018 [visited on 2019-05-12]. Available from: https://attack.mitre.org/techniques/T1003/.

31. DELPY, Benjamin. *Mimikatz* [software]. GitHub, 2019 [visited on 2019-05-12]. Available from: `https://github.com/gentilkiwi/mimikatz/wiki`.

32. MICROSOFT CORPORATION. *Microsoft Docs: Sysinternals: Windows Sysinternals* [online]. 2017 [visited on 2019-05-12]. Available from: `https://docs.microsoft.com/en-us/sysinternals/`.

33. MICROSOFT CORPORATION. *Microsoft Docs: Microsoft Security Bulletin MS14-025 - Important* [online]. 2014 [visited on 2019-05-12]. Available from: `https://docs.microsoft.com/en-us/security-updates/SecurityBulletins/2014/ms14-025`.

34. THE MITRE CORPORATION. *Techniques: Pass the Hash* [online]. © 2018 [visited on 2019-05-12]. Available from: `https://attack.mitre.org/techniques/T1075/`.

35. OFFENSIVE SECURITY. *PSExec Pass the Hash* [online]. 2019 [visited on 2019-05-12]. Available from: `https://www.offensive-security.com/metasploit-unleashed/psexec-pass-hash/`.

36. EWAIDA, Bashar. *Pass-the-hash attacks: Tools and Mitigation* [online]. 2010 [visited on 2019-05-12]. Available from: `https://www.sans.org/reading-room/whitepapers/testing/paper/33283`. Technical report. SANS Institute.

37. MICROSOFT CORPORATION. *Mitigating Pass-the-Hash and Other Credential Theft, version 2* [online]. © 2014 [visited on 2019-05-12]. Available from: `https://www.microsoft.com/en-us/download/details.aspx?id=36036`.

38. THE MITRE CORPORATION. *Techniques: Pass the Ticket* [online]. © 2018 [visited on 2019-05-12]. Available from: `https://attack.mitre.org/techniques/T1097/`.

39. METCALF, Sean. *Active Directory Security: Mimikatz and Active Directory Kerberos Attacks* [online]. © 2011-2017 [visited on 2019-05-12]. Available from: `https://adsecurity.org/?p=556`.

40. METCALF, Sean. *Active Directory Security: Detecting Forged Kerberos Ticket (Golden Ticket & Silver Ticket) Use in Active Directory* [online]. © 2011-2017 [visited on 2019-05-12]. Available from: `https://adsecurity.org/?p=1515`.

41. MICROSOFT CORPORATION. *Microsoft Docs: Event Logging* [online]. 2018 [visited on 2019-05-12]. Available from: `https://docs.microsoft.com/en-us/windows/desktop/eventlog/event-logging`.

42. MICROSOFT CORPORATION. *Microsoft Docs: PowerShell: Write-EventLog* [online]. 2019 [visited on 2019-05-12]. Available from: `https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/write-eventlog?view=powershell-5.1`.

43. MICROSOFT CORPORATION. *Microsoft Docs: Event Viewer* [online]. 2013 [visited on 2019-05-12]. Available from: `https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc766042(v=ws.11)`.

44. MICROSOFT CORPORATION. *Microsoft Docs: Advanced security auditing FAQ* [online]. 2017 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/advanced-security-auditing-faq.

45. MICROSOFT CORPORATION. *Microsoft Docs: Event Logs* [online]. 2013 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc722404(v=ws.11).

46. MICROSOFT CORPORATION. *Microsoft Docs: Security auditing* [online]. 2017 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/security-auditing-overview.

47. MICROSOFT CORPORATION. *Windows security audit events* [online]. 2016 [visited on 2019-05-12]. Available from: https://www.microsoft.com/en-us/download/details.aspx?id=50034.

48. MICROSOFT CORPORATION. *Microsoft Docs: Advanced security audit policy settings* [online]. 2017 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/advanced-security-audit-policy-settings.

49. MICROSOFT CORPORATION. *Microsoft Docs: Service Control Manager* [online]. 2018 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/desktop/services/service-control-manager.

50. MICROSOFT CORPORATION. *Microsoft Docs: PowerShell: Script Tracing and Logging* [online]. 2017 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/powershell/wmf/5.0/audit_script.

51. KOECHER, Ingmar. From PowerShell to P0W3rH3LL – Auditing PowerShell. *EventSentry Blog* [online]. 2018 [visited on 2019-05-12]. Available from: https://www.eventsentry.com/blog/2018/01/powershell-p0wrh11-securing-powershell.html.

52. MICROSOFT CORPORATION. *Microsoft Docs: Service Control Manager* [online]. 2018 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/desktop/ad/overview-of-change-tracking-techniques.

53. RUSSINOVICH, Mark. *Microsoft TechNet Blog: Mark's Blog: On My Way to Microsoft!* [online]. 2006 [visited on 2019-05-12]. Available from: https://blogs.technet.microsoft.com/markrussinovich/2006/07/18/on-my-way-to-microsoft/.

54. MICROSOFT CORPORATION. *Microsoft Docs: Sysinternals: Sysmon v9.0* [online]. 2017 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon.

55. CHUVAKIN, Anton; SCHMIDT, Kevin; PHILLIPS, Chris. Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management. In: Syngress, 2012, chap. 1. ISBN 978-1597496353.

56. KENT, Karen; SOUPPAYA, Murugiah. *Guide to Computer Security Log Management* [online]. 2006 [visited on 2019-05-12]. Available from: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-92.pdf. Technical report. National Institute of Standards and Technology. Special Publication 800-92.

57. ROBB, Drew. Top 10 SIEM Products. *eSecurity Planet* [online]. 2018 [visited on 2019-05-12]. Available from: https://www.esecurityplanet.com/products/top-siem-products.html.

58. KEARY, Tim. 8 Best SIEM Tools: A Guide to Security Information and Event Management. *Comparitech* [online]. 2019 [visited on 2019-05-12]. Available from: https://www.comparitech.com/net-admin/siem-tools/.

59. LARUE-LANGLOIS, Renaud. 6 Best Security Information and Event Management (SIEM) Tools Worth Checking Out in 2019. *AddictiveTips* [online]. 2019 [visited on 2019-05-12]. Available from: https://www.addictivetips.com/net-admin/siem-tools/.

60. GOLDMAN, Jeff. ArcSight vs Splunk: Top SIEM Solutions Compared. *eSecurity Planet* [online]. 2018 [visited on 2019-05-12]. Available from: https://www.esecurityplanet.com/products/arcsight-splunk-siem-solutions-compared.html.

61. HALE, Brad. *Estimating Log Generation for Security Information Event and Log Management* [online]. 2013 [visited on 2019-05-12]. Available from: http://content.solarwinds.com/creative/pdf/Whitepapers/estimating_log_generation_white_paper.pdf. Technical report. SolarWinds Worldwide, LLC.

62. MICROSOFT CORPORATION. *Microsoft Docs: Audit Policy Recommendations* [online]. 2017 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/audit-policy-recommendations.

63. SMITH, Randy Franklin. Why Workstation Security Logs Are So Important. *Randy's Blog* [online]. © 2012 [visited on 2019-05-12]. Available from: https://www.ultimatewindowssecurity.com/blog/default.aspx?p=de6ccc5f-bb23-4e64-9a2c-9406921a6ae6.

64. CARASSO, David. Exploring Splunk. In: [online]. New York: CITO Research, 2012, chap. 2-4 [visited on 2019-05-12]. ISBN 978-0-9825506-7-0. Available from: https://www.splunk.com/pdfs/exploring-splunk.pdf.

65. SPLUNK INC. *Distributed Deployment Manual* [online]. © 2005-2019 [visited on 2019-05-12]. Available from: https://docs.splunk.com/Documentation/Splunk/7.2.6/Deploy/Distributedoverview.

66. SPLUNK INC. *Splexicon: Forwarder* [online]. © 2005-2019 [visited on 2019-05-12]. Available from: https://docs.splunk.com/Splexicon:Forwarder.

67. SPLUNK INC. *Splunk Enterprise: Managing Indexers and Clusters of Indexers* [online]. © 2005-2019 [visited on 2019-05-12]. Available from: https://docs.splunk.com/Documentation/Splunk/7.2.6/Indexer/Aboutindexesandindexers.

68. SPLUNK INC. *Splunk Enterprise Admin Manual: Apps and add-ons* [online]. © 2005-2019 [visited on 2019-05-12]. Available from: `https://docs.splunk.com/Documentation/Splunk/7.2.6/Admin/Whatsanapp`.

69. SPLUNK INC. *Splunkbase: Splunk Enterprise Security* [online]. © 2005-2019 [visited on 2019-05-12]. Available from: `https://splunkbase.splunk.com/app/263/`.

70. SPLUNK INC. *Splunkbase: Splunk Add-on for Microsoft Windows* [online]. © 2005-2019 [visited on 2019-05-12]. Available from: `https://splunkbase.splunk.com/app/742/`.

71. SPLUNK INC. *Splunk Platform Comparison* [online]. © 2005-2019 [visited on 2019-05-12]. Available from: `https://www.splunk.com/en_us/software/features-comparison-chart.html`.

72. SPLUNK INC. *Developer Trial License* [online]. © 2005-2019 [visited on 2019-05-12]. Available from: `https://splunkbase.splunk.com/develop/`.

73. SPLUNK INC. *Splunk Docs* [online]. © 2005-2019 [visited on 2019-05-12]. Available from: `https://docs.splunk.com/Documentation`.

74. SPLUNK INC. *Splunkbase: Splunk ES Content Update* [online]. © 2005-2019 [visited on 2019-05-12]. Available from: `https://splunkbase.splunk.com/app/3449/`.

75. CENTER FOR INTERNET SECURITY. *CIS Controls* [online]. © 2019 [visited on 2019-05-12]. Available from: `https://www.cisecurity.org/controls/cis-controls-list/`.

76. SPLUNK INC. *Splunkbase: Splunk Add-on for Microsoft Active Directory* [online]. © 2005-2019 [visited on 2019-05-12]. Available from: `https://splunkbase.splunk.com/app/3207/`.

77. SPLUNK INC. *Splunkbase: Add-on for Microsoft Sysmon* [online]. © 2005-2019 [visited on 2019-05-12]. Available from: `https://splunkbase.splunk.com/app/1914/`.

78. SPLUNK INC. *Splunk Universal Forwarder: Forwarder Manual* [online]. © 2005-2019 [visited on 2019-05-12]. Available from: `https://docs.splunk.com/Documentation/Forwarder/7.1.3/Forwarder/InstallaWindowsuniversalforwarderfromaninstaller`.

79. METCALF, Sean. *Active Directory Security: The Most Common Active Directory Security Issues and What You Can Do to Fix Them* [online]. © 2011-2017 [visited on 2019-05-12]. Available from: `https://adsecurity.org/?p=1684`.

80. SMITH, Randy Franklin. *Security Log Encyclopedia: Windows Security Log Events* [online] [visited on 2019-05-12]. Available from: `https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/default.aspx`.

81. MICROSOFT CORPORATION. *Microsoft Docs: Security auditing: Audit Credential Validation* [online]. 2017 [visited on 2019-05-12]. Available from: `https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/audit-credential-validation`.

82. MICROSOFT CORPORATION. *Microsoft Docs: Security auditing: Audit Kerberos Authentication Service* [online]. 2017 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/audit-kerberos-authentication-service.

83. MICROSOFT CORPORATION. *Microsoft Docs: Security auditing: Audit Other Account Logon Events* [online]. 2017 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/audit-other-account-logon-events.

84. MICROSOFT CORPORATION. *Microsoft Docs: Security auditing: Audit Account Lockout* [online]. 2018 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/audit-account-lockout.

85. MICROSOFT CORPORATION. *Microsoft Docs: Security auditing: Audit Logoff* [online]. 2018 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/audit-logoff.

86. MICROSOFT CORPORATION. *Microsoft Docs: Security auditing: Audit Logon* [online]. 2017 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/audit-logon.

87. MICROSOFT CORPORATION. *Microsoft Docs: Security auditing: Audit Other Logon/Logoff Events* [online]. 2017 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/audit-other-logonlogoff-events.

88. MICROSOFT CORPORATION. *Microsoft Docs: Security auditing: Audit Special Logon* [online]. 2017 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/audit-special-logon.

89. MARGOSIS, Aaron. *Microsoft TechNet Blog: Microsoft Security Guidance blog: Configuring Account Lockout* [online]. 2014 [visited on 2019-05-12]. Available from: https://blogs.technet.microsoft.com/secguide/2014/08/13/configuring-account-lockout/.

90. MICROSOFT CORPORATION. *Microsoft Docs: Security auditing: 4740(S): A user account was locked out.* [online]. 2017 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4740.

91. MICROSOFT CORPORATION. *Microsoft Docs: Security auditing: 4768(S, F): A Kerberos authentication ticket (TGT) was requested.* [online]. 2017 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4768.

92. MICROSOFT CORPORATION. *Microsoft Docs: Security auditing: Audit Kerberos Service Ticket Operations* [online]. 2017 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/audit-kerberos-service-ticket-operations.

93. METCALF, Sean. *Active Directory Security: About* [online]. © 2011-2017 [visited on 2019-05-12]. Available from: https://adsecurity.org/?page_id=8.

94. METCALF, Sean. *Active Directory Security: Detecting Kerberoasting Activity Part 2 – Creating a Kerberoast Service Account Honeypot* [online]. © 2011-2017 [visited on 2019-05-12]. Available from: https://adsecurity.org/?p=3513.

95. KHEIRKHABAROV, Teymur. *Zero Nights 2017: Hunting for Credentials Dumping in Windows Environment* [online]. 2017 [visited on 2019-05-12]. Available from: https://www.slideshare.net/heirhabarov/hunting-for-credentials-dumping-in-windows-environment.

96. MICROSOFT CORPORATION. *Microsoft Docs: Process Security and Access Rights* [online]. 2018 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows/desktop/procthread/process-security-and-access-rights.

97. MICROSOFT CORPORATION. *Microsoft Docs: Vssadmin* [online]. 2018 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/vssadmin.

98. MICROSOFT CORPORATION. *Windows Dev Center: Win32_ShadowCopy class* [online]. 2019 [visited on 2019-05-12]. Available from: https://msdn.microsoft.com/en-us/library/windows/desktop/aa394428(v=vs.85).aspx.

99. MICROSOFT CORPORATION. *[MS-ADTS]: Active Directory Technical Specification: 5.1.3.2.1 Control Access Rights* [online]. 2019 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-adts/1522b774-6464-41a3-87a5-1e5633c3fbbb.

100. METCALF, Sean. *Active Directory Security: Mimikatz DCSync Usage, Exploitation, and Detection* [online]. © 2011-2017 [visited on 2019-05-12]. Available from: https://adsecurity.org/?p=1729.

101. SORIA-MACHADO, Miguel; ABOLINS, Didzis; BOLDEA, Ciprian; SOCHA, Krzysztof. Detecting Lateral Movements in Windows Infrastructure. *CERT-EU Security Whitepaper 17–002* [online]. 2017 [visited on 2019-05-12]. Available from: https://cert.europa.eu/static/WhitePapers/CERT-EU_SWP_17-002_Lateral_Movements.pdf.

102. GERZI, Eviatar. Detecting Pass-The-Hash with Windows Event Viewer. *Threat Research Blog Post* [online]. 2017 [visited on 2019-05-12]. Available from: https://www.cyberark.com/threat-research-blog/detecting-pass-the-hash-with-windows-event-viewer/.

103. SORIA-MACHADO, Miguel; ABOLINS, Didzis; BOLDEA, Ciprian; SOCHA, Krzysztof. Kerberos Golden Ticket Protection: Mitigating Pass-the-Ticket on Active Directory. *CERT-EU Security Whitepaper 2014–007* [online]. 2016 [visited on 2019-05-12]. Available from: http://cert.europa.eu/static/WhitePapers/UPDATED%20-%20CERT-EU_Security_Whitepaper_2014-007_Kerberos_Golden_Ticket_Protection_v1_4.pdf.

104. O'HARA, Sean. *The ThreatHunting Project: Finding Golden and Silver Tickets* [software]. GitHub, 2016 [visited on 2019-05-12]. Available from: https://github.com/ThreatHuntingProject/ThreatHunting/blob/master/hunts/golden_ticket.md.

105. MICROSOFT CORPORATION. *Microsoft Docs: PowerShell: ConvertTo-Json* [online]. 2019 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/convertto-json?view=powershell-3.0.

106. MICROSOFT CORPORATION. *Microsoft Docs: Microsoft Security Bulletin MS14-068 - Critical* [online]. 2014 [visited on 2019-05-12]. Available from: https://docs.microsoft.com/en-us/security-updates/securitybulletins/2014/ms14-068.

107. MONNÉ, Sylvain. *Python Kerberos Exploitation Kit* [software]. GitHub, 2017 [visited on 2019-05-12]. Available from: https://github.com/SecWiki/windows-kernel-exploits/tree/master/MS14-068/pykek.

108. METCALF, Sean. *Active Directory Security: Microsoft Advanced Threat Analytics (ATA) Overview* [online]. © 2011-2017 [visited on 2019-05-12]. Available from: https://adsecurity.org/?p=1583.

Appendix A

Acronyms

ACE Access Control Entry.

AD Active Directory.

AD DS Active Directory Domain Services.

API Application Programming Interface.

CIM Common Information Model.

DC domain controller.

DN Distinguished Name.

DNS Domain Name System.

DoS denial of service.

DRS Directory Replication Service.

ES Enterprise Security.

ESCU Splunk ES Content Update.

ESE Extensible Storage Engine.

FP false positive.

FQDN fully qualified domain name.

FSMO Flexible Single Master Operator.

GPP Group Policy Preferences.

GUI	graphical user interface.
GUID	Globally Unique Identifier.
IDS	intrusion detection system.
KDC	Kerberos Key Distribution Center.
LDAP	Lightweight Directory Access Protocol.
LSA	Local Security Authority.
LSASS	Local Security Authority Subsystem Service.
MMC	Microsoft Management Console.
NOS	Network Operating System.
NTLM	NT LAN Manager.
OPtH	Overpass the Hash.
OS	operating system.
OU	Organizational Unit.
PAC	Privilege Account Certificate.
PDC	Primary Domain Controller.
PtH	Pass the Hash.
PtT	Pass the Ticket.
RDN	Relative Distinguished Name.
RDP	Remote Desktop Protocol.
RID	Relative Identifier.
SAM	Security Account Manager.
SID	Security Identifier.
SIEM	Security Information and Event Management.
SOC	Security Operations Center.
SPL	Search Processing Language.

SPN Service Principal Name.

SQL Standard Query Language.

TA Technology Add-on.

TGS Ticket Granting Service.

TGT Ticket Granting Ticket.

TP true positive.

TTPs Tactics, Techniques, and Procedures.

VM virtual machine.

VSS Volume Shadow Copy Service.

WMI Windows Management Instrumentation.

Appendix **B**

Contents of enclosed CD

```
README.txt ....................................... the file with CD contents description
├── app ................................ the directory with Splunk ES Content Update Add-on
│   ├── ESCU.zip ............................... zip archive with the Add-on source files
│   ├── screenshots .................... the directory with screenshots from the Add-on
│   └── ESCU_README.txt .................... the file containing installation instructions
├── configuration ............................ the directory with auditing configuration files
│   ├── config_auditing.csv ......... CSV file with Advanced Audit Policy configuration
│   └── config_sysmon.xml ......................... XML file with Sysmon configuration
├── scripts .................................................. the directory with scripts
│   └── HuntGoldenTicket.ps1 .................. PowerShell script for the search S04D04
├── stories ............... the directory with exported Analytic Stories in PDF format
└── thesis ....................................................... the thesis text directory
    ├── BP_Kotlaba_Lukas_2019.pdf ....................... the thesis text in PDF format
    └── BP_Kotlaba_Lukas_2019.zip ...... zip archive with LaTeX source files of the thesis
```