An Analysis of the Threats Posed by Botnet Malware Targeting Vulnerable Cryptocurrency Miners

Joseph K. Wrieden
Department of Computer Science
University of York
York, United Kingdom
jw3007@york.ac.uk

Vassilios G. Vassilakis
Department of Computer Science
University of York
York, United Kingdom
vv573@york.ac.uk

***Abstract*—Since the invention and popularisation of blockchain technology, we have seen a recent surge of attacks targeting cryptocurrency infrastructure. Alongside this, botnet malware has become a staple within threat actors' toolkits, and have often been used to target a wide range of devices. This paper explores the threats that a custom built botnet poses when used to target cryptocurrency mining software. The botnet within this project is developed in the programming language Golang, due to its effective networking and utilisation in the malware development sector. The targets of the attack will be a cryptocurrency miner, and for ethical reasons a proof-of-concept miner will be used for testing. For evaluation purposes a fully virtualised network is used, with practical exploitation taking place to evaluate some metrics of performance for the botnet. With these metrics, the potential threats posed are then explored, with the main attack vector discovered being defined as "forced pooling". Through this attack vector we show how this unique threat facilitates a variety of different attacks, both on and off chain including a "51% attack" and password cracking; exploring how a potential distributed supercomputer can be used as an attack tool.**

***Index Terms*—Cryptocurrency, Botnet, Forced Pooling, Malware, Blockchain, Mining**

I. Introduction

Botnets are one of the most popular kinds of malware, with threat actors utilising large numbers of connected devices to conduct a wide range of attacks. Most botnets are commonly used to conduct attacks such as Distributed Denial of Service (DDoS), where large amounts of traffic is directed towards a specific targets to stop legitimate access [1]. Commonly these targets are comprised of servers and websites, however as the capabilities of botnets increase as does the target range.

Cryptocurrency and blockchain technology is something that has exploded in popularity over the past few years. Ever since Bitcoin's theorisation by Nakamoto [2], blockchain technology has been utilised to build an array of decentralised instruments. Because of this popularity, it has become the target of cybercriminals looking to profit from both the increased popularity and profitability of the currency. This has led to design and development of cryptocurrency orientated malware, often aiming to steal or deprive the victim of their funds.

Interestingly however, the application of botnets to cryptocurrency is fairly limited; with the majority of uses often boiling down to developing large botnets of disparate devices and then using that distributed computing power to mine the currency. Despite this, the concept of a customised malware, aimed at targeting devices already mining cryptocurrency is limited and one which could have serious consequences that do not only affect the cryptocurrency space. This is why our research aims to explore the potential threats that are present when a custom botnet is used to target devices within a blockchain network, such as cryptocurrency miners. Through this we hope to establish the level of threat that this kind of attack poses to the wider security space. While traditional botnets have been in use by threat actors for years, commonly such as was the case with the Mirai family of botnets [3] that utilised low-power Internet-of-things (IoT) devices. If a botnet of high-power mining machines, often containing multiple GPUs or CPUs, was able to be built the threats this may pose could provide an entirely different threat landscape providing a new insight into the capabilities of botnets.

The aim of this project is to evaluate the threats that a custom build malware used to target cryptocurrency mining devices poses, and the potential ramifications that a real version of this botnet may have. To achieve this we designed and developed a custom-built malware, which targets a vulnerable mining software to create a network of infected devices. This vulnerable mining software is a proof-of-concept, only simulating the process of mining and not interacting with a real chain. The aim is for the botnet to intelligently scan for vulnerable devices, infect them, and then perform some act of simultaneous execution on all of the hosts; all while not affecting the mining rate by more than 10%. This is the case, as due to the target being a mining device, one of the most noticeable impacts the botnet could have on the system is to reduce the hashrate. Once infection has taken place, the malware is able to simultaneously execute commands over the network, with a maximum time difference of around 0.1 second; such that it is reasonable to assume that this would not be an issue when it comes to executing attacks.

II. Related Work

A. Alternative Cryptocurrency Botnets and Malware

Cryptocurrency botnets are not a new concept, with many threat actors pivoting existing infrastructure to targeting the cryptocurrency space. We have seen botnets such as Sysrv-hello [4] utilising traditional botnet techniques to target and steal cryptocurrency. More generally the cryptocurrency malware space has grown, with large portions of them adopting mining into their techniques [5]. These "Malicious Miners" either utilise a social engineering technique, or exploit vulnerable services such as SMB, to gain access to a device and then mine cryptocurrency from it. Within the malware collection explored by Konoth et al. [5] there exists malware targeting both Linux and Windows PC's as well as mobile devices and IoT. Interestingly witin the malware surveyed it is not common for the malware to target multiple operating systems (OS), which is a feature that would allow a wider botnet to be created.

Another kind of cryptocurrency botnet that has begun to emerge is one that does not target the chain but instead operates utilising it. A paper which explored this was CoinBot [6], a covert botnet that operates using a blockchain for its command & control (C2) communication. This botnet design aims to be highly dependable, utilising the permanent nature of the blockchain to communicate with infected devices. This also ensures that the attacker is not required to host dedicated hardware for the C2 server, protecting them from having the server taken down. What makes this paper interesting is their practical approach to the analysis of this problem. This approach utilises the OP_RETURN output, which can be issued by the C2 leaving the command on the chain. This is then read by the bots utilising shortened URLs to retrieve the data from the chain. For testing this project, a proof-of-concept implementation of the C2 is written in the Python programming language and uses the cryptocurrency APIs of BlockCypher to send and read the transactions from the Bitcoin, Litecoin, and Dash blockchains. This practical approach enabled the researchers to test the efficiency of the C2 implementation, allowing them to accurately analyse the potential threats that this attack methodology poses. These include the difficulty of attribution, and the permanent nature of the C2 communications; from which the paper then briefly explores some potential mitigations.

B. Malicious Mining Detection

As outlined by Swedan et al. [7] there is a variety of different techniques which can be used to detect malicious mining. Some of the main methods that were identified include using anti-virus software, and blacklisting certain mining URLs; as well as their newly proposed solution. The one limitation with these defence mechanisms, is that it does not particularly defend against the hijacking of systems which are already mining cryptocurrency, as defence methods such as URL blacklisting would interfere with the standard operation of the device. Another effective detection method, proposed by Yazdinejad et al. [8] utilises a machine learning approach to identify malicious mining. This method analyses the operation codes within Windows applications to determine whether the mining software is malicious or benign. This methodology is highly effective, however one limitation with this, is the manipulation of traditional benign software may go undetected. For instance, our malware aims to manipulate a vulnerability within benign software, thus we are not inherently installing any malicious mining software onto the host. Instead we utilise a custom client side malware to operate the existing tools on the client system.

C. Existing Malware

When exploring related projects it was clear that a large amount of related material was found within real pieces of malware. Because of this, analysis of three notable pieces of malware was conducted, each being a different botnet. The three malware analysed were BotenaGo, GoBot2, and Mirai due to the source code of each being publicly available for analysis. Through analysis of each, some key features that an effective botnet must exhibit were identified, with each malware providing a unique and valuable feature.

The BotenaGo malware, first discovered in 2021 by AT&T Alien Labs [9], was famous for targeting over 30 different devices using a map of different exploit known as an Exploit Kit [10]. These were split into two categories, some designed for certain targets, and others used for unknown targets as part of an "exploit spray". By combining with a method such as operating system fingerprinting [11], the malware can identify and correctly exploit its target. This exploit kit is a powerful tool that enables the malware to not only be more efficient but also less overt in its exploitation.

The GoBot2 malware was developed by GitHub user SaturnsVoid [12] as a proof-of-concept Trojan and botnet. What makes this malware effective is its well designed C2 mechanism, utilising a single server which controls all bots within the network. This allows for operation via a well designed user interface, which enables specific actions to be conducted across the network. This is all combined with an SQL database which allows for the botnet data such as commands executed and bots connected to be stored. Overall the high level of control this feature provides the botmaster makes this botnet effective when used in targeted and coordinated attacks.

Mirai is potentially the most famous botnet family on account of being involved in some of the most famous DDoS attacks of the 2010s, against sites such as Krebs On Security [13]. October 2016 however, saw the source code of the botnet leaked on HackForums, which has since been archived on GitHub [14]. One feature that led the malware to its success, was its array of pre-built attacks that the botnet could conduct. This feature allows the botmaster to quickly deploy different kinds of DDoS attacks such as UDP flooding; ACK flooding; and DNS water torture. What makes this feature valuable is the pre-written attacks help ensure the attack chain is optimised, whilst also helping to remove the need to rely on external elements being present on the infected device.

III. Design

Through the analysis of other malware in Subsection II-C we have identified some core components that an effective botnet malware can exhibit. Using these combined with the botnet's goals, we can outline the design including any and all potential reasoning for the decisions made. It is also important to note that we are not aiming to include large amounts of obfuscation into the code as this is out of scope; this includes avoiding detection from traditional anti-virus or network detection software. Functionally the botnet follows the design shown in Figure 1. While the elements have no order of importance, the numbering shows the functional order which the elements would be used.

First among these, is that the botnet operates utilising a custom written C2 which communicates with the clients using TCP connections. What is unique about this C2 is that it also operates as a command center for the entire malware. This can be seen within Figure 1 as both the scanning and payload delivery mechanisms are controlled by the C2. This ensures that the majority of the execution is conducted on the server side of the operation, to ensure that the client code is as lightweight as possible. This enables the client to operate solely as a listener, accepting commands from the server and executing them. Because of this we are also able to get simultaneous execution across all clients, through sending commands from a centralised server; as opposed to the peer-to-peer method. The C2 is operated through an interface, that has a variety of functions enabling core operations such as scanning for new hosts; exploitation of hosts; execution of commands; and removal of bots from the network. The C2 server also utilises a database style back-end where the current status of each of the scanned or exploited bots can be stored and managed from. This back-end is operated by the server, enabling for configurations to be saved and loaded including the state of the botnet; such that the C2 can be exited, with the status saved.

For the malware to be successful, it must be able to efficiently scan a network for vulnerable devices; identifying vulnerable hosts, and discounting those which are not. Rather than scanning a network for all accessible devices and then queuing each for exploitation automatically, this malware is designed to operate with much less noise generated. The malware is able to scan for the vulnerable devices utilising known characteristics of the vulnerable mining software to identify hosts running it. This means that any hosts with unintentional similarities such as having the vulnerable port open without running the software, will not be mistaken; leading to a very small false positive rate. This malware design also decouples the scanning and exploitation phases. By doing this the botmaster is able to scan large numbers of devices before exploitation, or to exploit the devices in small batches to avoid creating too much noise. This all goes to create a usable and feature rich malware, that operates in a smart and decoupled manor.

With the payload delivery mechanism, one key feature

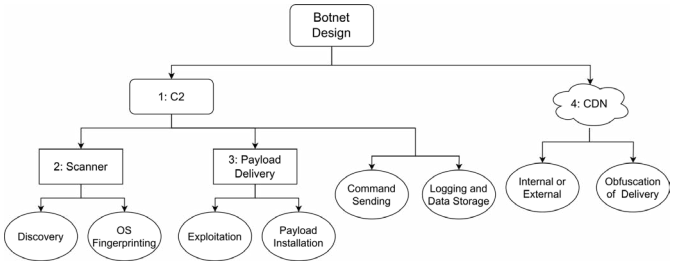

Fig. 1: Design of Botnet Malware

is that it must be able to target both Windows and Linux architectures. Due to Golang's cross-compilation features, the development of client-side malware for both parties is simple; enabling the server to distribute these to the client with ease. To achieve this, the botnet design utilises OS Fingerprinting and ensures that each bot within the network has an associated OS. This OS fingerprinting is conducted within the earlier discussed scanning phase; and the result stored within the C2's database back-end such that all functions within the malware can modify their execution appropriately. It is also important to note that the exploitation stage is built into an exploit kit, such that the process can be streamlined. As part of the payload delivery stage, the exploit will be delivered against the identified host, only if the target is assumed to be vulnerable; followed by the actual delivery and running of the malicious payload. This element of the botnet's design ensures that the malware has a robust and effective exploitation and payload delivery system; that ensures that all vulnerable devices are brought into the network.

Finally, what holds a small but key part of a botnet's ability to operate is the content delivery network (CDN). They provide an efficient way of distributing the delivery of content across multiple proxy servers. While this is not a core part of most botnets, we have seen malware utilising external CDNs from services such as Discord [15] to distribute their payloads. Because of the need for multiple different kinds of payload to be delivered to malicious hosts, it would be ideal to have the capability to decouple the C2 from the delivery server. Because of this the botnet's design ensures that both local and external content delivery is supported, enabling for external CDNs to be utilised. This ensures that the malware has been developed to professional standard, and that all modern malware features are accommodated.

IV. Development

The botnet is entirely written in Golang and is separated into two different elements: the server and the client. The server component of the malware is responsible for the majority of the actions of the botnet, including scanning, exploitation, and payload delivery. The client on the other hand is responsible for interaction with the exploited system. The botnet delivers a client side component to the infected device. By using this custom handler, we are able to not only lower the detection rate by not leaving the terminal gained via exploitation open for others to use; but it also allows us to disconnect and reconnect

from clients without the need for re-exploitation. As we will not be including source code for ethical reasons, to explain how the malware has been implemented we will analyse the core components identified in Figure 1.

A. Command and Control (C2)

The C2 developed for this malware has been implemented to allow for multiple interaction methods. The most basic of these is the terminal interaction method, allowing for a user to interact through a text based interface. This terminal interface allows for available options to be selected, as shown in Figure 2a, with any extra data being supplied using the text terminal. While this is functional, the C2 has an alternate mode where it can operate utilising a web interface. This is a feature that has been seen utilised by threat actors within their custom C2 and management infrastructure [16]. This web interface, allows for each component of the malware to be easily controlled and managed; such as listing the jobs run on the system (shown in Figure 2b); enabling for efficient management of the botnet during all stages: scanning, exploitation, and command execution.

Another previously discussed element which has been implemented is the inclusion of a database style backend. Because of the type of data stored, it was decided that there was no need for a true database to be built. The backend relies on a series of JSON files storing the current bot and job information as well as server configurations. Because of this feature choice, we are able to exit the botnet, not requiring constant communication so large botnets to be built over a longer period of time.

B. Scanner

The scanning component of the botnet utilises a smart scanning method to identify and profile each potential target. The scanning component can either take a specific IP address or IP address range, and will scan them accordingly. What makes the scanning implementation smart is the utilisation of Vulnerability Scanning and OS Fingerprinting functions. The first of these functions requests the version string from the vulnerable mining software. Using this request, the function can firstly identify whether the appropriate software is running and then whether it is the vulnerable version. If this is the case then the host will be considered vulnerable and will be added into the network. For those vulnerable devices the OS Fingerprinting module uses the version string of the software to identify the OS being run; due to it containing the OS details. Through this the newly scanned device's entry within the bot database will contain their associated OS such that each stage in the exploit and command execution chain can be easily modified for the required OS.

C. Payload Delivery

As previously mentioned the payload delivered to a vulnerable device is the client side of the malware. To do this the exploitation is split into multiple stages, first the delivery of the exploit itself; then connecting to the remote management

(a) Terminal Interface (b) Web Interface

Fig. 2: Botnet C2 Interface

terminal; downloading the payload onto the target; and finally executing it. The exploit itself is delivered through the server connection port in the vulnerable miner. Each function within the payload delivery is multi-OS. This is achieved using a map that takes the associated OS and returns the OS specific elements for the required command. The CDN, which is used within this stage, is specified within the botnets configuration files to ensure, that as long as the CDN is configured correctly, the delivery function will continue to operate as intended.

D. Command Execution

Finally the command execution component is handled utilising a socket connection between the server and the clients, each one being managed asynchronously. Each bot's client malware operates as a simple listener for commands from the server. The user can select one or multiple bots to send commands too having the option of either pre-written commands or custom ones. These pre-written commands similar to those seen used in Mirai enable for useful features, such as credential harvesting, to be conducted without requiring each command to be entered individually. This section also uses a command map, similar to those used in Subsection IV-C, where depending on the target's OS, different commands are delivered. This enables for pre-written commands to be sent to all bots, even if they are each running a different OS, ensuring that simultaneous execution can be achieved over the botnet. Overall the command execution implementation has been developed in a manner that ensures all commands sent, are executed as fast as possible; with as much complexity abstracted away.

V. Testing

A. Full Network Exploitation Test

For this test the network was composed as outlined in Table I with the C2 operating on an interactable device.

This test was a success proving that the malware was able to identify each of the red herring devices and only exploit those that were truly vulnerable. The malware identified both the patched mining version, and the red herring port and classed them as not vulnerable. It is also important to note that during this exploitation, the mining efficiency alarm was not raised raised during the process. This shows that the exploitation process is highly efficient and can infect machines without affecting the mining rate.

TABLE I: Virtual Machine Testing Configuration

VM ID	OS	Vulnerable	Notes
1	Linux	N/A	C2 Server
2	Windows	Yes	N/A
3	Windows	No	Has mining port open
4	Linux	Yes	Has other open ports
5	Linux	Yes	N/A
6	Linux	No	Running patched miner version

B. Simultaneous Execution Test

For the simultaneous execution test the bots exploited in Subsection V-A were used to execute the OS's relevant *time* command, using one of the pre-compiled exploits. While this is not a traditional exploit function, for the purposes of this test it enabled us to know down to the second how effective the command execution was across the network.

As can be seen in Table II the results were excellent with the max difference in execution time being never going above 00.06s. This means that the botnet would be more than capable of executing the attacks outlined in Section VI. This is because the difference in execution time is now removed as a potential risk factor to its success in executing attacks.

VI. Evaluation of Potential Threats

From the tests conducted, the malware outlined within this paper is utilising a technique more similar to that of traditional pooling than of the alternative cryptojacking. Within the cryptocurrency space it is common for users to not hold enough processing power to successfully mine on a chain. In these cases the users will form a collective known as a Mining Pool [17]. This is where multiple different crypto-miners pool together their mining resources such that they have a higher chance of being successful in their mining attempts. If the pool is successful, each member will get a portion of the rewards.

This attack is unconsensually pooling multiple mining devices together to make a powerful distributed mining computer; with a technique we have coined as "Forced Pooling". This technique has two versions of the attack, one active and another passive. The active attack forcefully installs new software or registers the devices into a pool creating a single powerful mining device. This method of the attack potentially gives the malicious actor more control over the mining activities of the cluster; also enabling the mining to be more directed and efficient. The other method, passive, utilises a similar technique to that of Satori-Coin-Robber [18] where the existing mining software configuration is modified such that the miner is now mining for the attackers wallet. This method of attack is by definition, a much quieter attack, leaving less footprint on the device.

Overall forced pooling theoretically presents a much more dangerous threat, when contrasted with attacks such as cryptojacking. Because of the high power level of the hardware within common mining rigs, pooling these resources together could pose the ability to build a distributed supercomputer. To discover what malicious capabilities this could provide, we explore two powerful attacks that this threat vector enables, and how the results shown in Section V would aid in the execution of them.

TABLE II: Simultaneous Execution Test Results (MM:SS.00)

Test ID	Command Send Time	Command Execution Time			Max Time Difference
		Bot1	Bot2	Bot3	
1	45:59.54	46:02.55	46:02.51	46:02.55	00:00.04
2	46:40.57	46:43.58	46:43.54	46:43.58	00:00.04
3	48:07.55	48:10.57	48:10.51	48:10.58	00:00.04
4	50:45.46	50:48.46	50:48.43	50:48.49	00:00.06

A. 51% Attack

The first and most important among these attacks to explore, is the potential to conduct the usually theoretical 51% attack [19]. The attack works when more than 50% of a chain's validating power is operated by a single user, and is harnessed for a malicious purpose. In our proof-of-work system, this validation power is the hashrate, and with this control a malicious actor is able to completely remove the integrity of the chain.

Specifically with our botnet, utilising the forced pooling attack; if the hashrate controlled by infected devices was over half of the chains total validating power, a 51% attack is definitely possible. Because of our ability to centrally control the infected devices through the C2 server, and the efficiency of the simultaneous execution this helps make this attack even more powerful. This is because, depending on the malicious actions needed, the chain-changing power can be wielded effectively but also carefully during the execution of targeted malicious activities utilising the 51% attack.

B. Password Cracking

While this paper is primarily focused on cryptocurrency based threats which this malware may enable, thanks to the power forced pooling provides this threat also seeps into traditional technology. The most obvious among these is the ability to crack passwords. Ironically when we revisit Nakamoto's original paper [2] within their explanation of the proof-of-work system they reference a paper from Hashcat [20], which at the time was utilised for DoS protection. Hashcat is now one of the most popular password cracking tools, that utilises a similar process to that of proof-of-work by reverse engineering the password hashing process to crack the passwords. This works by hashing inputs and comparing the output hashes with a target hash. Different to mining where a target quality within the hash is required, password cracking needs to completely match the target to provide what the inputs was prior to hashing.

As previously discussed the forced pooling attack is potentially able to pool large amounts of GPU compute power, as often the mining process is GPU intensive. Because of this, the botnet would be able to use this GPU compute power to distribute password cracking attacks such as brute-force and dictionary based attacks. This would be supported by the more complex features of the C2, ensuring that the task can be easily distributed, whilst enabling any required tools such as a password list, to be accessible on each device.

VII. Conclusion

Overall we believe that through this botnet, we have effectively evaluated the threats posed when botnets are used to target cryptocurrency miners. By taking a practical approach we were able to use real data to evaluate each threat. Not only was the botnet successful in its objectives, but by establishing forced pooling and its unique threats, this project provides a unique insight into a security problem.

A possible next step is to implement this attack scenario onto a real cryptocurrency chain, allowing for the threats explored in Section VI to be practically evaluated. Another element that could be enhanced is the obfuscation and evasion tactics used for the malware. It would be interesting to explore how this malware could bypass different security mechanisms. Finally, during the latter stages of research we discovered the widespread utilisation of application-specific integrated circuit (ASIC) miners [21]. These miners, which are built using custom hardware, would provide an interesting threat angle for this botnet to attack; with the potential for hardware based vulnerabilities.

References

[1] L. McNulty and V. G. Vassilakis, "IoT botnets: Characteristics, exploits, attack capabilities, and targets," in *13th IEEE/IET Int. Symp. on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, 2022, pp. 350–355.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, 2008.

[3] G. Kambourakis, C. Kolias, and A. Stavrou, "The mirai botnet and the iot zombie armies," in *IEEE Military Communications Conference (MILCOM)*, 2017, pp. 267–272.

[4] *Worm-like propagation of Sysrv-hello crypto-jacking botnet: Network traffic analysis and latest TTPs*, Darktrace. [Online]. Available: https://darktrace.com/blog/worm-like-propagation-of-sysrv-hello-crypto-jacking-botnet (visited on 03/29/2023).

[5] R. K. Konoth, R. van Wegberg, V. Moonsamy, and H. Bos, "Malicious cryptocurrency miners: Status and outlook," *arXiv preprint arXiv:1901.10794*, 2019.

[6] J. Yin, X. Cui, C. Liu, Q. Liu, T. Cui, and Z. Wang, "Coinbot: A covert botnet in the cryptocurrency network," in *22nd Int. Conf. on Information and Communications Security (ICICS), Copenhagen, Denmark*, 2020, pp. 107–125.

[7] A. Swedan, A. N. Khuffash, O. Othman, and A. Awad, "Detection and prevention of malicious cryptocurrency mining on internet-connected devices," in *2nd Int. Conf. on Future Networks and Distributed Systems*, 2018, pp. 1–10.

[8] A. Yazdinejad, H. HaddadPajouh, A. Dehghantanha, R. M. Parizi, G. Srivastava, and M.-Y. Chen, "Cryptocurrency malware hunting: A deep recurrent neural network approach," *Applied Soft Computing*, vol. 96, p. 106 630, 2020.

[9] O. Caspi, *AT&T Alien Labs finds new Golang malware (BotenaGo) targeting millions of routers and IoT devices with more than 30 exploits*, AT&T Alien Labs, 2021. [Online]. Available: https://cybersecurity.att.com/blogs/labs-research/att-alien-labs-finds-new-golang-malwarebotenago-targeting-millions-of-routers-and-iot-devices-with-more-than-30-exploits (visited on 12/23/2022).

[10] M. Hopkins and A. Dehghantanha, "Exploit kits: The production line of the cybercrime economy?" In *2nd IEEE Int. Conf. on Information Security and Cyber Forensics (InfoSec)*, 2015, pp. 23–27.

[11] F. Veysset *et al.*, "New tool and technique for remote operating system fingerprinting," *Intranode Software Technologies*, vol. 4, 2002.

[12] SaturnsVoid, *GoBot2*, version 8a6d4952279ad7ea71d5c6ff5ffbc95ccb65e731, GitHub, 2021. [Online]. Available: https://github.com/SaturnsVoid/GoBot2.

[13] B. Krebs, *KrebsOnSecurity Hit With Record DDoS*, KrebsOnSecurity, 2016. [Online]. Available: https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/ (visited on 01/28/2023).

[14] jgamblin, *Mirai-Source-Code*, version 3273043e1ef9c0bb41bd9fcdc5317f7b797a2a94, GitHub, 2017. [Online]. Available: https://github.com/jgamblin/Mirai-Source-Code.

[15] *Discord CDN: A Popular Choice for Hosting Malicious Payloads*, Zscaler: Security Insights, Feb. 2021. [Online]. Available: https://www.zscaler.com/blogs/security-research/discord-cdn-popular-choice-hosting-malicious-payloads (visited on 03/18/2023).

[16] J. Wrieden, *Who is Trickbot?* Cyjax. [Online]. Available: https://www.cyjax.com/app/uploads/2022/07/Who-is-Trickbot.pdf (visited on 03/29/2023).

[17] J. Frankenfield, *Mining Pool: Definition, How It Works, Methods, and Benefits*, Investopedia. [Online]. Available: https://www.investopedia.com/terms/m/mining-pool.asp (visited on 04/26/2023).

[18] RootKiter, *Art of Steal: Satori Variant is Robbing ETH BitCoin by Replacing Wallet Address*, NetLab, 2018. [Online]. Available: https://blog.netlab.360.com/art-of-steal-satori-variant-is-robbing-eth-bitcoin-by-replacing-wallet-address-en/ (visited on 01/28/2023).

[19] *51% Attacks*, MIT - Digita Currency Initiative. [Online]. Available: https://dci.mit.edu/51-attacks (visited on 05/01/2023).

[20] A. Back *et al.*, "Hashcash-a denial of service countermeasure," 2002.

[21] C. Tardi, *Application-Specific Integrated Circuit (ASIC) Miner*, Investopedia. [Online]. Available: https://www.investopedia.com/terms/a/asic.asp (visited on 04/26/2023).