



Assignment of bachelor's thesis

Title:	Unauthorized communication detection in modern application firewalls
Student:	Lukáš Hrdonka
Supervisor:	Ing. Josef Kokeš, Ph.D.
Study program:	Informatics
Branch / specialization:	Information Security 2021
Department:	Department of Information Security
Validity:	until the end of summer semester 2024/2025

Instructions

- 1) Describe basic security concepts of computer networks, particularly those based on the TCP/IP protocol.
- 2) Analyze various firewall types. Focus particularly on the application firewalls, i.e. those that apply rules to individual applications rather than protocols and ports. Research common evasion techniques for these firewalls.
- 3) Using a set of application firewalls of your choice (approved by the supervisor), design and implement a set of tests that would demonstrate the firewalls' ability (or inability) to detect and prevent unauthorized network access.
- 4) Evaluate and discuss the results received from these tests. Propose recommendations for the firewall setup that would maximize the security against unwanted communications.

Bachelor's thesis

**UNAUTHORIZED
COMMUNICATION
DETECTION IN MODERN
APPLICATION
FIREWALLS**

Lukáš Hrdonka

Faculty of Information Technology
Department of Information Security
Supervisor: Ing. Josef Kokeš, Ph.D.
May 12, 2024

Czech Technical University in Prague

Faculty of Information Technology

© 2024 Lukáš Hrdonka. All rights reserved.

This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis: Hrdonka Lukáš. *Unauthorized communication detection in modern application firewalls*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2024.

Contents

Acknowledgments	vii
Declaration	viii
Abstract	ix
List of abbreviations	x
Introduction	1
1 Traffic Filtering Overview	3
1.1 Network Access Layer	3
1.1.1 MAC Address	5
1.1.2 VLANs	5
1.1.3 Frame Check Sequence	5
1.2 Internet Layer	6
1.2.1 IP Address	6
1.2.2 IP Address Filtering	6
1.2.3 Header Checksum	7
1.3 Transport Layer	7
1.3.1 Transport Layer Protocols	7
1.3.2 Port Number	8
1.3.3 Port Number Filtering	8
1.4 Application Layer	9
1.4.1 Protocol HTTP	9
1.4.2 Filtering Options	11
2 Application Firewall	13
2.1 Phases of Attack	13
2.1.1 Cyber Kill Chain	13
2.2 Principle of an Application Firewall	16
2.2.1 Trusted Application Definition	16
2.2.2 Trusted Application Verification	17
2.3 Common Evasion Techniques	17
2.4 Popular products	20
2.4.1 ESET Internet Security	20
2.4.2 Avast Free Antivirus	21
2.4.3 Windows Firewall	22
2.4.4 TinyWall	23
2.4.5 ZoneAlarm Firewall	25

3	Testing Environment Description	27
3.1	Operating System and Installed Programms	27
3.1.1	Victim	27
3.1.2	Attacker	27
3.2	Application Firewalls	28
4	Substitution	31
4.1	Attack Description	31
4.1.1	Executable's Name and Path	31
4.1.2	Digital Certificates	32
4.1.3	Attack Options	38
4.2	ESET Internet Security	39
4.3	Avast Free Antivirus	43
4.4	Windows Firewall	45
4.5	TinyWall	47
4.6	ZoneAlarm Firewall	49
5	Injection	53
5.1	Attack Description	53
5.1.1	Code Explanation	57
5.1.2	Practical Demonstration	59
5.2	ESET Internet Security	61
5.3	Avast Free Antivirus	63
5.4	Windows Firewall	65
5.5	TinyWall	66
5.6	ZoneAlarm Firewall	67
6	Discussion of Results	69
6.1	Substitution	69
6.2	Injection	70
6.3	Discussion	70
6.4	Vendors Reaction	72
7	Conclusion	75
	Contents of the Attachment	83

List of Figures

1.1	Filtering Options Overview	4
1.2	Comparson of TLSv1.2 and TLSv1.3 Handshake	11
2.1	Cyber Kill Chain Model with Offensive and Defensive Actions	14
2.2	Common Principle of the Attacks	18
4.1	Substitution Attacks Principle	32
4.2	Digital Signature of OneDrive Executable	35
4.3	OneDrive Certificate Chain	36
4.4	Export Certificate to PFX Format	37
4.5	Additional Parameters of Certificate Export	38
4.6	Executable Signed By Untrusted Organization	39
4.7	Rule for Trusted OneDrive Application in ESET Internet Security	41
4.8	Warning about Application Modification in ESET Internet Security	41
4.9	Rule for Untrusted OneDrive Application in ESET Internet Security	42
4.10	Application Excluded from Modification Detection in ESET Internet Security	42
4.11	Rule for Trusted OneDrive Application in Avast Free Antivirus	43
4.12	Details of Default Rule for Trusted OneDrive Application in Avast Free Antivirus	43
4.13	Rule for Untrusted OneDrive Application in Avast Free Antivirus	44
4.14	Untrusted Certificate Detection by Avast Free Antivirus	45
4.15	Default Settings of Windows Firewall	46
4.16	General Tab of Rule for Trusted OneDrive Application in Windows Firewall	46
4.17	Details of Rule for Trusted OneDrive Application in Windows Firewall	47
4.18	Rule for Trusted OneDrive Application in TinyWall	48
4.19	Details of Rule for Trusted OneDrive Application in TinyWall	49
4.20	Rule for Untrusted OneDrive Application in TinyWall	49
4.21	Rule for Trusted OneDrive Application Definition in ZoneAlarm	50
4.22	Rule for Untrusted OneDrive Application Definition in ZoneAlarm	51
5.1	Injection Attacks Principle	54
5.2	Legitimate Communication of Windows Explorer	55
5.3	Information about IP Address 52.109.28.46 in Shodan	56
5.4	Certificate of Signed DLL	57
5.5	Launching the Injection Attack and Verification of the Result in the Task Manager	59
5.6	Listing of Linked DLLs in the Process Explorer	60
5.7	Communication Captured by Wireshark	60
5.8	Communication of Trusted OneDrive Application in ESET Internet Security	61
5.9	Communication of Injected OneDrive Application in ESET Internet Security	62
5.10	Exception Details in Avast Free Antivirus	64
5.11	Attack Detection by Avast Free Antivirus	64
5.12	Configuration of Code Integrity Guard In Windows Security Center	66
5.13	List of Active Connection in TinyWall	67
5.14	Sensitivity Level Settings in ZoneAlarm Firewall	67

List of Tables

1.1	Port Number Classification	8
1.2	Selected Well-known Ports	9
3.1	Popular Application Firewalls	29
4.1	Common Attributes of Distinguished Name	33
4.2	Executables Used for Substitution Attack	39
4.3	Terminology Used during the Testing	40
6.1	Firewalls' Ability to Detect Substitution Attacks	73
6.2	Firewalls' Ability to Detect Injection Attacks	73
6.3	Firewall Vendors Reactions	73

List of Listings

1.1	Mutliple IP Addresses for Single Domain	7
4.1	Create New Self-signed Certificate	37
4.2	Sign Executable File Using Visual Studio Tools	38
5.1	Process Injection	57

I want to thank my thesis supervisor, Ing. Josef Kokeš, Ph.D., for the invaluable guidance, patience, and help provided while writing the thesis. I also want to thank my family and friends for supporting me not only during my studies.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Czech Technical University in Prague has the right to conclude a licence agreement on the utilization of this thesis as a school work pursuant of Section 60 (1) of the Act.

In Prague on May 12, 2024

Abstract

The thesis provides an analysis of vulnerabilities in modern application firewalls.

In the research part, the various ways of traffic filtering are introduced. The common principles behind the application firewall and the most commonly used evasion techniques are described, too.

The practical part shows the implementation of substitution and injection attacks written in C++ programming language using standard Microsoft Windows API functions. The tests of the substitution and injection attacks are performed against five application firewalls and the results are then discussed.

The main findings are that the application firewalls can detect substitution attacks as expected, although the injection attack remains undetected in most application firewalls.

Keywords cybersecurity, penetration test, malware, network traffic filtering, application firewall, firewall leak test, code injection, Microsoft Windows, DLL, C++

Abstrakt

Tato práce se zabývá analýzou zranitelností moderních aplikačních firewallů.

V teoretické části jsou představeny možnosti filtrace síťového provozu, klíčové principy aplikačního firewallu a nejčastěji používané techniky pro testování jeho bezpečnosti.

Praktická část navazuje implementací útoků typu substitution a injection za využití standardních API funkcí operačního systému Microsoft Windows. Tyto ukázky jsou implementovány v programovacím jazyce C++. Vytvořené programy jsou dále testovány za využití celkem pěti aplikačních firewallů. Výsledky testů jsou diskutovány v závěru práce.

Hlavním zjištěním v rámci této práce je, že aplikační firewally se chovají v souladu s předpokládanou funkcionalitou při detekci útoků typu substitution. Většina aplikačních firewallů však není schopna detekovat útoky typu injection.

Klíčová slova kybernetická bezpečnost, penetrační testování, malware, filtrování síťového provozu, aplikační firewall, firewall leak test, code injection, Microsoft Windows, DLL, C++

List of abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
ARPANET	Advanced Research Projects Agency Network
C&C	Command and Control
CA	Certificate Authority
DARPA	Defense Advanced Research Projects Agency
DDE	Dynamic Data Exchange
DHCP	Dynamic Host Configuration Protocol
DiD	Defense in Depth
DLL	Dynamic Link Library
DLP	Data Leakage Prevention
DNS	Domain Name System
FCS	Frame Check Sequence
FP	False Positive
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ISG	Intelligent Security Graph
ISO/OSI	International Organization for Standardization / Open Systems Interconnection
MITM	Man-in-the-Middle Attack
ML	Machine Learning
MSRC	Microsoft Security Response Center
NAT	Network Address Translation
NTP	Network Time Protocol
OS	Operating System
PID	Process Identifier
PKI	Public Key Infrastructure
PoC	Proof of Concept
QUIC	Quick UDP Internet Connections
RAM	Random Access Memory
RDP	Remote Desktop Protocol
RFC	Request For Comments
SCTP	Stream Control Transmission Protocol
SOHO	Small Office Home Office Network
SSH	Secure Shell
TCP/IP	Transmission Control Protocol / Internet Protocol
Telnet	Teletype Network
TLS	Transport Layer Security
UDP	User Datagram Protocol
URL	Uniform Resource Locator
UWP	Universal Windows Platform
VLAN	Virtual Local Area Network
WDAC	Windows Defender Application Control
WSUS	Windows Server Update Services

Introduction

With the growth in the usage of computers in recent years, the security of these end devices has to be taken into account. In rough numbers, billions of malware instances were reported during the last years [1]. Based on that number, one can see that a proper detection system is needed in order to detect attacks and perform proper defense against them. However, proper detection of the attack is extremely complicated. As written in IBM's report for 2023 [2], the average time needed to detect an attack was 277 days (almost 9 months).

Attackers often use vulnerabilities in a system to deliver and run their malware. The delivery method may depend on various factors. For instance, attackers can abuse improper detection mechanisms or use some form of social engineering. When the malware is installed on a device, it usually needs to communicate with the attacker. This communication can be inspected by application firewalls that apply rules to specific applications and thus allow traffic filtering based on the origin of the application. From a practical point of view, users can specify which applications are allowed to communicate with other devices in the network (or the internet in general) and which ones are blocked from communication. The mentioned type of firewalls is often used as a part of complex antimalware products.

This thesis shows common attacks used to bypass rules defined in the application firewalls, with demonstrations performed on solutions used nowadays. A common concept of attacks is to abuse a trusted application (i.e., one that is allowed to communicate) to hide malicious communication.

It should be mentioned that these types of attacks were initially described in the first decade of this century. However, for more than 15 years, only a minimal amount of information regarding this type of attack has been found. Thus, one of the thesis's goals is to analyze various approaches to firewall leak testing and verify whether modern application firewalls are capable of detecting the described attacks.

Selected techniques are demonstrated in 5 application firewalls. It was decided to test the application firewall delivered within antimalware by ESET company as it is commonly used software in the industry, then Avast as a representative of free and popular antimalware for home environments and Windows Defender because it is delivered as a part of the Microsoft Windows operating system. As representatives of dedicated application firewalls, TinyWall and ZoneAlarm Firewall were used.

All tests are performed using the default configuration of the mentioned application firewalls as it is expected that the vast majority of the users have only a limited understanding of the configuration of the application firewall, antimalware, or even cybersecurity in general. Based on that, the thesis shows how these implementations protect regular users against selected attacks.

It is worth mentioning that this thesis focuses primarily on application firewalls (even though they are often part of the complex antimalware program). Thus, the outcomes presented at the end of the thesis inform about the ability of the actual application firewalls to detect potentially malicious communication. The thesis does not solve the problem of delivering these executables to the victim. It is thus possible that the executable files provided in the attachment would be detected upon their delivery or while extracting the archive.

Traffic Filtering Overview

As outlined in the Introduction part of the thesis, various ways of traffic filtering exist. This chapter provides a classification of the filtering options based on the TCP/IP network stack and thus presents how data can be filtered on different layers of TCP/IP.

At the beginning of this chapter, the difference between commonly used network protocols TCP/IP and ISO/OSI is described while explaining why the decision to map filtering options to the first one was made. [3, 4]

The OSI protocol stack was initially created to describe the concept of computer networks and divide their functionality into seven logical layers. As the ISO organization designed it, the protocol became an international standard. Its main advantage is the universality. However, over time, it became clear that this protocol is unsuitable for practical applications as it is too complicated and sometimes one service is spread between more than one layer.

Based on the needs of the ARPANET, the TCP/IP was designed by DARPA. The main difference between these protocols is that the TCP/IP has only four layers (whereas ISO/OSI has seven layers). Another advantage of the TCP/IP could be great scalability or the support for various routing protocols. [5]

In summary, the ISO/OSI remained a theoretical (often called reference) model, whereas TCP/IP became the most used protocol in current networks. Thus, TCP/IP became a standard of modern computer networks and is used as a base in this document.

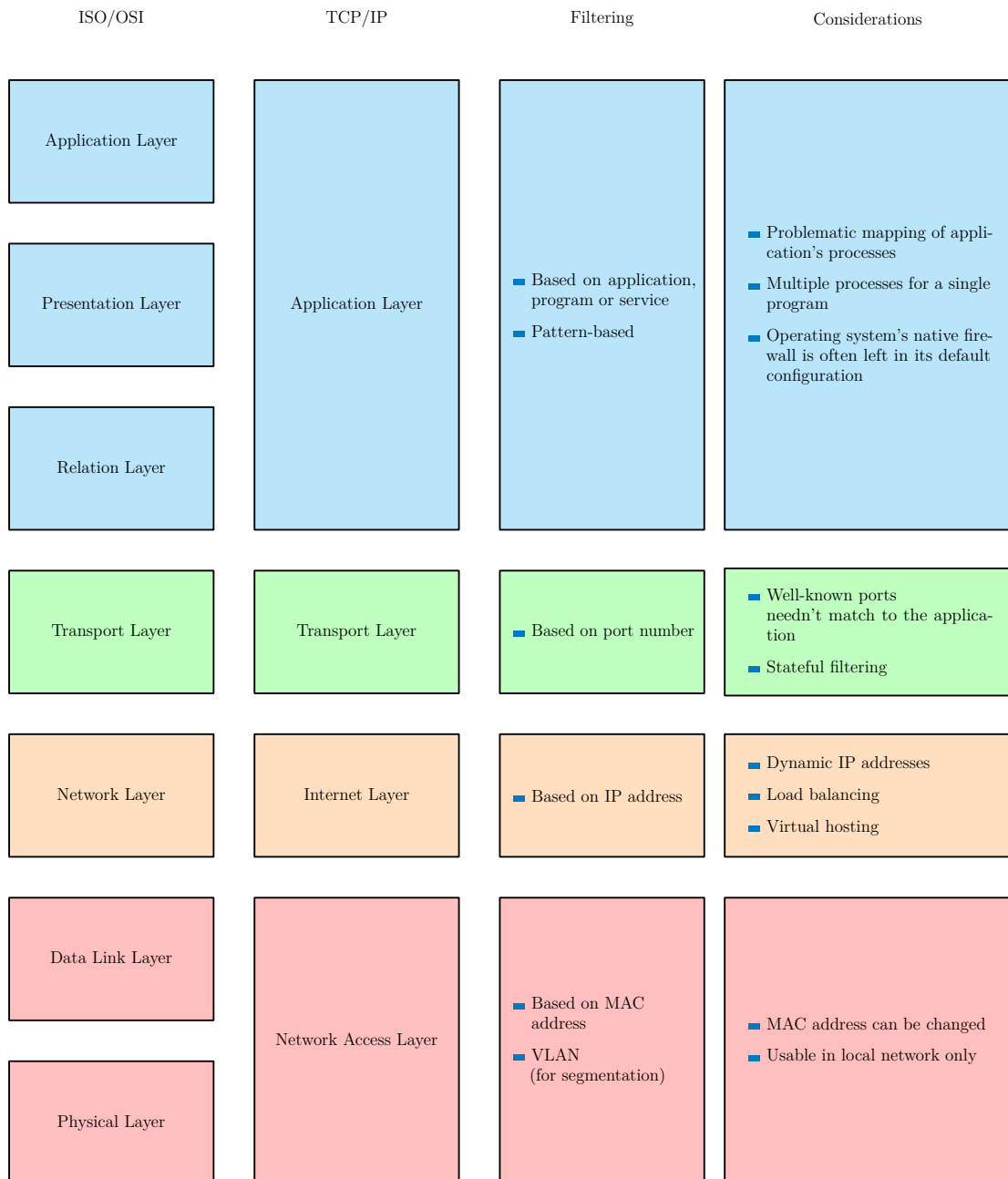
Figure 1.1 shows common differences between the ISO/OSI and TCP/IP models, as well as filtering options on their layers. This figure is used only as an overview as the concepts are explained in more detail later in the chapter. [6]

1.1 Network Access Layer

The Network Access layer is the lowest layer of the TCP/IP model. Encapsulated data on this layer are called Ethernet frames (or sometimes simply frames).

The most important parts of the frame's header¹ are the MAC addresses. Similar to the further layers described in this chapter, there are two types of addresses: destination and source. However, there is a crucial difference between the other layers – the destination MAC address precedes the source MAC address. This solution was chosen to speed up the process of switching.

¹of course, from the filtering's point of view



■ **Figure 1.1** Filtering Options Overview [3]

1.1.1 MAC Address

MAC address consists of 48 bits. While the first 24 bits are used to identify the vendor, the other 24 bits are used as a serial number. Based on that, the MAC address clearly identifies a device on the local network. As a device can be identified this way, filtering based on the MAC address is intended to be used.

This solution became extremely handy, especially in larger LANs like corporate or campus networks. However, one can also use it in smaller networks². These networks are often designed so that users connect their devices to the network switch. In such a design, it is very beneficial to determine devices that could be connected to the switch. If this limitation is not applied, every device can communicate within the network, which is valuable for potential attackers.

Filtering of unknown devices can be based on MAC addresses. The ideal way would be to define a set of trusted MAC addresses and deny access from any other. However, the mentioned solution has two significant problems.

First, it can be nearly impossible to maintain such a set of allowed MAC addresses in large environments, especially in combination with the BYOD³ (Bring Your Own Device) principle. Some switch vendors thus implement the option to allow only a limited number of MAC addresses on physical switch ports, while ones beyond this threshold will be prevented from communication. This principle is often called switch port security based on the number of MAC addresses and is commonly used to prevent MAC flooding attacks. [7]

The second problem is that the attacker can change the MAC address of her computer and thus bypass the rules. On the other hand (and with proper configuration), this attack can be successful only if the attacker knows the allowed MAC address.

1.1.2 VLANs

When writing about the security on the Network Access layer, the concept of VLANs should also be mentioned, although it is not a filtering option in the original way. [8]

VLAN can be defined as a logical partitioning of LAN on the Network Access layer of TCP/IP. This concept is typically used to split different logical parts of the organizations, such as company departments. Thanks to this, it is also possible to manage communication between the logical parts⁴. It is usually done by adding an 802.1Q tag to the frame header.

Based on that, it is commonly used for segmenting logical parts within the organizations (various company departments, for instance) and managing communication between them.

It is especially beneficial from the view of network design as the broadcast domain is reduced. In the adequately configured network, it implies that the attacker has more limited attack options. Consequently, VLANs have the potential to make the lateral movement more challenging.

1.1.3 Frame Check Sequence

Frame Check Sequence (FCS) is a value calculated from the entire frame and placed at its end. The CRC-32 checksum algorithm computes the FCS value which is an error detection code based on the polynomial division in the Galois field.

From the above mentioned, it is clear that it cannot be used for checking the integrity of the message. For a better explanation, let us consider a message that an attacker intercepted. The attacker can then change the content of the message. Because CRC-32 is a publicly known algorithm, the new message could be created so that the FCS value remains the same as in the original message. Based on that, the FCS can only be used for error detection during transmission (caused by some interference on the medium, for instance).

²so-called SOHO networks

³users bring their own devices and use them in the corporate environment, for instance

⁴limit the communication between them, for instance

1.2 Internet Layer

From the information mentioned above, the Network Access layer is intended to connect the devices to the local network and then identify these devices based on their MAC address. A motivation for using the Internet layer of TCP/IP is to group those devices into an IP network and perform routing⁵.

1.2.1 IP Address

The Internet layer uses the IP address to identify the device. Combined with a subnet mask⁶, it forms a broadcast domain. A similar approach is impossible with MAC addresses alone as the first part of the MAC address identifies the vendor. However, in one network, devices made by various manufacturers could be used.

There are two commonly used types of IP addresses in today's networks – IPv4 and IPv6. IPv4 was described in RFC 791 (dated September 1981) and consists of 32 bits. It means that there are 2^{32} possible IPv4 addresses (in reality, this number is slightly less as some of them cannot be used because they are reserved for specific usage). With the massive growth of IoT in the recent years, it became clear that more IP addresses were needed. [9]

There are at least two possible solutions to this problem. The first solution is to use modern IPv6 addresses. The significant difference to IPv4 is that their length is 128 bits. Thus, there are 2^{128} possible IPv6 addresses, at least in theory. However, as this was a new type of address, some routers did not understand them in the past. The second solution is to continue using IPv4 and implement NAT. Its principle is to create a set of private and public IPv4 addresses and perform mapping between them. Whereas private IPv4 addresses can be used only in local networks, public IPv4 addresses are primarily used for routing purposes. [10]

1.2.2 IP Address Filtering

Filtering on the Internet layer can be done by filtering the packet's source or destination IP address. This type of firewall is often deployed on the border of a local network. This approach is called packet filter or stateless firewall. A typical representative is Cisco's standard ACL (Access Control List).

However, IP address filtering has limitations, too. These are introduced below with a demonstration of HTTP and DNS protocols.

- Dynamic IP address. The IP address of the origin can be changed. Then the administrator has to reconfigure the firewall. This can be difficult if a long list of firewall entries exists.
- Load balancing. In load balancing, the requested service is deployed on more than one physical device⁷. For instance, let us have a user who tries to access the website `www.microsoft.com`. According to the DNS response listed in Code listing 1.1, five different IPv4 addresses can be used for communication. Firewalls must be configured so that all communication is handled in the same way.
- Virtual hosting. It is a popular principle used by web hosting where multiple websites use one public IP address. The actual website is determined by port number, or more often by the `Host:` header used in HTTP protocol. From the filtering perspective, preventing one IP address from communicating can consequently affect multiple domains.

⁵i.e., finding the best path to the destination

⁶divides the IP address to network and host parts

⁷often deployed in different geographical locations

```
1 $ $ nslookup -type=A microsoft.com
2 Server:          147.32.88.4
3 Address:         147.32.88.4#53
4
5 Non-authoritative answer:
6 Name:   microsoft.com
7 Address: 20.76.201.171
8 Name:   microsoft.com
9 Address: 20.112.250.133
10 Name:  microsoft.com
11 Address: 20.231.239.246
12 Name:  microsoft.com
13 Address: 20.236.44.162
14 Name:  microsoft.com
15 Address: 20.70.246.20
```

■ **Code listing 1.1** Multiple IP Addresses for Single Domain

1.2.3 Header Checksum

Similarly to the Network Access layer, there is a checksum that can be used for error detection in the header. It is computed as a 2's complement sum of all 16-bit words in the header⁸. When the packet arrives at another device, it can calculate the checksum in the same way and then compare the results.

As this is only a sum, its value can be easily changed. Thus, it does not bring any additional security.

1.3 Transport Layer

The previous sections described how the device can be identified in the local network based on its MAC address, as well as the concept of IP addresses. The motivation behind the Transport layer is to identify specific applications (or sometimes called services) running on the particular device.

Without this mechanism, only one application could communicate simultaneously, which would be extremely slow and inefficient.

1.3.1 Transport Layer Protocols

Historically, two main protocols have been used on this layer – TCP and UDP. In modern applications, SCTP or QUIC⁹ can also be used. [11]

Protocol TCP is connection-based and thus brings reliability into the connection. Before the actual data are sent over the network, there is a so-called 3-way handshake intended to establish a channel between the source and destination. After the channel is established, the data are divided into parts and sent over the network. Those parts of data are tagged by their sequence number, which is used to acknowledge that this part successfully arrived at the destination and to present the parts in the correct order¹⁰. When all the data are transferred, the channel

⁸i.e., the header is divided into parts of 16 bits and then summed

⁹QUIC is included in this section, although sometimes it is called an extension of UDP and included in the set of Application layer protocols

¹⁰as parts might be delivered in a different order

between the source and destination is closed using a so-called 4-way handshake.

On the other hand, UDP is a connectionless protocol, which, rather than creating a channel¹¹, just sends the data over the network. This protocol does not care whether the data arrives at the destination at all or in which order. Typical usage of this protocol is live-streaming or voice conferences.

Protocol SCTP was designed to support the simultaneous parallel transfer of multiple independent flows. The core principle of SCTP is UDP, but data can be acknowledged. This protocol is commonly used in telecommunications, especially in 5G networks.

As SCTP was designed for telecommunication networks, protocol QUIC was created to improve the efficiency of HTTP communications. It is a relatively new protocol which was described in RFC 9000 only a few years ago. A significant built-in mechanism of QUIC is connection migration. This means that the device can change the IP address¹², which does not affect the whole communication process. This protocol is used in protocol HTTP/3. [12, 13]

1.3.2 Port Number

A specific application running on the system is, from the perspective of network communication, identified by its port number. The port number is a 16-bit identifier. Thus, there can be at most 2^{16} applications that communicate at the same time.

As listed in Table 1.1, port numbers can be divided into three categories.

■ **Table 1.1** Port Number Classification

Name	First port number	Last port number
well-known ports	0	1023
registered ports	1024	49151
private and dynamic ports	49152	65535

Table 1.2 shows selected well-known services and their port numbers. However, the actual usage of ports can vary based on the administrator's decision¹³. In reality, it can happen that SSH¹⁴ would communicate via port 80, a well-known port for HTTP. On the other hand, keeping the port number for well-known applications is a recommended practice.

1.3.3 Port Number Filtering

Filtering on this layer is based on the source or destination port number, commonly used in combination with IP addresses on the lower layer. This combination can determine a specific network service running on the particular device. However, there are a few limitations that have to be taken into account.

- Well-known ports. As described in the previous section, it could happen that the service does not match a well-known port. Administrators have to set up the filtration by taking this into account.
- Stateful filtering. In this section, various protocols operating on the Transport layer were introduced. Especially for protocol TCP, confirmation messages are sent back, and those should also be allowed. If they are not allowed, an acknowledgment message does not arrive to the sender and data needs to be retransmitted. As a consequence, failure to do so usually ends with a timeout of the communication.

¹¹i.e., performing a handshake between the source and destination

¹²due to switching between Wi-Fi and mobile data connection, for instance

¹³the administrator can decide which port number is used for a particular application

¹⁴which should be communicating via port 22

■ **Table 1.2** Selected Well-known Ports [14]

Port number	Transport Layer Protocol	Application	Short Description
22	TCP	SSH	secure remote login and command execution
23	TCP	Telnet	unencrypted text communications
53	TCP or UDP	DNS	translation of domain names to IP address
67, 68	UDP	DHCP	automatic assignment of IP address
80	TCP or UDP or QUIC	HTTP	communication between a web client and a server
123	UDP	NTP	time synchronization
443	TCP or UDP or QUIC	HTTPS	same as HTTP, but communication is encrypted using TLS

- Combination with Internet layer. It is common to combine filtration on this layer with IP addresses (if there is no such interaction with the lower layer, the whole service is blocked¹⁵). Thus, the disadvantages written in the previous section must also be considered.

1.4 Application Layer

A specific application used in a network communication is identified on the Transport layer by its port number. The main goal of the Application layer is to provide an interface through which users can communicate with the particular network application. The inputs from the users are then transformed into the format that supports lower layers of the TCP/IP model and then transported over the network.

1.4.1 Protocol HTTP

As regular users are often in contact with the HTTP protocol, the principle of this layer is demonstrated on its basis. Another aspect is that this protocol typically has less strict rules than others. In comparison, when using the DNS, only a significantly reduced set of servers can be contacted and thus it is possible to whitelist them. However, this can be challenging for HTTP because users usually want to access various sites. The actual set is unknown in advance and is nearly impossible to predict.

To better understand the motivation behind the Application layer, let us imagine a situation where a user would like to view some website. The user initially opens some web browser, such as Google Chrome, Microsoft Edge, or others. Opening the web browser starts at least one process¹⁶. The user enters a domain name of the requested site and the web browser then performs the necessary steps to initiate HTTP communication – it converts the user’s request into a HTTP request which is then processed by the lower layers of the TCP/IP model.

When the response arrives, the goal of the Application layer is to decode the message and then present it to the user by using the defined application (a web browser, in this example).

¹⁵no device listening on the particular port number cannot be reached

¹⁶in reality, often more than one

1.4.1.1 Methods

As one can see, HTTP is a complex protocol that covers multiple actions for a given resource. In reality, users often need to view a website's content, but sometimes, they also need to send some data, for instance, when they would like to log into their account¹⁷. Various types of requests are served by HTTP methods. Below is a list of the most common methods. It should be mentioned that the listing is not complete as there are other, rarely-used methods. [15]

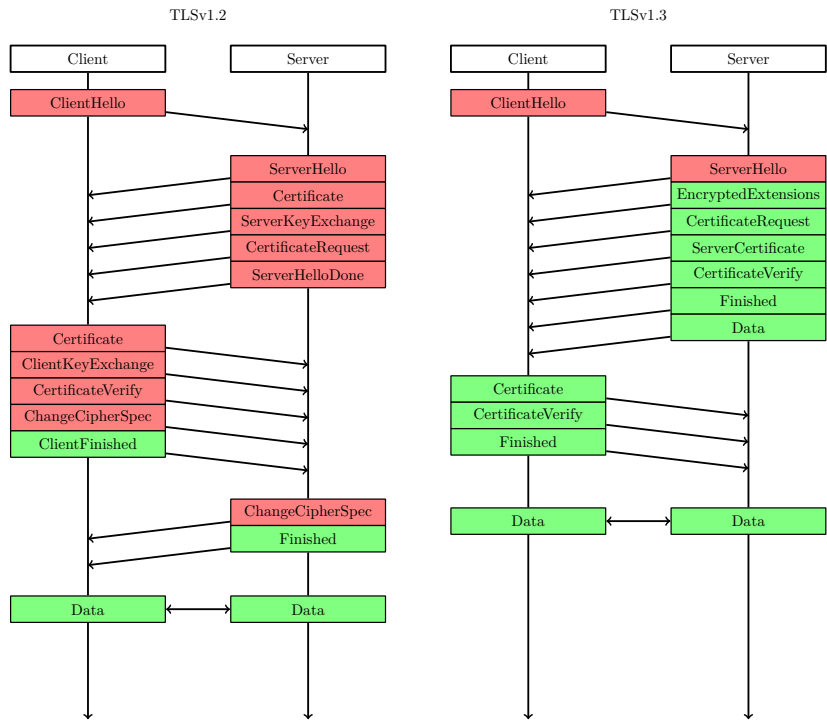
- GET. This method is used to retrieve data from the specified resource. It is possible to send some users' data with this method, but it should be used only in reasoned cases as the actual data can only be sent as a part of the URL and often in plain text. Furthermore, the URL is usually present in the network device logs. Attackers can then obtain sensitive data when the logs leak. Another aspect is that the data length is limited. It is not a good idea to send sensitive information using this method.
- HEAD. This method has a similar meaning to the GET method described above. The only difference is that the response only contains the headers, not the body with the actual resource representation. This method can be used alongside a caching mechanism. For example, a browser can send a HEAD request and thus obtain current headers. Based on the obtained information (especially in the header `Last-Modified:`), the browser can decide whether the current representation of the resource has been already stored in the browser's cache or whether the GET request is needed. This method is commonly used as it can save some time.
- POST. This method can be used to transfer users' data to the specified resource. The actual data are transferred in the body of the message and thus attackers cannot obtain them as easily as in the GET method. Also, the length of the actual data is extended. It is the most recommended method for carrying user data.
- PUT. The PUT method creates or updates the representation of specified resources. For instance, users can upload some images on the server using this method.
- DELETE. As the name indicates, this method deletes a particular resource on the server.

1.4.1.2 Return Codes

As users need to be informed about the results of their requests, HTTP return codes are used for this purpose. The protocol describes the following classes of return codes (the class is identified with the first number in the 3-digit code). [16]

- 1xx. These return codes are used only for information purposes.
- 2xx. This class of return codes indicates that the request was successfully processed and is often sent within the actual representation of the resource.
- 3xx. Return codes that begin with number 3 indicate that the location of the resource was (either temporarily or permanently) moved. A client then needs to request its new location.
- 4xx. This class represents client-side errors. It can cover situations where the user does not have enough rights to view the resource, authentication is required, or the resource cannot be found.
- 5xx. The 5xx messages represent server-side errors and are often sent due to serious problems with the server or its configuration.

¹⁷generally, provide some data



■ **Figure 1.2** Comparison of TLSv1.2 and TLSv1.3 Handshake [17]

1.4.1.3 Security

As HTTP is a text-based protocol, additional security must be applied to ensure the integrity and confidentiality of the carried data.

For that purpose, the TLS is used. Its main functionality is to establish a secure channel for ongoing HTTP communication. The actual data transported in the channel is then enciphered using symmetric cryptography. Keys for this cipher are exchanged during the initial phase of TLS (typically using the Diffie-Hellman key exchange).

Protocol TLS is also responsible for authentication, meaning that the server needs to prove its identity¹⁸. It can be done by using certificates or, more precisely, the concept of PKI.

Currently, two versions of TLS are used in modern applications – TLSv1.2 and TLSv1.3. Although TLSv1.2 was released in 2008, it can still be used in current applications. In protocol TLSv1.3, messages are sent in a different order, and thus, data can be enciphered earlier in the communication. The order of messages can be seen in Figure 1.2. Despite TLSv1.2 supporting a large variety of cryptographic functions, in TLSv1.3, many of them were removed to prevent weak algorithms from being used. [17, 18, 19]

1.4.2 Filtering Options

There are also a few different approaches to data filtration on the Application layer.

- Type of application. This approach is typically used in close integration with the Transport layer. This means that the specific network protocol¹⁹ is prevented from communication. From the point of view of configuration, this is usually done by whitelisting or blacklisting the specific port used.

¹⁸in some cases, it is needed to prove the client’s identity as well

¹⁹such as HTTP, DNS, DHCP, etc.

- Pattern-based filtering. Some protocols have a unique sequence or format of messages used in the communication. The format of DHCP messages or messages for establishing a TLS channel can be used as an example. However, this filtering type is rarely used, mainly because it is a time-consuming and thus ineffective solution.
- Filtering based on application, program, or service. As described in the previous section, there are always some applications that present actual data to users. Managing communication only from, or to a trusted set of processes is sometimes beneficial. However, this approach has limitations as it is challenging to map all processes responsible for communication. For web browsers, these include the core, rendering engine, and even more. This principle is discussed in more detail further in the thesis.
- Windows services. Although Microsoft Windows is a popular operating system, its service structure could sometimes be challenging to understand. Microsoft Windows comes with a prepared antimalware solution called Microsoft Defender, which also has its own application firewall called Windows Firewall. However, even with the cleanly installed operating system, many rules are defined that apply to communication. As administrators usually have only a limited understanding of the actual function of those services, the initial set-up is often left in the system. In consequence, this can provide an opportunity for attackers.

Application Firewall

In this chapter, the main principle of the application firewalls is described, as well as a motivation for using the application firewalls. Alongside that, the most common evasion techniques are shown. This chapter also introduces popular application firewalls used in the industry.

2.1 Phases of Attack

The previous chapter of the thesis shows the common filtering mechanisms used on various layers of TCP/IP. This section provides a slightly different approach to presenting the motivation for using the application firewalls as it introduces the Cyber Kill Chain model. It was initially designed by Lockheed Martin company to describe the actual steps taken by attackers to perform successful attacks.

2.1.1 Cyber Kill Chain

In 2021, Lockheed Martin company introduced the Cyber Kill Chain framework¹ as a part of the Intelligence Driver Defense model. Cyber Kill Chain was designed to help organizations through a deeper understanding of the attackers' steps that need to be performed to make an attack successful². [20]

As the attackers' steps are described with this model, it can also help to understand how to prevent attacks and thus create a more secure environment. However, prevention has to be considered in all steps of the Cyber Kill Chain to ensure a so-called defense in depth (DiD) approach. It means preparing an environment in which attackers must overcome all of the security mechanisms to achieve their objectives. These mechanisms thus can make the attack more challenging.

The stages of the Cyber Kill Chain framework within the offensive and defensive actions taken on particular stages are shown in Figure 2.1.

¹inspired by a military concept called Kill Chain

²typically means gain control of the device or its part



■ **Figure 2.1** Cyber Kill Chain Model with Offensive and Defensive Actions [20, 21]

There are seven steps³ described in this framework. [21]

- **Reconnaissance.** In this phase, the attacker is trying to obtain as much information about the target as possible. This step can be done offline or online. While offline means researching pieces of information that are disclosed to the public⁴, online is done by communication with the actual device maintained by the target organization⁵. The target organization has only a limited set of countermeasures at this stage. However, the most recommended steps are disclosing only reasonably acceptable information and applying proper mechanisms to detect network scans.
- **Weaponization.** Based on the information obtained at the previous stage, the actual malicious code is prepared on the attackers' side. These exploits typically use some vulnerability or weakness found in the system. Common automated tools⁶ can help attackers prepare the exploit accordingly. On the side of the target, proper patch management, vulnerability analysis, and regular penetration testing have to be performed to maximize the level of security at this step.
- **Delivery.** At this stage, malware is delivered to the target device. This can be achieved with the help of social engineering, phishing emails⁷, or malicious USB sticks, for instance. From the target's point of view, this is the first step where the actual attack can be effectively detected as it enables the use of filters in communication, endpoint security, or training users to handle suspicious activities properly.
- **Exploitation.** As the malware has already been delivered to the target devices, the system's vulnerability is exploited. In order to prevent exploitation, a proper detection mechanism should be applied. For instance, when opening the file, it should be at least checked by antimalware or opened in sandbox.
- **Installation.** When the actual vulnerability in the system is exploited, the next goal is to achieve persistence⁸. Sometimes, this is called a backdoor and can be gained using hijacking, injection, or changes in registry settings. From the defense point of view, proper process isolation and following the principle of the least privilege is needed.
- **Command and Control (C2, C&C).** At this step, malware initializes the communication with the attacker as it opens the channel to the attacker's infrastructure. That communication is often tunneled over known protocols, such as HTTP or DNS. From the target's point of view, turning off remote access tools (such as SSH or RDP) alongside proper traffic filters should be used to detect and prevent such communication. Thus, this is a nice use case for application firewalls.
- **Actions on objectives.** This is the last stage that attackers need to complete to achieve the attack's goal. It can mean leakage of sensitive data, encrypting data using ransomware, or a lateral movement to infect more devices. This is also the last step where the attacker can be detected. A proper detection system with Data Leakage Prevention (DLP) or proper network design can be helpful when mitigating the risk.

³sometimes called stages

⁴including social networks or specialized sites such as Shodan

⁵often with the help of the Nmap tool to find open ports, for instance

⁶also known as "Malware as a Service" (MaaS)

⁷typically in combination with insufficient spam filters

⁸attackers will still have access to the device even when the device is restarted, for instance

2.2 Principle of an Application Firewall

As written in the previous chapter, filtering based on the network protocol can be deployed to limit communication. However, the main disadvantage of this solution is that it is based on the type of the application⁹. [22]

Another aspect is that there is only a minority of people familiar with these protocols as a vast majority of the users only use the Graphical User Interface (GUI), which transforms their requests into the language of the network protocol. For instance, users open the web browser to access the website and usually do not care that there are some protocols, such as DNS or HTTP, in the background.

While referring to the explanation given in the above sections, the principles in this section are also demonstrated mainly on the HTTP protocol. The reason is that attackers often abuse this protocol to hide their communication because the HTTP protocol often has less strict rules than other protocols.

From the perspective of end devices, the source application of the communication could be identified as there is always some running process responsible for the communication. In an ideal world, it is thus possible to detect whether the communication is initiated from a trusted web browser or some malware.

2.2.1 Trusted Application Definition

In order to perform filtration based on the source application, a list of trusted applications should be created. The actual set of rules can be generated in several ways.

- All outgoing communication is permitted. Even though no filtering of outgoing communication is in place, this is the one that is usually used as a default configuration of a vast majority of solutions used nowadays. This option does not bring any additional security to outgoing traffic as the rules are applied only to incoming traffic.
- Users have to create a list of trusted applications manually. Initially, there is an empty list of trusted applications and users must decide which application is trusted and which is not. This option has a tendency to be the most secure one. However, it has to be said that a deep knowledge about the system is needed to manage rules correctly as a large number of applications usually needs to communicate¹⁰. Thus, users often need to confirm many applications during the initial phase. This is an ideal opportunity to overlook the malicious ones.
- Predefined set of trusted applications. Vendors of application firewalls usually come with a predefined set of essential applications. This often includes the most commonly used web browsers, as well as system processes. When the users want to add other applications to the list, it can be done similarly to the previous option.

⁹HTTP, DNS, DHCP, and even more

¹⁰various processes of Microsoft Windows operating system, for instance

2.2.2 Trusted Application Verification

When traffic filtering based on the rules created by one of the described ways is present, those rules must be applied to the communication. There are a few ways in which it can be decided whether a trusted application is communicating.

- Name of the process. When the rule is created, the name of the trusted process is stored in the internal database. When a communication is detected, its source is compared to the records in the database. This is the most naive approach that can be easily bypassed because the attacker can prepare a malicious executable with the same name as the trusted one.
- Location of the binary file. The actual location, from which the binary is launched, is stored in the database alongside its name. This can be considered an extension of the previous one. On the other hand, it is more secure solution as the location of the binary file is often in the more protected filesystem's part. Local administrator rights are typically needed to write to these locations.
- Hash of the binary file. When the rule is created, a hash of the binary file is stored in the database. This hash is then compared with the actual communication attempt. This approach has the main disadvantage in that the rule must be set up again when an application update is installed (as the hash of the executable is then changed).
- Digital signature of the binary file. This principle uses the PKI mechanisms. The main idea is that the vendor digitally signs the binary file. Then, applications from known publishers can be considered as trusted ones. Digital signature thus overcomes the problems with updates described in the previous point.

2.3 Common Evasion Techniques

An “Evasion technique” is a term used to describe the concept of bypassing the defined rules in firewalls. It is also known as a leak test.

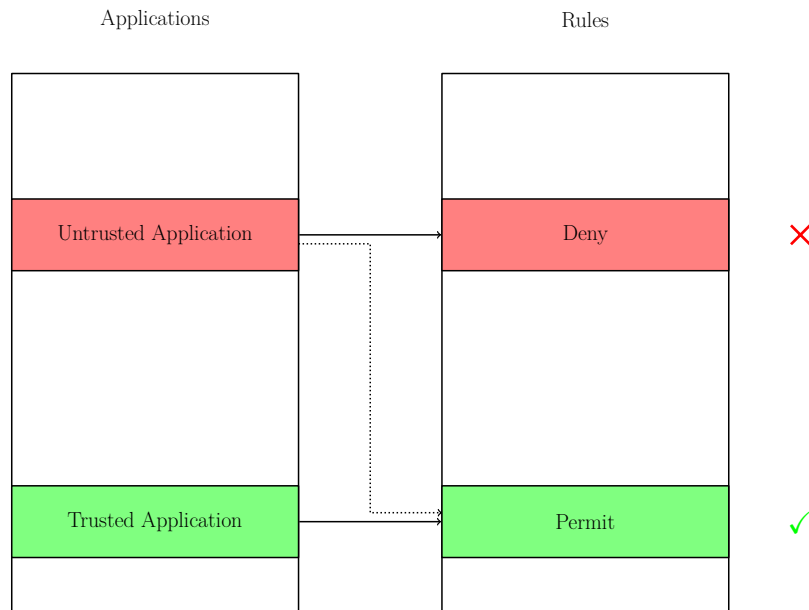
As outlined in the previous section, application firewalls need a defined set of rules for their operation. These rules typically include a list of trusted applications permitted to communicate with other devices over the network. A core principle of the test is to abuse these trusted applications to hide malicious communication. This principle is shown in the Figure 2.2. The goal of attacks can be achieved in the ways described further in the section. [23, 24]

- Substitution. The principle behind this technique is that the untrusted application tries to identify itself as a trusted one. As the name indicates, this usually includes changing the name of its process to the trusted one, replacing the executable file on the drive, or substituting processes' data in the random access memory (RAM). The actual success depends on the implementation of the firewalls. Consequently, users typically have only a very limited¹¹ set of possibilities for mitigation by some configuration. Firewalls can use, for instance, hash functions to identify the application and thus help to verify a particular application¹². To mitigate the risk of potential attacks, it is recommended to store executable files in the location where regular users have limited write permissions¹³ and use as many digitally signed applications as possible. An additional check of their vendors can then be applied by using certificates, or more precisely, the concept of public key infrastructure (PKI).

¹¹or sometimes none

¹²based on their irreversibility

¹³default settings of Microsoft Windows operating system is to use C:\Program Files\



■ **Figure 2.2** Common Principle of the Attacks. In the ideal situation, permit rules are applied to trusted applications only, while untrusted applications are prevented from communication by applying the deny rule. In the figure, this is highlighted by solid lines. However, a common goal of the attackers is to overcome the firewall in the way highlighted by the dotted line.

- **Launching.** This method is based on the fact that some of the applications¹⁴ have also implemented a command line interface and it is thus possible to initiate a defined behavior from the command prompt. Based on that, it is possible to add some parameters to the command prompt that can help to bypass the rules. Another subtype of this attack is to launch the untrusted executable from the trusted one. For instance, while the process `explorer.exe` can be defined as a trusted one, it could be possible to launch an untrusted application in its context. From a command line point of view, it is entered as a parameter.
- **DLL Injection.** In this scenario, attackers create a malicious dynamic link library (DLL) which is then loaded into the address space of the trusted application's process. When the DLL is loaded, it acts as a part of the trusted application and thus has the same rights. This primarily means applying the same firewall rules as for a trusted application. It is one of the most commonly used evasion techniques.
- **Code Injection.** This technique is similar to the previous one, although the executable code is injected into the address space instead of using DLL. In theory, this can be done manually as the attackers modify the actual address space of the process. Sometimes, creating a remote thread (by the `CreateRemoteThread` Windows API function, for instance) and running malicious code in this thread can be more valuable. As in the previous technique, it runs as a part of a trusted application, and thus, the rules of the trusted application are applied.
- **Race conditions.** As application firewalls may need the process identifier (PID) to apply particular rules, the concept of this technique is to change the PID of the untrusted process to one that is trusted. In improper implementations, an application firewall may consider this communication as legitimate. PID is a number that uniquely identifies the running process. It is usually given by the OS and can be changed by restarting the process.

¹⁴at least most of the browsers

- **Browser services.** This approach combines built-in functionalities of the OS and web browsers responsible for interaction between processes. For instance, attackers can send Windows messages to the web browser window and thus actually change its state. Another example used primarily in the earlier implementations, browsers' Dynamic Data Exchange (DDE), which was responsible for data exchange between two processes, could be abused.
- **System services.** In general, this technique is almost the same as the previous one, although it uses program interfaces given by the OS. Those include services like Windows Server Update Services (WSUS), which is responsible for downloading the updates in the background. Also, DNS API functions can be abused to encapsulate outgoing messages to the DNS request.
- **Default rules.** Many firewalls come with a predefined set of rules applied to particular applications. These would contain essential services of the OS¹⁵ or the most common web browsers. For the vast majority of users, this is the configuration they use. From an attackers' perspective, it means they can install the application firewall on their device and have a deeper look at them. Thus, it is not the evasion technique in its original meaning, but it can help attackers prepare for the actual attack.
- **Custom protocol driver.** While the main functionality of the application firewall is based on the TCP/IP protocol stack, another stack would be used in this method. In the core principle of this low-level technique, attackers can develop their own protocol stack that will communicate directly with the network adapter. Thus, the parallel protocol can be created to bypass the filtration mechanism of the deployed application firewall. As an example, the mentioned protocol can be based on the Winpcap library.
- **Unhooking.** In general, packet filtering hooks are used to intercept system functions to provide the basis for the subroutines responsible for the actual filtering. If this intercept is disabled, the application firewall cannot work correctly and thus does not effectively apply rules on the traffic. Although it is not an evasion technique in its original meaning, it had been decided to include it in the listing as it is a common goal of the attacker to turn off the firewall completely. [25]

¹⁵responsible for DHCP, DNS, and even more

2.4 Popular products

Application firewalls can be deployed as standalone software or, most commonly, as a part of complex antimalware solutions. This section presents the most popular application firewalls.

2.4.1 ESET Internet Security

ESET is a very popular security product vendor, at least in Central Europe. Internet Security is the leading product of this Slovak company.

The product offers a 30-day trial version that supports all the features included in the regular version. After 30 days, users can decide whether they want to pay for the subscription or use other products. Based on that, it was decided to perform the test during the trial period and thus test whether the additional fee provides an adequate level of security. [26]

2.4.1.1 Application Firewall Modes

An application firewall called ESET Internet Security Firewall is delivered as part of the Internet Security.

Before its filtering modes are presented, the order of the evaluation for the rules should be explained. In contrast to the older versions of the software, where the actual order depended on the defined priority level, the most recent products evaluate the rules in the order they appear in the rules list while applying first-match concept¹⁶.

Users can define several modes in which the ESET's application firewall operates. These are listed below within a short explanation.

- Automatic mode. This mode is the default one and is intended for users who are not interested in the actual setup but accept a limited level of security. Its main disadvantage is that all outgoing connections are enabled, while rules are applied only to incoming connections. This could be an ideal opportunity for several malware families, such as Trojan horses.
- Policy-based mode. In this scenario, a defined set of rules is evaluated within each communication. If any rule matches the communication requirements, it is applied. If no such rule is found, then the communication is blocked by the firewall.
- Interactive mode. This one is similar to the previous one, but there is a difference at the end of the evaluation. In the previous mode, communication was blocked when no matching rule was found. In this case, the message is then displayed to prompt users to confirm whether they trust the application that attempted to communicate. Users can view additional information about the application¹⁷ and decide whether to create permit or deny rule.
- Learning mode. This mode does not bring any additional security as it automatically creates rules for each application that is trying to communicate with the outside world. Although the vendor recommends using this mode in the initial phase of firewall setup, it is worth mentioning that when the computer is infected with some malware, its malicious communication is added as a rule to the list without any notification to the user.

¹⁶the first match that for the rule is applied and thus the actual evaluation ends

¹⁷such as its certificate

2.4.1.2 Application Modification Detection

In the Internet Security settings, there is a built-in feature called Application Modification Detection, which aims to detect changes in evaluated applications.

In the default configuration, this feature is enabled, although there is another limitation. The “Allow modification of signed (trusted) applications” checkbox is also enabled. Thus, this additional check is not applied for signed applications¹⁸.

If such modification is found, the product shows the information to users and lets them decide whether they still trust the modified application. Even though attackers can modify the executable file, the root cause can be legitimate. It can be caused by the application’s update, for instance.

2.4.2 Avast Free Antivirus

While ESET products are popular within organizations, Avast products are often considered a good option for end users. From the perspective of end users, the most significant benefit of the product is the stable free version. Even though there is no additional fee for the Avast Free Antivirus product, it is often listed at the top of antivirus rankings.

Another positive aspect could be that Avast is often called to be one of the biggest propagators of using artificial intelligence (AI) and machine learning (ML) for behavioral detection. As the company writes on its site, it takes only a couple of hours to create an actual malware definition model for a new threat. [27]

2.4.2.1 Firewall Rules

As well as other solutions, the Avast Free Antivirus Firewall has several operation modes. In principle, these are the same as in ESET Internet Security, although there are some minor differences. [28]

- Smart mode. In this mode (which is the default one), Avast decides whether the communication is blocked or permitted based on the application’s trustworthiness. We can only guess how this trustworthiness is defined as no reliable source describes the process. It is considered a business secret.
- Allow mode. As the name indicates, this option allows communication from all unknown applications. It means that the actual rule list is evaluated, and when no matching rule is found, the application’s communication is enabled.
- Block mode. This operation mode is complementary to the previous one. In its principle, communication is blocked when no matching rule is found.
- Ask mode. The fourth mode lets users decide whether to trust newly discovered applications. Thus, users must decide whether to permit or deny communication for the new applications.

2.4.2.2 Behavior Shield

As an additional layer of security, Behavior Shield can be deployed. While enabled in the default configuration, it offers real-time scanning for the applications’ suspicious behavior, indicating the presence of malware in the system. In its principle, Behavior Shield is trying to map suspicious activity to known threats. [29]

There are three options in which Behavior Shield can be configured in the Avast product.

- Always ask. If the threat is detected, Behavior Shield may ask the users to decide whether the detection is true positive. Thus, users can manually decide whether any or no action has to be taken.

¹⁸most of the system processes or web browsers

- Automatically move detected threats to Quarantine. The actual behavior of the application is compared to the list of known threats. If there is a similarity, then the application is moved to Quarantine. Quarantine can be defined as a safe place used for storing and testing potentially malicious files which is completely isolated from the rest of the operating system.
- Automatically move known threats to Quarantine. As with the previous method, actual behavior is compared to the list of known threats. If there is a match, then the application is moved to Quarantine. If there are doubts or similarities, the users are asked to make the final decision. Users can thus decide whether the application is safe or has to be moved to the Quarantine.

2.4.3 Windows Firewall

Windows Defender is the built-in and enabled-by-default antivirus in any current Microsoft Windows operating system edition. It is a complex software that includes Windows Firewall used for network traffic filtering based on IP address, port number, or the application. [30]

As the vendor is the same for the product and operating system, one can say that it should match the system's requirements best, compared to other solutions. On the other hand, even though a brand new Windows operating system is installed, the Windows Firewall contains many rules. Those are applied mainly to Windows processes and thus are often left in the default configuration because regular users typically have only a limited understanding of these processes.

2.4.3.1 Firewall Profiles

The Windows Firewall has three predefined profiles that specify rules applied in a defined environment. For example, let us say there is a corporate network with some database server that can be accessed only from that corporate network. On the client computer, there is some database client. It may make sense to enable communication from the client only if connected to the corporate network and not a public one.

As already mentioned, there are three profiles defined in the Windows Firewall. These are listed below.

- Domain network. This profile is used for a corporate network. Apart from the others, it cannot be assigned manually. The profile becomes active when the computer joins the Active Directory network or, more precisely, when the domain controller (DC) for the particular network becomes available.
- Private network. This profile is intended for home networks and usually means that less strict rules are applied to decrease latency caused by additional inspection. Thus, this mode should be active only in the network users trust. Users with local administrator rights can manually set it on a network interface.
- Public network. It is a default profile that is applied when connected to unidentified networks. It is supposed to be the environment with higher security risk as it covers public wireless networks at various places or Wi-Fi hotspots.

2.4.3.2 Firewall Rules

Just like the other solutions, Windows Firewall needs a list of defined rules.

Unlike the other products introduced in this section, Windows Firewall enables the identification of actual communication on several layers at once as a rule can combine the application, service, source and destination IP address, common service, transport layer protocol, or even interface type. The principle behind rule evaluation is that the incoming traffic is blocked but can be enabled by a rule, and all outgoing traffic is allowed but can be disabled by some matching a rule.

As written in the official manual for Windows Defender, it is recommended to allow all communication from the device. “When designing a set of firewall policies for your network, it’s a recommended practice to configure allow rules for any networked applications deployed on the host. Having the rules in place before the user first launches the application helps to ensure a seamless experience.” [31] However, with the growth of allowed applications, the attack surface is increasing and the probability of the attack’s success is also more significant.

In the default setup, new rules for particular applications are added automatically during the installation process of the new application. When not added for whatever reason, the users are asked to allow or block the communication when the application tries to communicate for the first time. The local administrator’s rights are needed to add a permanent rule to the list.

2.4.3.3 Windows Defender Application Control

When talking about filtering network traffic based on the application, the Windows Defender Application Control (WDAC) concept should be mentioned. [32]

The WDAC was introduced within the Microsoft Windows 10 operating system. It allows controlling which applications can run on the device and which cannot¹⁹. WDAC can be deployed in all of the currently supported versions and brings the ability to control applications based on the attributes of the codesigning certificate, other attributes of the binary file, or the process that launched the actual binary.

A set of rules can be defined and evaluated. More often, Microsoft’s Intelligent Security Graph (ISG) is used to define the application’s reputation.

Before the introduction of WDAC, AppLocker was used. This one was introduced in Windows 7 and brings only limited possibilities compared to WDAC. Thus, Microsoft recommends using WDAC instead of AppLocker whenever possible.

2.4.4 TinyWall

TinyWall is used as a representative of dedicated application firewalls as it is not part of a complex antimalware product. As written on the vendor’s site, it should be an application firewall that is lightweight, user-friendly, secure, and fully compatible with Microsoft Windows operating systems. TinyWall’s target groups are computers deployed in small office or home office (SOHO) networks. It should provide an additional level of security without the need for a complicated setup from the users’ side. [33]

While evaluating the rules, the firewall in default settings blocks all communication except these defined by the rules. It is a crucial difference from other solutions described in the chapter as they often leave outgoing communication unblocked.

On the other hand, based on the non-intrusive operation feature, the users are not informed about the actual result of the evaluation. Thus, if a user launch an untrusted application, the communication is automatically blocked but no information is displayed to the user. Then, if the untrusted application has not implemented error messages regarding invalid network communication, troubleshooting may become challenging.

¹⁹e.g., limitation not only from networking view

2.4.4.1 Trusted Application Definition

If the users want to enable communication from untrusted applications, it is possible to do so in the TinyWall Firewall settings on the “Application Exception” tab. In the default configuration, there are only a few rules defined²⁰.

Users can decide whether the actual exception rule would be defined based on the following parameters.

- **Process.** In this option, users can choose a process on the computer whose communication should have been permitted. Internally, a specific process is identified by its executable file²¹. As a consequence, this option is almost the same as the following one.
- **File.** Users can create exceptions for particular executable files whose processes should have been allowed to communicate. These executable files can be stored in any location in the filesystem.
- **Service.** In this option, users can define exceptions for various Microsoft Windows operating system services. These include Remote Desktop Services or Windows Updates, for instance. Their executables are often stored in the location `C:\Windows\system32\`.
- **Universal Windows Platform (UWP) application.** This option enables the creation of exceptions for particular UWP packages. UWP is a Microsoft application development platform. Applications developed in UWP can be used not only on Microsoft computers but even on mobile phones or tablets. Actual applications can be distributed via Microsoft Store, for instance. [34]

2.4.4.2 Operation Modes

Similarly to other solutions described in this chapter, TinyWall comes with several modes in which the firewall operates. These are described below.

- **Normal protection.** This is the vendor’s default and most recommended mode. While this mode is enabled, all communication²² is blocked unless a matching exception exists.
- **Block all.** This mode blocks all of the device’s communication. Consequently, no application on the device is allowed to communicate over the network.
- **Allow outgoing.** This mode is similar to the ones recommended by other vendors. It allows all outgoing communication and thus applies rules only to incoming traffic.
- **Disable firewall.** With this option, all communication is allowed. Thus, it is the same situation as if the application firewall was not installed at all.
- **Autolearn.** In this operation mode, all communication is allowed, and the firewall automatically adds the rules to its list. It is recommended to run this mode only when having clean installation of the operating system and not for a long time. In this mode, if there is some malware installed on the device, it can be automatically added as an exception.

²⁰applied to Internet Explorer or Microsoft Edge and its updater

²¹more precisely path to the executable

²²both, outgoing and incoming

2.4.5 ZoneAlarm Firewall

ZoneAlarm Firewall is another representative of standalone application firewalls. This product is very popular and awarded one. It is intended for end users with limited knowledge about the problematics. On the other hand, users have only extremely limited possibilities of the actual setup. Just like some other products described in this chapter, ZoneAlarm enables all outgoing communication. [35]

When a new application wants to perform network communication, creating a new rule by hand is impossible as there is no such option in the product's configuration. The only possible workaround is to let ZoneAlarm add exception for the application. Then, the new application is shown in the exception list with "Allow connection" property. Users can then change this property to the "Block connection".

2.4.5.1 Filtering Modes

The actual filtering rules are applied based on the network where the device is located. Two types of networks can be set in the firewall – Public and Trusted. While networks in the Trusted zone are defined in the firewall's settings, the Public zone covers all the remaining networks.

As written above, users have only limited settings options in this product. Regarding actual filtering, users can only decide on the filtration level they would like to use. [36]

- High. All outgoing communication is enabled in this mode, while defined rules are applied only to incoming communication. It is also the default configuration for Public zones.
- Medium. This mode is almost the same as the previous one, but it lets users use some network-sharing services. For instance, other devices on the network can access shared folders on the device, which is protected by this application firewall. It is the default filtering mode for Trusted networks.
- Off. When applying this option, the firewall is not in operation. Thus, no additional protection is applied.

Testing Environment Description

This chapter provides an overview of actual programs and operating systems used during the testing. Whenever possible, a link to download the product is given. Due to license terms, it is not allowed to provide the virtual machine used for testing, although these pieces of information are sufficient to create the same testing environment.

3.1 Operating System and Installed Programms

There are two virtual machines used in the practical part of the thesis. For further description, these are called Victim and Attacker.

3.1.1 Victim

For the Victim machine, the Microsoft Windows operating system was chosen as the most used operating system for desktops and laptops. All the following tests are performed on edition Windows 11 Pro, version 10.0.22621. The actual version of the Microsoft Windows operating system is available on the vendor's website¹.

For a compilation of the source codes for Windows devices, Microsoft Visual Studio Community 2022 was used (version 17.9.4). The online installer is available on Microsoft's site².

If Visual Studio is not installed on the device, at least Microsoft Visual C++ Redistributable has to be installed to run the executables provided. For testing purposes, version 14.38.33135.0 was used. Microsoft Visual C++ Redistributable is available on its vendor site³.

3.1.2 Attacker

For the Attacker machine, Kali Linux 2024.1 was used as it offers preinstalled software needed for security testing. Kali Linux can be downloaded from the official website⁴. However, it should be mentioned that any other Linux distributions can be used. The only required tools for the actual testing are the g++ compiler and Wireshark to monitor network traffic. It should be mentioned that both tools can be obtained using the package manager of the particular Linux distribution.

¹<https://www.microsoft.com/software-download/windows11>

²<https://visualstudio.microsoft.com/>

³<https://learn.microsoft.com/en-GB/cpp/windows/latest-supported-vc-redist?view=msvc-170>

⁴<https://www.kali.org/get-kali/>

3.2 Application Firewalls

As described in the previous chapter, application firewalls can be deployed as a standalone program or as a part of robust antimalware products. In Table 3.1, there is a listing of the application firewalls used in this thesis for vulnerability testing. It should be mentioned that Table 3.1 is the basis of the following chapters, as the first column (“Name”) is used as an alias for the particular firewall in the text.

■ **Table 3.1** Popular Application Firewalls

Name	Vendor	Product	Version	Licence	Download link
Eset Internet Security	ESET	Internet Security	17.0.16.0	Free (30-day trial)	Download ⁵
Avast Free Antivirus	AVAST	Free Antivirus	24.2.6105 (virus definitions 240327-2)	Free	Download ⁶
Windows Firewall	Microsoft	Defender Firewall	4.18.24020.7 (virus definitions 1.407.664.0)	Windows Built-in	N/A
TinyWall	TinyWall	TinyWall	3.3.1	Free	Download ⁷
ZoneAlarm Firewall	ZoneAlarm	Free Firewall	4.2.712.19773	Free	Download ⁸

⁵https://download.eset.com/com/eset/apps/home/eis/windows/latest/eis_nt64.exe

⁶<https://www.avast.com/en-us/free-antivirus-download>

⁷<https://tinywall.pados.hu/download.php>

⁸<https://www.zonealarm.com/software/free-firewall/download>

Substitution

This chapter describes the substitution attack in more detail. In this chapter, the mentioned application firewalls are tested, whether they are able to detect such type of attack, where attackers completely substitute trusted executables with untrusted ones.

4.1 Attack Description

As written in the chapter abstract, the principles described in this chapter closely follow the information described in the theoretical part of the thesis that focuses on the trusted application definition.

To briefly refresh the information written in the previous chapters, there are several options for how the application firewalls can save pieces of information about the executables. For instance, it is possible to identify these applications based on their process name, the location of the executable on the filesystem, or by using PKI¹. It should be mentioned that we can only guess the internal implementation of the tested application firewalls because it is considered a business secret.

The common principle of this type of attack is shown in Figure 4.1. The attack options are described further in the thesis.

4.1.1 Executable's Name and Path

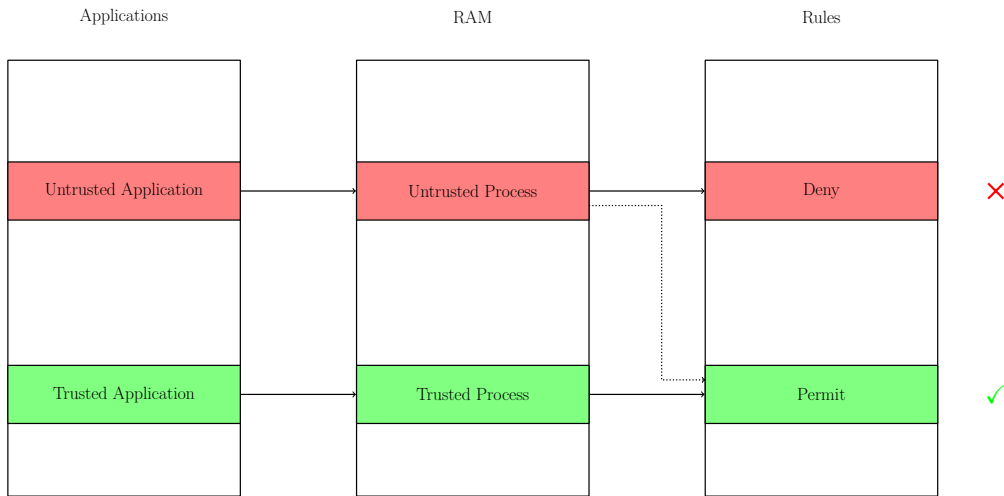
As the rules are often identified by application name, one of the possible approaches is to verify the actual application based on the executable's name. As one can see, this is the most naive approach that can be easily bypassed.

For instance, let us have an executable `OneDrive.exe`. When the application is launched, at least one process is created. As OneDrive seems to be a trusted application, it could have a permit rule set in the list. On the other hand, attackers have a malicious application that can be easily renamed to `OneDrive.exe`. In improper implementations, application firewalls can consider the malicious application trusted (they have the same name) and thus allow malicious communication.

A partial mitigation against this attack can be checking the whole path to the executable, not just the name. Modern operating systems are designed so that the executables are stored in more protected parts of the system where regular users have only limited access rights² to these locations.

¹more precisely, using some items listed in executables' digital certificate

²only read permissions, in most cases



■ **Figure 4.1** Substitution Attacks Principle. The figure shows the common principle behind substitution attacks. It should be mentioned that trusted and untrusted applications often have the same name. Sometimes, more circumstances need to be achieved to perform the attack successfully.

Consequently, local administrator rights are typically needed to change³ the applications. Based on the need for evaluated rights, the attack is more difficult for the attacker.

However, as described in the section on Cyber Kill Chain, attackers often need to establish a communication channel with the victim's device. With the local administrator rights, it should be possible to entirely turn off the firewall functionality or add new rules to the list. For potential attackers, it is often more beneficial to abuse the current state of security mechanisms to remain undetected. There is a higher probability that the attackers become detected if new rules are added to the list. Instead, attackers can use the executable stored in the filesystem and change it to another one with the same name. If the original functionality of the application remains the same after the executables switch, the attack need not be detected.

4.1.2 Digital Certificates

Before diving into the usage of digital certificates for the digital signing of executables, the key concept behind digital certificates should be explained.

A digital signature is a mechanism that can be used to ensure the document's authenticity based on the private key ownership. To explain the previous sentence, let us consider a scenario where we must digitally sign a file. At the beginning, the hash code⁴ of the file is calculated using a hash function. Its result is a fixed-size⁵ binary string. While using asymmetric cryptographic mechanisms, this hash code is then enciphered with the private key. The final value is known as a digital signature and can be sent to other users alongside the original file. As asymmetric cryptography is based on a pair of public and private keys, the public one can be disclosed.

When the second party receives the file with its digital signature, it is possible to calculate the hash value using the same hash function. Because the first party's public key was disclosed, it is also possible to decipher the digital signature and thus obtain the hash value. Then, if these two hash values are equal, the file's legitimacy is proven.

The principle described above has one significant problem that needs to be considered – the distribution of public keys. If their legitimacy is unchecked, an attacker could set herself

³or install, for instance

⁴sometimes called a digest

⁵actual size depends on the hash function used

into the so-called Man-in-the-Middle (MITM) position. In this way, the attacker can eavesdrop or even modify the actual communication. This problem can be solved by the use of digital certificates.

A digital certificate is a structure that consists of a public key and other items needed to identify the resource⁶, which are verified and signed by a trusted third party called Certificate Authority (CA). It can be said that the digital certificate is a proof of private key⁷ ownership. The digital certificate can be sent publicly, similarly to the digital signature described in previous paragraphs.

It was mentioned that the certificate contains several items. The ones that are relevant to the further sections are listed here.

- **Public Key.** This field contains the actual public key for which the certificate was created.
- **Subject.** This field is used for resource identification. It is in the form of a Distinguished name that describes the resource in further detail.
- **Issuer.** This field is used for signer identification. As with the previous one, the Distinguished name format is used.
- **Not before.** This one represents the date before which the certificate is not valid.
- **Not after.** This one represents the date after which the certificate is not valid.
- **Key usage.** In this field, the purpose of the certificate can be specified. It could include information on whether it is possible to use it for signing files or signing further certificates⁸.

As written above, the information about the Subject and Issuer is stored in the format of a Distinguished name. Table 4.1 summarizes the most used fields of the format.

■ **Table 4.1** Common Attributes of Distinguished Name

Abbreviation	Full Name	Usage
CN	Common Name	Subject name (typically DNS name for server certificates, or name and surname for personal certificates)
O	Organization	Name of the organization
L	Locality	Name of the city
S	State or Province	Name of the state
C	Country	State code according to ISO 3166 (CZ = Czech Republic, US = United States, etc.)
OU	Organizational Unit	Department of company

When the digital certificate arrives on the device, it must be checked whether the public key provided can be considered as trusted. It can be done by verifying all the certificates in the chain. It should be mentioned that each operating system has its certificate storage with defined trusted root certificates.

For the purposes of the thesis, discussing all the aspects of certificate verification is unnecessary as the process is quite complicated. It should be only mentioned that the initial certificate is trusted if and only if all the certificates in the chain to some of the trusted root certificates are successfully verified. Otherwise, the certificate is untrusted.

⁶name, organizational unit, location, and even more

⁷corresponding to public key included in the certificate

⁸and thus create a so-called certificate chain

4.1.2.1 Microsoft Certificate Manager

As the main part of the testing is performed on the Microsoft Windows operating system, its certificate store, Certificate Manager, should be introduced.

Certificate Manager is a built-in tool in every modern Microsoft Windows operating system edition. It consists of two different parts⁹ described below. [37, 38]

- Local Machine. This one covers certificates for the whole device which are valid for every user logged in to the device. Because the changes in these settings affect all device users, local administrator rights are needed to add, remove or modify the certificates. The settings are stored under the `HKEY_LOCAL_MACHINE` root in the registry.
- Current User. This one covers certificates valid for the particular user¹⁰. Thus, certificates under this component could be different for every device user. On the other hand, the basis should be the same as the Current User component inherits¹¹ from the Local Machine component. While talking about the registry, these settings are stored under the `HKEY_CURRENT_USER` root.

These components consist of several containers¹² that classify the particular types of certificates. The most commonly used of them are briefly introduced below.

- Personal (My). This container is used for the certificates for which the user or machine owns the private key.
- Trusted Root Certification Authorities. This container contains certificates of trusted certification authorities. In more detail, certificates in this container are self-signed and provide a trusted anchor for certificate validation. Users and administrators should carefully decide which certificates could be added as trusted root and which not.
- Intermediate Certification Authorities. Intermediate certification authorities are used to create a certificate chain. In reality, the client certificates are not often signed directly by a root certification authority but some intermediate authority. This intermediate authority has its certificate issued by other intermediate authority or a root certificate authority itself. This principle is called certificate chain and refers to all the certificates that should be verified to ensure trustworthiness of a client certificate.
- Trusted Publishers. In this container, trusted signing certificates are stored. The crucial difference to the Trusted Root Certification Authorities container is that adding the certificate to the Trusted Publishers container does not make all certificates issued by this certificate authority trusted. [39]
- Untrusted Certificates. This container includes explicitly untrusted certificates. Thus, if the user does not trust the particular certificate, it could be moved to this container.
- Smart Card Trusted Roots. Trusted root certificates for smart cards are kept in this store. This one is extremely handy, especially in corporate networks where users use smart cards as an authentication factor.

⁹sometimes called components

¹⁰who is currently logged in to the device

¹¹except for the Personal (My) certificates

¹²sometimes called stores

4.1.2.2 Code signing

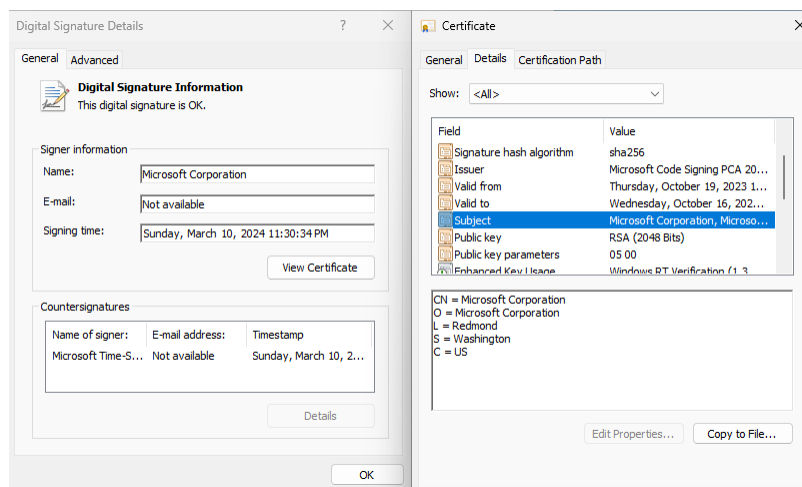
The concept behind the digital signature is often used to sign executable files to ensure their authenticity and integrity. It is highly beneficial because executables run their code, which can be potentially malicious. However, the digital signature is used only as an additional and optional security feature.

When the executable is launched, the operating system tries to find information about the digital signature. The digital signature is an optional feature, so it does not need to be included. If the signature is found, then the trustworthiness of its signer can be verified. It is done using the verification process of the signer's certificate.

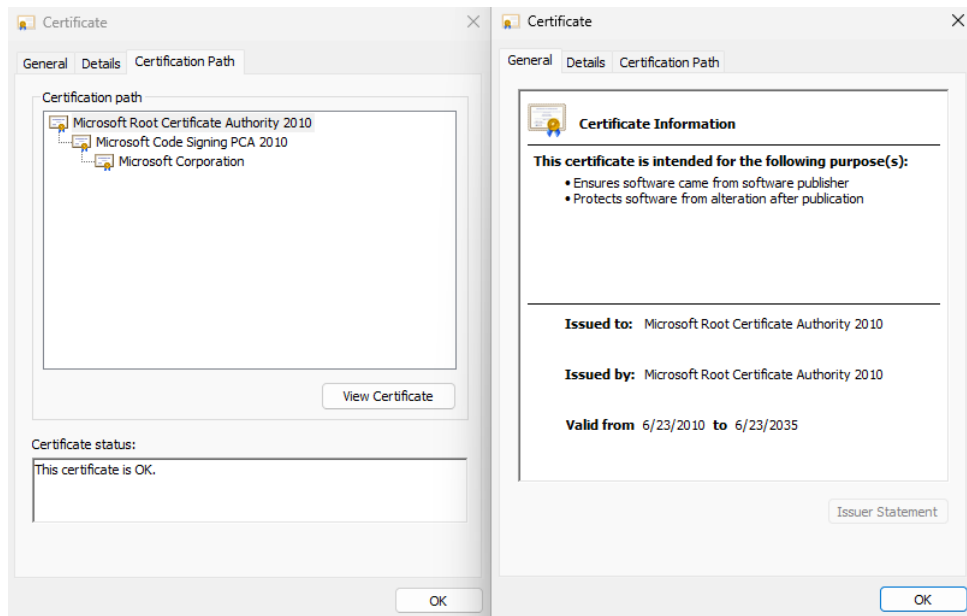
Information about the signer can be used as a parameter of trusted application identification. For instance, while having enabled the mode in which users need to decide on rule creation, information about the signer can be displayed to the user. Some products can also indicate whether the signer's certificate can be successfully verified.

This approach can be helpful in cases where the trusted application is often updated. During the testing, it was seen that some application firewalls use the validity of the signer's certificate to decide whether the application matches the rule. As an example, let us consider the application OneDrive. As it can be seen from Figure 4.2, the `OneDrive.exe` is signed by Microsoft Corporation. The certificate chain of OneDrive executable can be seen in Figure 4.3. When the OneDrive application is updated, and the same signer signs this version, it can happen that the updated one will still match the rules. On the other hand, the signer's name does not need to be unique.

The question is what happens when attackers prepare the executable named `OneDrive.exe` signed by any other certificate with the same subject as the original certificate. This concept is tested in this chapter.



■ **Figure 4.2** Digital Signature of OneDrive Executable. On the left side of the figure, one can see that the signature for this executable is valid, and it is signed by Microsoft Corporation. Its certificate can be shown by clicking on the View Certificate button. Additional details about the certificate are shown on the right side of the figure. In the Subject field, there is information about the signer in the form of a Distinguished name. Other items discussed earlier in the thesis can also be seen in the certificate.



■ **Figure 4.3** OneDrive Certificate Chain. On the left side of the figure, the actual certificate chain of the OneDrive executable can be seen. On the right side, one can see the self-signed certificate of the root CA.

4.1.2.3 Signing Demonstration Using Microsoft Tools

In this section, detailed instructions on how to set up an environment for code signing are given. Thus, further in the section, creating a new self-signed certificate¹³ is shown, as well as signing executable binary using the certificate created.

In the thesis attachment, there is a demonstration C++ code with the corresponding executable. Its code is similar to the `dllmain.cpp` described further in the thesis, although any other binary can also be used for demonstration purposes. On the other hand, the binary must communicate with some resources over the network to verify that the application matches the rules of the original application.

As the first step, creating the certificate, whose purpose is code signing, is necessary. Microsoft Windows tools can be used as there is only a need for one certificate, which is self-signed. On the other hand, if we need to create a certificate chain, it would be more convenient to use an open-source utility called OpenSSL.

To create a new self-signed certificate, Microsoft Powershell launched with local administrator rights should be used. The only thing needed is to run the cmdlet `New-SelfSignedCertificate` with the parameters described in the Code listing 4.1. For more information on how the distinguished name can be defined, it is possible to refer to Figure 4.2 where the original certificate of Microsoft Corporation is shown and also to Table 4.1 where the format of a Distinguished name is explained.

When the cmdlet finishes successfully, the newly created certificate should appear in the Certificate Manager. If `Cert:\CurrentUser\My` was used, it should be located in the Personal certificates as displayed in Figure 4.4.

Figure 4.4 also shows the initial phase of exporting the certificate to the PFX format supported by the `signtool` utility used for actual signing later.

To export the certificate from the Certificate Manager, it is necessary to right-click the one that needs to be exported, select “All tasks” and then the “Export” button. As a next step, users

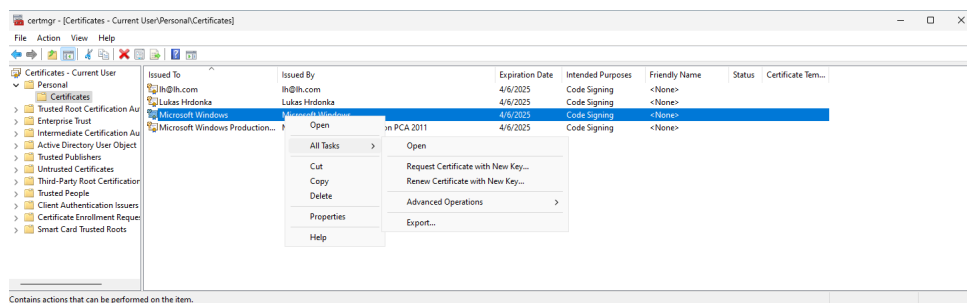
¹³meaning that the subject and issuer are the same


```

1 # <Distinguished-Name> = information shown in the certificate subject and
  ↳ issuer fields
2 # -Type codesigning = certificate is used for signing executables
3 # Cert:\CurrentUser\My = location where the certificate appears
4
5 PS> New-SelfSignedCertificate -Subject "<Distinguished-Name>" -Type
  ↳ CodeSigning -CertStoreLocation Cert:\CurrentUser\My

```

■ Code listing 4.1 Create New Self-signed Certificate



■ Figure 4.4 Export Certificate to PFX Format

have to select that the private key would be included in the exported structure¹⁴ and choose PFX format. Also, marking the checkbox “Export all extended properties” is needed, as shown in Figure 4.5. This ensures that each property defined in the certificate is exported. As a last step, the location of the exported certificate and the password for future usage should be specified. The PFX file should be found in the specified location when the export is completed.

The example certificates used in the thesis are also included in the attachment. The author is fully aware that the private key and the password for the certificate structure should never be published. On the other hand, these certificates were explicitly created for demonstration purposes and do not have any additional value.

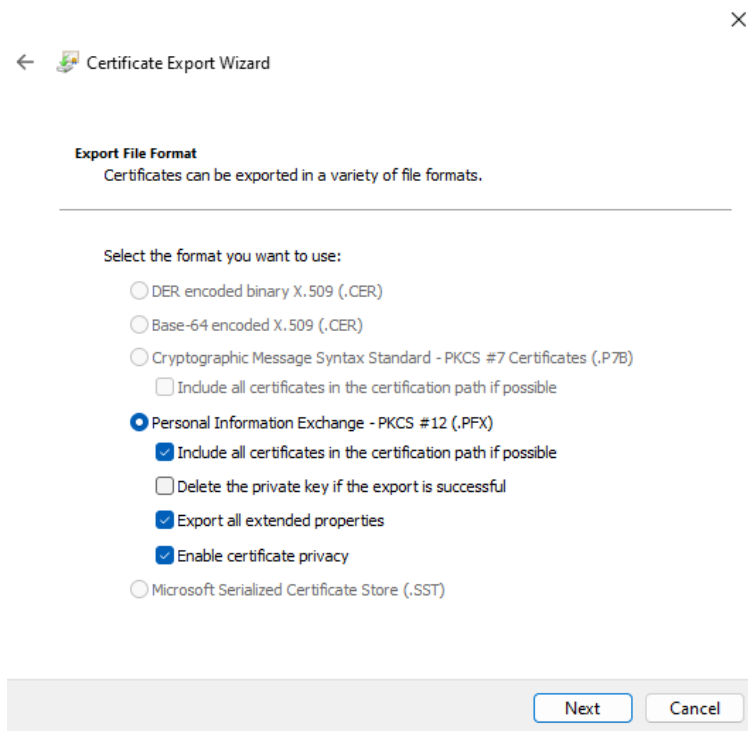
While having the certificate in the PFX format, it is possible to perform the actual signing of the executable, which is done using the `signtool` utility. The tool could be accessed from Developer Powershell for VS 2022, installed as a part of Microsoft Visual Studio. To digitally sign an executable, the syntax is shown in Code listing 4.2.

However, there is a security issue while using this command. As one can see, the password is entered as a plain text. Thus, while the command is running, the password can be visible in the Process Monitor and later also searchable in the command history. For sensitive certificates, a better solution is to create a batch file containing this command and run this batch file.

By the time the command successfully finishes, the executable is digitally signed. It can be verified with the file properties on the “Digital Signature” tab. An example of a digitally signed binary by an untrusted organization is shown in Figure 4.6.

In the screenshot, it can be seen the executable is signed by Microsoft Corporation. However, this fake self-signed certificate is, by default, untrusted in the operating system.

¹⁴needed for actual signing



■ **Figure 4.5** Additional Parameters of Certificate Export

```

1 # <Path-to-Certificate> = path to certificate in PFX format
2 # <Certificate-Password> = password created when exporting certificate
3 # <Hash-Function> = hash function used for digital signature, SHA256 can be
  ↪ used in most cases
4 # <Executable-to-Sign> = path to executable that will be signed
5 # /t <Timestamp-Authority-URL> = optional parameter specifying location of
  ↪ timestamp authority
6
7 PS> signtool.exe sign /v /f <Path-to-Certificate> /p <Certificate-Password>
  ↪ /fd <Hash-Function> /t <Timestamp-Authority-URL> <Executable-to-Sign>

```

■ **Code listing 4.2** Sign Executable File Using Visual Studio Tools

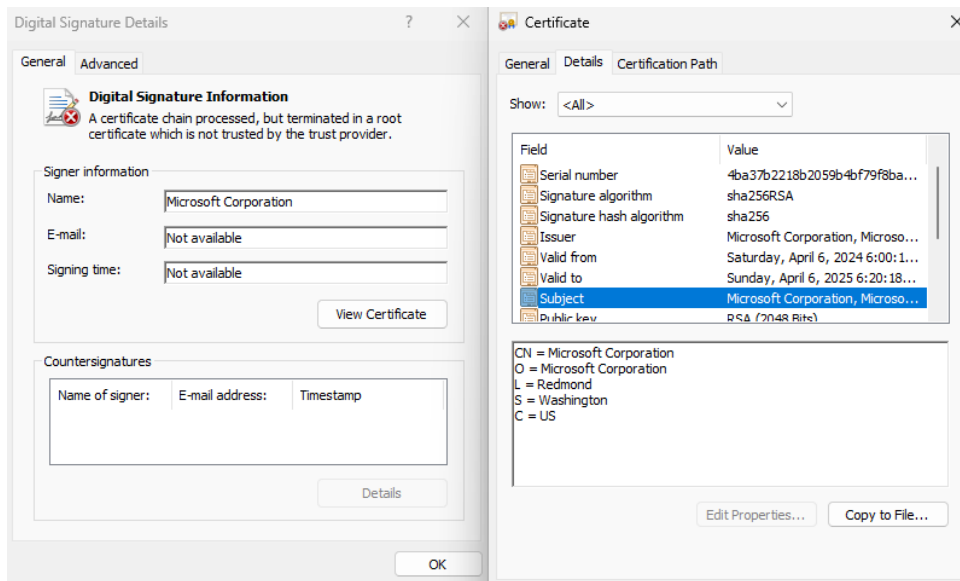
4.1.3 Attack Options

From the pieces of information written above, it can be seen that many variants of the attack exist. These are tested in further sections of the thesis.

Before the actual testing, executable files used should be introduced. It is worth mentioning that the actual executables are the same. The only difference is their digital signature. These differences are shown in Table 4.2.

It can be seen that fake Microsoft certificates are used for testing purposes. The reason is that the tests are focused on trusted executables natively presented in the Microsoft Windows

¹⁵timestamp authority is used for confirmation of the file state at the time of signing



■ **Figure 4.6** Executable Signed By Untrusted Organization

■ **Table 4.2** Executables Used for Substitution Attack

Folder in the Attachment	Digital Signature Used
substitution1	None
substitution2	Lukas Hrdonka (self-signed)
substitution3	Lukas Hrdonka (self-signed), with timestamp authority verification ¹⁵
substitution4	Microsoft Corporation (self-signed)
substitution5	Microsoft Code Signing PCA 2010 (self-signed)

operating system. In the real Microsoft application¹⁶, the certificate of “Microsoft Corporation” is used as the end certificate in the certificate chain. “Microsoft Code Signing PCA 2010” is then used as the issuer of the original “Microsoft Corporation” certificate.

It was decided to perform all the tests on the OneDrive application. Thus, executables given in the particular folders in the attachment will be launched. Each application will be launched twice – from the untrusted location (such as the Desktop folder) and a trusted location (where the operating system expects the legitimate executable `OneDrive.exe`¹⁷). Then, falsified root certificates will be added to the relevant stores of the Microsoft Certificate Manager and the tests will be repeated.

Table 4.3 provides an overview of the terminology used further in the chapter during the testing.

4.2 ESET Internet Security

In this section, the ESET Internet Security application is tested to determine whether it is capable of detecting substitution attacks.

At the beginning of the testing, there is a valid permit rule for the OneDrive application defined in the product. This one was created based on the legitimate traffic and is shown in Figure 4.7.

¹⁶such as `OneDrive.exe`, `msedge.exe`, and even more

¹⁷`C:\Users\\AppData\Local\Microsoft\OneDrive\OneDrive.exe`, for current versions

■ **Table 4.3** Terminology Used during the Testing

Definition	Meaning
trusted application	original OneDrive application digitally signed by real Microsoft Corporation certificate
untrusted application	one of the applications provided in the thesis attachment
trusted location	location where trusted application should be placed
untrusted location	any location on the filesystem

In general, if a new permit rule for the executable is created and its digital signature is present, ESET stores the information about it. In the future, the same signer is required to match this rule. If no digital signature is provided when creating the rule, then the rule in the list is identified only based on the application location and name.

For the OneDrive application, the rule identification consists of the executable name, location, and digital signature.

As a first test, all the provided applications were launched from potentially untrusted locations, such as **Documents**, **Downloads**, or **Desktop**. While having “Interactive mode” enabled¹⁸, all the attempts end with a request to create a new rule.

The next set of tests was performed using the trusted location where the operating system expects the valid OneDrive executable. Provided malicious files were thus copied to the trusted location. At this time, no request to add a new rule is displayed to the users, but there is always a message saying that the application was modified. Then, it is up to the user to decide whether to proceed with the communication. This situation is shown in Figure 4.8.

If the user decides to keep the initially defined rules, the new rule is created as shown in Figure 4.9. As it can be seen from the screenshot, ESET Internet Security did not verify the certificate successfully and thus created a rule based only on the location and name of the executable file.

A scenario where the certificates used were added as root certificates to the Windows Certificate Manager was also tested. Even though the root certificate was included in all the relevant stores of Microsoft Certificate Manager, the attack remained detected. It thus seems that the ESET Internet Security comes with its own set of trusted root certificates independent of the ones defined in the Certificate Manager.

Based on the tests above, the ESET Internet Security seems to work according to the expectations and can detect such attacks in its default configuration.

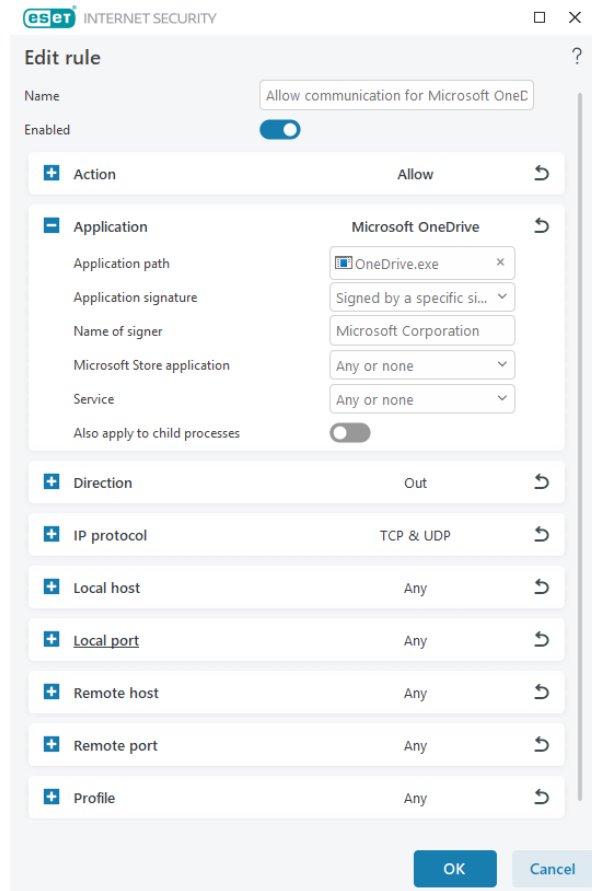
On the other hand, in the default configuration, application modification¹⁹ is enabled only for digitally signed executables. However, the product allows manual exclusion of executables from modification detection. This seems to be a reasonable alternative for applications that are not directly trusted by ESET Internet Security but need to be updated frequently without the need to prompt the user every time.

While testing this functionality, it was found that the rules in “List of applications excluded from detection”, that should enable the modification of the listed executable files, consists only of the name and path of the executable files as shown in Figure 4.10. On the other hand, the decision to adopt this approach is quite logical as ESET Internet Security could not verify the certificate accordingly based on the root certificates defined in the product.

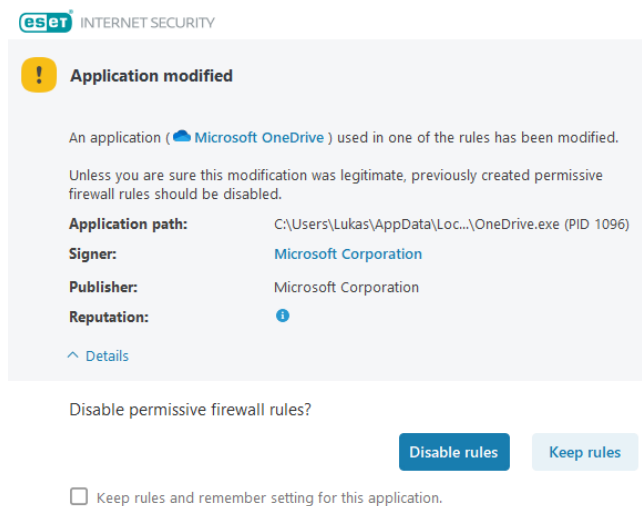
On the other hand, the possibility of adding some trusted root certificates became extremely handy in corporate networks as users often need to use corporate applications that are trusted from the company’s point of view but not from the general perspective of ESET Internet Security. If there is a need for customization of root certificates, the company could use ESET PROTECT Server which can be used for centralized management of devices from a security perspective and also for the definition of custom root certificates.

¹⁸to clearly see if the previously created rule matches or not

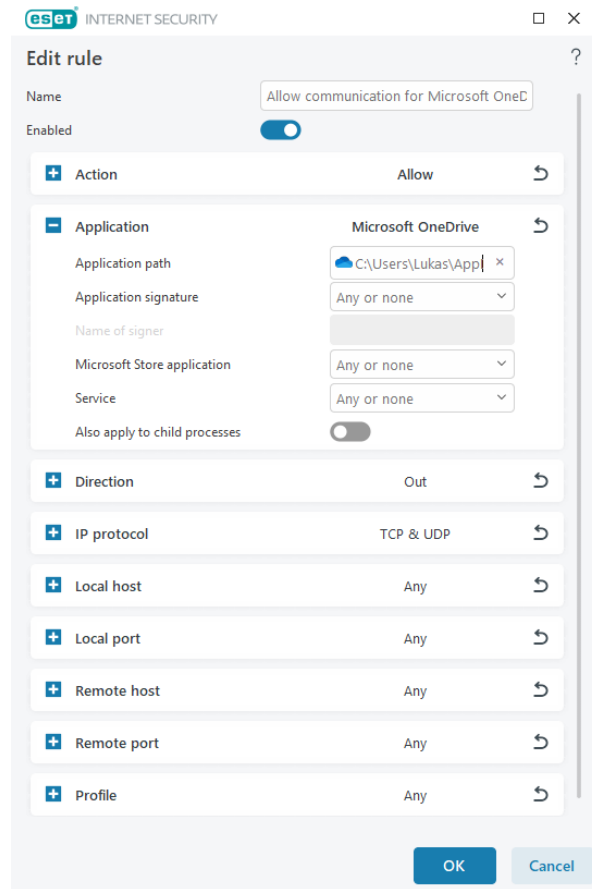
¹⁹which detected the attack



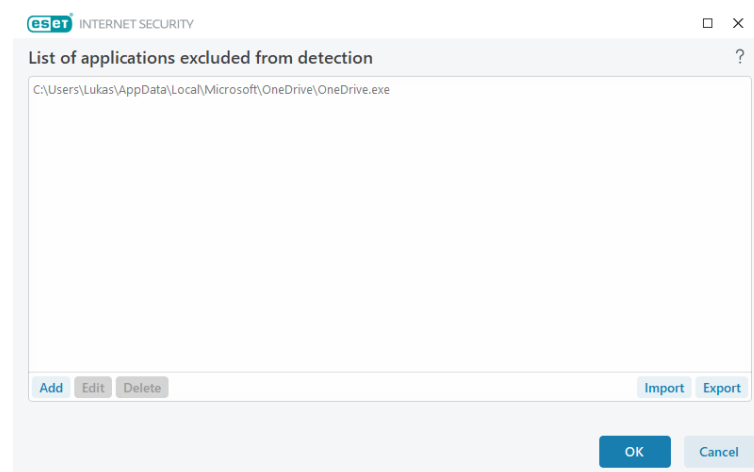
■ **Figure 4.7** Rule for Trusted OneDrive Application in ESET Internet Security. From the figure, one can see that the ESET Internet Security successfully verified the trustworthiness of the OneDrive application as the information about the digital certificate is included in the rule. There is one other important observation from the figure – the default behavior of ESET Internet Security is to enable outgoing communication with every device using every available port number.



■ **Figure 4.8** Warning about Application Modification in ESET Internet Security



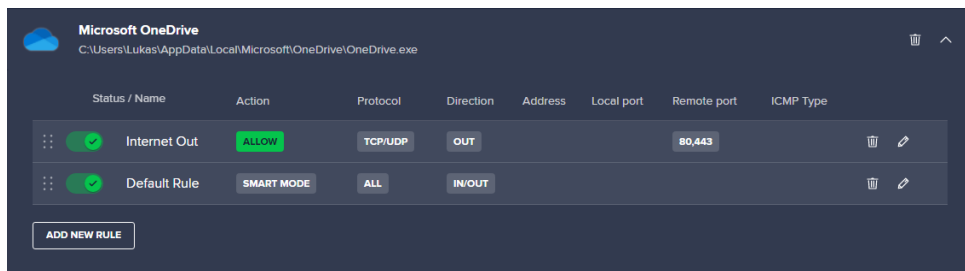
■ **Figure 4.9** Rule for Untrusted OneDrive Application in ESET Internet Security. As can be seen, ESET Internet Security did not successfully verify the certificate of the fake OneDrive executable, and the rule is thus based only on the name and location of the executable.



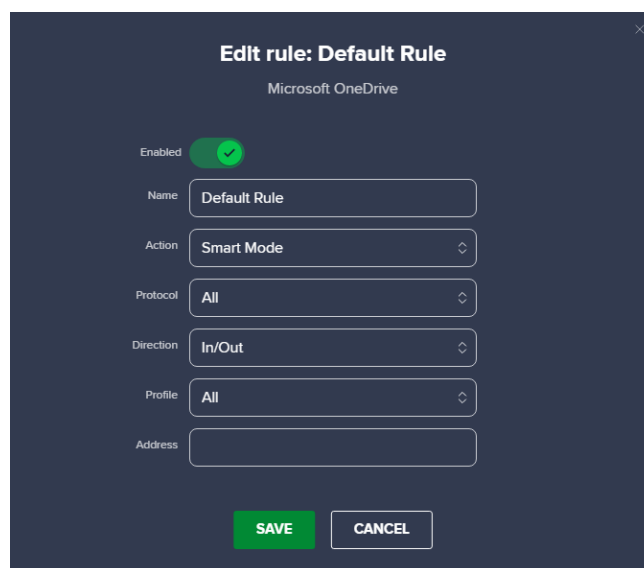
■ **Figure 4.10** Application Excluded from Modification Detection in ESET Internet Security. The figure shows a list of example applications excluded from modification detection. As can be seen, the applications are identified based on their name and location only.

4.3 Avast Free Antivirus

Just like with ESET Internet Security, a valid record about the trusted application OneDrive was created. The exception rule can be found in Figure 4.11. There is also one similarity to the ESET Internet Security – in the default configuration, whole communication for the OneDrive executable is enabled by default. In Avast Free Antivirus, the situation is quite worse as it enables both, incoming and outgoing, communication by default²⁰. This can be seen in Figure 4.12 where the details of “Default Rule” are shown.



■ **Figure 4.11** Rule for Trusted OneDrive Application in Avast Free Antivirus



■ **Figure 4.12** Details of Default Rule for Trusted OneDrive Application in Avast Free Antivirus

In contrast to the solution provided by ESET, the rule created by Avast Free Antivirus does not record any information about the digital certificate used. The reason is that Avast software uses another approach to match the rules.

Unlike the majority of solutions tested, Avast Free Antivirus comes with its own list of trusted certificate authorities. It does not use the host operating system’s certificate management. Instead of the traditional approach, Avast uses its Avast Threat Labs.

Thus, if publishers want to make their product trusted by Avast software, they have to provide the executable to the Avast team for further investigation. If it is not considered malware²¹, the

²⁰ESET Internet Security enables only outgoing in its default configuration

²¹and meets other Avast’s requirements

executable file can be whitelisted. Then, Avast Free Antivirus might not report any issues while using the application. This concept is called File whitelisting. [40]

Another option for the File whitelisting is the approach based on the digital certificate. If the publisher is publicly known, it is possible to request the whitelisting of all executables signed by this publisher. However, obtaining this type of exception may be challenging as it is provided only to a limited set of publishers.

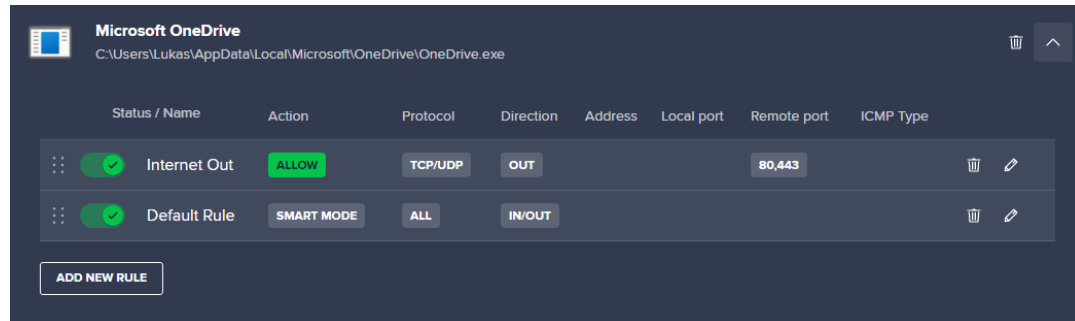
Before the actual testing, the product's settings were changed to ask if a new application initialized network communication. This was done to make sure that the rules for the original (trusted) application were applied.

The first set of tests using this application firewall consists of launching the executable files from untrusted locations. It was found that the application firewall is capable of detecting this type of attack as the product only shows requests to create rules for a new application.

Launching the untrusted executable files from the trusted locations was also tested. Based on the fact that OneDrive is considered a trusted application and probably has the exception set directly in Avast's definitions²², every attempt to execute the untrusted executable from the trusted location was unsuccessful. The program only displays the information that an executable with an untrusted certificate is trying to communicate. Based on that, applications provided in the attachment can be detected in trusted locations.

It was also tested the addition of root certificates of the fake executable files to Certificate Manager. Even though the root certificates were placed to the Certificate Manager, this did not make the attack successful, which corresponds to the philosophy of File whitelisting described in more detail in the previous paragraphs.

However, when the rule for untrusted application is created, it seems almost the same as a trusted one. The only difference that can be seen is the icon of the application. While the trusted OneDrive application shows its icon²³, the rule for untrusted OneDrive application shows only the default icon. This rule can be seen in Figure 4.13.



■ **Figure 4.13** Rule for Untrusted OneDrive Application in Avast Free Antivirus

On the other hand, the situation became interesting for the applications that Avast does not directly whitelist. For those executables, Avast is capable of showing the executable's signer. However, the certificate is always marked as untrusted²⁴. This detection can be seen in Figure 4.14.

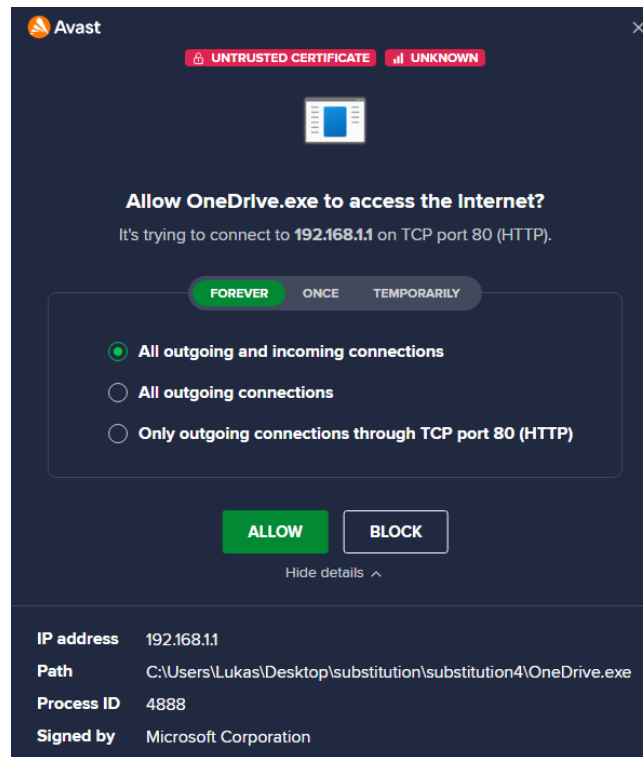
As mentioned earlier, Avast stores only the information about the executable's name and location on the filesystem. No other information was found in the rules. As an example, let us consider an application that Avast does not directly whitelist. Then, this executable can be replaced by any other and match the original rules. In this concept, whether the second application is digitally signed or not does not matter.

In summary, Avast uses quite a different approach from ESET, which is capable of detecting the substitution attack of executables verified by Avast Threat Labs. On the other hand,

²²not managed by regular users

²³a blue cloud

²⁴even in Microsoft Windows operating system can be the certificate verified in a standard way



■ **Figure 4.14** Untrusted Certificate Detection by Avast Free Antivirus

a substitution attack can be performed without any problems for applications considered untrusted by this team.

4.4 Windows Firewall

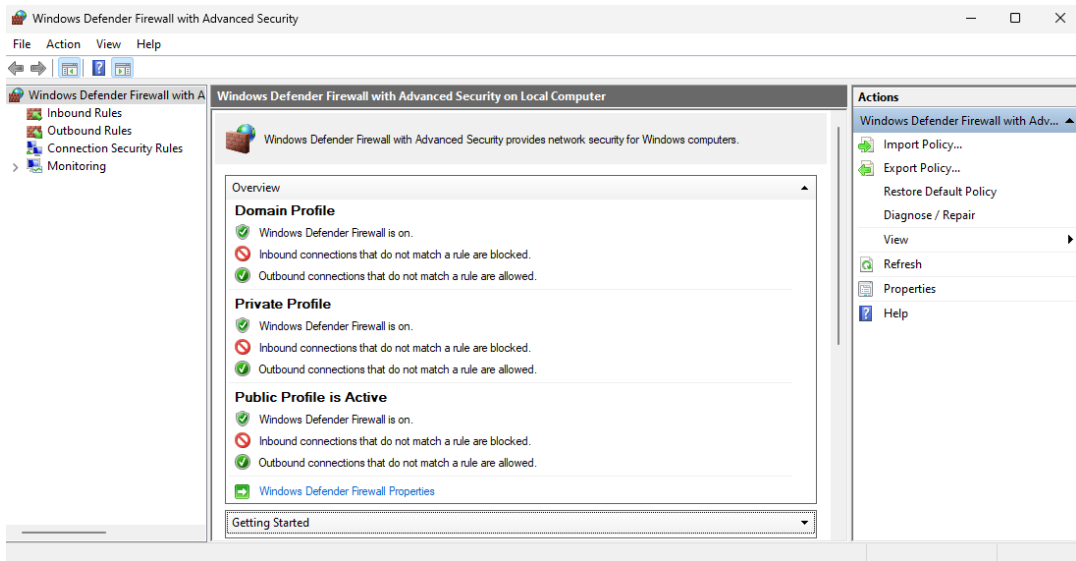
While the Windows Firewall is in operation, the default settings enable all outgoing traffic unless it matches a deny rule and prevent all incoming traffic unless it matches a permit rule. It should be kept in mind that Microsoft recommends keeping this setting to maximize user experience from the system²⁵. These default settings can be seen in Figure 4.15. On the other hand, this approach provides more possibilities for potential attackers. [31]

On the other hand, users can change the firewall's settings to apply a default deny rule and permit only whitelisted applications. It can be done by right-clicking on the "Windows Defender Firewall with Advanced Security" button situated in the left pane of Figure 4.15, selecting "Properties" and then specifying the required settings. Microsoft recommends sufficient testing of the firewall's functionality if the decision to adopt this approach is made. The primary worries from the vendor's side are probably caused by dependencies between the applications and services used within the Microsoft Windows operating system.

During the tests performed in the thesis, the firewall was set to block every communication attempt unless it came from trusted applications. This change in settings was made to determine whether an untrusted application could match the rules initially created for a trusted one.

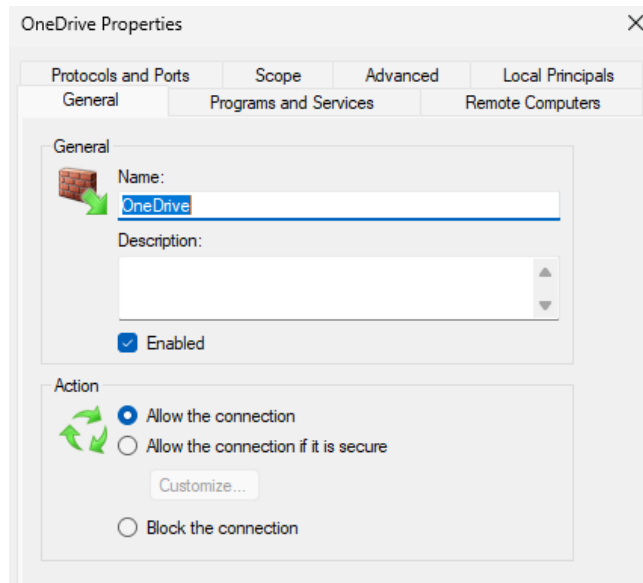
During the testing and manual creation of the new rules, it was seen that the rules are based only on the application name and its path. This can be seen in Figures 4.16 and 4.17 where the rule for the trusted application OneDrive was created. While Figure 4.16 shows

²⁵i.e., preventing communication from trusted applications and associated troubleshooting



■ **Figure 4.15** Default Settings of Windows Firewall

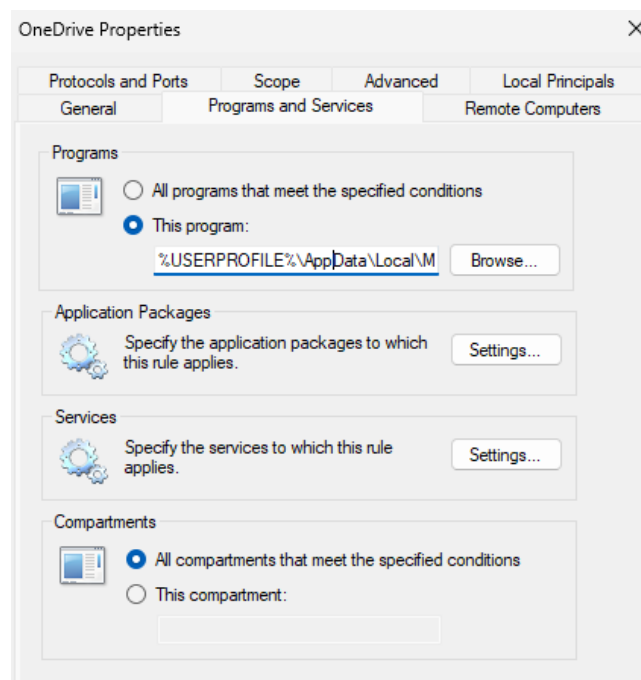
that the communication is enabled for the OneDrive application, Figure 4.17 shows that the rule is identified only based on the executable's name and path. It should be mentioned that the path in Figure 4.17 shows the abbreviation %USERPROFILE% which is later expanded to C:\Users\\.



■ **Figure 4.16** General Tab of Rule for Trusted OneDrive Application in Windows Firewall

The first set of tests is used to determine whether the application firewall can detect the launching of untrusted applications from untrusted locations. From the rule format defined in Windows Firewall, one can think that the attack ends with an unsuccessful. This hypothesis was confirmed during the testing as the attack was always detected.

The situation became more interesting for locations where the application firewall expects the trusted executables. When the trusted one is substituted with the untrusted one in this



■ **Figure 4.17** Details of Rule for Trusted OneDrive Application in Windows Firewall

location, the attack may stay undetected.

The last set of tests is performed after adding the root certificates of untrusted executables to the Certificate Manager. Based on the format of firewall rules, it is not a surprise that this addition did not change the result of the attack – the attack remained undetected from the trusted location.

Thus, if the attacker can write to a trusted location²⁶, it is possible to perform the attack successfully. The firewall does not check any additional properties of the executable.

There is also one consideration that can prevent this type of attack. While having the complex antimalware solution called Windows Security enabled as a whole, it is possible that the anti-malware engine would detect the malicious executable. During this process, the trustworthiness of the code signing certificate can be checked using standard Windows tools and the result can be presented to the users.

On the other hand, the firewall should detect application modification, as its rules can contain additional information to better identify trusted processes.

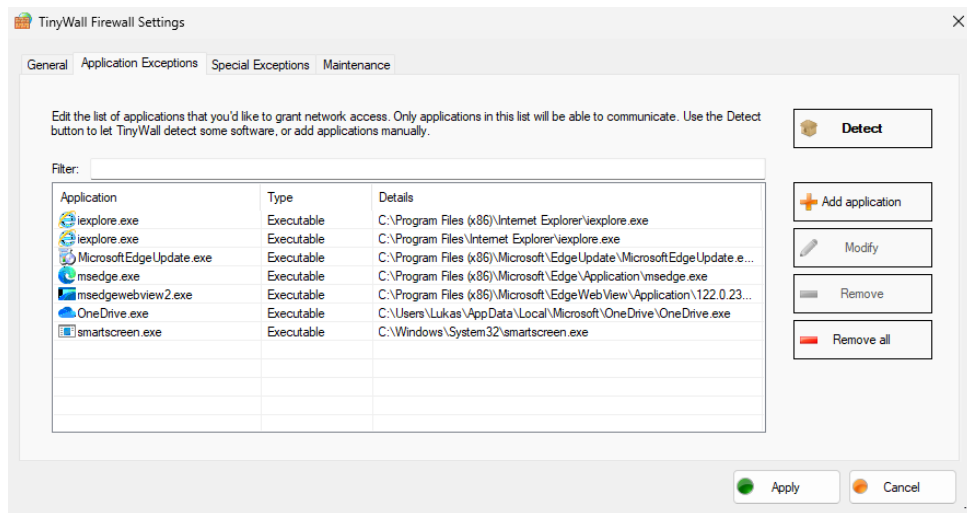
4.5 TinyWall

This application firewall covers only a limited number of applications permitted in the default configuration. Furthermore, new rules for the application must be created manually or by using the Autolearn mode.

Autolearn mode was used to create the rule for the trusted OneDrive application. The rule can be seen in Figure 4.18 alongside only a few other applications included in the default configuration of the product.

The details of the rule for trusted application are then shown in Figure 4.19. As seen from the screenshot, the application is identified only based on the path and name, and the default

²⁶where the firewall expects trusted code



■ **Figure 4.18** Rule for Trusted OneDrive Application in TinyWall

state is to permit communication with any other resource²⁷ on the internet.

From the information obtained from Figure 4.19, it can be clear that no information about the digital signature is used during application identification.

Before describing the actual testing, it should be revised to say that TinyWall does not inform the user about the actual result of rule evaluation. As a consequence, there are no settings that can help with the determination of success as in the previous application firewalls tested. On the other hand, communication can always be eavesdropped on the attacker's device²⁸ to confirm the result of the attack.

Just like the other application firewalls, this one was also tested to detect attacks that substitute only the name of the process. It was found that this attack was unsuccessful.

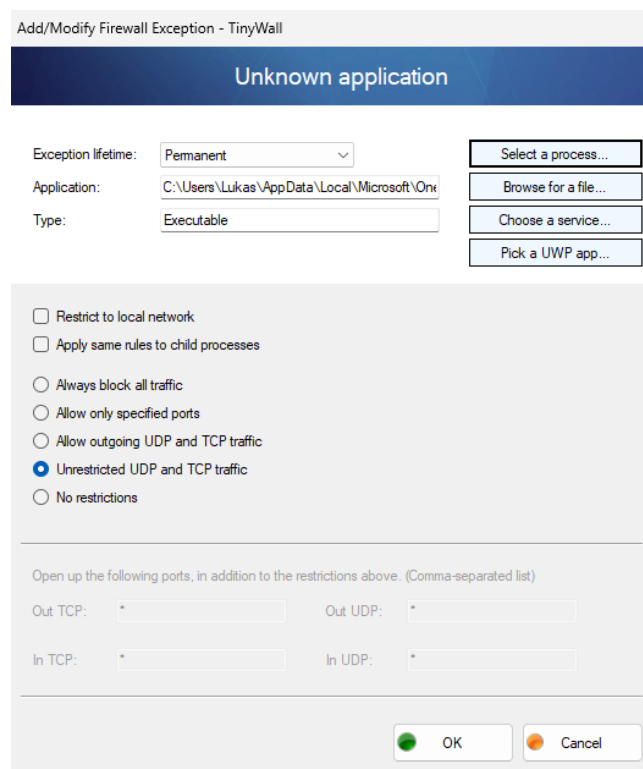
Similarly to Windows Firewall, TinyWall cannot detect the attack where the untrusted application is placed in the location defined in the rule for the trusted one. Then, the attack is undetected.

However, there is one other observation that must be mentioned. While the rule for the trusted application is created (as shown in Figure 4.18), then its executable is substituted with the untrusted one and launched, the actual rule is changed in TinyWall settings as the icon of the executable is changed. Other information remains unchanged. This can be seen in Figure 4.20. Based on this observation, it seems that TinyWall updates its rules every time the application is launched.

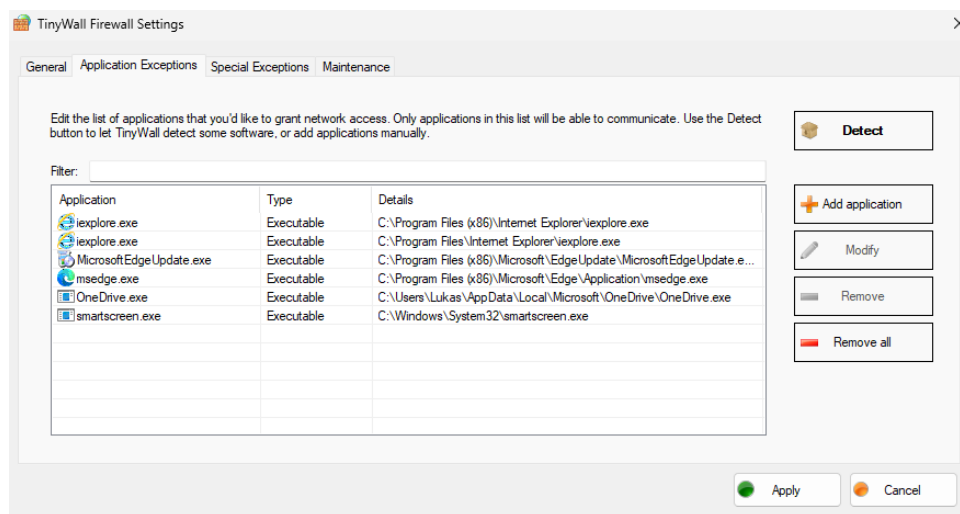
In summarisation of the information provided above, the attack is possible when the attacker obtains access to the locations where TinyWall expects the executable for which the original rule was created. Then, the only thing needed is to place an executable with the same name in this trusted location. TinyWall does not implement any additional checks based on the trustworthiness of the signer or publisher of the executable file.

²⁷i.e., any IP address and port number

²⁸using Wireshark software, for instance



■ **Figure 4.19** Details of Rule for Trusted OneDrive Application in TinyWall



■ **Figure 4.20** Rule for Untrusted OneDrive Application in TinyWall

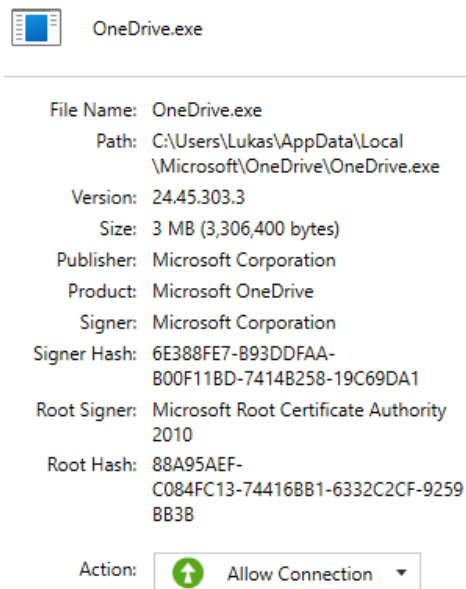
4.6 ZoneAlarm Firewall

As mentioned earlier in the thesis, in ZoneAlarm Firewall, regular users have only a minimal possibility of setting up the product. Another aspect is that all the outgoing communication is enabled by default, and users can only blacklist applications that had been at least once communicating in the past.

Based on the information provided in the previous paragraph, this type of attack is always possible as a new permit rule is created for every application that initializes the communication from the device. When a new rule is created, users can only change a switch and thus deny the communication. As it can be seen in Figure 4.21, on the bottom of the image, there is a button saying “Allow Connection”. Then, switching this one to “Block Connection” is possible.

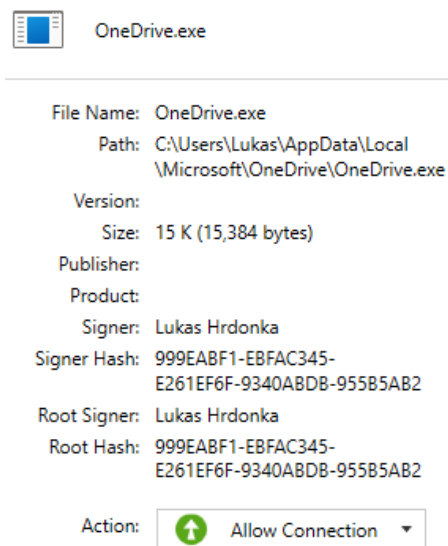
As shown in Figure 4.21, ZoneAlarm Firewall seems to be capable of working with digital signatures as in the valid rule of the trusted OneDrive application, information about the executable’s signer and root signer are presented alongside their hash values. Thus, when another application²⁹ is placed instead of this one, the new rule is created and the communication is then permitted. This situation is shown in Figure 4.22 where the executable file is signed by “Lukas Hrdonka”.

As previously mentioned, all types of attacks tested in this part of the thesis were performed with success. The worst thing behind this type of attack is that no option was found in the product that was capable of detecting application modification on the filesystem level.



■ **Figure 4.21** Rule for Trusted OneDrive Application Definition in ZoneAlarm

²⁹different version, signer, root signer, etc.



■ **Figure 4.22** Rule for Untrusted OneDrive Application Definition in ZoneAlarm

Injection

In this chapter, application firewalls are tested to determine whether they can detect the injection of untrusted code into the trusted application. At the beginning of the chapter, there is a detailed description of the attack alongside an explanation of critical parts of the code used for exploitation. This malicious code is then launched on the device running the above-described application firewalls.

5.1 Attack Description

To explain this attack, let us consider having some trusted application. In most cases, it should be some web browser or some native application installed in the operating system. Based on the information provided in the previous chapters, when these applications are launched, at least one process is created¹. When considering the Microsoft Edge web browser, several processes named `msedge.exe` start, as well as `msedgewebview2.exe`² when launching the application. [41]

As a representative of native Windows applications, we can use File Explorer and its process named `explorer.exe`. Process `explorer.exe` has a significant property for attackers – if the computer with Microsoft Windows is running, there is almost always at least one such process. If there is no such process, users typically encounter a blackscreen³.

The principle of this attack is thus to abuse these well-known and trusted applications to run malicious code in their context. Then the communication may seem like an actual part of the trusted application wants to communicate, not the malicious one.

It may be possible to gain access to the trusted process. In Microsoft's language, it means to create a handle for the process. While opening the handle, the requested rights have to be provided. It should be mentioned that opening the handle to the process has legitimate usage as debuggers often use it.

At this moment, it is possible to allocate additional memory for the process. Theoretically, machine code can be written into this allocated memory space and run. The attack described in this thesis shows another option, as the allocated space is abused to link a malicious DLL library. It means that the further malicious code is written in this dynamically linked library, which may then run malicious code on the device. For instance, it can initialize network communication with the attacker's servers.

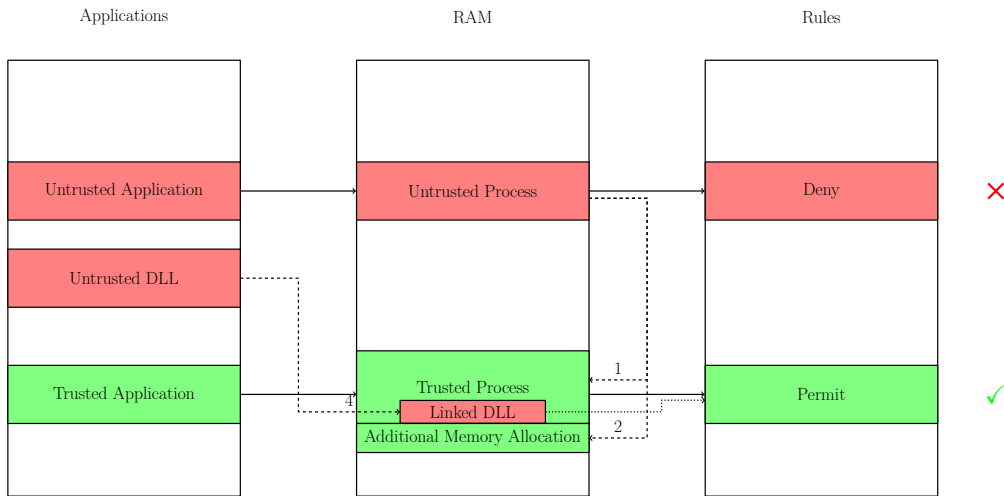
The principle behind this attack is summarized in Figure 5.1.

However, some aspects can prevent the attack. These are listed below.

¹these can be seen in the Task Manager, for instance

²specialized process responsible for additional features not only for Microsoft Edge but other Microsoft applications as well

³due to corrupted system files or incorrect Windows configurations, for instance



■ **Figure 5.1** Injection Attacks Principle. The figure shows the core principle behind this attack. In an ideal world, only solid lines are applied. The attacker’s goal is to perform communication using the permit rule on the firewall, as shown by the dotted line in the picture. In order to do so, attacker has to open the handle to process, expand its memory space and link the malicious DLL. These steps are demonstrated using dashed lines with numbers which correspond to Code listing 5.1 described below in the thesis.

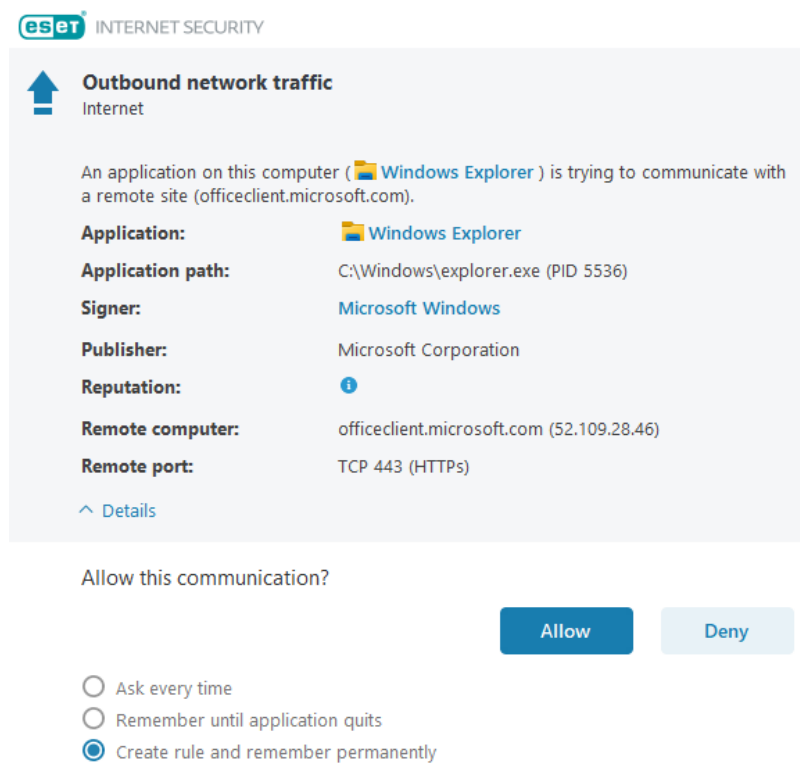
- Handle creation. The handle to the trusted process may not be created successfully. During the testing, it was found that the `OpenProcess` function returned the error code 5 (0x05) when the handle had not been created. This error code was obtained using the `GetLastError` function. The mentioned error code generally means that access to the specified process was denied (`ERROR_ACCESS_DENIED`). This behavior was seen while accessing the system processes and processes that were launched using another user account. According to the official documentation of the `OpenProcess` function, this is expected behavior. The documentation also says: “If the specified process is the System process or one of the Client Server Run-Time Sub-system (CSRSS) processes, this function fails and the last error code is `ERROR_ACCESS_DENIED` because their access restrictions prevent user-level code from opening them.” [42] This behavior was also seen during the testing. [43]
- Additional memory allocation. Some applications have built-in mechanisms that can recognize unauthorized access to their memory and thus prevent the allocation of additional memory space.
- Unsuccessful DLL linkage. Even though the `CreateRemoteThread` function returns handle to the created thread⁴, the actual linkage of DLL could be unsuccessful. This situation is also described in the documentation of the `CreateRemoteThread` function: “Note that `CreateRemoteThread` may succeed even if `lpStartAddress` points to data, code, or is not accessible. If the start address is invalid when the thread runs, an exception occurs, and the thread terminates.” [44] It should be mentioned that the `lpStartAddress` represents the address of the `LoadLibraryA` function in Code listing 5.1. This behavior was especially seen when trying to inject the DLL into the code of the most popular web browsers.
- Firewalls’ application modification detection. Application firewalls should implement mechanisms that recognize a modification of the trusted application. In an ideal world, the application firewall should be the last point of attack detection in case the others fail.

⁴signaling the success of the function

These limitations were seen during the testing as some processes are thus not vulnerable to this type of attack. The applications installed by default within the operating system were tested primarily as no additional action is needed from the users' side. Based on that, the probability of a successful attack is higher.

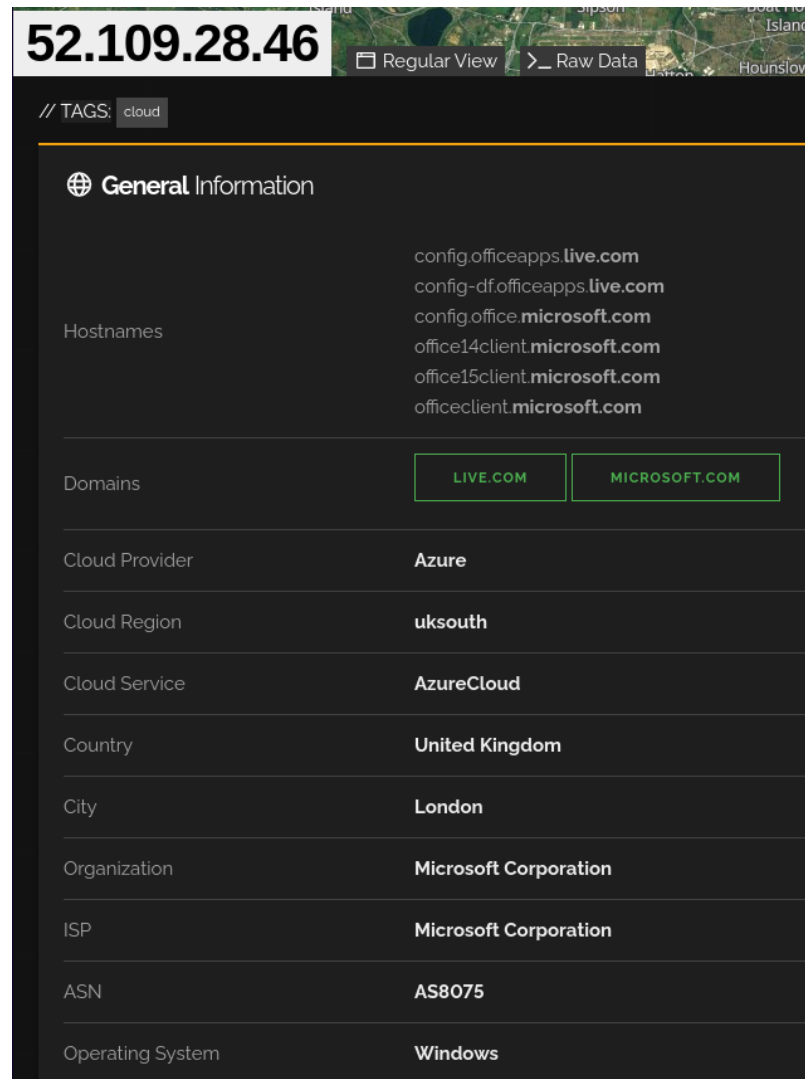
While having all the native Windows security features disabled, it was found that all of the following processes are vulnerable to injection attacks – `explorer.exe`, `msedgewebview2.exe`, `msteams.exe`, `cmd.exe`, `powershell.exe`, `svchost.exe`, `dllhost.exe`, `OneDrive.exe`, `notepad.exe`, and even more. In consequence, attackers may run any code within the vulnerable process, of course, with the rights of the vulnerable process's owner.

Based on the listing of processes in the previous paragraph, one can suspect that processes such as `notepad.exe` or `explorer.exe` need not communicate with other resources on the internet. Surprisingly, while testing, it was observed that at least process `explorer.exe` initiated a legitimate network connection to the Microsoft server with IP address 52.109.28.46. This communication was seen while testing ESET Internet Security and the screenshot of the communication request is shown in Figure 5.2. According to Shodan⁵, Microsoft Corporation uses the IP address mentioned. Information obtained from Shodan can be seen in Figure 5.3.



■ **Figure 5.2** Legitimate Communication of Windows Explorer

⁵search engine used to gather information about devices connected to the internet, available from <https://www.shodan.io/>



52.109.28.46

// TAGS: cloud

General Information

Hostnames	config.officeapps.live.com config-df.officeapps.live.com config.office.microsoft.com office14client.microsoft.com office15client.microsoft.com officeclient.microsoft.com
Domains	LIVE.COM MICROSOFT.COM
Cloud Provider	Azure
Cloud Region	uksouth
Cloud Service	AzureCloud
Country	United Kingdom
City	London
Organization	Microsoft Corporation
ISP	Microsoft Corporation
ASN	AS8075
Operating System	Windows

■ **Figure 5.3** Information about IP Address 52.109.28.46 in Shodan

Besides the applications mentioned above, various web browsers were also tested. The test included Google Chrome version 123.0.6312.59, Mozilla Firefox version 124.0.1, and Microsoft Edge version 122.0.2365.92. During the testing, it was discovered that most of their processes were not vulnerable to this type of attack. However, there can be some processes that are still vulnerable. Thus, the attack is possible, at least in theory, but it is not guaranteed that it is always successful⁶.

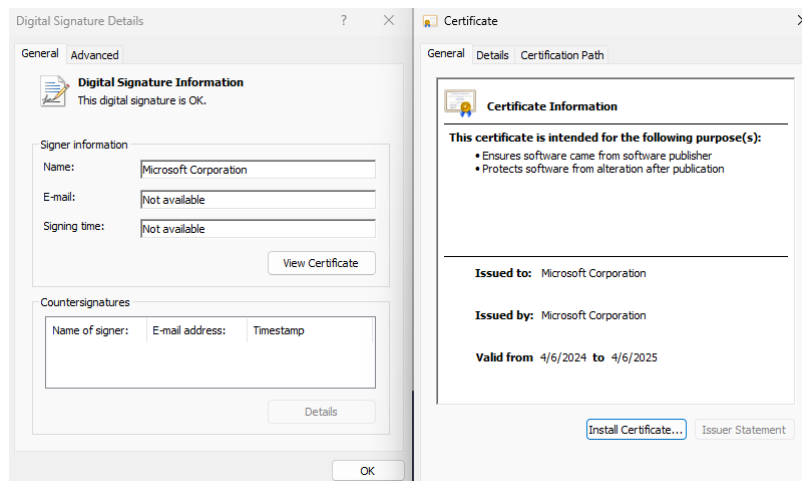
According to the blog post on Microsoft's website, the vendor implemented protection against DLL injection into the Microsoft Edge application in 2015⁷. Since then, every DLL linked into the Microsoft Edge must be Microsoft-signed or WHQL-signed. WHQL⁸ is used to certify Windows-compatible devices and ensure the legitimacy of their drivers. Even though no official information about implementing such a mechanism in other web browsers was found, it is more than likely that they have implemented it as well. [45, 46]

⁶as it is for Windows processes listed above

⁷more precisely, alongside the introduction of EdgeHTML 13

⁸Windows Hardware Quality Labs

This observation was also included in the tests as the malicious DLL was signed using a fake certificate of Microsoft Corporation created in Chapter 4. For the actual signing, the `signtool.exe` was used as described in Code listing 4.2 in the previous chapter. The digital certificate was then added to the relevant store of the Certificate Manager⁹. The validation of the digital signature alongside the certificate details can be seen in Figure 5.4. This DLL was tested while injecting `msedge.exe` process but the digital signature did not make the attack successful as the DLL was not linked to the `msedge.exe` process.



■ **Figure 5.4** Certificate of Signed DLL

5.1.1 Code Explanation

Based on the information above, the actual malicious code is divided into two parts. While the first one is responsible for the injection into the trusted process, the code of DLL is the actual malicious code.

5.1.1.1 Inject.cpp

The code listed in Code listing 5.1 shows the main principle of injection to the code of the trusted application.

```

1 HANDLE hProcess = OpenProcess(PROCESS_VM_WRITE | PROCESS_VM_OPERATION |
  ↳ PROCESS_CREATE_THREAD, FALSE, pid);
2 auto p = VirtualAllocEx(hProcess, nullptr, 1 << 12, MEM_COMMIT | MEM_RESERVE,
  ↳ PAGE_READWRITE);
3 auto w = WriteProcessMemory(hProcess, p, argv[2], strlen(argv[2]) + 1,
  ↳ nullptr);
4 auto hThread = CreateRemoteThread(hProcess, nullptr, 0,
  ↳ (LPTHREAD_START_ROUTINE) GetProcAddress(GetModuleHandle(L "kernel32.dll"),
  ↳ "LoadLibraryA"), p, 0, nullptr );

```

■ **Code listing 5.1** Process Injection

⁹decribed in more detail in Section 4.1.2.1

The first line shows the handle opening for the trusted process. The actual process is defined by its process identifier (PID), which is used as the last parameter of the `OpenProcess` function. The first attribute of the function represents the permissions needed to perform the attack successfully. In this case, the `PROCESS_VM_WRITE` and `PROCESS_VM_OPERATION` are needed to make the necessary changes in the memory space of the trusted process. Then, the `PROCESS_CREATE_THREAD` is needed to create a new thread in the context of the target process. [42, 47]

When the handle to the process is obtained, the code tries to allocate additional memory space by using the `VirtualAllocEx` function on line 2. As a third parameter, the `VirtualAllocEx` function expects the actual size of additional virtual memory space to be allocated. In this case, the value of 2^{12} is used¹⁰. This value corresponds to 4096 Bytes, which is the default size of a memory page used in the 64-bit architecture of the Microsoft Windows operating system. As the fourth parameter, the type of memory allocation is expected. In Code listing 5.1, a combination of `MEM_RESERVE` and `MEM_COMMIT` is used. Whereas `MEM_RESERVED` is used to reserve a range within the virtual address space, `MEM_COMMIT` commits the reserved memory pages. `MEM_COMMIT` also ensures that the initial content of allocated memory is filled by zeros. It should be mentioned that the physical pages are not allocated until the virtual addresses are accessed. As the last parameter, memory protection specification is needed. `PAGE_READWRITE` is used in this example as there is a need to write some data into the allocated address space and then read these data when creating a new thread. [48, 49]

The absolute path to the malicious DLL is then written in this allocated memory. In this case, the path is entered as a second parameter of the executable.

The last line of the listing is quite complicated, so it is described in more detail. The `CreateRemoteThread` function is used to create a new thread that runs in the address space of another process. As a fourth argument, the function expects an entry point to the function to be executed. In this scenario, the function `LoadLibraryA`, whose address is obtained from `kernel32.dll` with the help of the `GetModuleHandle` function, is used. It has to be mentioned that the `kernel32.dll` is linked to each process running on the Microsoft Windows operating system, and `LoadLibraryA` function is included in it. As the fifth parameter of the `CreateRemoteThread`, there is the newly allocated memory space address containing the absolute path to malicious DLL. This one is used as a parameter to the function defined as the fourth argument. [44]

Eventually, a new thread is created. This thread runs the `LoadLibraryA` function. Thanks to this, the malicious DLL is loaded into the context of the trusted process.

As part of the thesis, the executable version of this source code is provided¹¹. It can be run on a Microsoft Windows 11 device using the command line interface. The application expects the PID of the trusted process as the first parameter and then the absolute path to the malicious DLL library as the second one.

5.1.1.2 Dllmain.cpp

The code of `dllmain.cpp` gives an example of malicious DLL. In this particular case, the `Winsock2` library is used to create a communication channel with the attacker's server. An example `Winsock` client code available on the Microsoft site was used as a base for this code. [50]

The main modification is that this code waits in its while loop for the commands from the attacker. These commands are launched using the `WinExec` function. In the example scenario, the attacker can send the message "`calc.exe`" and this application is then launched on the victim's device. Consequently, complete code is a straightforward version of the reverse shell.

The compiled DLL library is presented in the thesis attachment¹², too. The DLL is currently

¹⁰written with the use of binary left shift operator

¹¹file `Inject.exe` in the attachment

¹²file `DLLSocket.dll` in the attachment

preconfigured to initiate a communication channel with server 192.168.1.1:80. This parameter can be changed in the provided source code, and then a new version of DLL can be created using Microsoft Visual Studio 2022 built-in tools.

5.1.1.3 Server.cpp

As the name suggests, the source code of `server.cpp` is used to run on the attacker's machine. Its purpose is to listen to the specified port on any device's IP address and process the requests. For simplicity, the server will read the request from the client and reply with the string "calc.exe". As written in the previous parts, this string is then used as a parameter of the `WinExec` function on the client side.

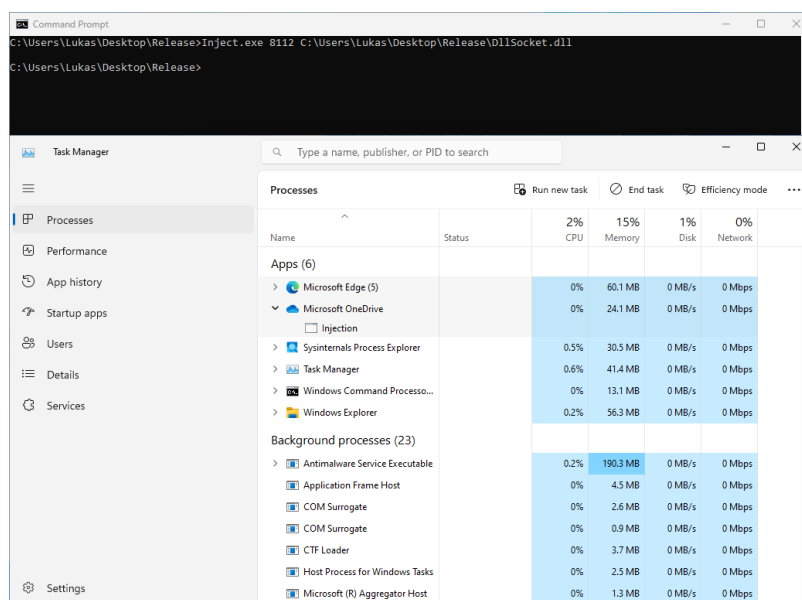
Actual executable binary code is included as the attachment of the thesis¹³. It is possible to run the application using the command line interface of the Linux distribution. The executable expects one parameter – the port on which the server listens.

5.1.2 Practical Demonstration

In this section, a practical demonstration of the attack is shown alongside the screenshots that provide proof of a successful attack. The binary files used in the demonstration can be found in the thesis attachment.

The first step of the attack is to identify the PID of the trusted process. This information can be obtained from the Task Manager on the "Details" tab. As mentioned above, several processes are vulnerable to this type of attack. `OneDrive.exe` was decided to be used for demonstration purposes as well as in the previous chapter.

While having the PID of the trusted process noted, `Inject.exe` can be launched from the command line interface. From the upper part of Figure 5.5, one can see its actual launching. The first argument represents the PID of trusted process¹⁴, and the second one is the absolute path to the malicious DLL. Process `OneDrive.exe`, with its PID 8112, is used there.



■ **Figure 5.5** Launching the Injection Attack and Verification of the Result in the Task Manager

¹³file `Server.out` in the attachment

¹⁴obtained from the previous steps

When the code is successfully launched, a message box appears in the context of the trusted application. This can be seen in the lower part of Figure 5.5 as there is the application named “Microsoft OneDrive” and its context window named “Injection”.

It is also possible to verify the result of linking of the malicious DLL in the Process Explorer. The Process Explorer is software developed by Sysinternals that provides more detailed information about processes running on a particular device. The newest version of the application can be obtained from the vendor’s site. [51]

As can be seen in Figure 5.6, there is a highlighted row¹⁵ with process `OneDrive.exe` and its PID 8112. Further, in the listing of linked DLLs, our `DLLSocket.dll` is shown.

The screenshot shows Process Explorer with the 'DLLs' tab selected for the `OneDrive.exe` process (PID 8112). The process list shows `OneDrive.exe` with 36,156 K private bytes and 694 Sysinternals Process Explorer company name. The DLL list below shows various system and application DLLs, with `DLLSocket.dll` highlighted in blue.

Name	Description	Company Name	Path
DataExchange.dll	Data exchange	Microsoft Corporation	C:\Windows\System32\DataExchange.dll
DateTimePicker.dll	React Native Date Time Picker for...	Microsoft native community	C:\Users\Lukas\AppData\Local\Microsoft\OneDrive\24.04...
dcocmp.dll	Microsoft DirectComposition Library	Microsoft Corporation	C:\Windows\System32\dcocmp.dll
dhcpcsvc.dll	DHCP Client Service	Microsoft Corporation	C:\Windows\System32\dhcpcsvc.dll
dhcpcsvc6.dll	DHCPv6 Client	Microsoft Corporation	C:\Windows\System32\dhcpcsvc6.dll
DiagnosticDataSettings.dll	Microsoft Windows Diagnostic Dat...	Microsoft Corporation	C:\Windows\System32\DiagnosticDataSettings.dll
directmanipulation.dll	Microsoft Direct Manipulation Com...	Microsoft Corporation	C:\Windows\System32\directmanipulation.dll
DirectXAppx.sdb			C:\Windows\appxhost\DirectXAppx.sdb
directxdatasehelper.dll	DirectXDatabaseHelper	Microsoft Corporation	C:\Windows\System32\directxdatasehelper.dll
DLLSocket.dll			C:\Users\Lukas\AppData\Local\Microsoft\OneDrive\24.04...
dnsapi.dll	DNS Client API DLL	Microsoft Corporation	C:\Windows\System32\dnsapi.dll
dpapi.dll	Data Protection API	Microsoft Corporation	C:\Windows\System32\dpapi.dll
dsreg.dll	AD/AAO User Device Registration	Microsoft Corporation	C:\Windows\System32\dsreg.dll
dsmanapi.dll	Microsoft Desktop Window Manag...	Microsoft Corporation	C:\Windows\System32\dsmanapi.dll
DWrite.dll	Microsoft DirectX Typography Serv...	Microsoft Corporation	C:\Windows\System32\DWrite.dll
DXCORE.dll	DXCore	Microsoft Corporation	C:\Windows\System32\DXCORE.dll
diag.dll	DirectX Graphics Infrastructure	Microsoft Corporation	C:\Windows\System32\diag.dll
edpui.dll	EDP UI	Microsoft Corporation	C:\Windows\System32\edpui.dll
FamilySafetyExt.dll	FamilySafety ChildAccount Extensi...	Microsoft Corporation	C:\Windows\System32\FamilySafetyExt.dll
FileSync.LocalizedResources.dll	Microsoft OneDrive	Microsoft Corporation	C:\Users\Lukas\AppData\Local\Microsoft\OneDrive\24.04...
FileSync.LocalizedResources.dll	Microsoft OneDrive	Microsoft Corporation	C:\Users\Lukas\AppData\Local\Microsoft\OneDrive\24.04...
FileSync.LocalizedResources.dll	Microsoft OneDrive	Microsoft Corporation	C:\Users\Lukas\AppData\Local\Microsoft\OneDrive\24.04...
FileSync.LocalizedResources.dll	Microsoft OneDrive	Microsoft Corporation	C:\Users\Lukas\AppData\Local\Microsoft\OneDrive\24.04...
FileSync.Resources.dll	Microsoft OneDrive	Microsoft Corporation	C:\Users\Lukas\AppData\Local\Microsoft\OneDrive\24.04...

Figure 5.6 Listing of Linked DLLs in the Process Explorer

The whole client-server communication can be captured using Wireshark software. Its output is shown in Figure 5.7. The highlighted line shows the command "`calc.exe`", which is sent from the server to the client. Captured communication is provided as an attachment to the thesis, too.

The screenshot shows a Wireshark capture of network traffic. The packet list pane shows several TCP segments. Packet 10 is highlighted in blue, showing a segment from 192.168.1.100 to 192.168.1.100. The packet details pane shows the payload of this segment as `0000 08 00 27 ad 53 a3 0a 00 27 00 00 00 00 00 45 00`. The packet bytes pane shows the hex and ASCII representation of the data, with the command `calc.exe` visible in the ASCII column.

Figure 5.7 Communication Captured by Wireshark

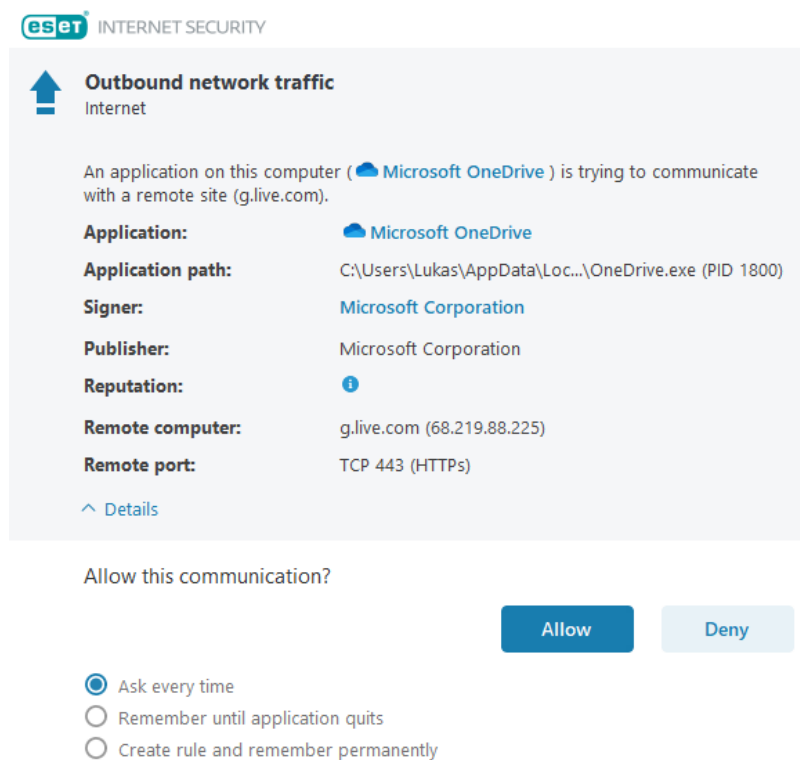
¹⁵row number 3 in the upper part

5.2 ESET Internet Security

The ESET Internet Security product includes the first application firewall tested in this chapter.

Just like in the previous chapter, the rule for trusted OneDrive application was created. This communication request can be seen in Figure 5.8. It can be seen that the OneDrive executable tries to communicate with remote host 68.219.88.225 using port number 443. Based on the information from the screenshot, the IP address 68.219.88.225 is used by Microsoft Corporation¹⁶ and the communication seems to be enciphered by using TLS which was discussed in more detail in Section 1.4.1.3.

While referring to Figure 5.8, changing the switch to “Create rule and remember permanently” and hitting the “Allow” button results in the creation of a rule such as the one shown in Figure 4.7.



■ **Figure 5.8** Communication of Trusted OneDrive Application in ESET Internet Security

In the beginning, the mentioned code was tested using the default configuration of the product. The test discovered that the ESET’s application firewall is unable to prevent this type of attack in the default configuration settings.

The “Application modification detection” feature can be¹⁷ used to prevent this type of attack. During the testing, it was discovered that this detection is insufficient for this type of attack as it probably checks for changes only at the start of the trusted application. Consequently, it is possible to successfully perform the attack even if the “Application modification detection” is enabled.

Another important thing that has to be mentioned is that in the default configuration, “Application modification detection” is enabled but only for applications without valid certificates.

¹⁶also verified using Shodan

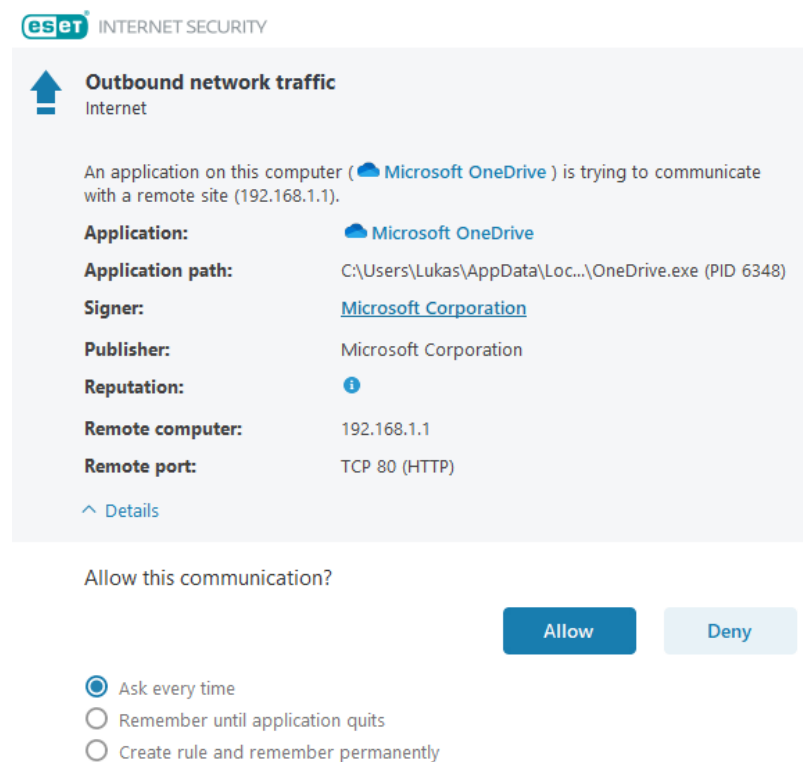
¹⁷at least according to the vendor

If the application has a valid certificate at the time of rule creation, when later applying the actual rule it is only checked whether the same digital certificate is used in the digital signature¹⁸.

Various product settings were also tested, such as the aggressivity of the detections or additional detection mechanisms. However, there was no such option in the firewall's settings found that is able to prevent this type of attack.

Another aspect related to the topic is the actual information about the untrusted application when creating the rule. As shown in Figure 5.9, it can be seen that despite the fact the `OneDrive.exe` was injected, there is no information regarding reduced trustworthiness. This screenshot was obtained after the rule for the trusted OneDrive executable had been deleted. The figure thus simulates the initial communication of the injected executable.

From the regular user perspective, it seems that the legitimate application, which is digitally signed with a valid certificate of Microsoft Corporation, would like to communicate with some remote address. The only suspicious thing is that the remote computer is set to `192.168.1.1`. In reality, an attacker should use some public IP address. On the other hand, regular users typically have a minimal understanding of IP addresses and are typically unable to verify their trustworthiness¹⁹. Another aspect is that doing this check for every communication attempt could be demanding and time-consuming. Thus, end users could consider this communication legitimate and create a permanent rule.



■ **Figure 5.9** Communication of Injected OneDrive Application in ESET Internet Security

¹⁸vendor arguments with potential problems with updating the trusted application

¹⁹i.e., whether it is maintained by a trusted party

5.3 Avast Free Antivirus

In this section, the result of Avast Free Antivirus testing on the vulnerability of injection type is given.

First of all, it was verified that the rule for the trusted OneDrive application is included in the product's list of rules. The same rule for trusted application was used as in Figure 4.11.

Then, just like the other implementation, the code was initially launched while the default settings were set in the product. The attack was not successful because the Behavior shield detected it.

A Behavior shield is an additional feature that should provide an additional level of security as it detects anomalies in trusted binaries. It was originally introduced in 2017 and has been included in all Avast antimalware products since then. On the other hand, at least at its beginning, there was a higher false positive (FP) rate, as may be seen on Avast's forums. However, over time the FP ratio started decreasing as the detection mechanism was about to begin using artificial intelligence for behavioral detection. It saves the default behavior of trusted applications during the initial phase²⁰ and then compares this with the actual state. [52]

This feature is enabled by default in the product. However, it is still possible to turn off the feature. It can be done in the "Protection tab", by selecting "Core Shields" and then toggling the button under "Behavior Shield". During the testing, this feature was temporarily disabled to prove that the Behavior shield is the only mechanism that can detect the attack.

Another option to a disable specific shield for a particular application is using the "Exceptions". The actual settings can be found under "Settings" on the "General" tab. It should be mentioned that the actual exceptions are identified using the name and path of the executable. As can be seen in Figure 5.10, several shields could be excluded for a particular executable. These are explained in the listing below. [53, 54]

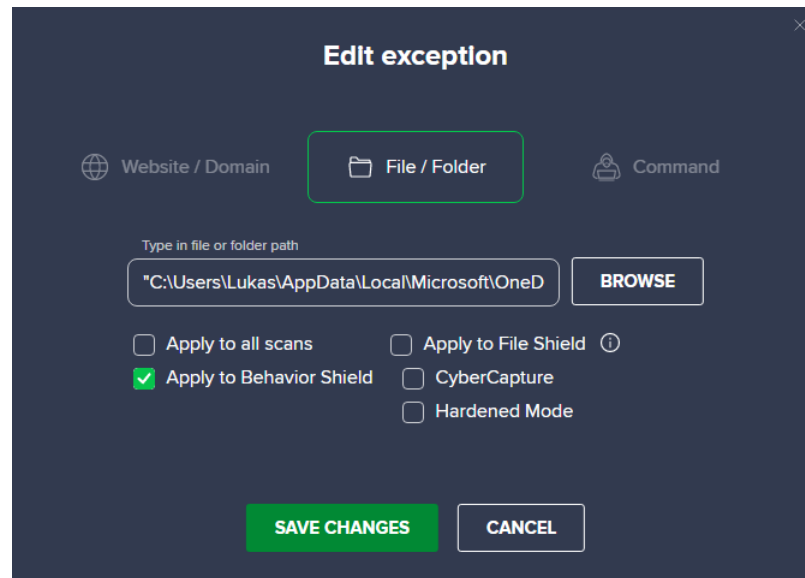
- Apply to all scans. By selecting this one, all the scans of the defined executable are disabled.
- Apply to Behavior Shield. Selecting this one results in disabling the Behavior shield. This means that the executable file defined has no longer been checked for runtime changes.
- Apply to File Shield. By selecting this item, the specified program is not scanned for malicious threats anymore when the file is opened, run or even modified.
- CyberCapture. CyberCapture is used to analyze unrecognized files and warn about new threats.
- Hardened Mode. Hardened Mode should be used to determine which executable files are safe to open based on their reputation.

In Figure 5.11, the actual detection can be seen. The executable file was also moved to some potentially trusted locations, but the only difference was the name of the detection category – from the "IDP.Generic", it changed to "IDP.ALEXA.54". However, no official documentation regarding the detection classes was found. On the other hand, when trying to search for more information about the specified detection class, it can be found that this one has sometimes been considered as FP detection, at least in the past.

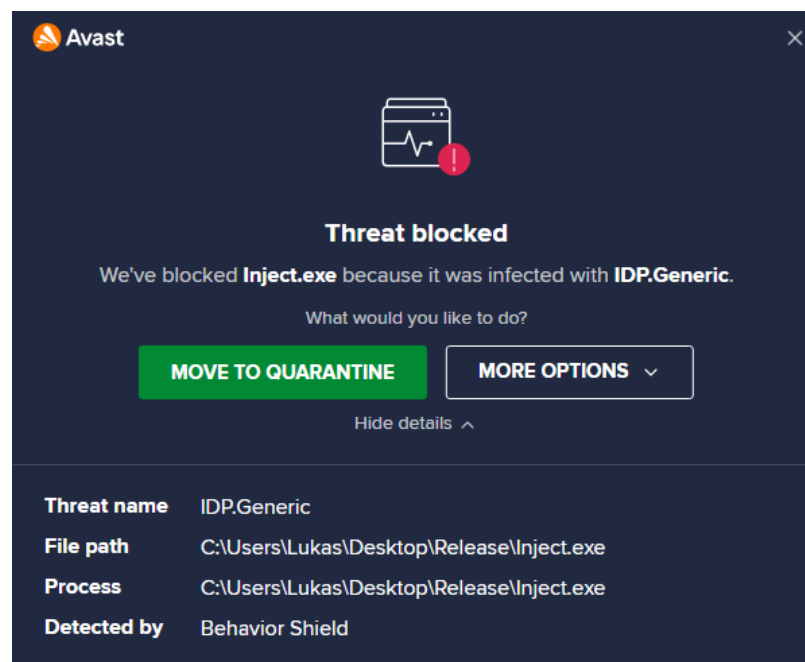
When is the message in Figure 5.11 shown, users can decide whether to move the threat to the Quarantine or create an exception. The exception means that Avast stores the information that this action is expected and updates the expected behavior in its structure.

If the users decide to move the executable to the Quarantine, the file is placed in a special environment where the files are entirely isolated from the rest of the operating system. As a consequence, they cannot be accessed by any other processes. From the users' perspective, the executable disappears from its original location. [55]

²⁰initial scan right after the installation of Avast product



■ **Figure 5.10** Exception Details in Avast Free Antivirus



■ **Figure 5.11** Attack Detection by Avast Free Antivirus

There is one other limitation that should be taken into account. When the users click the “Move to Quarantine” button, they are prompted to restart the computer to complete the action. It was tested that the DLL library is successfully loaded into the address space of the trusted process, and it can communicate with the attacker’s device until the computer is restarted. Thus, if the users decide to restart the computer later, the malicious DLL can run on the device until that moment.

5.4 Windows Firewall

Binaries introduced in the previous sections were tested on the Windows Defender and its firewall, Windows Firewall, too.

Just like the other solutions tested in this chapter, a rule for the trusted OneDrive executable was created in the product. This rule is the same as the one shown in Figures 4.16 and 4.17.

While having default configuration enabled, the attack can be completed successfully. Based on that, attackers can link malicious DLLs into the trusted process and create a communication channel with the attackers' devices without causing any violation from the side of native Windows security tools.

On the other hand, some settings can be helpful in attack prevention. These settings are located in the Windows Security Center on the "App & Browser Protection" tab. When selecting "Exploit Protection", there are two tabs with additional security features. While "Program settings" are applied only to the specified applications, "System settings" are applied to all remaining applications. Based on the fact that all settings relevant to the problematics of injection are enabled on the "System settings" tab by default, further paragraphs are focused on the options under "Program settings".

However, these additional features are enabled for each application separately. Thus, when we consider having ten applications and want to enable the same security feature for all of them, it has to be done in ten places in the product configuration.

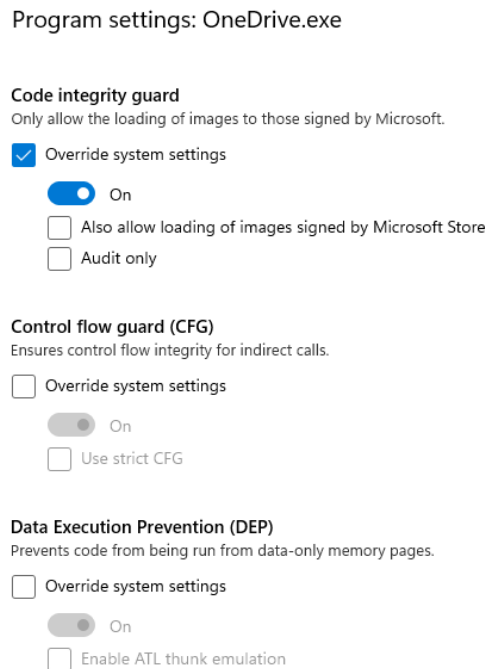
A list of additional security features is provided further in the chapter. These may be used to ensure an additional level of security. It should be mentioned that all of them are disabled by default. [56, 57]

- **Arbitrary Code Guard.** This option should enable additional checks regarding code page modification. However, during the testing, it was found that this additional setting does not prevent this attack.
- **Validate Handle Usage.** An additional check regarding the handle reference is applied when this option is enabled. It may raise an exception if an invalid handle reference is detected.
- **Validate Image Dependency Integrity.** It may be used to detect code substitution for statically linked DLLs by Windows binaries.
- **Code Integrity Guard.** It ensures that all executables loaded into a trusted process are digitally signed by the same publisher. Consequently, if some custom application requires loading its own DLLs to a digitally signed binary, then this guard should prevent the attack. However, it is not intended for all applications²¹. It is the only option that can prevent this type of attack. During testing, it was verified that allocating additional memory is impossible. Consequently, attackers need to find other ways to link malicious DLLs because the procedure described in this chapter cannot be used.

The actual configuration of the Code Integrity Guard for the `OneDrive.exe` can be found in Figure 5.12.

In summary, the attack can be successfully performed in the default configuration of Windows Firewall. There is an additional security feature called Code Integrity Guard that can prevent a successful attack.

²¹for instance, third-party applications may not work correctly while having this feature enabled



■ **Figure 5.12** Configuration of Code Integrity Guard In Windows Security Center

5.5 TinyWall

TinyWall's default configuration has the potential to prevent this type of attack. However, the only thing that helps prevent the attack is that the default configuration contains only a limited set of trusted applications. This set does not include applications that are reliably vulnerable to this type of attack. To explain the word “reliably” in the previous sentence, it should be mentioned that the list includes processes such as `msedge.exe`. As written above, most of Microsoft Edge's processes are not vulnerable to injection attacks, but some may still be vulnerable.

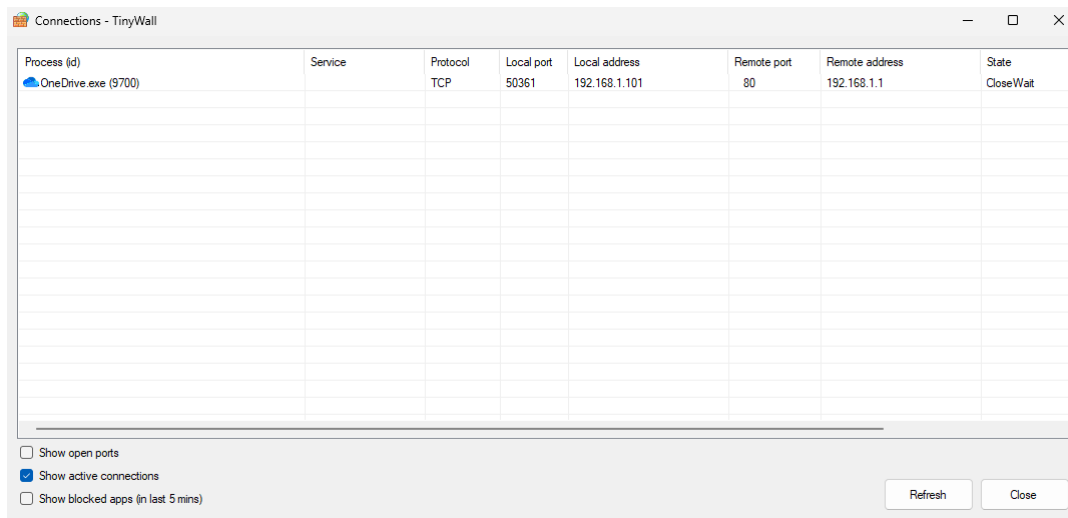
On the other hand, it is still possible to add new rules for applications that users consider as trusted. When applications such as Microsoft Teams or Microsoft OneDrive are added to the exception list, the attack is successfully performed using these applications. The rule created for OneDrive application is shown in Figures 4.18 and 4.19.

Just like in the other solutions tested in this chapter, the attack was performed using the trusted OneDrive executable. The attack's success can be verified using the list of active connections within the Tinywall as shown in Figure 5.13.

The actual settings of the TinyWall firewall are relatively simple. Users thus have only limited possibilities for settings. Consequently, no option regarding the detection of application modification was found in the firewall settings.

Another aspect of this solution is that it is not intended to work with digital certificates. In the competing solutions, it is possible to define application certificate details which are validated when matching the rules. In these solutions, for instance, the certificate's validity and its issuer can be additionally checked. No such option was found in TinyWall's settings.

In summary, the application firewall can, using the default settings, prevent this type of attack to some extent. On the other hand, when applications such as Microsoft OneDrive are added, the firewall cannot detect the attack.

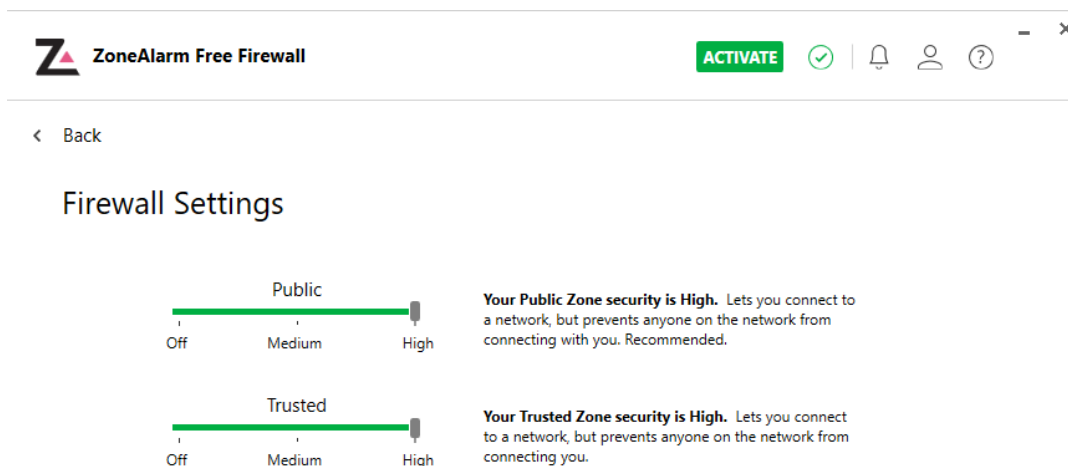


■ **Figure 5.13** List of Active Connection in TinyWall

5.6 ZoneAlarm Firewall

This application firewall comes with only fairly limited configuration as it targets the users who would prefer to spend less time setting up the firewall.

While testing its configuration, it was discovered that the attack was successful using the firewall’s default configuration. As mentioned above, users have only minimal setting options. These only include the specification of network prefixes, which are considered private, and the sensitivity of the firewall. Even if the sensitivity is set to the highest value, this attack remains undetected. These settings can be seen in Figure 5.14.



■ **Figure 5.14** Sensitivity Level Settings in ZoneAlarm Firewall

In this implementation, all outgoing communication is always permitted. Thus, applications are always automatically added as rules with the predefined option “Allow communication” enabled. It can be quite confusing for the regular users. When they would like to block some applications, it has to be performed in a unique way – when the application is added to the list, users can toggle the button to value “Block communication”. Based on these pieces of information, ZoneAlarm Firewall works in the blocklisting mode, which is not a good security principle.

Due to its simple configuration, no attack mitigation option was found. For instance, no options to enable something like detection of application modification can be set.

Thus, this attack is always possible in whatever configuration is deployed, and there is no possible mitigation using this application firewall.

Discussion of Results

In this chapter, the results of the tests described in more detail in the previous chapters are provided alongside their discussion. The chapter also provides a quick overview of firewalls' ability to prevent the mentioned attacks, recommendations for firewall setup, and reactions from the firewall vendors.

6.1 Substitution

Chapter 4 provides a detailed overview of substitution attacks. In this section, the results of the testing are given. As already written, many variants of the attack exist. The actual ability of tested firewalls to prevent particular attack variants is shown in Table 6.1.

Based on the mentioned chapter's information, the testing uncovers how the mentioned application firewalls evaluate their rules. For clarification, it should be reminded that the default settings of tested firewalls were changed to ask if communication from a new application is detected¹.

As seen in Table 6.1, Avast Free Antivirus and ESET Internet Security are capable of detecting all of the attempts described above. One of the pros behind these solutions is that they probably use their own set of trusted root certificates, independent of the ones defined in the operating system. Thus, the user can always be informed about the communication originating from untrusted applications.

In order to prepare a similar environment for testing, the Windows Firewall was set to block all outgoing communication until it matched the allowed rule (as the default settings of the product allow all outgoing communication). While an implicit block for outgoing communication is enabled, the attack remains undetected for locations where Windows Firewall expects a valid executable. Based on that, it is clear that the product uses only the executable's path and name when evaluating the rule.

On the other hand, all three application firewalls described in the previous paragraphs are delivered as a part of complex antimalware products. Thus, the executable may be detected as malicious by the antimalware engine. Then the attack is not completed successfully.

TinyWall uses an approach in which the rule is identified only based on the executable's path and name. Thus, if the attacker obtains access to the location where TinyWall expects a trusted executable, it can be changed to another, and the product cannot detect this change.

¹if this option is available in the product

ZoneAlarm Firewall uses a similar approach to Windows Firewall in its default configuration as it enables all the outgoing communication. The critical difference to Microsoft's solutions is that users cannot change this setting. Based on that, the untrusted executable can be launched from any location on the filesystem and the application can communicate over the network.

6.2 Injection

In Chapter 5, the principle of injection attack and the actual testing in selected application firewalls are provided. In this section, an analysis of the results is given.

Before diving into the analysis, it may be mentioned that the ability of tested firewalls to detect this attack is summarised in Table 6.2 listed below in the thesis. As seen from Table 6.2, most of the application firewalls tested cannot detect this attack.

The core principle of the attack is to open the handle to a trusted process and perform necessary changes to it in order to link a malicious DLL to it. The DLL can then perform network communication and application firewalls often consider this traffic legitimate because it originates from a trusted application.

During the testing, it was found that only Avast Free Antivirus can detect this type of attack. More precisely, its Behavior shield detected the attack. However, the steps after the detection could be handled better as the product moves only the executable performing linking of DLL, not the DLL itself, to the Quarantine. Thus, the malicious DLL can be successfully loaded into the trusted process and run until the computer is restarted. This time could be extremely long for servers in a production environment, for instance.

The Windows Firewall was the next product tested in the thesis. It was found that this application firewall cannot detect this attack in its default configuration. On the other hand, there is an additional check called Code Integrity Guard. While this guard is enabled, the attack cannot be performed successfully. However, the main con is that the Code Integrity Guard must be set up separately and manually for every application.

In other products tested in the thesis, ESET Internet Security, TinyWall, and ZoneAlarm Firewall, the attack can be performed successfully without any detection from the application firewall or even the antimalware solution itself². Furthermore, no additional settings were found to help with the attack detection and, thus, its mitigation.

Overall, only two of five³ application firewalls are capable of detecting this type of attack. However, the detection seems not to be the domain of application firewalls but the antimalware solution behind them. This opinion is based on the observation that both detection engines, the Behavior shield used by Avast and the Code Integrity Guard used by Microsoft, are part of the antimalware program, not the application firewall itself.

6.3 Discussion

During the testing of the substitution attack, it was seen that most of the products behaved as expected.

However, Avast's and ESET's approaches should be highlighted as they use their own set of trusted root certificates. Thus, the attackers cannot bypass the rules by adding the trusted root certificate to the certificate store located directly in the operating system. However, for publishers of new applications, the deployment process can be more difficult as they have to prove that their product is not considered malicious. It usually includes deeper testing on the side of firewall vendors and their malware laboratories.

It was also seen that some of the products enable all outgoing communication in the default configuration. As a Microsoft Windows operating system vendor, Microsoft also recommends this

²if delivered within the product

³only one of five in the default configuration

approach to ensure a seamless user experience from the system. This approach extends the attack surface and thus gives more options for potential attackers. Thus, the probability of a successful attack is also higher. This approach is an excellent example of the contrast between security and user-friendliness of the application. Generally, this dependency says that the application's security decreases with higher user-friendliness and vice versa.

After testing of substitution attack⁴, the thesis also shows an attack which focuses on the change of the trusted process' code while it is running. Based on the testing, a deeper check of the trusted application's integrity seems to happen only when the application is launched. This means that if the application is changed and then launched, it can be detected in most products. On the other hand, when some changes are made during the trusted application run, these changes often remain undetected. Thus, the application firewall does not check which part of the trusted application communicates within the internet.

In this discussion, the detection used by Avast Free Antivirus should be highlighted as this product is the only one that can detect this type of attack in its default configuration. As the product was tested, it was seen that its part called Behavior shield, which uses ML and AI principles, can detect this attack with certainty. On the other hand, the product detects only the untrusted executable file, not the malicious DLL itself. Thus, the main recommendation while using this product is to follow the instructions given by the product. If those are followed⁵, the product fully protects the device against this attack. On the other hand, when the instructions are not followed⁶, the attack can continue.

Furthermore, the ability to detect attacks from the side of the Windows Firewall may be discussed, too. Even though the firewall is not able to detect the attack in its default configuration, an additional setting called Code Integrity Guard can be helpful with the detection. The main disadvantage behind this setting is that it must be done for each application separately, as no setting in the configuration was found to enable it globally for any application running in the operating system. While Code Integrity Guard is enabled, the attack can be detected, although the setup can be pretty annoying. It would be better if the vendor provided the possibility to enable this one globally.

At this point, it should be mentioned that the Behavior shield and the Code Integrity Guard are not the features of application firewalls but the antimalware program as a whole. Consequently, the attack is not detected by the application firewall but by the more complex antimalware solution. This detection should be done directly in the application firewall during the rules evaluation.

Other application firewalls tested in the thesis could not detect this attack. Even more, no settings in their configuration were found that could help with attack detection. Based on that, the attack can be successfully performed in these firewalls without any notification to the user. As the users cannot prevent this attack by changing the product configuration, the only possible solution is to appeal to firewall vendors to consider this vulnerability and prepare adequate patches for their products.

As has already been mentioned in the thesis, all of the vendors were confronted with the test results. Their reaction can be found in a separate section further in the chapter.

Based on the information above, it seems the tested programs can only detect a situation where the executable is changed before the initial launch. While it is running, changes to its code can be done using the principles of injection. Except for Avast Free Antivirus, this attack ends with success in the firewall's default configuration.

⁴considered as being detected in expected way

⁵means restarting the device after attack detection

⁶it may be impossible to restart servers in the production environment, for instance

6.4 Vendors Reaction

All the weaknesses found in the thesis were reported to firewall vendors. At the time of submitting the thesis, only three vendors responded to the reported findings. More information can be found in the following overview.

- ESET Internet Security. The vendor of this product, Slovak company ESET, was informed about the potential vulnerability found in their product. The finding was identified to be out of scope. Thus, the code presented in the thesis was also sent using the form for false negative detections. Besides the general, automatically generated confirmation of successful delivery, no other response from the vendor has been received until the submission date.
- Avast Free Antivirus. Avast was also informed about the improper detection. I want to remind that the Avast product successfully detected the attack. However, if the user does not follow the instructions from the product⁷, the attack can continue. According to the information provided after submitting the form, Avast should update the database of malicious files within 48 hours. No other information from the vendor has been delivered since then.
- Windows Firewall. Microsoft, as a vendor of Windows Firewall, was informed about the vulnerability using the Microsoft Security Response Center (MSRC). After delivering all the executables, source codes, and other materials relevant to reproducing the issue, the vulnerability is under the vendor's deeper investigation.
- TinyWall. The email was sent to the email address obtained from the product's official website. However, no response has been delivered until the submission of the thesis.
- ZoneAlarm Firewall. The information about the testing was sent to the vendor of this product, too. The procedure was almost the same as Microsoft's. The vendor asked for additional information, PoC, which is needed for vulnerability reproducing. The issue is now in the phase of deeper investigation on the vendor's side.

All the information described in this section is summarised in Table 6.3, which can be used as a quick overview of the vendor's reaction.

⁷especially to restart the device

■ **Table 6.1** Firewalls' Ability to Detect Substitution Attacks

Executable Location	Executable Signed By	Root Certificate Added to Store	Eset Internet Security	Avast Free Antivirus	Microsoft Defender	TinyWall	ZoneAlarm Firewall
Any	Not signed	No	✓	✓	✓	✓	✗
Any	Lukas Hrdonka	No	✓	✓	✓	✓	✗
Any	Lukas Hrdonka	Yes	✓	✓	✓	✓	✗
Any	Microsoft Corp.	No	✓	✓	✓	✓	✗
Any	Microsoft Corp.	Yes	✓	✓	✓	✓	✗
Same as trusted	Not signed	No	✓	✓	✗	✗	✗
Same as trusted	Lukas Hrdonka	No	✓	✓	✗	✗	✗
Same as trusted	Lukas Hrdonka	Yes	✓	✓	✗	✗	✗
Same as trusted	Microsoft Corp.	No	✓	✓	✗	✗	✗
Same as trusted	Microsoft Corp.	Yes	✓	✓	✗	✗	✗

■ **Table 6.2** Firewalls' Ability to Detect Injection Attacks

Attack Variant	Eset Internet Security	Avast Free Antivirus	Microsoft Defender	TinyWall	ZoneAlarm Firewall
Default configuration	✗	✓	✗	✗	✗
Additional detection enabled (if available in the product)	✗	✓	✓	✗	✗

■ **Table 6.3** Firewall Vendors Reactions

Event	Eset Internet Security	Avast Free Antivirus	Microsoft Defender	TinyWall	ZoneAlarm Firewall
First Response Time	2 hours	N/A	24 hours	N/A	24 hours
Current status	declared as out of scope	N/A	under investigation	N/A	under investigation

As the thesis shows only two of many possible attacks, more attack types could be added to create a complex tool capable of evaluating the firewall's ability and effectiveness in preventing leaks. The idea for future work is to create an application that users launch, and the strength of the firewall configuration is evaluated and shown to the users.

Last but not least, the essential parts of the thesis will be used as a basis for materials used in lectures and tutorials on subjects taught by the Department of Information Security at the Faculty of Information Technology, Czech Technical University in Prague.

Bibliography

1. PETROSYAN, Ani. *Number of malware attacks per year 2022* – Statista — *statista.com* [online]. 2023. Available also from: <https://www.statista.com/statistics/873097/malware-attacks-per-year-worldwide/>. [Accessed 09-03-2024].
2. IBM, Corp. *Cost of a data breach 2023* – IBM — *ibm.com* [online]. 2023. Available also from: <https://www.ibm.com/reports/data-breach>. [Accessed 09-03-2024].
3. CISCO SYSTEMS, Inc. *Cisco Internetworking Basics* — *cisco.com* [online]. 2002. Available also from: https://www.cisco.com/E-Learning/bulk/public/tac/cim/cib/using_cisco_ios_software/linked/tcpip.htm. [Accessed 09-03-2024].
4. CISCO SYSTEMS, Inc. *TCP/IP Overview* — *cisco.com* [online]. 2005. Available also from: <https://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13769-5.html>. [Accessed 09-03-2024].
5. KESSLER, Gary C. *Overview of TCP/IP and the Internet* — *garykessler.net* [online]. 2019. Available also from: <https://www.garykessler.net/library/tcpip.html>. [Accessed 09-03-2024].
6. ALI, Firkhan. A study of technology in firewall system. In: 2011, pp. 232–236. ISBN 978-1-4577-1548-8. Available from DOI: 10.1109/ISBEIA.2011.6088813.
7. CISCO SYSTEMS, Inc. *Security – Configuring Port Security* [Support] — *cisco.com* [online]. 2011. Available also from: https://www.cisco.com/en/US/docs/general/Test/dwverblo/broken_guide/port_sec.html. [Accessed 09-03-2024].
8. LEISCHNER, Garrett; TEWS, Cody. Security through VLAN segmentation: Isolating and securing critical assets without loss of usability. In: *proceedings of the 9th Annual Western Power Delivery and Automation Conference, Spokane, WA*. 2007.
9. POSTEL, Jon. *Internet Protocol* [RFC 791]. RFC Editor, 1981. Request for Comments, no. 791. Available from DOI: 10.17487/RFC0791.
10. HINDEN, Bob; DEERING, Dr. Steve E. *Internet Protocol, Version 6 (IPv6) Specification* [RFC 2460]. RFC Editor, 1998. Request for Comments, no. 2460. Available from DOI: 10.17487/RFC2460.
11. DOSTÁLEK, Libor. *Computer Networks II* [online]. 2023. Available also from: https://gitlab.fit.cvut.cz/BI-SPS/bi-sps/-/raw/eeb5ee03d6b120cc4caad086d986a5e38279af74/files/TCP-IP_II_CZ_v08.pdf. [Accessed 09-03-2024].

12. LANGLEY, Adam; RIDDOCH, Alistair; WILK, Alyssa; VICENTE, Antonio; KRASIC, Charles, et al. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In: *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. Los Angeles, CA, USA: Association for Computing Machinery, 2017, pp. 183–196. SIGCOMM '17. ISBN 9781450346535. Available from DOI: 10.1145/3098822.3098842.
13. IYENGAR, Jana; THOMSON, Martin. *QUIC: A UDP-Based Multiplexed and Secure Transport* [RFC 9000]. RFC Editor, 2021. Request for Comments, no. 9000. Available from DOI: 10.17487/RFC9000.
14. CISCO SYSTEMS, Inc. *Cisco Content Hub – Addresses, Protocols, and Ports Reference – content.cisco.com* [online]. 2019. Available also from: https://content.cisco.com/chapter.sjs?uri=/searchable/chapter/www.cisco.com/content/en/us/td/docs/interfaces_modules/services_modules/ace/vA4_1_0/configuration/rtg_brdg/guide/rtbrgdgd/subnets.html.xml. [Accessed 09-03-2024].
15. ANONYM. *HTTP Methods GET vs POST – w3schools.com* [online]. 2024. Available also from: https://www.w3schools.com/tags/ref_httpmethods.asp. [Accessed 09-03-2024].
16. BAŘINKA, Lukáš. *HTTP – Apache httpd Web Server Administration – lukasbarinka.gitlab.io* [online]. 2021. Available also from: <https://lukasbarinka.gitlab.io/apache/http.html>. [Accessed 09-03-2024].
17. KOKEŠ, Josef. *Sockets security – 12th lecture of BI-BEK.21 (Secure Code) course* [online]. 2023. Available also from: <https://courses.fit.cvut.cz/BI-BEK/media/lectures/bek12en.pdf>. [Accessed 09-03-2024].
18. RESCORLA, Eric; DIERKS, Tim. *The Transport Layer Security (TLS) Protocol Version 1.2* [RFC 5246]. RFC Editor, 2008. Request for Comments, no. 5246. Available from DOI: 10.17487/RFC5246.
19. RESCORLA, Eric. *The Transport Layer Security (TLS) Protocol Version 1.3* [RFC 8446]. RFC Editor, 2018. Request for Comments, no. 8446. Available from DOI: 10.17487/RFC8446.
20. CORPORATION, Lockheed Martin. *Cyber Kill Chain® – lockheedmartin.com* [online]. 2024. Available also from: <https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html>. [Accessed 12-03-2024].
21. KARKI, Supragya. *Cyber Kill Chain – Offensive and Defensive Approach – medium.com* [online]. 2021. Available also from: <https://medium.com/cryptogenepal/cyber-kill-chain-offensive-and-defensive-approach-22033e37a340>. [Accessed 12-03-2024].
22. CHOPRA, Aakanksha. Security Issues of Firewall. *International Journal of P2P Network Trends and Technology*. 2016, vol. 22, pp. 4–9. Available from DOI: 10.14445/22492615/IJPTT-V22P402.
23. GREBENNIKOV, Nikolay. *Using leak tests to evaluate firewall effectiveness – securelist.com* [online]. 2007. Available also from: <https://securelist.com/using-leak-tests-to-evaluate-firewall-effectiveness/36182/>. [Accessed 16-03-2024].
24. MATOUSEK, David. Firewall leak testing. *Hackin9 – IT Security Magazine*. 2007, pp. 62–67.
25. KARL-BRIDGE-MICROSOFT; DJM00N, [Ryazantcev Dimitriy]; V-KENTS, [Kent Sharkey]; DCTHEGEEK, [David Coulter]; DREWBATGIT, [drew batchelor]; MIJACOBS, [Mike Jacobs]; MSATRANJR, [Michael Satran]. *Hooks – Win32 apps – learn.microsoft.com* [online]. 2021. Available also from: <https://learn.microsoft.com/en-us/windows/win32/winmsg/hooks>. [Accessed 16-03-2024].
26. ESET, spol. s r.o. *Configuring and using rules – ESET Internet Security – ESET Online Help – help.eset.com* [online]. 2023. Available also from: https://help.eset.com/eis/15/en-US/idh_config_epfw_view_rule.html?idh_config_epfw_basic_group.html. [Accessed 19-03-2024].

27. GEN DIGITAL, Inc. *AI & machine learning – Technology – Avast* — *avast.com* [online]. 2024. Available also from: <https://www.avast.com/en-us/technology/ai-and-machine-learning>. [Accessed 20-03-2024].
28. GEN DIGITAL, Inc. *Advanced Firewall Settings* — *businesshelp.avast.com* [online]. 2024. Available also from: https://businesshelp.avast.com/Content/Products/AfB_Antivirus/ConfiguringSettings/FWSettings.htm. [Accessed 20-03-2024].
29. GEN DIGITAL, Inc. *Behavior Shield* — *businesshelp.avast.com* [online]. 2024. Available also from: https://businesshelp.avast.com/Content/Products/AfB_Management_Consoles/ConfiguringSettingsandPolicies/BehaviorShield.htm. [Accessed 20-03-2024].
30. [PAOLO MATARAZZO], paolomatarazzo. *Windows Firewall overview – Windows Security* — *learn.microsoft.com* [online]. 2024. Available also from: <https://learn.microsoft.com/en-us/windows/security/operating-system-security/network-security/windows-firewall/>. [Accessed 21-03-2024].
31. PAOLOMATARAZZO, [Paolo Matarazzo]. *Windows Firewall rules – Windows Security* — *learn.microsoft.com* [online]. 2024. Available also from: <https://learn.microsoft.com/en-us/windows/security/operating-system-security/network-security/windows-firewall/rules>. [Accessed 21-03-2024].
32. JSUTHER1974; PAOLOMATARAZZO, [Paolo Matarazzo]; VINAYPAMNANI-MSFT, [Vinay Pamnani]. *WDAC and AppLocker Overview – Windows Security* — *learn.microsoft.com* [online]. 2024. Available also from: <https://learn.microsoft.com/en-us/windows/security/application-security/application-control/windows-defender-application-control/wdac-and-applocker-overview>. [Accessed 21-03-2024].
33. PADOS, Károly. *TinyWall – A free, lightweight and non-intrusive firewall* — *tinywall.pados.hu* [online]. 2023. Available also from: <https://tinywall.pados.hu/>. [Accessed 23-03-2024].
34. QUINNRADICH, [Quinn Radich]; ALEXBUCKGIT, [Alex Buck]; STEVEWHIMS, [Steven White]; HICKEYS, [Shawn Hickey]; SPHINXKNIGHT; CMCCLISTER, [Christopher McClistler]; DMATTWOJO, [Matt Wojciakowski]. *What's a Universal Windows Platform (UWP) app? – UWP applications* — *learn.microsoft.com* [online]. 2023. Available also from: <https://learn.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide>. [Accessed 23-03-2024].
35. CHECK POINT SOFTWARE TECHNOLOGIES, Inc. *Free Firewall* — *zonealarm.com* [online]. 2024. Available also from: <https://www.zonealarm.com/software/free-firewall>. [Accessed 23-03-2024].
36. CHECK POINT SOFTWARE TECHNOLOGIES, Inc. *ZoneAlarm Firewall* — *zonealarm.com* [online]. 2024. Available also from: <https://www.zonealarm.com/learning-center/firewall>. [Accessed 23-03-2024].
37. TEDHUDEK, [Ted Hudek]. *Digital Certificates - Windows drivers* — *learn.microsoft.com* [online]. 2021. Available also from: <https://learn.microsoft.com/en-us/windows-hardware/drivers/install/digital-certificates>. [Accessed 28-04-2024].
38. TEDHUDEK, [Ted Hudek]; MHOPKINS-MSFT, [Mark Hopkins]. *Local Machine and Current User Certificate Stores - Windows drivers* — *learn.microsoft.com* [online]. 2023. Available also from: <https://learn.microsoft.com/en-us/windows-hardware/drivers/install/local-machine-and-current-user-certificate-stores>. [Accessed 28-04-2024].
39. TEDHUDEK, [Ted Hudek]; ELIOTSEATTLE, [Eliot Graff]. *Trusted Publishers Certificate Store - Windows drivers* — *learn.microsoft.com* [online]. 2022. Available also from: <https://learn.microsoft.com/en-us/windows-hardware/drivers/install/trusted-publishers-certificate-store>. [Accessed 28-04-2024].

40. GEN DIGITAL, Inc. *Avast Threat Labs – File whitelisting – Official Avast Support* — *support.avast.com* [online]. 2022. Available also from: <https://support.avast.com/en-ww/article/229/>. [Accessed 09-04-2024].
41. MSEDGETEAM; MIKEHOFFMS, [Michael Hoffman]. *Web View2 end-user FAQ – Microsoft Edge Developer documentation* — *learn.microsoft.com* [online]. 2023. Available also from: <https://learn.microsoft.com/en-us/microsoft-edge/webview2/concepts/end-user-faq>. [Accessed 24-03-2024].
42. KARL-BRIDGE-MICROSOFT. *OpenProcess function (processthreadsapi.h) - Win32 apps* — *learn.microsoft.com* [online]. 2022. Available also from: <https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-openprocess>. [Accessed 27-04-2024].
43. KARL-BRIDGE-MICROSOFT. *GetLastError function (errhandlingapi.h) - Win32 apps* — *learn.microsoft.com* [online]. 2024. Available also from: <https://learn.microsoft.com/en-us/windows/win32/api/errhandlingapi/nf-errhandlingapi-getlasterror>. [Accessed 27-04-2024].
44. KARL-BRIDGE-MICROSOFT. *CreateRemoteThread function (processthreadsapi.h) - Win32 apps* — *learn.microsoft.com* [online]. 2022. Available also from: <https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createremotethread>. [Accessed 24-03-2024].
45. COWAN, Crispin. *Protecting Microsoft Edge against binary injection* — *blogs.windows.com* [online]. 2015. Available also from: <https://blogs.windows.com/msedgedev/2015/11/17/microsoft-edge-module-code-integrity/>. [Accessed 27-04-2024].
46. MHOPKINS-MSFT, [Mark Hopkins]; CMCCLISTER, [Christopher McClister]; SAJEANP. *WHQL Test Signature Program - Windows drivers* — *learn.microsoft.com* [online]. 2022. Available also from: <https://learn.microsoft.com/en-us/windows-hardware/drivers/install/whql-test-signature-program?source=recommendations>. [Accessed 27-04-2024].
47. KARL-BRIDGE-MICROSOFT; HENKE37; S1CKB0Y1337, [Nikos Katsiopoulos]; DREWBATGIT; V-KENTS, [Kent Sharkey], et al. *Process Security and Access Rights - Win32 apps* — *learn.microsoft.com* [online]. 2022. Available also from: <https://learn.microsoft.com/en-us/windows/win32/procthread/process-security-and-access-rights>. [Accessed 27-04-2024].
48. KARL-BRIDGE-MICROSOFT. *VirtualAllocEx function (memoryapi.h) - Win32 apps* — *learn.microsoft.com* [online]. 2022. Available also from: <https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualallocex>. [Accessed 27-04-2024].
49. ALVINASHCRAFT, [Alvin Ashcraft]; REEMSABAW1, [Reem Sabawi]; DREWBATGIT; V-KENTS, [Kent Sharkey]; DCTHEGEEK, [David Coulter], et al. *Memory Protection Constants (WinNT.h) - Win32 apps* — *learn.microsoft.com* [online]. 2022. Available also from: <https://learn.microsoft.com/en-us/windows/win32/Memory/memory-protection-constants>. [Accessed 27-04-2024].
50. STEVEWHIMS, [Steven White]; V-KENTS, [Kent Sharkey]; MSATRANJR, [Michael Satran]. *Complete Winsock Client Code - Win32 apps* — *learn.microsoft.com* [online]. 2021. Available also from: <https://learn.microsoft.com/en-us/windows/win32/winsock/complete-client-code>. [Accessed 24-03-2024].

51. MARKRUSS; FOXMSFT, [Alex Mihaiuc]; MARIOHEWARDT, [Mario Hewardt]; JOHNSTEP, [John Stephens]; ALEXBUCKGIT, [Alex Buck]; STEPHENBRENTPETERS, [Stephen Peters], et al. *Process Explorer – Sysinternals — learn.microsoft.com* [online]. 2023. Available also from: <https://learn.microsoft.com/en-us/sysinternals/downloads/process-explorer>. [Accessed 26-03-2024].
52. VLČEK, Ondřej. *Behavior Shield: our newest behavioral analysis technology — blog.avast.com* [online]. 2017. Available also from: <https://blog.avast.com/behavior-shield-our-newest-behavioral-analysis-technology>. [Accessed 27-03-2024].
53. GEN DIGITAL, Inc. *Excluding certain files or websites from scanning in Avast Antivirus and Avast One — Official Avast Support — support.avast.com* [online]. 2022. Available also from: <https://support.avast.com/en-us/article/antivirus-scan-exclusions/>. [Accessed 29-04-2024].
54. GEN DIGITAL, Inc. *Adjusting settings for Avast Antivirus Core Shields — Official Avast Support — support.avast.com* [online]. 2022. Available also from: <https://support.avast.com/en-ww/article/antivirus-shield-settings/>. [Accessed 29-04-2024].
55. GEN DIGITAL, Inc. *Quarantine – Getting Started – Official Avast Support — support.avast.com* [online]. 2024. Available also from: <https://support.avast.com/en-us/article/use-antivirus-quarantine/>. [Accessed 27-03-2024].
56. SIOSULLI, [Sinead O’Sullivan]; DANSIMP, [Daniel Simpson]; V-EMILYPR, [Emily Prindeville]; JUSTPIES, [Justin Piesco], et al. *Customize exploit protection — learn.microsoft.com* [online]. 2022. Available also from: <https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/customize-exploit-protection?view=o365-worldwide>. [Accessed 27-03-2024].
57. SIOSULLI, [Sinead O’Sullivan]; DENISEBMSFT; CHRISDA, [Chris Davis]; AMERICAN-DIPPER, [Jeff Borsecnik]; DANSIMP, [Daniel Simpson], et al. *Exploit protection reference — learn.microsoft.com* [online]. 2023. Available also from: <https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/exploit-protection-reference?view=o365-worldwide>. [Accessed 27-03-2024].

Contents of the Attachment

readme.txt	file with a brief overview of attachment
thesis.pdf	the thesis in the PDF format
implementation	source codes used in the thesis
injection	the directory with source codes used in the injection attack
capture	Wireshark capture of network traffic
exe	executable files used for testing
attacker	executable file of the attacker (server) part
victim	executable files of the victim (client) part
src	source codes written in C++
attacker	source code of the attacker (server) part
victim	source codes of the victim (client) part
substitution	the directory with source codes used in the substitution attack
certificates	example certificates in the PFX format
src	source code of demonstration application written in C++
substitution1	unsigned executable file
substitution2	executable file signed by Lukas Hrdonka
substitution3	executable file signed by Lukas Hrdonka with TSA verification
substitution4	executable file signed by Microsoft Corporation
substitution5	executable file signed by Microsoft Code Signing PCA 2010
thesis	directory of L ^A T _E X source codes of the thesis
text	actual text of the thesis in the L ^A T _E X format
images	images included in the thesis
vendors	communication with the vendors
Avast	communication with Avast
ESET	communication with ESET
Microsoft	communication with Microsoft
TinyWall	communication with TinyWall
ZoneAlarm	communication with ZoneAlarm