Trend
Research

# Red Team Tools in the Hands of Cybercriminals and Nation States

## Developing a New Methodology for Managing Open-Source Threats

Stephen Hilt, Aliakbar Zahravi

# Contents

Published by
**Trend Research**

Thanks to
**Lord Remorin**

For **Raimund Genes (1963-2017)**

The constantly evolving threat landscape has necessitated the integration of innovative methodologies into cybersecurity practices. Red teaming, a simulation-based approach, has emerged as a crucial component for organizations to fortify their defenses against sophisticated attacks. By using the expertise of red teams to identify vulnerabilities and provide critical insights, organizations can enhance their security measures and stay ahead of emerging threats.

Despite its benefits, the dual-use nature of red team tools also presents significant risks, which highlights the importance of ethical guidelines and robust detection capabilities. Malicious actors have increasingly been using these tools to launch attacks on unsuspecting victims, making it necessary for organizations to adapt their defense strategies accordingly.

The exploitation of open-source software in the supply chain underscores the complexity and challenges in maintaining secure software ecosystems. In this research, we will explore the importance of balancing innovation with ethics and discuss the integration of AI-driven analysis into traditional red teaming tactics to enhance detection and response capabilities. We will also examine the need for organizations to stay ahead of the curve and address the growing threat posed by malicious actors exploiting red team tools.

# Overview of the Project

The development of advanced malware involves malicious actors employing their available resources to create a sophisticated environment. Advanced persistent threat (APT) groups, which are often state-sponsored or highly skilled organizations, usually engage in prolonged cyberespionage campaigns using complex malware.

Meanwhile, independent malware authors typically create malicious software for financial gain, personal satisfaction, or to sell to other criminals. Both types of developers continuously make use of their shared knowledge and tools to improve their malware.

Red teams, composed of cybersecurity professionals simulating real-world attacks, can inadvertently contribute to this ecosystem by developing sophisticated attack methods and tools during testing and assessment processes.

Malware development relies on diverse sources for gathering necessary tools and information. For example, GitHub, one of the primary online platforms for sharing code, can be exploited for malicious purposes. Other critical resources include the Exploit Database, leaked source code, proof of concept (PoC) code, blogs, and hacking forums. These platforms provide a wealth of information and raw materials.

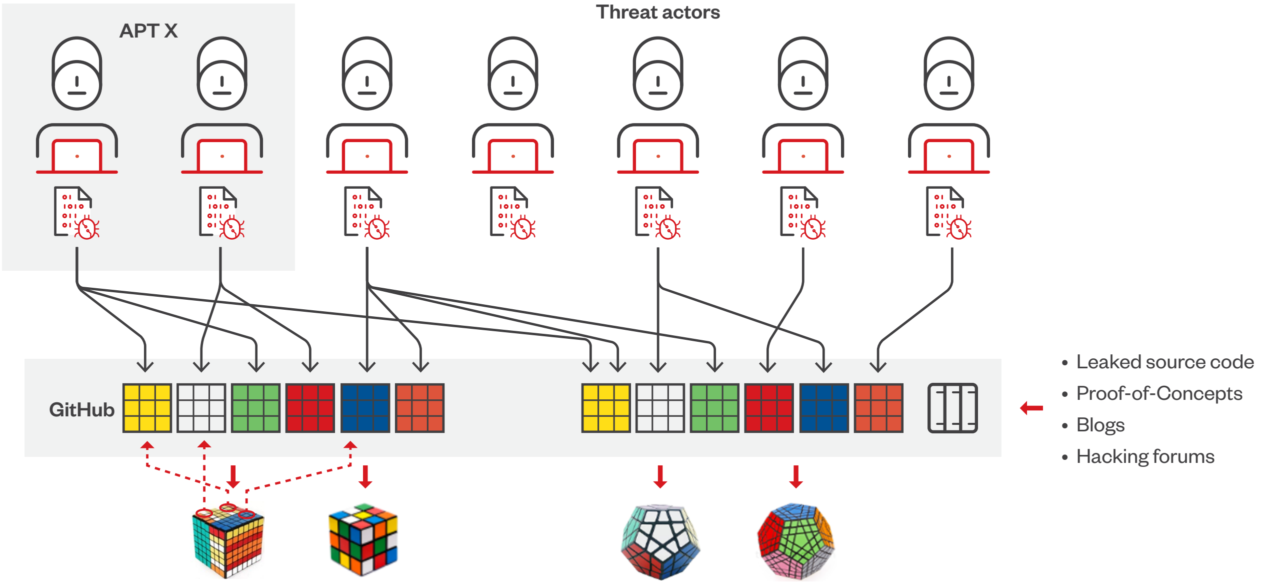


Figure 1. The malware development process

The development process involves a flow of information from diverse sources to malware creators, highlighting its collaborative nature. Technical vulnerabilities, practical implementation guides, and other resources enable developers to create malware that is effective and difficult to counter.

Red teams, by their very nature, are focused on strengthening cybersecurity defenses, which can also contribute to the pool of knowledge and techniques that can be repurposed for malicious motives. It's essential to distinguish between the benign and malicious use of red team tools, emphasizing the importance of intent and context in their application.

Effective cybersecurity strategies must ensure these tools are used ethically and responsibly, while also acknowledging the challenges faced in combating malware within the ever-evolving landscape of cybersecurity.

In summary, malware development is a complex process involving various actors, resources, and platforms. It's essential to understand the collaborative nature of this endeavor and the potential risks associated with the misuse of red team tools to develop effective countermeasures against malicious activities.

# What is Red Teaming?

Red teams are specialized groups that use techniques, tactics, and procedures that allow them to emulate attackers to test and improve organization security. The primary objective of red teams is to enhance security by identifying weaknesses and verifying the effectiveness of security measures before actual attackers can exploit them. These teams provide an essential perspective to organizations by adopting an adversarial approach to challenge planning, operations, and systems.

## History of Red Teams

The concept of red teaming has its roots in military strategy. Historically, armies would simulate battles by having one team (the "red team") play the role of the enemy to provide realistic opposition to the forces being trained (typically represented by the "blue team"). This practice allowed military strategists to explore different scenarios and tactics from an opponent's perspective.

The adoption of red teaming in cybersecurity began to gain traction during the late 1990s and early 2000s as organizations realized the value of proactive security assessments. Initially, these practices were primarily adopted by government and military organizations to protect national security and critical infrastructure. Over time, as cyber threats evolved and became more sophisticated, the private sector also began to implement red team exercises as part of their security protocols.

## Evolution and Role in Cybersecurity

In the realm of cybersecurity, red teaming has evolved from basic penetration testing to comprehensive programs that include a variety of attack simulations, social engineering, physical security assessments, and more. Red teams often use tools and techniques that are at the cutting edge of technology to simulate potential attacks as realistically as possible.

Red teams are particularly valuable in industries that are highly regulated or have significant security needs (such as finance, healthcare, and critical infrastructure). They help organizations not only detect potential vulnerabilities but also understand the real-world implications of having these weaknesses exploited. This circles back to organizations to help them prioritize the mitigation of potential exploitable risks.

The feedback and insights provided by red teams are used to refine security policies, strengthen systems, and train personnel to better defend against actual cyberthreats. This iterative process of testing and improvement plays a crucial part in maintaining an organization's resilience against evolving cybersecurity challenges.

By combining their expertise with advanced tools such as Metasploit, Bloodhound, Sliver, and Havoc, red teams can simulate complex attacks that mimic those used by sophisticated threat actors. These open-source tools are developed by companies that also offer professional red teaming services, allowing for a unique synergy between the two. For example, Rapid7's Metasploit is used to spearhead its penetration testing engagements, while BSquare's Bloodhound is designed to be used with their red teaming services. This integration of open-source tools and expert red team groups enable organizations to gain valuable insights into their security posture and stay ahead of the latest threats.

# GitHub: The Ultimate Knowledge Base for Red and Blue Teamers

Originally conceived as a platform for collaborative software development, GitHub now hosts a vast array of resources that cybersecurity professionals can use to enhance their skills, share techniques, and collaborate on cutting-edge projects. From repositories containing tools and scripts to detailed documentation and research on the latest vulnerabilities, GitHub offers an unparalleled repository of collective knowledge. For red teamers, this means access to a dynamic, constantly updated pool of information that is critical for conducting security assessments and improving overall defense mechanisms.

SOC analysts, blue teamers, and defenders can greatly benefit from studying and examining red team repositories on code repositories such as GitHub. By doing so, they can identify the latest techniques, including previously unknown methods. This proactive approach allows defenders to anticipate and understand the tactics, techniques, and procedures that threat actors might employ. Consequently, they can enhance their detection capabilities, improve their security products, and bolster their defenses against potential cyberattacks. Leveraging the insights gained from red team repositories, SOC teams can develop more effective detection strategies for their organizations.

# Current Trends and Future Directions

With the increasing prevalence of cyberattacks, red teaming has become an integral part of cybersecurity strategies for many organizations. The future of red teaming may see even more integration with artificial intelligence and machine learning to simulate attacks and predict potential breaches with greater accuracy.

## Dual Use of Red Team Tools

Red team tools are often dual-use technologies. While they are primarily developed for the legitimate purpose of testing and improving security, their capabilities make them attractive to malicious actors. These tools can automate the discovery of vulnerabilities, perform sophisticated penetration tests, and simulate social engineering attacks, among other functions. When such tools fall into the wrong hands, they can be used to conduct malicious activities against the target, including performing reconnaissance, gaining unauthorized access, and executing attacks.

## Cybercriminal Exploitation

Cybercriminals can use red team tools to streamline their operations. By employing these sophisticated tools, they can efficiently scan for vulnerabilities across a wide range of systems and exploit them at scale. This not only increases the effectiveness of attacks, but also reduces the amount of time and resources required to achieve objectives. For example, tools that automate the exploitation of known vulnerabilities can enable malicious actors to compromise several systems before organizations can apply necessary patches.

An example of such criminal misuse can be seen in the Water Dybbuk campaign.[1] In this instance, malicious actors adapted and used advanced red teaming tools to carry out targeted attacks. The campaign involved using phishing email attachments to lure users into divulging their usernames and passwords, eventually leading to financial theft and espionage.

The email attachment received by the user is obfuscated using an open-source JavaScript obfuscator (*obfuscator.io*). Once the attachment is opened, it employs a criminal toolkit to perform several checks on the target's machine to ensure it is indeed the intended target. If the check is successful, the user is redirected to a phishing page containing *Evilginx2*, another open-source tool designed for red teams.

This example highlights how red team tools, initially designed for securing networks, can be repurposed by cybercriminals to conduct sophisticated and damaging cyberattacks. In this case, it was a BEC threat actor using multiple open-source tools to perform their attacks against targets around the world.

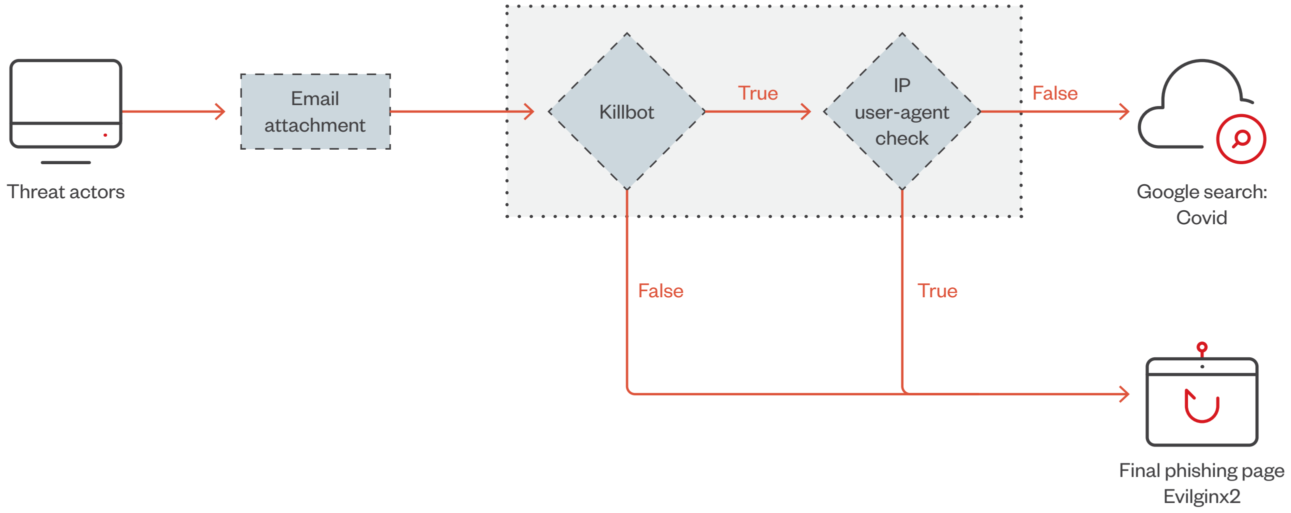


Figure 2. How the attack works with a targeted email to the target in an attempt to phish their credentials

# Integration of Red Team Toolkits by Nation-State Threat Actors

Nation-state actors, who often have greater access to resources and advanced capabilities, can use red team tools for strategic objectives such as espionage, sabotage, or influence.

These threat actors might customize red team tools to create tailored attack vectors that target specific technologies or organizations. Such targeted attacks can be part of a broader set of cyberwarfare strategies and are particularly dangerous due to their potential to disrupt critical infrastructure, steal sensitive information, or undermine national security. Moreover, the accessibility of these tools lowers the barrier to entry for lesser funded nation-states, enabling them to engage in cyber activities that were previously beyond their capabilities. This widespread accessibility of tools means that even smaller nations can pose a significant threat, increasing the complexity and scope of the global cybersecurity landscape.

Nation states and threat actors that make use of red teaming tools can effectively blend into the digital landscape, making detection and attribution challenging. By using widely-available tools, they can disguise their activities among the legitimate security testing traffic generated by ethical hackers and security researchers, or by other malicious actors who also employ these tools. Additionally, open-source tools often have well-documented capabilities and behaviors, allowing threat actors to manipulate their operations to avoid detection.

For example, the threat actor APT41 uses a red teaming tool that employs Google infrastructure in its operations, which are aimed at stealing sensitive information.[2] Google Command and Control (GC2) uses various Google's services, such as Google Sheets or Google Drive, which are used to facilitate communication between a malware-infected device and the attacker's server. This technique allows cybercriminals to use legitimate and trusted Google infrastructure to manage and control compromised systems, often making detection and blocking by security systems more difficult. Recent reports demonstrate these kinds of tactics.[3]

The following table provides an overview of open-source red teaming tools that have been used by nation-state actors over the past few years. The table serves as a resource for understanding how these tools are leveraged in real-world scenarios, offering insights into the techniques and capabilities employed by APT groups. Note that this is only an example of the tools that are being used and is not meant to be an inclusive list.

| Tool | Group | Description |
|---|---|---|
| Empire[4, 5] | Various nation-state actors | Post-exploitation framework that includes a pure PowerShell 2.0 Windows agent and a Python 2.7 Linux/ OS X agent |
| AresRAT[6, 7, 8] | Fishing Elephant | Ares is an open-source remote access tool (RAT) developed in Python, designed to offer remote control of systems across multiple platforms, including Windows, and Linux. It features reverse shell functionality, file upload/download capabilities, and the ability to execute commands remotely on compromised systems. |
| Google Command and Control (GC2)[9, 10] | APT41 | Tool for testing and debugging Web Receiver applications used in targeted information-stealing attacks |
| Sliver[11, 12, 13] | Various nation-state actors (eg. TA551) | Sliver is an open-source, cross-platform command and control (C2) framework written in Go |
| Impacket[14, 15] | Various nation-state actors | Python classes for working with network protocols, used for lateral movement and privilege escalation |
| Metasploit[16]<br>Meterpreter[17, 18] | Various nation-state actors (eg. APT 41) | Framework for developing, testing, and executing exploits against various targets |
| Mimikatz[19] | Various nation-state actors | Tool for extracting plaintext passwords, hashes, PIN codes, and kerberos tickets from memory |
| Nishang[20] | Various nation-state actors | Offensive security framework for PowerShell |
| PowerSploit[21] | Various nation-state actors | PowerShell modules aiding penetration testers and red teamers |
| Pupy[22] | Various nation-state actors | Cross-platform remote administration tool (RAT) used for command-and-control operations |
| SILENTTRINITY[23, 24] | Various nation-state actors | Cross-platform  Post-exploitation command-and-control framework powered by Python and .NET's DLR |
| BlueShell[25, 26] | Dalbit | Cross-platform backdoor malware written in Go that targets Windows, Linux, and macOS systems; supports remote command execution and file transfer, and acts as a Socks5 proxy, using encrypted communication to avoid detection |
| DNS-Persist[27, 28] | REF2924 | Post-exploitation agent that uses DNS for command and control. |
| ligolo[29, 30] | MuddyWater | Reverse tunneling made easy for pentesters, by pentesters |
| Antak[31, 32] | Various nation-state actors (e.g APT39, ATK15:Emissary Panda) | An ASP.NET web shell designed for post-exploitation on Windows servers that allows attackers or penetration testers to execute system commands, upload/download files, and interact with processes via a web interface |

Table 1.  Tools used by nation-state actors

# Ethical and Legal Considerations

The use of red team tools by malicious actors raises important ethical and legal questions. It underscores the need for a balanced approach that supports the development and dissemination of these tools for defensive purposes while implementing measures to prevent their misuse. Organizations and developers may need to consider ethical guidelines, user verification processes, and possibly even restrictions on certain features or capabilities to mitigate the risks of misuse.

While red team tools are important to cybersecurity defenses, their power and capabilities also make them susceptible to misuse by criminals and nation-state actors. Addressing this dual-use challenge requires careful consideration of security, ethical standards, and looking into regulatory frameworks to ensure that these tools continue to serve as a force for good rather than becoming instruments of malicious intent.

# Evolution of Red Teaming Methodologies and Detection Techniques

Red teaming methodologies have undergone significant transformations as the cybersecurity environment has grown in complexity. Initially focused on penetration testing, these methodologies were mainly geared towards identifying and exploiting technical system vulnerabilities. This approach was designed to simulate perimeter hacking attempts to test the robustness of network defenses.

However, this traditional approach has been criticized for its limited scope, which primarily focuses on direct system vulnerabilities without incorporating a broader security perspective. As a result, modern red teaming has expanded to include a wider array of tactics, such as social engineering, physical breach attempts, insider threat simulations, and complex multi-vector attack scenarios.

## Criticisms and Evolution of Detection Strategies

A significant limitation of earlier red teaming approaches is their narrow focus on offensive tactics without integrating robust detection strategies for red team tools and activities. According to a blog post[33] by cybersecurity expert Florian Roth, many of the traditional red team exercises lack sufficient alignment with broader security strategies and fail to address how organizations can detect and respond to the effective use of red team tools.

This critique points out that while red teams often successfully highlight security flaws, the exercises do not necessarily lead to substantial or long-lasting security enhancements because they do not adequately focus on improving detection and response mechanisms.

Florian also emphasizes that glorifying red team breaches without improving detection capabilities can lead to repetitive cycles of vulnerability exploitation without meaningful security enhancement. This issue underscores the need for red teams to not only test defenses but also improve how organizations detect and neutralize red team tools during simulated attacks.

## Integrating Red Teaming with Enhanced Detection Capabilities

To move past these limitations, there is an increasing emphasis on enhancing detection methodologies alongside traditional red team tactics. This integrated approach includes:

- **Alignment of red team exercises with detection strategies:** Ensuring that red teaming is not just about breaching defenses but also about testing and refining the organization's ability to detect and respond to the tools and tactics used by the red team.
- **Collaborative efforts between teams:** Promoting a collaborative environment where red teams and blue teams (defensive teams) work together to understand each other's tactics, with a particular focus on the detection of red team activities, will greatly enhance the organization’s overall security posture.

- **Feedback loops for continuous improvement:** Developing mechanisms within an organization to ensure that both the successes and failures of red team exercises feed into continuous improvements in detection techniques, ensuring more effective identification of, and defense against vulnerabilities.

By addressing the critiques raised by experts and incorporating more sophisticated detection techniques, the role of red teaming can be elevated from merely testing security defenses to also enhancing the organizational capability to identify and respond to real-world cyberthreats. This holistic approach ensures that red teaming remains a crucial component of a comprehensive cybersecurity strategy.

# Adapting Open-Source Red Teaming Tools into the Cyberattack Ecosystem

## Overview

Malware attacks are constantly becoming more sophisticated and complex with increasingly elaborate attack chains across a range of systems. APT groups and malicious actors often seek to improve their capabilities by leveraging the growing ecosystem of open-source tools. Rather than building malware components from scratch, they frequently turn to repositories like GitHub, where security researchers and developers have already done much of the groundwork in creating effective and evasive modules and tools. By repurposing these open-source resources, malicious actors can streamline their attack preparations, efficiently integrating pre-built components into their arsenal and customizing them to fit their specific needs.

This tactic allows attackers to avoid the time-consuming process of researching and developing advanced malware tools from the ground up. Instead, they borrow code from existing red teaming tools that are designed for ethical hacking and adapt them for malicious purposes. This trend has been accelerating as malware authors recognize the convenience and versatility of using these readily available tools. While the initial intent of publishing such software is to improve cybersecurity defenses, malicious actors have increasingly been exploiting these resources, further enhancing the complexity and effectiveness of their attack strategies. The result is a cybersecurity landscape where defenders must continually adapt to an ever-evolving ecosystem of threats, as adversaries increasingly incorporate these open-source tools into their attack methodologies.

This rise in the adoption of open-source red teaming tools presents a significant challenge to cybersecurity professionals. Organizations must defend themselves against attackers who are well-equipped with highly capable malware built on freely available code.

This freely available code can be invaluable for security vendors and researchers by providing insights into the attacker techniques, helping them understand the internal mechanisms of attacks and enabling the development of effective detection and prevention methods against these tactics. Additionally, it is not just about detection – analyzing these open-source red-teaming repositories can provide significant insights into various techniques to penetrate systems and bypass security measures. By studying these resources, researchers can devise innovative tools and strategies to identify and block potential attacks before they are exploited by cybercriminals and adversaries.

Traditional security measures often fall short due to the dynamic and customizable nature of these adapted tools. Consequently, defenders are required to strengthen their incident response capabilities, invest in threat intelligence, and implement proactive measures such as behavioral analysis and network monitoring to detect unusual activities that are indicative of malicious intent. Currently, open-source code is frequently found in various cyberattacks and complex malware. In these instances, the attackers borrow code or complete modules from legitimate open-source tools (such as custom network communication modules) and adapt these components for malicious purposes within their attack infrastructure.

# Common Strategies for Malicious Actors Using GitHub Red-Teaming Tools

Malicious actors typically employ red-teaming tools in three ways. Note that this is a non-exhaustive list, and that there are other methods that are not listed here.

1. **Using the Complete Project**

The malicious actors use open-source tools exactly as they are, taking advantage of their fully developed capabilities. However, when a repository is well-known (with a high number of forks and stars), the detection rate tends to be high, making immediate usage less effective. In such cases, attackers often employ their own custom loader, with the primary payload fully encrypted and loaded directly into memory, resulting in a fileless attack. C2 Matrix[34] compiles a comprehensive list of open-source C&C frameworks.

2. **Using the Repository as a Groundwork and Adding Custom Code**

In this scenario, malicious actors use the repository as a groundwork or a template, then add their own code based on their specific needs. In such cases, they typically apply obfuscation techniques to the code or modules, such as string obfuscation or Windows API call hashing, to evade pattern detection, complicate reverse engineering, and disguise the tools. Alternatively, they might remove unnecessary functions and code to streamline the tool for a specific purpose, making it leaner and more focused on a particular task.

For instance, in our February 2023 blog post, Enigma Stealer Targets Cryptocurrency Industry with Fake Jobs,[35] we highlighted how the suspected Russian threat actor employed various open-source red teaming projects to design a sophisticated and evasive attack scheme. This attack aimed to infect systems and steal sensitive information, specifically targeting the cryptocurrency industry. The following image highlights the section that originated from the open-source projects for this attack:

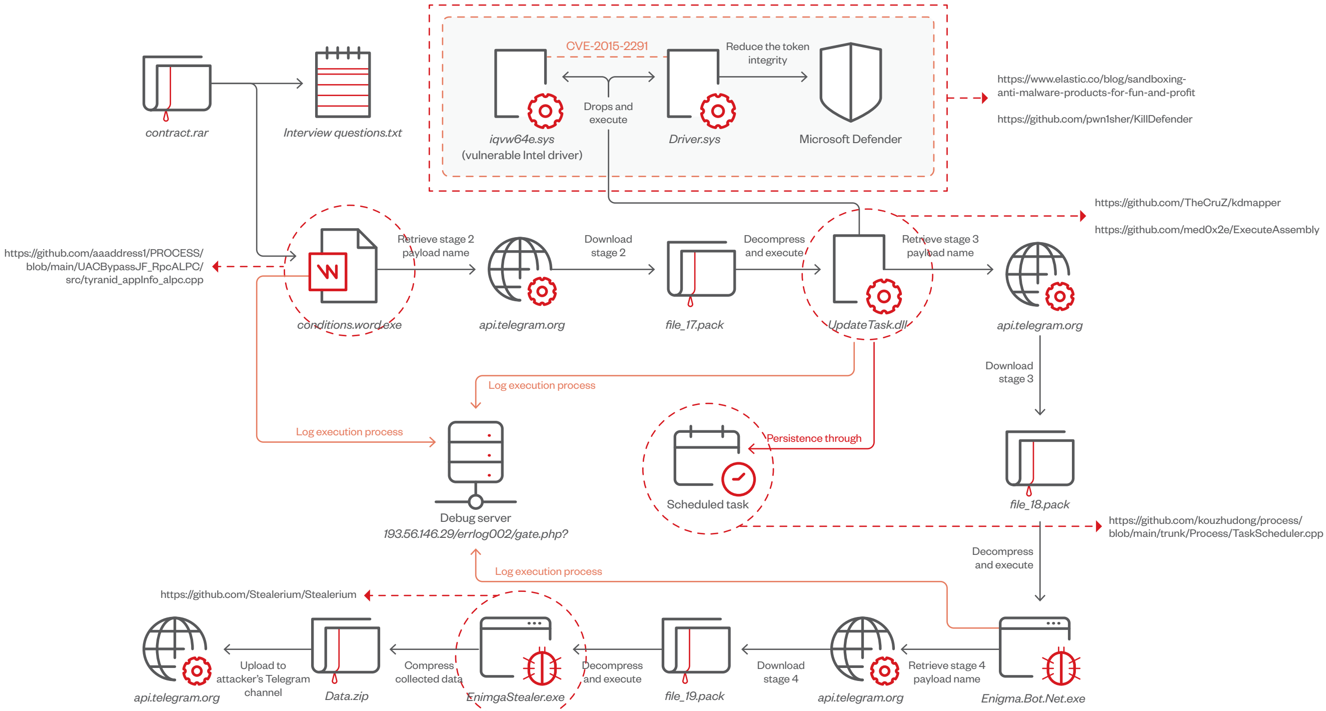

contract.rar
Interview questions.txt
CVE-2015-2291
Drops and execute
iqvw64e.sys
(vulnerable Intel driver)
Driver.sys
Reduce the token integrity
Microsoft Defender
https://www.elastic.co/blog/sandboxing-anti-malware-products-for-fun-and-profit
https://github.com/pwn1sher/KillDefender
https://github.com/aaaddress1/PROCESS/blob/main/UACBypassJF_RpcALPC/src/tyranid_appInfo_alpc.cpp
conditions.word.exe
Retrieve stage 2 payload name
api.telegram.org
Download stage 2
file_17.pack
Decompress and execute
UpdateTask.dll
Retrieve stage 3 payload name
api.telegram.org
https://github.com/TheCruZ/kdmapper
https://github.com/med0x2e/ExecuteAssembly
Log execution process
Log execution process
Debug server
193.56.146.29/errlog002/gate.php?
Persistence through
Scheduled task
Download stage 3
file_18.pack
https://github.com/kouzhudong/process/blob/main/trunk/Process/TaskScheduler.cpp
Decompress and execute
Log execution process
https://github.com/Stealerium/Stealerium
api.telegram.org
Upload to attacker's Telegram channel
Data.zip
Compress collected data
EnimgaStealer.exe
Decompress and execute
file_19.pack
Download stage 4
api.telegram.org
Retrieve stage 4 payload name
Enigma.Bot.Net.exe

Figure 3. The attack kill chain used by the Enigma Stealer operator

One of the tools they used, called KDMapper,[36] manually maps unsigned or self-signed drivers into memory by exploiting the *iqvw64e.sys* Intel driver. This technique was used to disable Windows Defender, providing attackers free rein to execute their malicious activities without being detected.

The perpetrators remove all unnecessary sections from the original code, making it more efficient and compact, then apply string and Windows API call obfuscation to bypass pattern detection.

The driver's sole purpose is to patch the integrity level of Microsoft Defender (*MsMpEng.exe*) and forcibly reduce it from system to untrusted integrity. The reduction of the integrity level to untrusted impedes the process of accessing secure resources on the system for the victim, silently disabling it without terminating the process.

This driver is a modified version of an open-source proof-of-concept (POC),[37] which was created based on an Elastic Security Lab article titled "Sandboxing Antimalware Products for Fun and Profit."[38]

The malicious actor uses specific functions and modules as a foundation, modifying and updating the code to fit the intended requirements.

To execute the final stage payload in memory (written in C#), the attacker employs another open-source project called ExecuteAssembly.[39] This tool serves as an alternative to Cobalt Strike's execute-assembly module, built with C/C++ to load or inject .NET assemblies.

The final stage malware of this attack chain is modified version of Stealerium, which is an open-source project written in C# and markets itself as a stealer, clipper, and keylogger with logging capabilities using the Telegram API.

To achieve persistence and bypass User Account Control (UAC), the malicious actor uses open-source code, including TaskScheduler.cpp and tyranid_appInfo_alpc.cpp.[40] These tools facilitate the execution of malicious activities by exploiting existing vulnerabilities and enabling elevated privileges without user intervention.

### 3. **Using the Repository as a Knowledge Hub**

GitHub repositories are an invaluable resource for knowledge sharing and collaboration, acting as virtual libraries of code, documentation, and innovative ideas. Many repositories go beyond basic functionality, showcasing cutting-edge red teamer techniques that push the boundaries of cybersecurity.

For example, some repositories contain implementations of advanced techniques like the Process Doppelgänging attack,[41] a code injection technique that was unveiled at the Black Hat Europe conference by Tal Liberman and Eugene Kogan of enSilo lab in December 2017.[42] This technique abuses the NTFS transaction mechanism in Windows to create a process from a legitimate executable file, but then replaces the executable image with malicious code during process creation. This allows the malicious code to run under the guise of a legitimate process, making it difficult for security tools to detect.

Shortly after this technique was released, many GitHub repositories started showcasing POCs for it. Various threat actors and malware authors quickly began employing this technique.

Kaspersky Lab researchers, Anton Ivanov, Fedor Sinitsyn, and Orkhan Mamedov OV, discovered a new variant of the SynAck ransomware trojan in April 2018.[43] The trojan uses the doppelgänging technique to bypass antivirus security by hiding in legitimate processes. This is the first time that the doppelgänging technique had been observed being used in a ransomware attack in the wild.

Another example of this technique's use is when Symantec uncovered operations by a threat actor named Leafminer. Leafminer targeted a broad list of government organizations and businesses in the Middle East.[44] The group used process doppelgänging to evade security software while deploying malicious tools on compromised systems.

GitHub repositories serve as knowledge hub  for cybersecurity professionals, enabling them to explore novel evasion techniques, discover new red teaming tools and C&C frameworks, and understand emerging security threats.

# Red Teaming Tools for An Open Source Software Supply Chain

One of the most concerning cybercrime trends in recent years is the rise of supply chain attacks – particularly those that compromise codebases – as a critical global concern within the cybersecurity landscape. According to a report released by the European Union Agency for Cybersecurity (ENISA), 39% to 62% of organizations were affected by a third-party cyber incident, with only 40% of surveyed organizations saying that they understood their third-party cyber-and privacy risks.[45] Even more alarming, 38% of surveyed organizations said they were not able to know when (or whether) a cyber issue arises from a third-party component. Recent high-profile supply chain cyberattacks include malware and vulnerabilities involving heavily used software and tools such as the XZ Utils backdoor, Apache Log4j, SolarWinds Orion, and 3CX's 3CXDesktopApp, all of which shed light on how costly supply-chain attacks can be.

In modern software development, developers rely on third-party components to streamline development processes. This allows developers to create cost-effective, efficient, and feature-rich applications. However, what happens when one of these trusted components is compromised?

Attackers can indirectly infiltrate systems by targeting the less secure elements of an organization's supply chain, such as third-party vendors or software repositories. This approach allows them to compromise even trusted components, gaining a foothold in larger, more secure environments. These attacks typically involve malicious code being embedded within what appear to be legitimate software repositories. When developers integrate these components – believing them to be genuine – they unknowingly introduce vulnerabilities and other cybersecurity risks into their core systems.

During our monitoring and research, we observed that a significant number of payloads delivered at the end of the infection chain are derived from open-source projects.  In this type of attack, the perpetrator typically builds several downloader and dropper (stager) components with multi-layered obfuscation to deliver malicious code to an infected system.

In a blog entry we published in October 2023, titled "Exposing Infection Techniques Across Supply Chains and Codebases," we examined the intricate methods threat actors used to implant malicious payloads within seemingly legitimate applications and codebases.[46] We also provided hunting queries for blue teams, SOCs, and threat hunters to identify and find these kinds of techniques.

There are numerous tactics and techniques at an attacker's disposal to distribute malicious code to consumers of OSS packages – from typosquatting and dependency confusion attacks to the creation of malicious merge requests for legitimate Git repositories.[47]

The following is a list of common techniques used by malicious in OSS supply chain threats:[48]

| Threat type | Real example | Mitigation via OSS SSC Framework |
|---|---|---|
| Accidental vulnerabilities in OSS code or containers | SaltStack | Apply automated patching, display OSS vulnerabilities as pull requests |
| Intentional vulnerabilities/backdoors in OSS | phpMyAdmin | Perform proactive security review of OSS |
| Compromise of a known good OSS component | ESLint incident | Block ingestion via malware scan, single feed, all packages are scanned for malware prior to download |

| Threat type | Real example | Mitigation via OSS SSC Framework |
|---|---|---|
| Creation of a malicious package similar to a popular OSS component | Typosquatting | Perform OSS provenance analysis, single feed, all packages are scanned for malware prior to download |
| Compromise of the compiler used in an OSS build | CCleaner | Rebuild OSS on trusted build infrastructure to ensure that packages do not have anything injected at build time |
| Dependency confusion, package substitution attacks | Dependency confusion | Single feed, securely configure package source mapping |
| Addition of new malicious dependencies to an OSS component | Event-Stream incident | Scan all packages for malware prior to download, single feed |
| Tampering with the integrity of an OSS package after build | How to tamper with Electron apps | Apply digital signature or hash verification, SBOM validation |
| Removal or takedown of upstream source affecting builds | left-pad | Use package-caching solutions, mirror a copy of OSS source code to an internal location for BCDR scenarios |
| Vulnerability not fixed by upstream maintainer within a desired timeframe | Prototype Pollution in lodash | Implement a change in the code to address a zero-day vulnerability, rebuild, deploy to organization, and confidentially contribute the fix to the upstream maintainer |
| Compromise of a package manager account | Ua-parser-js | Perform OSS provenance analysis, single feed, scan OSS for malware |

Table 2. Common techniques used in OSS supply chain threats

We have compiled a list of common GitHub repositories used in OSS supply chain attacks:

| Name | | |
|---|---|---|
| W4SP Stealer[49, 50] | Creal Stealer[51, 52] | BlackCap-Grabber[53] |
| Luna-Grabber[54] | empyrean[55] | Kematian-Stealer[56] |
| Vare-Stealer[57] | Blank-Grabber[58] | TurkoRat[59] |

Table 3. Common GitHub repositories used in OSS supply chain attacks

The following is a list of common open source obfuscators for Python scripts and JavaScript. Some of these tools are frequently used by malware authors in supply chain attacks. Notable examples include BlankOBF for Python and javascript-obfuscator for JavaScript files.

| Name | | |
|---|---|---|
| BlankOBF[60] | javascript-obfuscator[61] | Vare-Obfuscator[62, 63] |
| Intensio-Obfuscator[64] | Hyperion[65] | Kramer[66] |
| gnirts[67] | jfogs[68] | javascript-obfuscator[69] |

Table 4. Examples of tools used in supply chain attacks

# Trending Repository Detection

Monitoring and detecting trending red teaming repositories on GitHub is crucial for staying updated with the latest malware and red teaming tool developments.  This section explores how to detect trending red teaming tools on GitHub based on star and fork counts, providing insights into their popularity and relevance.

## Importance of Trending Repositories

Trending repositories reflect the community's interest in tools and techniques. By identifying these trends, security researchers, SOC analysts, and blue teamers can track the popularity of red teaming tools through star and fork counts, as well as by monitoring discussions in underground forums. This enables them to identify emerging threats and techniques and allows them to deploy detections early and respond more effectively to new challenges.

## Methodology

Repository Categorization: The first step involves categorizing repositories. We focus on four main categories: Post-Exploitation, AV-Evasion, Credential Access (stealers), and Phishing Kits. Once the category is identified, we begin collecting repositories.

We quickly realized that simple topic-based search queries (e.g., topic:stealer) are not sufficient. Many repositories do not include topics (tags), making them difficult to find through these methods alone. In fact, many repositories only contain code or minimal descriptions, which causes us to miss them when relying solely on GitHub's topic modifiers.

To overcome this limitation, we conducted an in-depth analysis of repositories within each specific category. By identifying common patterns – such as typical API calls, strings, descriptions, file/function names, and other characteristics – we gained a better understanding of what to look for. Based on this knowledge, we crafted more refined GitHub queries tailored to each category, allowing us to discover additional repositories. These queries are then integrated into our automation system, which continuously monitors GitHub for new releases.

Trend Analysis: Our primary focus in this step is to identify and analyze repositories that suddenly gain a significant number of stars and forks. One automated script is responsible for running the queries, filtering out irrelevant repositories, and downloading the relevant ones into a local database. Another script continuously monitors the repositories in the database on a daily basis, tracking changes in stars and forks. When a repository's stars or forks exceed a predefined threshold, the script alerts the researchers for further analysis.

Visualization: Visualizing the data through graphs and charts helps in understanding the growth and popularity of a repository. This makes it easier to spot significant spikes in activity and overall trends.

Cumulative Growth: Tracking the cumulative growth of stars and forks provides a long-term view of the repository's popularity. This helps in distinguishing between short-lived hype and sustained interest.

# Early Detection and Response: Case Studies in Threat Hunting on GitHub

Let's put our discussion into practice. Since 2020, we have observed a significant increase in the number of GitHub repositories hosting Antivirus/ endpoint detection and response (EDR) evasion techniques, crypters/packers, code and binary obfuscators, and malware loaders.

Most of these repositories claim that their tools can bypass EDR solutions. While not all these claims are necessarily true, some might be. Even if a tool gets detected, threat actors can modify the code. With sufficient tweaking and modifications, they may be able to achieve successful evasion.

It is important to identify these types of repositories, thoroughly examine their contents, and understand their inner mechanisms. By targeting the core components and understanding how these tools operate, we can develop effective detection strategies. Analyzing the use and implementation of techniques allows us to write precise detection rules and countermeasures, ultimately enhancing defense against potential threats. Regular monitoring and updating of these detection methods are essential to staying ahead of evolving evasion tactics.

The following chart illustrates the growing number of repositories for AV/EDR evasion, code obfuscators, and trojan loaders. Note that this chart only covers until May 2024, which explains the dip from 2023 to 2024. We expect this to grow as we have seen with the previous years.

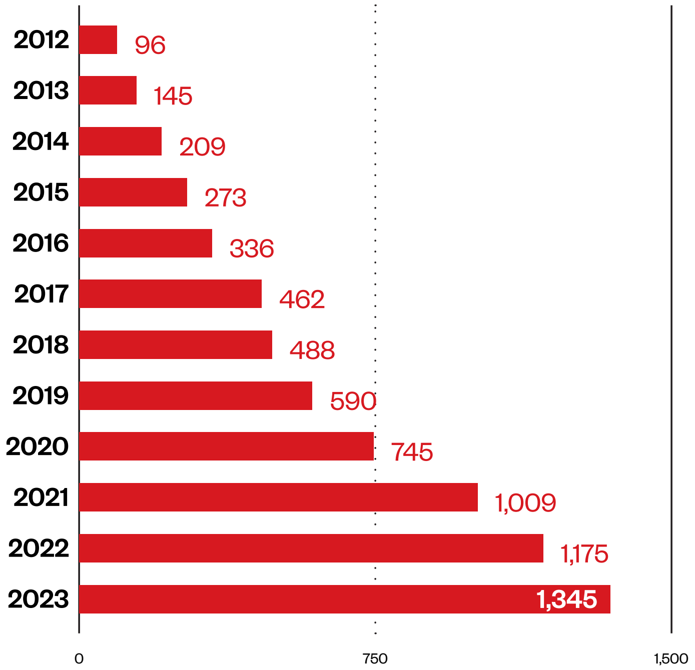


Figure 4. Count of repositories for AV/EDR evasion, code obfuscators, and trojan loaders from 2012 to 2023

Our research focuses on several categories, including GitHub repositories related to the "post-exploitation" topic from 2018 to 2023. The data also includes the annual growth percentage for each year within this period.

A total of 1,205 related repositories were created over these six years. The number of repositories grew each year, starting with 147 in 2018 and increasing to 255 by 2023. Notable growth rates include a 20.41% increase in 2019 and a 16.44% increase in 2023. The growth percentages varied, with the highest growth in 2019 and the lowest in 2022 at 2.82%.

This trend underscores the growing interest and development in post-exploitation topics within the GitHub community over the past six years.

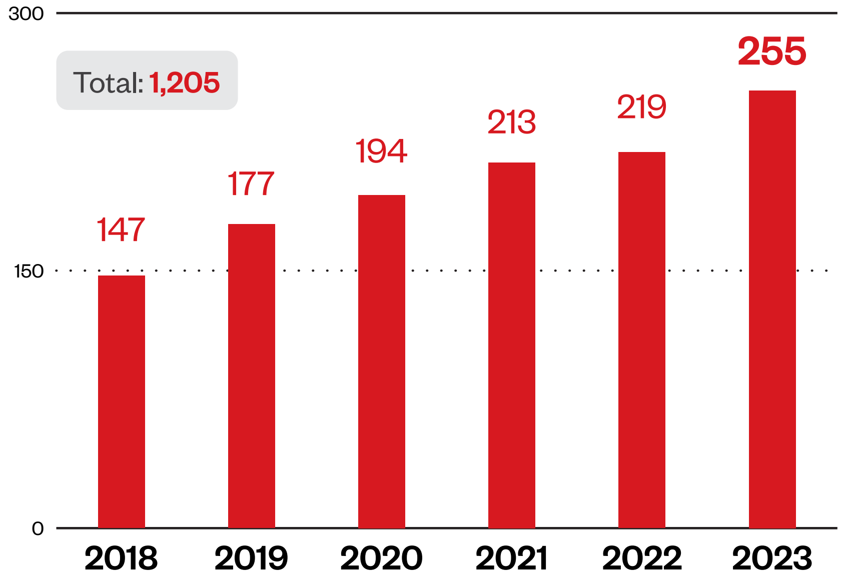
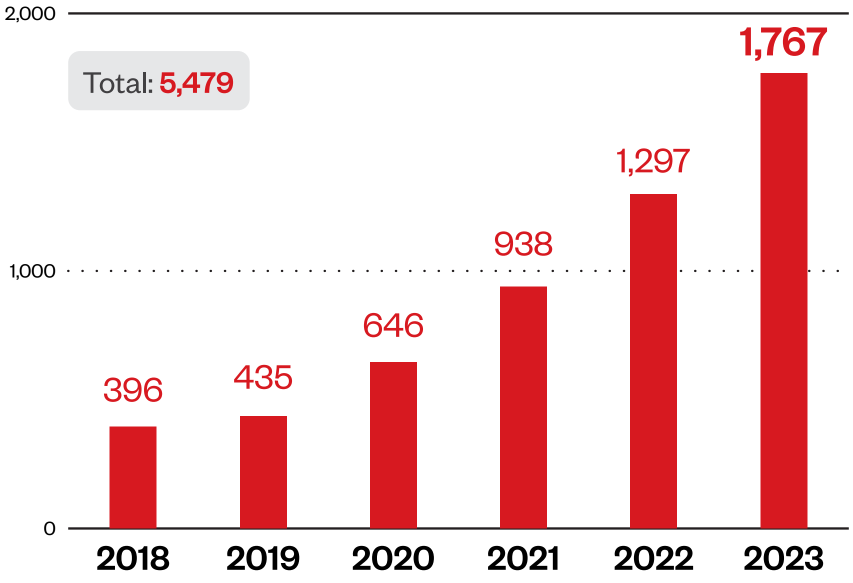


Figure 5. GitHub repositories on the "post exploitation" topic created per year (2018-2023)

The same can be applied to the categories of "information stealers," "phishing," and "AV evasion." Each of these categories is experiencing year-over-year growth. This increases the likelihood that a threat actor will begin using a tool that is relatively unknown to the industry, for which no detections or preventions have been established.



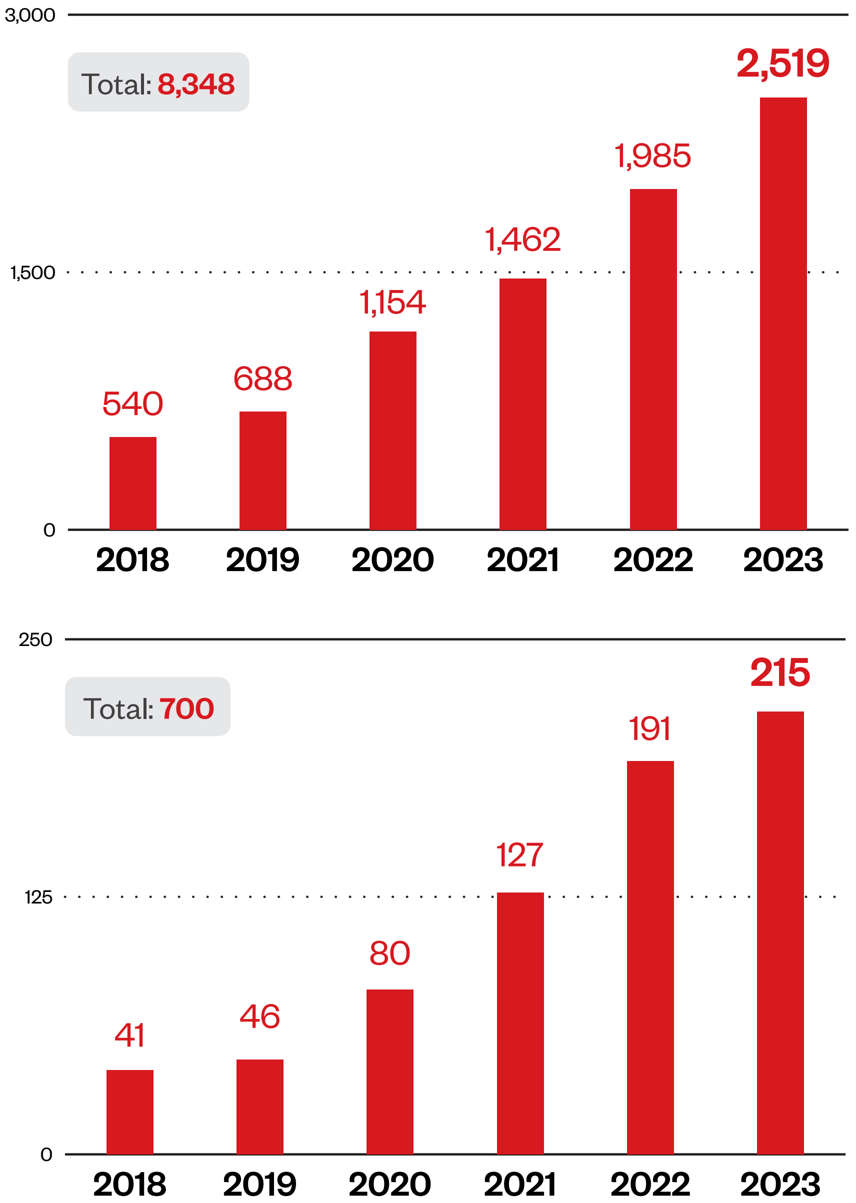


Figure 6. GitHub repositories on the "stealer," "phishing," and "AV evasion" topics created per year (2018 - 2023)

The most common languages used for these categories are illustrated in the chart below. This data reveals the preferred programming languages for developing AV/EDR evasion tools, code obfuscators, and trojan loader builders.

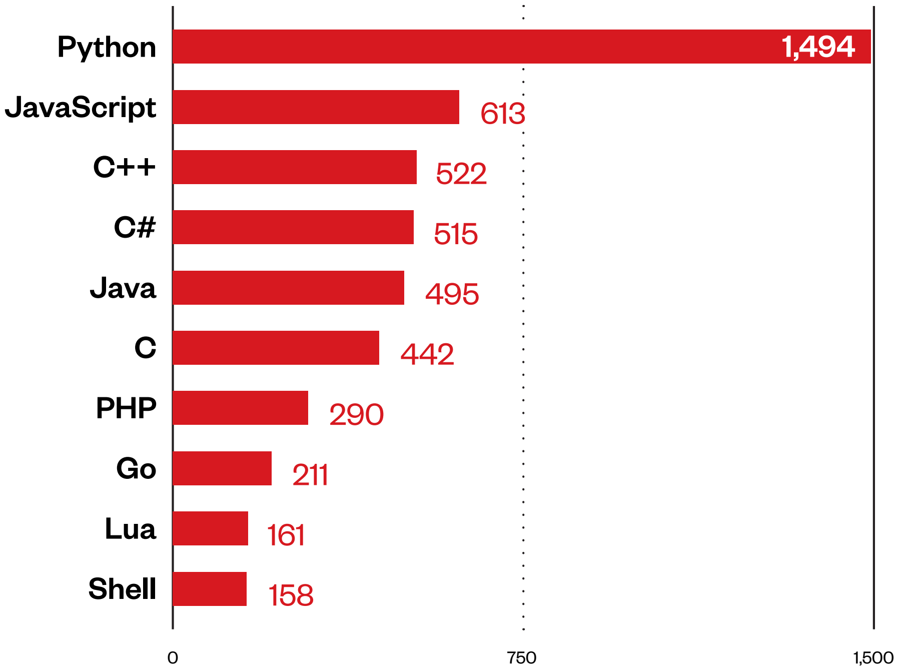


Figure 7. The top 10 programming languages used for developing AV/EDR evasion tools, code obfuscators, and trojan loader builders

The following chart categorizes offensive security tool repositories used in malicious projects, breaking them down into various cybersecurity and penetration testing topics, and displaying the number of repositories for each topic. This visualization offers an overview of the distribution and prevalence of different offensive security tools within the GitHub ecosystem. Key topics include reconnaissance, penetration testing frameworks, remote access tools, botnet/DDoS utilities, and various forms of malware. Note that this data is a snapshot from December 6, 2023.

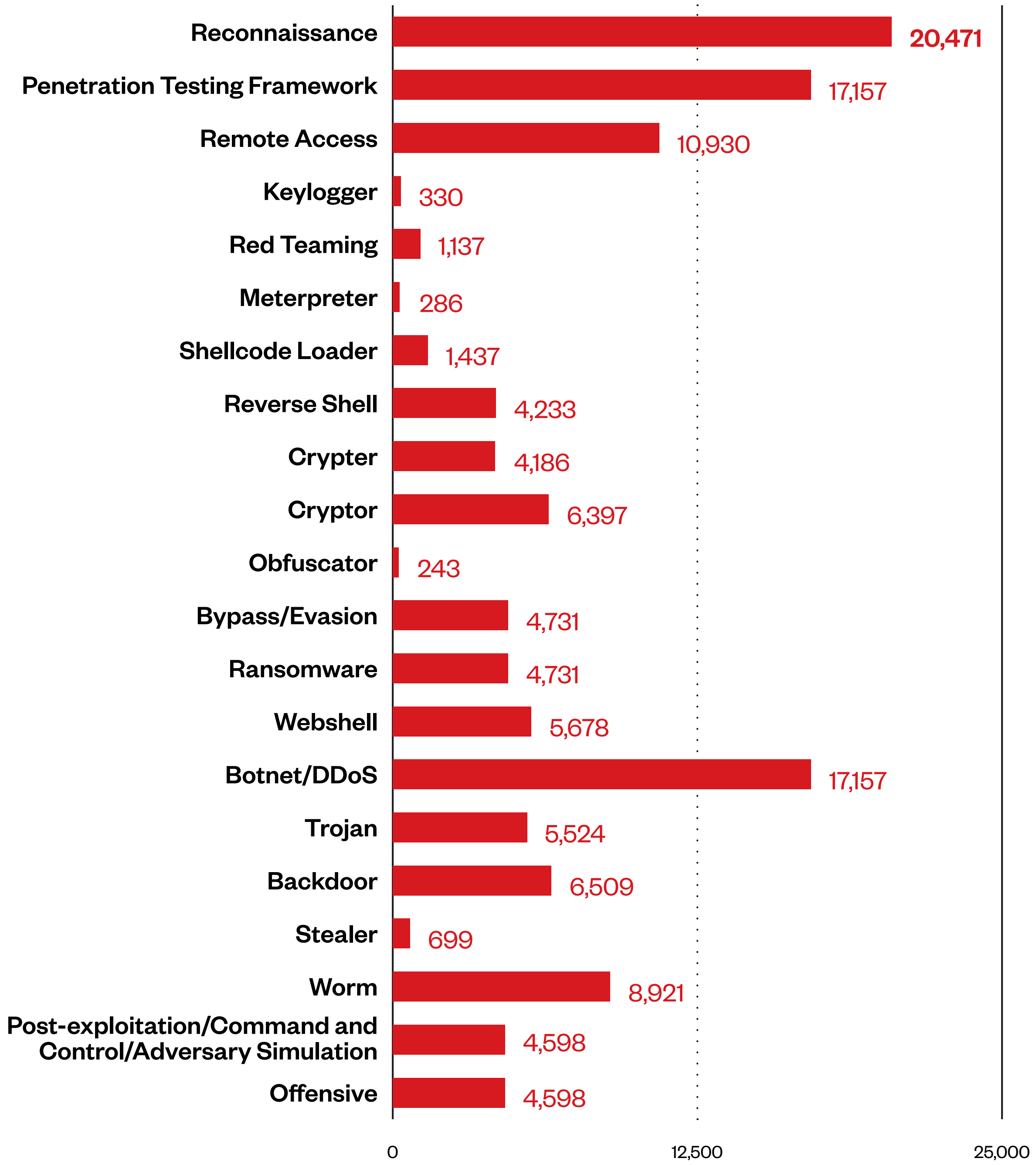


Figure 8. A breakdown of topics by offensive security tool repositories used in malicious projects

Let's examine one of these repositories called fileless-elf-exec, a tool for executing ELF binaries on Linux systems without writing to disk by leveraging the memfd_create syscall. It creates an anonymous in-memory file, maps the ELF binary, and executes it directly from memory. The loader compresses the payload using zlib and encodes it with Base64.

The following snippet shows the stars and forks of this project on GitHub.

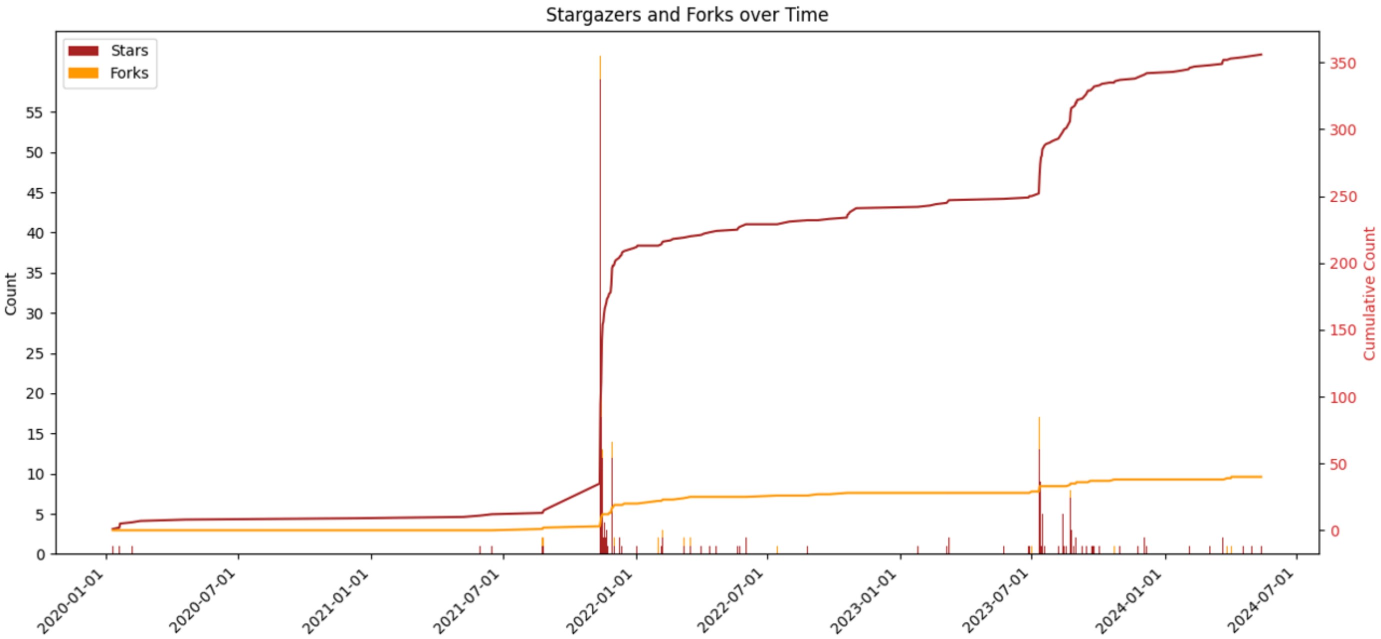
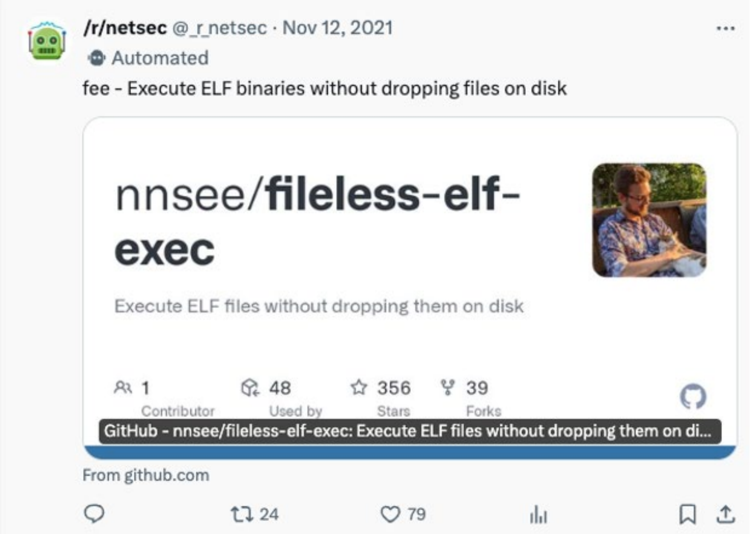


Figure 9. Stars and forks for the fileless-elf-exec repository on GitHub

On November 12, 2021, the tools were shared on Twitter and various Telegram channels, which led to the first spike in activity.


/r/netsec @_r_netsec · Nov 12, 2021

Automated

fee - Execute ELF binaries without dropping files on disk

nnsee/**fileless-elf-exec**

Execute ELF files without dropping them on disk

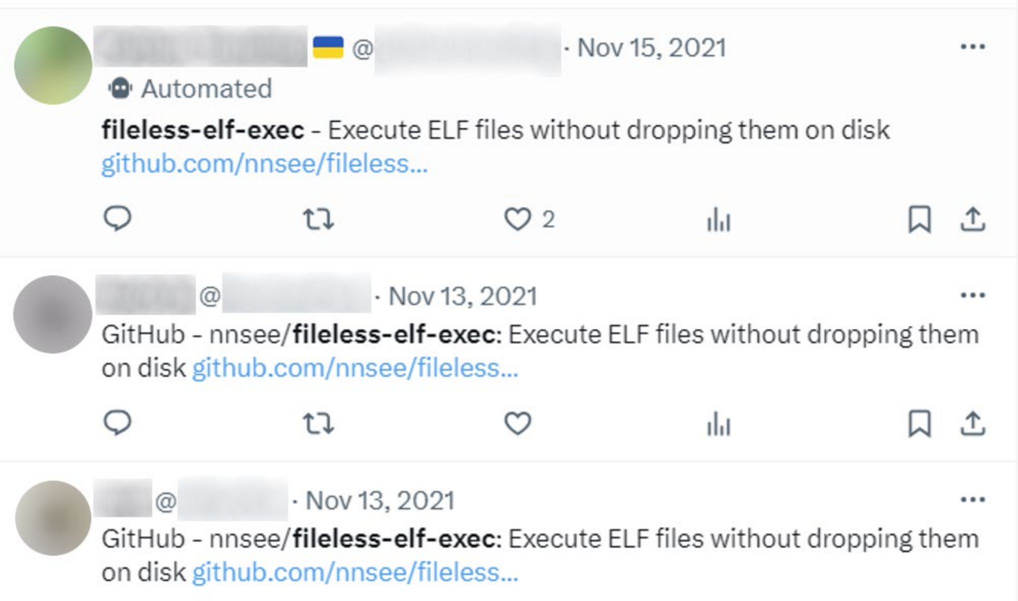
1 Contributor · 48 Used by · 356 Stars · 39 Forks

GitHub - nnsee/fileless-elf-exec: Execute ELF files without dropping them on di...

From github.com

Figure 10. The fileless-elf-exec tools being shared on Twitter

Following this, we observed a few more spikes on November 13 and November 15 when the project was reshared on Twitter.



[redacted] 🇺🇦 @[redacted] · Nov 15, 2021

Automated

**fileless-elf-exec** - Execute ELF files without dropping them on disk github.com/nnsee/fileless...

[redacted] @[redacted] · Nov 13, 2021

GitHub - nnsee/**fileless-elf-exec**: Execute ELF files without dropping them on disk github.com/nnsee/fileless...

[redacted] @[redacted] · Nov 13, 2021

GitHub - nnsee/**fileless-elf-exec**: Execute ELF files without dropping them on disk github.com/nnsee/fileless...

Figure 11. Resharing the fileless-elf-exec tools

The first public attack using these tools was reported in July 2023.[70] The Wiz blog post on PyLoose details a Python-based fileless malware that targets cloud workloads using *fileless-elf-exec*. This malware exploits the *memfd* Linux feature to load an XMRig cryptominer directly into memory, bypassing file-based detection mechanisms. The attack focuses on Jupyter Notebooks with weak security, executing system commands via Python's modules such as os and subprocess. PyLoose represents a sophisticated approach, leveraging cloud environment flexibility and the capabilities of Python for stealthy execution.

It has been reported that this is the first publicly documented Python-based fileless attack targeting cloud workloads in the wild, and evidence shows close to 200 instances where this attack was used for crypto mining.

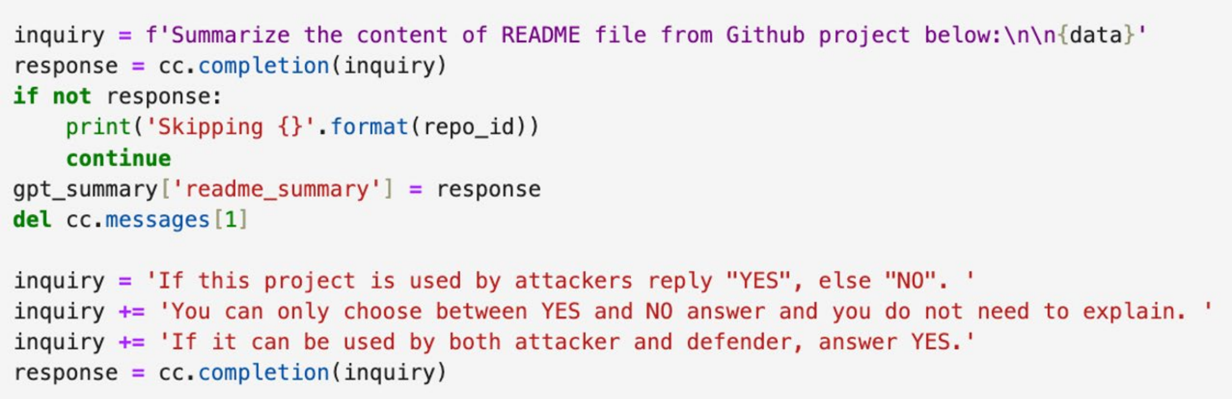
During our investigation, we identified a few new cryptocurrency miner attacks with zero detections (at the time or observation) that were delivered via the fileless ELF execution method. These attacks use an open-source miner called *cpuminer-opt*.[71]

# AI-Based Identification Method

The research project employed artificial intelligence (AI) in two significant ways to streamline the evaluation of thousands of GitHub projects, ultimately reducing the workload for human threat analysts. These AI-driven methods focused on filtering down to potentially malicious tools and generating comprehensive reports about the projects.

The first application of AI involved automating the assessment of GitHub projects to determine their potential use for malicious activities. This process began with the extraction of README files from each project repository. This file often contains crucial information about the project's purpose, functionality, and usage instructions. By analyzing this data, we aimed to identify projects that could be used for harmful purposes.

To achieve this, we employed large language models (LLMs) to evaluate the content of each README file. The LLMs were tasked with determining whether the described project had the potential for malicious use. The criteria for this assessment included the presence of keywords and phrases commonly associated with offensive security tools, hacking utilities, or any explicit mention of malicious intent.



```
inquiry = f'Summarize the content of README file from Github project below:\n\n{data}'
response = cc.completion(inquiry)
if not response:
    print('Skipping {}'.format(repo_id))
    continue
gpt_summary['readme_summary'] = response
del cc.messages[1]

inquiry = 'If this project is used by attackers reply "YES", else "NO". '
inquiry += 'You can only choose between YES and NO answer and you do not need to explain. '
inquiry += 'If it can be used by both attacker and defender, answer YES.'
response = cc.completion(inquiry)
```

Figure 12. Prompts used to summarize the README file and determine the project’s purpose

We manually reviewed projects flagged by the LLMs as potentially malicious confirm their intent. This automated pre-screening effectively weeded out tools meant for defensive purposes, allowing us to focus on those that warranted closer examination.

The second AI application we implemented involved the creation of detailed threat analyst reports for each project. These reports provided in-depth insights into the project's codebase, functionality, and potential security implications. The AI-generated reports included several key components:

1. **Project Summary:** An overview of the project's stated purpose and primary functions, extracted from the README file and other documentation.
2. **Code Analysis:** A detailed examination of the code, highlighting any suspicious or potentially harmful code segments. This analysis includes identifying the use of known vulnerabilities or exploitation techniques.
3. **Usage Context:** Information about how the project is typically used, derived from both the README file and usage examples within the repository.

4. **Community Activity:** An analysis of the project's activity on GitHub, including the frequency of updates, the number of contributors, and any reported issues related to security concerns.

These AI-generated reports provide a comprehensive understanding of each project, allowing threat analysts to make informed decisions about associated potential risks. By automating the initial analysis and report generation, we significantly reduced the number of projects requiring manual review, thus optimizing the threat assessment process.

By integrating AI into the research project, we successfully streamlined the evaluation of thousands of GitHub projects. The automated identification of potentially malicious tools and the generation of detailed threat analyst reports reduced the workload for human analysts and enhanced the accuracy and efficiency of the threat assessment process. These AI-driven methodologies ensured a focused and effective approach to identifying and mitigating potential security threats in the vast landscape of open-source projects. For example, after completing our README analysis, we reduced the number of projects flagged as malicious via AI from approximately 2,000 to just 292. This reduction was achieved by applying additional filters to minimize AI costs by analyzing the smallest possible number of projects.

# Unique Triage Methodology

To determine the most important projects for evaluation, we developed a unique methodology incorporating several key steps. This methodology includes using AI to assist in the process, assessing the project's popularity within the criminal underground, and considering its popularity on GitHub.

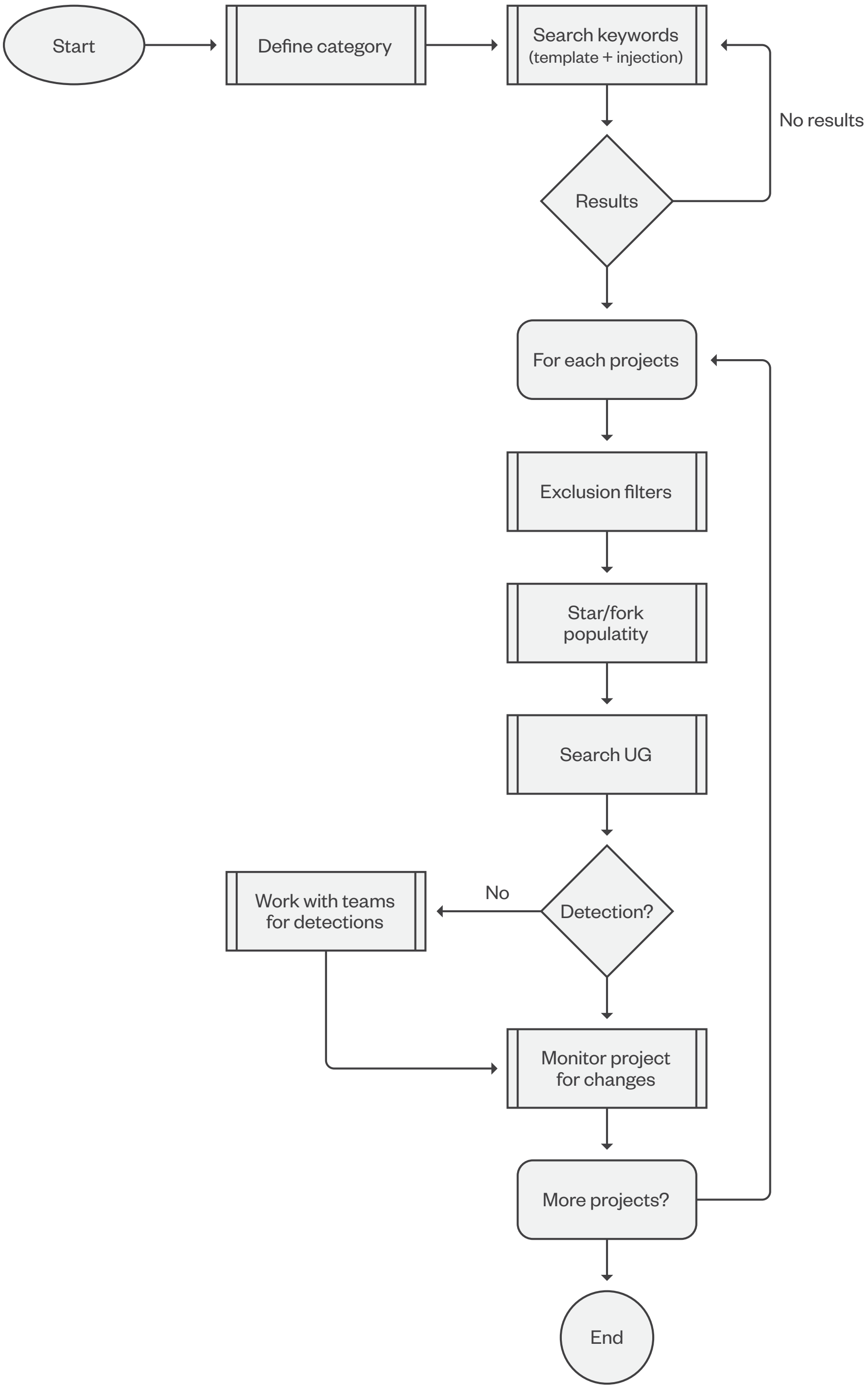


Figure 13. Flow chart for the triage process

The process begins with defining the category of interest, which sets the stage for subsequent steps. This involves clearly identifying the scope and criteria for the projects being targeted. The category could be based on technology, industry, type of solution, or any other relevant classification that helps narrow down the search.

| Search | Result |
|---|---|
| path:"/sites/paypal" OR path:"/sites/dropbox" OR path:"/sites/github" OR path:"/sites/twitter" OR path:"/sites/linkedin" OR path:"/sites/netflix" OR ……. | 464 |
| antibot.php OR antibot.ini OR block_bot.txt | 15 |
| (profile.html OR login.html) AND phishing | 9 |
| 'phising OR phish OR phishing kit OR phish kit' | 207,000 |
| "victim" AND "phish" | 94 |
| "usernames.txt" AND " $_POST" | 696 |
| "target" AND "phish" | 124 |
| "educational" AND "phish" | 143 |
| "'ip.php';" AND "login" | 159 |
| "$email = $_POST['email'];" AND "fwrite" | 489 |

Table 5. Sample phishing kit category searches and keywords

The next step involves searching for keywords using a predefined template combined with specific keyword injections. This method ensures a comprehensive and systematic approach to discovering relevant projects. The search aims to gather a broad set of results that fit within the defined category.

Upon obtaining the search results, the process branches depending on whether results are found. If no results are available, the process terminates. However, if results are found, the process moves forward to evaluate each project individually.

Exclusion filters are applied for each project identified within the search results. These filters help eliminate projects that do not meet certain criteria (such as relevance, activity level, or specific attributes that disqualify them from further consideration). This step ensures that only the most pertinent projects are included in the subsequent analysis.

Following the popularity assessment, comes the search for cybercriminal underground content. This step includes exploring criminal forums and other platforms where discussions and feedback about the projects might be found. This search aims to gather qualitative insights and real-world usage information that can provide valuable context to the view of a project from the perspective of malicious actors.

If significant findings or relevant user-generated content are detected, the two previously-mentioned AI processes work to narrow down the projects that require attention. The AI then investigates and analyzes these projects to ensure that findings are thoroughly examined and that any necessary actions are taken based on the insights gained.

If no immediate detections are made, the projects are monitored for any changes over time. This ongoing monitoring ensures that any new developments, updates, or shifts in the project status are promptly identified and addressed.

We tested this process using four categories: AV Evasion, Phishing, Post Exploitation Frameworks, and Credential Access. In total, based on the queries we put in place, we were able to pinpoint 1,949 projects that potentially needed further review. Our triage methodology allowed us to further narrow this amount down to 72 projects.

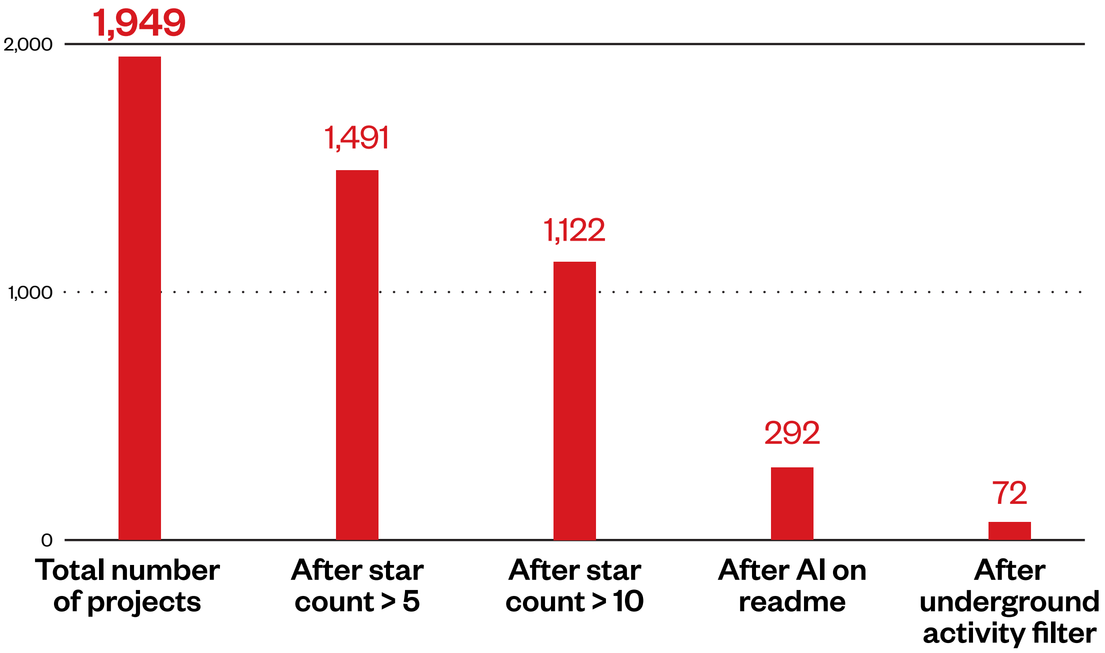


Figure 14. Reduction of projects based on filters

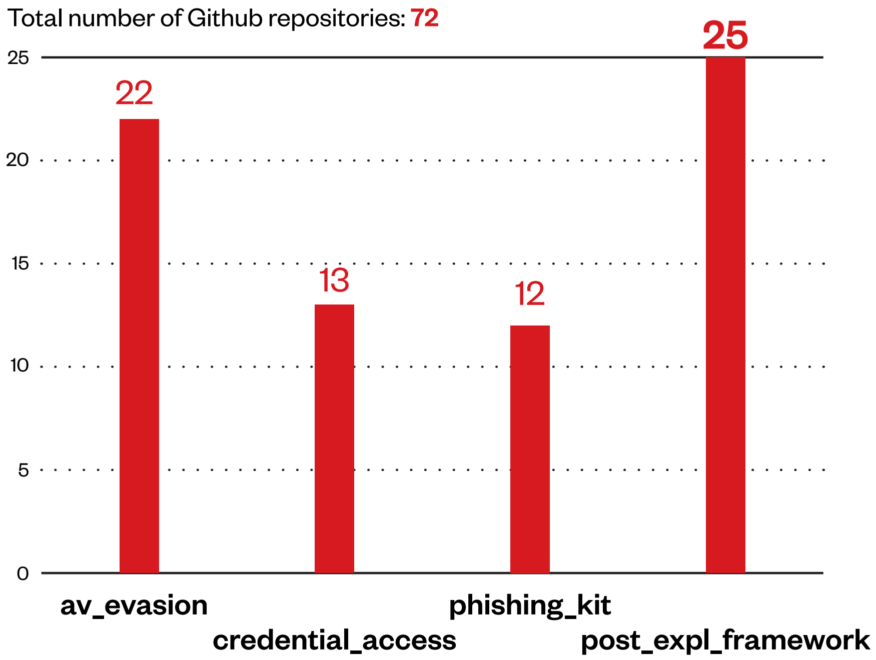


Figure 15. Total remaining projects by category

The next step involves assessing the popularity of each project by examining the number of stars and forks. These metrics provide a quantitative measure of how widely accepted and utilized a project is within the community. A higher star and fork count generally indicates a higher level of interest and engagement from users.

At this point, the process evaluates whether there are more projects to be processed. If there are additional projects, the process loops back to the step involving the individual evaluation of each project. If there are no more projects to consider, the process concludes.

This systematic approach ensures a thorough and continuous evaluation of projects within a defined category. It combines quantitative measures of popularity with qualitative insights from user-generated content. It also involves ongoing collaboration and monitoring to stay updated with any changes.

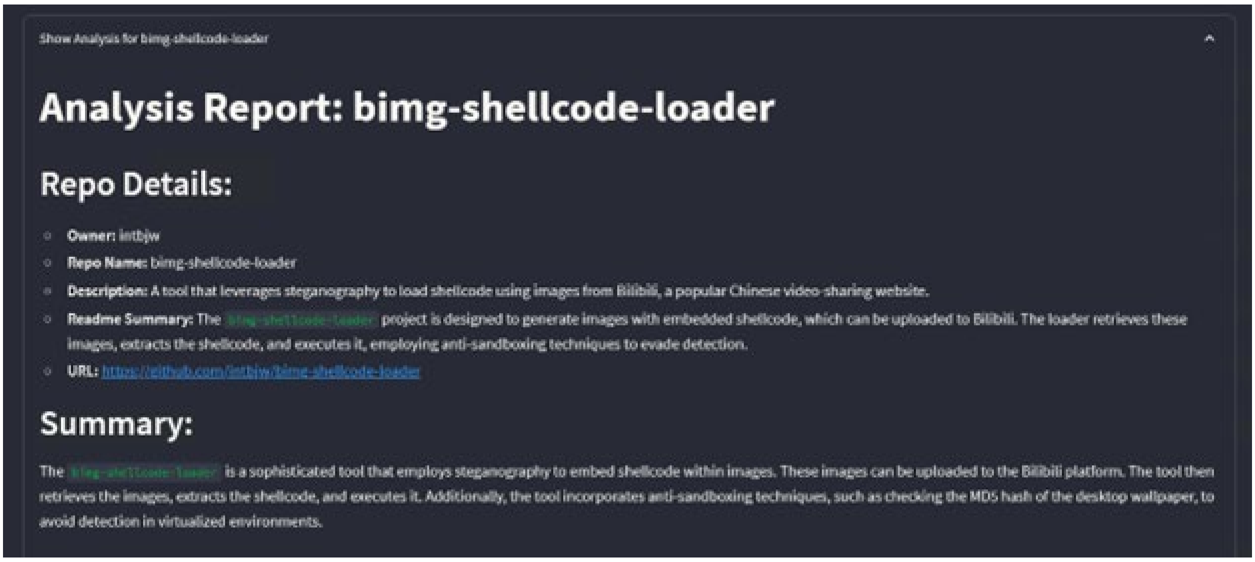
# Accelerating Analysis with AI Auto-Generated Code Analysis Reports

Due to the high number of red teaming repositories on GitHub, which also includes daily new releases and additions, analyzing these repositories to identify high-quality projects and trends has become very challenging. To address this issue, we took advantage of OpenAI's capabilities for code analysis and trend identification. By using advanced AI tools, we can efficiently sift through the vast array of repositories, ensuring that we stay updated with the best and most relevant projects in the field of red teaming. This approach not only saves time, but also enhances the accuracy of our analysis, providing us with valuable insights into the latest trends and developments within our chosen topic of interest.

To address this need, we developed an internal beta platform that monitors GitHub repositories. The platform filters out irrelevant repositories and retains only new and appropriate red teaming tools using the prompts we design on OpenAI. For prioritization, the system tracks project stars. If a repository experiences a significant spike in stars within a short timeframe, the system notifies the research team, adding the repository to the priority queue for analysis. The platform then generates an analysis report.

Researchers can also manually submit specific repositories to the system. Upon submission, the system downloads the repository, conducts an analysis, and produces a report. Depending on the configuration, the repository can be added to a watch list for future updates. If the repository receives a new commit, the system notifies the researcher of the latest changes. This feature is particularly useful when a malware updates its C&C communication, changes its encryption algorithm, or adds new methods of attack. Researchers can then update detection patterns or create new ones accordingly.

The following snippet is a malware analysis report generated by AI for the *bimg-shellcode-loader* repository:[72]



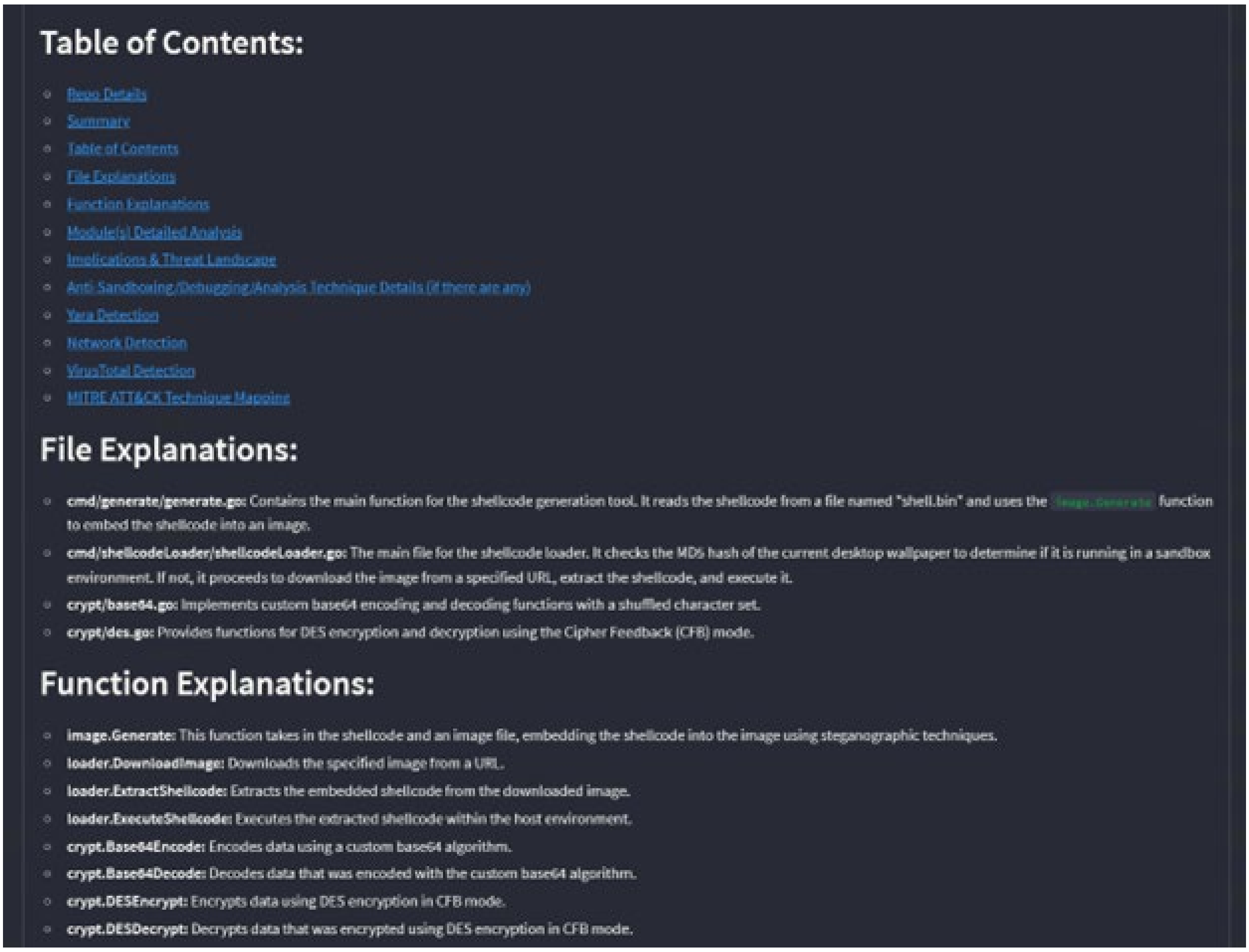
**Analysis Report: bimg-shellcode-loader**

**Repo Details:**

- **Owner:** intbjw
- **Repo Name:** bimg-shellcode-loader
- **Description:** A tool that leverages steganography to load shellcode using images from Bilibili, a popular Chinese video-sharing website.
- **Readme Summary:** The `bimg-shellcode-loader` project is designed to generate images with embedded shellcode, which can be uploaded to Bilibili. The loader retrieves these images, extracts the shellcode, and executes it, employing anti-sandboxing techniques to evade detection.
- **URL:** https://github.com/intbjw/bimg-shellcode-loader

**Summary:**

The `bimg-shellcode-loader` is a sophisticated tool that employs steganography to embed shellcode within images. These images can be uploaded to the Bilibili platform. The tool then retrieves the images, extracts the shellcode, and executes it. Additionally, the tool incorporates anti-sandboxing techniques, such as checking the MD5 hash of the desktop wallpaper, to avoid detection in virtualized environments.



## Table of Contents:

- Repo Details
- Summary
- Table of Contents
- File Explanations
- Function Explanations
- Module(s) Detailed Analysis
- Implications & Threat Landscape
- Anti-Sandboxing/Debugging/Analysis Technique Details (if there are any)
- Yara Detection
- Network Detection
- VirusTotal Detection
- MITRE ATT&CK Technique Mapping

## File Explanations:

- **cmd/generate/generate.go:** Contains the main function for the shellcode generation tool. It reads the shellcode from a file named "shell.bin" and uses the `image.Generate` function to embed the shellcode into an image.
- **cmd/shellcodeLoader/shellcodeLoader.go:** The main file for the shellcode loader. It checks the MD5 hash of the current desktop wallpaper to determine if it is running in a sandbox environment. If not, it proceeds to download the image from a specified URL, extract the shellcode, and execute it.
- **crypt/base64.go:** Implements custom base64 encoding and decoding functions with a shuffled character set.
- **crypt/des.go:** Provides functions for DES encryption and decryption using the Cipher Feedback (CFB) mode.

## Function Explanations:

- **image.Generate:** This function takes in the shellcode and an image file, embedding the shellcode into the image using steganographic techniques.
- **loader.DownloadImage:** Downloads the specified image from a URL.
- **loader.ExtractShellcode:** Extracts the embedded shellcode from the downloaded image.
- **loader.ExecuteShellcode:** Executes the extracted shellcode within the host environment.
- **crypt.Base64Encode:** Encodes data using a custom base64 algorithm.
- **crypt.Base64Decode:** Decodes data that was encoded with the custom base64 algorithm.
- **crypt.DESEncrypt:** Encrypts data using DES encryption in CFB mode.
- **crypt.DESDecrypt:** Decrypts data that was encrypted using DES encryption in CFB mode.

Figure 16. A malware analysis report for the "bimg-shellcode-loader" repository

The malware analysis report contains the following sections:

- Repository details
  - Owner
  - Repository name
  - Description
  - README summary
  - URL
- Summary
- Table of contents
- File explanations
- Function explanations
- Detailed module(s) analysis
- Implications and threat landscape

- Anti-sandboxing/debugging/analysis technique details (if any)
- Yara detection
- Network analysis
- VirusTotal detection
- MITRE ATT&CK technique mapping

The following is an example of an OpenAI prompt to generate analysis report – note that some sections, such as Yara Detection and VirusTotal Detection, are processed by other scripts and automation systems, and the JSON.XML result will be processed here:

You are an AI specialized in malware analysis. Generate a comprehensive malware analysis report for a tool provided by the user from a GitHub repository. The report should include the following sections:

```
**Repo Details:**

–    **Owner:** Retrieve the owner of the repository.

–    **Repo Name:** Retrieve the name of the repository.

–    **Description:** Provide a brief description of the repository.

–    **Readme Summary:** Summarize the key points from the repository's README file.

–     **URL:** Provide the URL of the repository.
```

```
**Summary:**

–    Provide a brief summary of what the tool in the repository does. Mention its purpose, how it
     operates, and any special techniques it uses, such as steganography, anti-sandboxing, or other
     evasion techniques.
```

```
**Table of Contents:**

–    List all the sections of the report in a structured manner.
```

```
**File Explanations:**

–    Describe the purpose and functionality of key files in the repository. Include details for the main
     files involved in the tool's operation.
```

**Function Explanations:**

- Provide detailed explanations of important functions within the key files. Explain their roles and how they contribute to the overall functionality of the tool.

**Module(s) Detailed Analysis:**

- Conduct an in-depth analysis of the key modules in the tool. Explain their purpose, how they interact with each other, and their role in the overall functionality of the tool.

**Implications & Threat Landscape:**

- Discuss the broader implications of using such tools. Explain the threat landscape, potential targets, and how this tool could be misused.

**Anti-Sandboxing/Debugging/Analysis Technique Details (if there are any):**

- Explain any techniques used by the tool to avoid sandboxing, debugging, or analysis.
- Describe how these techniques help the tool evade detection or analysis.
- Include technical details and examples of how these techniques are implemented within the code.
- Discuss the effectiveness of these techniques and any potential countermeasures that could be employed to detect or mitigate them.

**Yara Detection:**

- <Results from other scripts are processed in this section>

> **Network Analysis:**
>
> - Describe any network communications it initiates or participates in, such as contacting command and control (C2) servers, downloading additional payloads, or exfiltrating data.
> - Include details on the protocols and ports used, the nature of the data transmitted, and any indications of malicious network activity.
> - Provide insights into how this network activity can be detected and mitigated, including potential indicators of compromise (IOCs) and recommended monitoring strategies

> **VirusTotal Detection:**
>
> - <Results from other scripts are processed in this section>

> **MITRE ATT&CK Technique Mapping:**
>
> - Map the techniques used by the tool to the MITRE ATT&CK framework. Provide details on which techniques are used and how they align with the framework's tactics and techniques.
>
> Generate the report in markdown format, ensuring each section is clearly marked with appropriate headings and subheadings.
>
> The user will provide the GitHub repository URL for analysis.

By structuring the analysis report in this manner, we gain a comprehensive overview of the malware, detailing its functions and potential impact. This approach enhances our understanding of the project and facilitates a more effective response to the threat it poses.

The following snapshots illustrate the trend for the DarkPulse shellcode loader/obfuscator repository.[73] When configured to detect a threshold of over 20 stars per day, the system marks it as a trending repository and notifies the researcher accordingly.

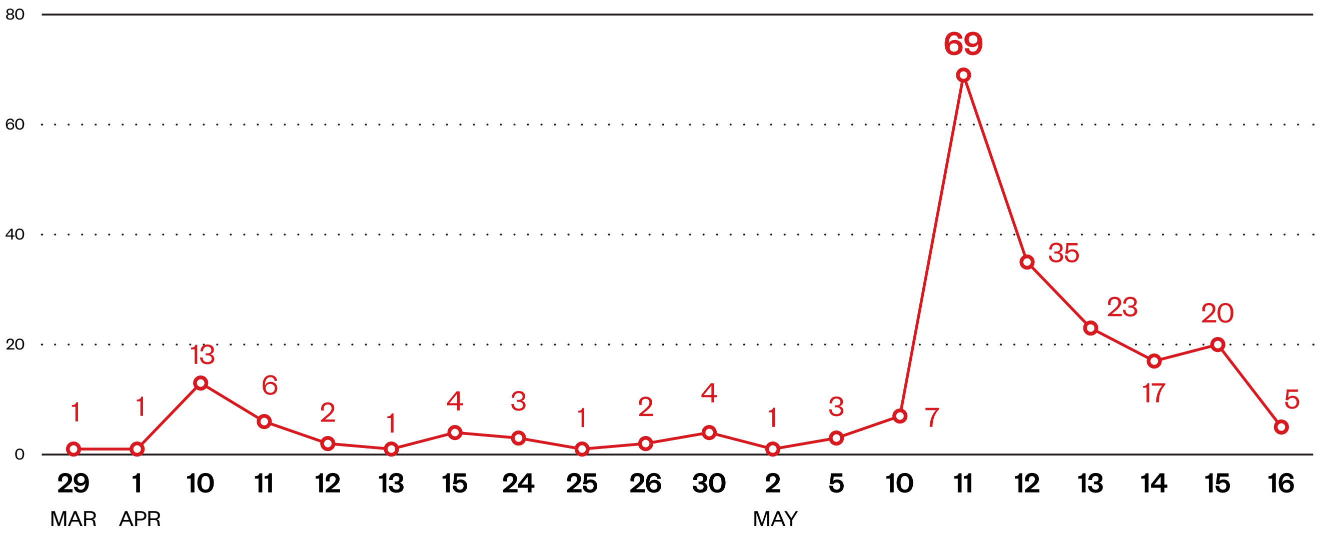


Figure 17. Trend for the DarkPulse shellcode loader/obfuscator repository based on stars earned (daily threshold of 20)

We can change the data aggregation to a weekly limit of 100 stars. Once the GitHub repository goes over 100 stars and hits the limit, the system will create a notification and add the repository to the analysis queue.

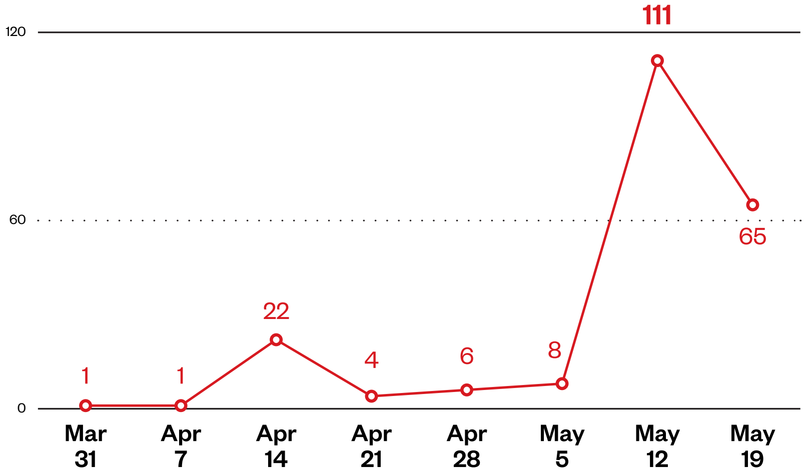


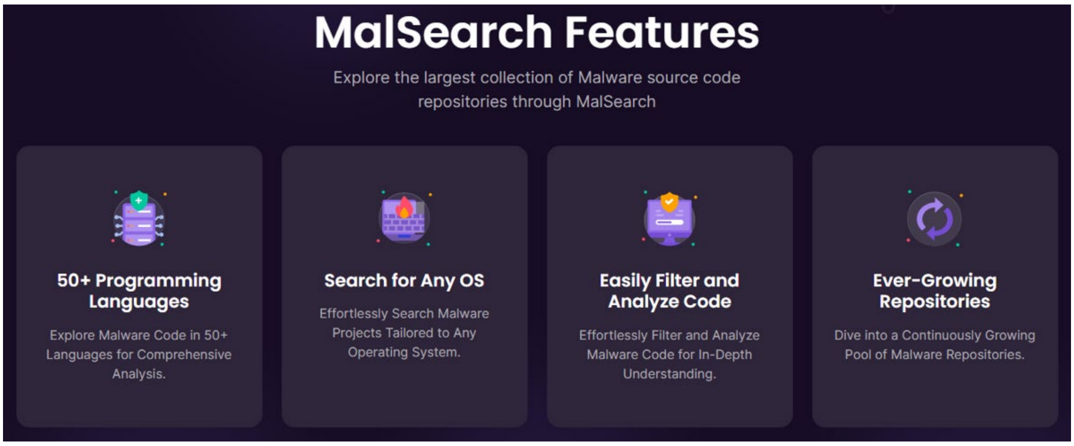
Figure 18. Trend for the DarkPulse shellcode loader/obfuscator repository based on stars earned (Weekly threshold of 100)

Once the repository has been thoroughly analyzed, the researcher changes its status to remove it from the analysis queue. Depending on the outcome, it will then be placed in either the monitoring queue for ongoing future commits or the done queue, indicating that the analysis is complete.

# The Future of Open Source Red Teaming

Due to the increasing frequency of leaks and the widespread sharing of red teaming code (along with the large number of related techniques and code snippets), managing and categorizing these resources has become an essential task. The need to effectively label and make this information searchable has grown significantly.

In response to this demand, specialized paid services like Malsearch,[74] that can organize and streamline the management of such data, have emerged to address these issues. These services play a crucial role in helping professionals efficiently navigate and utilize the vast amount of available information in the field of cybersecurity.



MalSearch Features

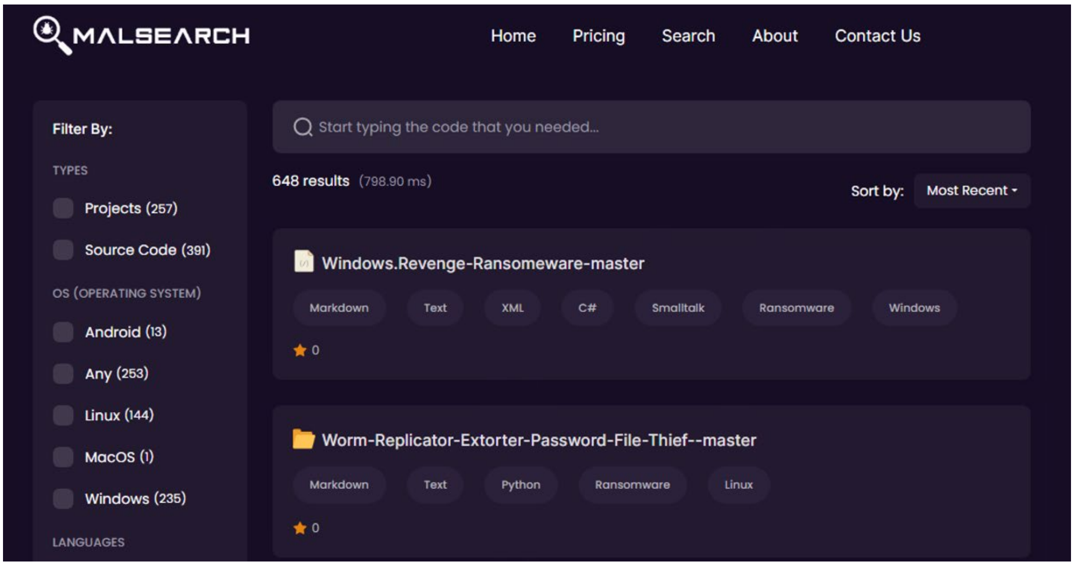
Explore the largest collection of Malware source code repositories through MalSearch

**50+ Programming Languages**
Explore Malware Code in 50+ Languages for Comprehensive Analysis.

**Search for Any OS**
Effortlessly Search Malware Projects Tailored to Any Operating System.

**Easily Filter and Analyze Code**
Effortlessly Filter and Analyze Malware Code for In-Depth Understanding.

**Ever-Growing Repositories**
Dive into a Continuously Growing Pool of Malware Repositories.



Filter By:

TYPES
- □ Projects (257)
- □ Source Code (391)

OS (OPERATING SYSTEM)
- □ Android (13)
- □ Any (253)
- □ Linux (144)
- □ MacOS (1)
- □ Windows (235)

LANGUAGES

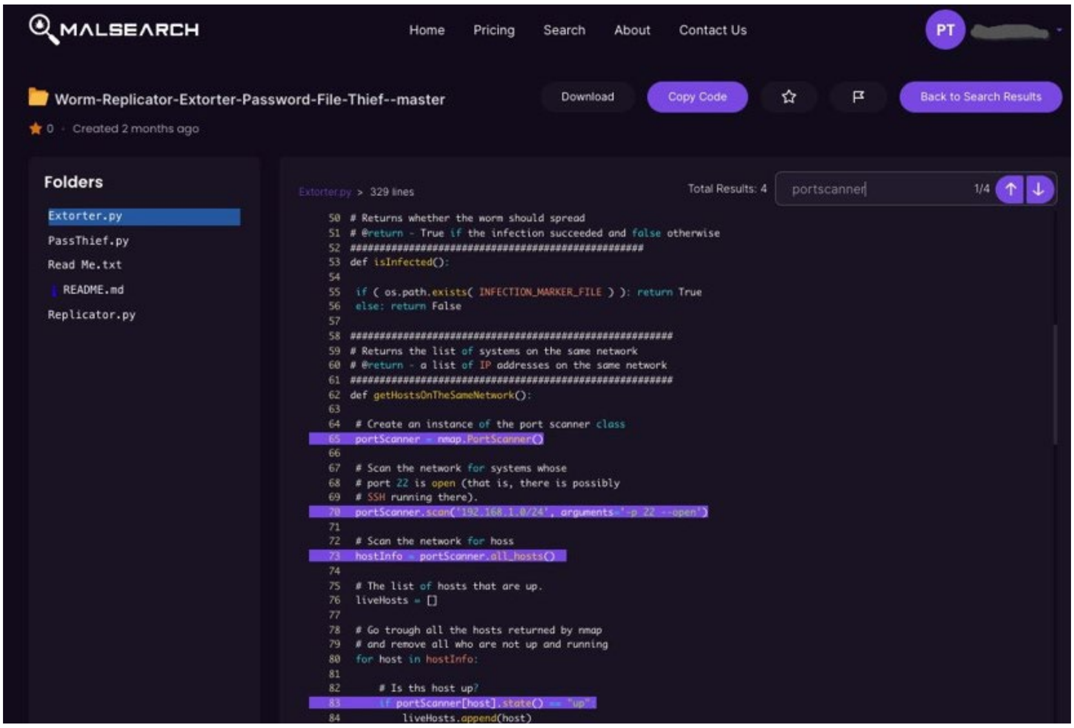
Start typing the code that you needed...

648 results (798.90 ms)

Sort by: Most Recent

Windows.Revenge-Ransomeware-master
Markdown | Text | XML | C# | Smalltalk | Ransomware | Windows
★ 0

Worm-Replicator-Extorter-Password-File-Thief--master
Markdown | Text | Python | Ransomware | Linux
★ 0



```
# Returns whether the worm should spread
# @return - True if the infection succeeded and false otherwise
####################################################
def isInfected():

 if ( os.path.exists( INFECTION_MARKER_FILE ) ): return True
 else: return False

#########################################################
# Returns the list of systems on the same network
# @return - a list of IP addresses on the same network
#########################################################
def getHostsOnTheSameNetwork():

 # Create an instance of the port scanner class
 portScanner = nmap.PortScanner()

 # Scan the network for systems whose
 # port 22 is open (that is, there is possibly
 # SSH running there).
 portScanner.scan('192.168.1.0/24', arguments='-p 22 --open')

 # Scan the network for hoss
 hostInfo = portScanner.all_hosts()

 # The list of hosts that are up.
 liveHosts = []

 # Go trough all the hosts returned by nmap
 # and remove all who are not up and running
 for host in hostInfo:

     # Is ths host up?
     if portScanner[host].state() == "up":
         liveHosts.append(host)
```

Figure 19. The Malsearch tool

Malsearch is a search engine specifically designed for red teaming code, offering a well-organized and easily accessible repository. These platforms categorize and manage red teaming and penetration testing code, providing security professionals with a centralized resource, allowing them to quickly find and use the tools they need.

Based on this, we can see a shift from open source to a service and paid subscription-based model.  With this new approach, organizers can curate tools, enhance them, and add their own methods to create a more exclusive community.

Trend Micro researchers David Sancho and Vincenzo Ciancaglini published a blog entry on the misuse of generative AI by malicious individuals.[75] In the article, they examine new criminal LLMs, criminal services with ChatGPT-like capabilities, and deepfakes being offered on criminal sites.

These knowledge bases and code repositories can be used to train private AI models, leading to the creation of new services designed to generate red teaming tools and enabling the creation of sophisticated and evasive malware in the future. This evolution is already underway and is expected to lead to the creation of increasingly advanced services.

The following is a list of the services currently available:

| Name | Description |
|---|---|
| MalSearch[76] | MalSearch is a dedicated platform that provides advanced search capabilities for malware analysts and researchers, facilitating easy access to a vast database of malware samples and related information. |
| Maldev Academy[77] | Maldev Academy is a comprehensive malware development course that focuses on x64 malware development, providing knowledge from basic to advanced level.<br>Maldev Academy Code Search  is a paid service with over 7000+ lines, 300+ snippets in total with new snippets being added every month. |
| Vx Underground[78] | Platform and community dedicated to the collection, analysis, and dissemination of malware samples and threat intelligence for educational and research purposes |
| Unprotect Project[79] | A comprehensive resource that facilitates the search and exploration of various evasion techniques, code snippets, and detection rules related to cybersecurity. |
| theZoo[80] | TheZoo is a GitHub project that offers a publicly accessible repository of live malware samples, intended for educational and research purposes in the field of malware analysis. |

Table 7. A list of currently available services that can be used by both red and blue teams

# Challenges and Limitations

## Technical Challenges

One of the most significant challenges we encountered at the outset of this research was the overwhelming number of non-relevant repositories that appeared when searching for generic keywords like "phishing" and "loader." The results of these searches were excessively high and inundated with irrelevant code, which significantly slowed down our process. To address this issue, we developed an OpenAI prompt along with several pattern-matching methods designed to filter out non-red teaming tools from our dataset.

This approach not only streamlined our research process but also enhanced the accuracy and relevance of the data we were analyzing. To speed up the process, we first passed the GitHub project description and *README.md* file to the AI for processing. For those projects that don't have a description or *README.md* file, we handled them as follows:

1. Get the project programming language
2. Extract the files (excluding non-relevant ones)
3. Pass these files to the AI for processing.

The challenge lies in the cost of query and processing for large projects. Due to their complexity and numerous files, the AI might struggle to accurately determine if the project is a red team tool and to generate precise results for project analysis. However, for small to medium-sized repositories (with around 1-10 files), the AI system can produce relatively acceptable results. The accuracy of the query and processing depends on the category being analyzed. For instance, information stealers tend to yield more accurate results compared to the phishing category.

## Dual Hooking

The ethical concerns surrounding red team tools are further complicated by the risk of dual hooking by threat actors. Dual hooking occurs when malicious actors alter the code of red team tools to surreptitiously steal the information collected during a penetration testing engagement.

This manipulation can lead to severe privacy breaches, as sensitive data intended for security assessment can be exfiltrated by unauthorized entities. Such actions not only compromise the integrity of the red team exercise but also pose significant risks to the organization's data security. Detecting these covert modifications requires advanced threat detection capabilities and continuous monitoring to ensure that the tools used for cybersecurity do not become attack vectors for further exploitation. Addressing dual hooking necessitates stringent code review processes, enhanced security measures for tool deployment, and ongoing vigilance to safeguard against the manipulation of cybersecurity resources by malicious actors.

# Future Directions for This Methodology

## Potential improvements in AI models

Potential improvements in AI models can play a crucial role in combating the use of red team tools by nation states and cybercriminals. Advanced AI-driven cybersecurity systems can be designed to identify and neutralize threats with greater speed and accuracy. By enhancing machine learning algorithms, these systems can better recognize patterns associated with red team tools and sophisticated cyberattacks.

For example, AI models can be trained on vast datasets that include various attack vectors and behaviors exhibited by malicious actors. This allows the models to predict and detect new and previously unseen TTPs. Furthermore, integrating natural language processing (NLP) capabilities enables AI systems to analyze threat intelligence reports and communications in real-time, providing insights that can preemptively counteract potential cyber threats.

## Expanding the triage criteria

Expanding the triage criteria is another vital strategy in mitigating the impact of red team tools used by malicious actors. Traditional triage processes often focus on immediate and high-severity threats, potentially overlooking subtle or emerging risks. By expanding the triage criteria, organizations can incorporate a more nuanced assessment of threats, which includes considering the context, potential impact, and historical patterns of similar attacks.

This involves integrating behavioral analytics and anomaly detection into the triage workflow, allowing security teams to prioritize not just based on severity but also on the likelihood of exploitation and the strategic importance of targeted assets. Enhanced triage criteria should also include indicators of compromise (IOCs) and TTPs specific to red team tools, ensuring a more comprehensive and proactive defense posture.

## Collaborations and community involvement

Collaborations and community involvement are critical components in stymying the sophisticated use of red team tools by nation states and cybercriminals. Cybersecurity is a collective effort that benefits immensely from shared knowledge and resources. Establishing partnerships between private sector companies, government agencies, and international organizations can lead to the development of unified threat intelligence platforms. These platforms can facilitate real-time sharing of information on emerging threats and vulnerabilities, as well as best practices for individuals and organizations to follow and implement.

Moreover, community-driven initiatives, such as open-source threat databases and collaborative cybersecurity research projects, empower organizations of all sizes to enhance their defensive capabilities. Encouraging active participation in cybersecurity forums, workshops, and training programs helps build a resilient and informed community capable of effectively responding to evolving cyberthreats. Through these collaborative efforts, the collective cyberdefense mechanisms becomes stronger, more adaptive, and better equipped to counteract the misuse of red team tools.

# Conclusion

The integration of red teaming methodologies into cybersecurity practices has proven essential in enhancing the robustness and resilience of organizational defenses amid an evolving threat landscape. By simulating adversarial attacks, red teams can identify vulnerabilities and provide critical insights that help organizations fortify their security measures.

However, the dual-use nature of red team tools, while beneficial for security testing, also presents significant risks as malicious actors can repurpose these tools for cyberattacks. This underscores the importance of balancing the development and use of these tools with ethical guidelines and robust detection capabilities.

The exploitation of open-source software in the supply chain highlights the complexity and challenges in maintaining secure software ecosystems. Cybercriminals and nation-state actors can capitalize on the openness of open-source software to introduce vulnerabilities and carry out widespread attacks, making it imperative for organizations to implement stringent security measures and continuously monitor for emerging threats. The integration of AI and machine learning in identifying and triaging potential threats from open-source repositories offers a promising direction for enhancing detection and response capabilities.

This process significantly enhances scalability, enabling the efficient triage of numerous projects simultaneously. By leveraging the two AI processes we discussed in this research, we can swiftly filter out less important projects and focus on those that require immediate attention. This automation dramatically reduces the time and effort needed for analysis.

In our testing, tasks that would take a human a week to accomplish were done in about five minutes. This efficiency is due to the ability of AI to quickly detect significant findings and relevant user-generated content, combined with intelligent filters that prioritize projects based on importance. As a result, the process not only saves time but also ensures that critical projects receive prompt and thorough evaluation.

Our research emphasizes the need for a holistic approach that combines traditional red teaming tactics with advanced detection and response strategies. By fostering collaboration between red and blue teams and integrating AI-driven analysis with manual triage processes, organizations can better prepare for and mitigate the impact of cyberthreats. Future directions should focus on refining AI models, expanding triage criteria, and involving the cybersecurity community in collaborative efforts to enhance overall security.

Ultimately, the continuous evolution of red teaming methodologies, coupled with proactive detection and ethical considerations, will be crucial in staying ahead of malicious actors, and protecting critical infrastructure and sensitive information from sophisticated attacks. Moving from a reactive approach, which involves handling tools as they gain popularity with cybercriminals to a proactive approach where defenders are constantly monitoring for upcoming and popular tools with malicious actors will lead to a stronger defensive posture for organizations around the world.

# Endnotes

1 Stephen Hilt and Lord Alfred Remorin. (Feb. 2, 2023). *Trend Micro*. “What SOCs Need to Know About Water Dybbuk, A BEC Actor Using Open-Source Toolkits.” Accessed on Sep. 26, 2024, at: Link.

2 Hara Hiroaki and Ted Lee. (Aug. 24, 2023). *Trend Micro*. “APT41 Resurfaces as Earth Baku With New Cyberespionage Campaign.” Accessed on Sep. 26, 2024, at: Link.

3 Office of the CISO. (April, 2023). *Google Cloud*. “Threat Horizons: April 2023 Threat Horizons Report.” Accessed on Sep. 26, 2024, at: Link.

4 EmpireProject. (n.d.). *GitHub*. “EmpireProject.” Accessed on Sep. 26, 2024, at: Link.

5 Cybersecurity and Infrastructure Security Agency. (June 30, 2023). *CISA*. “Publicly Available Tools Seen in Cyber Incidents Worldwide.” Accessed on Sep. 26, 2024, at: Link.

6 SweetSoftware. (n.d.). *GitHub*. "Ares." "Accessed on Sep. 26, 2024, at: Link.

7 Global Research and Analysis Team. (April 14, 2020). *Securelist*. "APT Trends Report Q1 2020". Accessed on Sep. 26, 2024, at: Link.

8 Sathwik Ram Prakki (Dec. 01, 2023). *SEQRITE*. "Operation RusticWeb targets Indian Govt: From Rust-based malware to web service exfiltration". Accessed on Sep. 26, 2024, at: Link.

9 Loo Ciprian. (n.d.). *GitHub*. "GC2-sheet." Accessed on Sep. 26, 2024, at: Link.

10 Lawrence Abrams. (April 17, 2023). *Bleeping Computer*. "Hackers Abuse Google Command and Control Red Team Tool in Attacks." Accessed on Sep. 26, 2024, at: Link.

11 Bishop Fox. (n.d.). *GitHub*. "Sliver". Accessed on Sep. 26, 2024, at: Link.

12 Bryan Campbell, Selena Larson, and the Proofpoint Threat Insight Team. (Oct. 20, 2021). *Proofpoint*. "TA551 Uses Sliver Red Team Tool in New Activity." Accessed on Sep. 26, 2024, at: Link.

13 Cybereason Global SOC and Incident Response Team. (n.d.). *Cybereason*. "Sliver C2 Leveraged by Many Threat Actors." Accessed on Sep. 26, 2024, at: Link.

14 Fortra. (n.d.). *GitHub*. "Impacket." Accessed on Sep. 26, 2024, at: Link.

15 Cybersecurity and Infrastructure Security Agency. (Oct. 5, 2022). *CISA*. "Impacket and Exfiltration Tool Used to Steal Sensitive Information from Defense Industrial Base Organization.” Accessed on Sep. 26, 2024, at: Link.

16 Rapid7. (n.d.). *GitHub*. "Metasploit Framework." Accessed on Sep. 26, 2024, at: Link.

17 r00t-3xp10it. (n.d.). *GitHub*. "Meterpeter." Accessed on Sep. 26, 2024, at: Link.

18 Internet Crime Complaint Center. (Sept. 16, 2020). *Internet Crime Complaint Center*. "2020 IC3 Internet Crime Complaint Center Report." Accessed on Sep. 26, 2024, at: Link.

19 Cybersecurity and Infrastructure Security Agency. (November 15, 2018). *CISA*. "Publicly Available Tools Seen in Cyber Incidents Worldwide." Accessed on Sep. 26, 2024, at: Link.

20 Microsoft (March 2, 2021). *Microsoft*. "HAFNIUM targeting Exchange Servers with 0-day exploits." Accessed on Sep. 26, 2024, at: Link.

21 Group-IB (Jan. 11, 2023). *Group-IB*. "Dark Pink." Accessed on Sep. 26, 2024, at: Link.

22 Eduard Kovacs. (Feb. 16, 2017). *Security Week*. "Iranian Spies Target Saudi Arabia with Magic Hound Attacks." Accessed on Sep. 26, 2024, at: Link.

23 byt3bl33d3r (n.d.). *GitHub*. "SILENTTRINITY." Accessed on Sep. 26, 2024, at: Link.

24 PT Expert Security Center. (Aug. 19, 2024). *Positive Technologies*. "IronPython, darkly: how we uncovered an attack on government entities in Europe." Accessed on Sep. 26, 2024, at: Link.

25 restran (n.d.). *GitHub*. "BlueShell." Accessed on Sep. 26, 2024, at: Link.

26 AhnLab (Sept. 5, 2023). *AhnLab*. "BlueShell Used in APT Attacks Against Korean and Thai Targets." Accessed on Sep. 26, 2024, at: Link.

27 chouaibhm (n.d.). *GitHub*. "DNS-Persist." Accessed on Sep. 26, 2024, at: Link.

28 Salim Bitam. (March 22, 2023). *Elastic Security Labs*. "Not Sleeping Anymore: SomniRecords' Wakeup Call". Accessed on Sep. 26, 2024, at: Link.

29 sysdream (n.d.). *GitHub*. "Ligolo." Accessed on Sep. 26, 2024, at: Link.

30 Trellix (July 29, 2022). *Trellix*. " Trellix Insights: Multiple campaigns attributed to MuddyWater APT group ." Accessed on Sep. 26, 2024, at: Link.

31 samratashok (n.d.). *GitHub*. "Antak WebShell." Accessed on Sep. 26, 2024, at: Link.

32 Malpedia (n.d.). *Malpedia*. "Antak." Accessed on Sep. 26, 2024, at: Link.

33 Florian Roth. (Nov. 23, 2019). *Medium*. "The Problems with Today's Red Teaming." Accessed on Sep. 27, 2024, at: Link.

34 C2 Matrix. (n.d.). *The C2 Matrix*. "The C2 Matrix." Accessed on Sep. 27, 2024, at: Link.

35 Aliakbar Zahravi and Peter Girnus. (Feb. 9, 2023). *Trend Micro*. "Enigma Stealer Targets Cryptocurrency Industry with Fake Jobs." Accessed on Sep. 27, 2024, at: Link.

36 TheCruZ. (n.d.). *GitHub*. "kdmapper." Accessed on Sep. 27, 2024, at: Link.

37 pwn1sher. (n.d.). *GitHub*. "KillDefender." Accessed on Sep. 27, 2024, at: Link.

38 Gabriel Landau. (Feb. 21, 2023). *Elastic Security Labs*. "Sandboxing Antimalware Products." Accessed on Sep. 27, 2024, at: Link.

39 med0x2e. (n.d.). *GitHub*. "ExecuteAssembly." Accessed on Sep. 27, 2024, at: Link.

40 aaaddress1. (n.d.). *GitHub*. "tyranid_appInfo_alpc.cpp." Accessed on Sep. 27, 2024, at: Link.

41 hasherezade. (n.d.). *GitHub*. "process_doppelganging." Accessed on Sep. 27, 2024, at: Link.

42 Tal Liberman and Eugen Kogan. (Dec. 2017). *Black Hat*. "Lost in Transaction: Process Doppelgänging." Accessed on Sep. 27, 2024, at: Link.

43 Anton Ivanov, Fedor Sinitsyn, and Orkhan Mamedov. (May 7, 2018). *Securelist*. "SynAck Targeted Ransomware Uses the Doppelgänging Technique." Accessed on Sep. 27, 2024, at: Link.

44 Symantec Threat Hunter Team. (July 25, 2018). *Symantec*. "Leafminer: Espionage Campaigns Targeting Middle Eastern Regions." Accessed on Sep. 27, 2024, at: Link.

45 European Union Agency for Cybersecurity. (June 21, 2023). *ENISA*. "Good Practices for Supply Chain Cybersecurity." Accessed on Sep. 27, 2024, at: Link.

46 Aliakbar Zahravi and Peter Girnus. (Oct. 5, 2023). *Trend Micro*. "Exposing Infection Techniques Across Supply Chains and Codebases." Accessed on Sep. 27, 2024, at: Link.

47 Henrik Plate. (Feb. 28, 2024). *Endor Labs*. "Detect Malicious Packages Among Your Open Source Dependencies." Accessed on Sep. 27, 2024, at: Link.

48 Microsoft. (n.d.). *Microsoft*. "Open Source Software (OSS) Supply Chain Threats." Accessed on Sep. 27, 2024, at: Link.

49 Im4wasp. (n.d.). *GitHub*. "W4SP-Stealer-Sourcecode." Accessed on Sep. 27, 2024, at: Link.

50 Im4wasp. (n.d.). *GitHub*. "W4SP-Stealer-V2." Accessed on Sep. 27, 2024, at: Link.

51 Ayhuuu. (n.d.). *GitHub*. "Creal-Stealer." Accessed on Sep. 27, 2024, at: Link.

52 Felworl22. (n.d.). *GitHub*. "Creal-Stealer." Accessed on Sep. 27, 2024, at: Link.

53 Inplex-sys. (n.d.). *GitHub*. "BlackCap-Grabber-NoDualHook." Accessed on Sep. 27, 2024, at: Link.

54 Smug246. (n.d.). *GitHub*. "Luna-Grabber." Accessed on Sep. 27, 2024, at: Link.

55 addi00000. (n.d.). *GitHub*. "empyrean." Accessed on Sep. 27, 2024, at: Link.

56 ChildrenOfYahweh. (n.d.). *GitHub*. "Kematian-Stealer." Accessed on Sep. 27, 2024, at: Link.

57 saintdaddy. (n.d.). *GitHub*. "Vare-Stealer." Accessed on Sep. 27, 2024, at: Link.

58 Blank-c. (n.d.). *GitHub*. "Blank-Grabber." Accessed on Sep. 27, 2024, at: Link.

59 cryst4linqq. (n.d.). *GitHub*. "TurkoRat." Accessed on Sep. 27, 2024, at: Link.

60 Blank-c. (n.d.). *GitHub*. "BlankOBF." Accessed on Sep. 27, 2024, at: Link.

61 javascript-obfuscator. (n.d.). *GitHub*. "javascript-obfuscator." Accessed on Sep. 27, 2024, at: Link.

62 saintdaddy. (n.d.). *GitHub*. "Vare-Obfuscator." Accessed on Sep. 27, 2024, at: Link.

63 saintdaddy. (n.d.). *GitHub*. "Vare-Obfuscator2.0." Accessed on Sep. 27, 2024, at: Link.

64 Hnfull. (n.d.). *GitHub*. "Intensio-Obfuscator." Accessed on Sep. 27, 2024, at: Link.

65 billythegoat356. (n.d.). *GitHub*. "Hyperion." Accessed on Sep. 27, 2024, at: Link.

66 billythegoat356. (n.d.). *GitHub*. "Kramer." Accessed on Sep. 27, 2024, at: Link.

67 anseki. (n.d.). *GitHub*. "gnirts." Accessed on Sep. 27, 2024, at: Link.

68 zswang. (n.d.). *GitHub*. "jfogs." Accessed on Sep. 27, 2024, at: Link.

69 nbsp1221. (n.d.). *GitHub*. "javascript-obfuscator." Accessed on Sep. 27, 2024, at: Link.

70 Avigayil Mechtinger, Oren Ofer, and Itamar Gilad. (July 11, 2023). *Wiz*. "PyLoose: Python-based fileless malware targets cloud workloads to deliver cryptominer." Accessed on Sep. 27, 2024, at: Link.

71 JayDDee. (n.d.). *GitHub*. "cpuminer-opt." Accessed on Sep. 27, 2024, at: Link.

72 intbjw. (n.d.). *GitHub*. "bimg-shellcode-loader." Accessed on Sep. 27, 2024, at: Link.

73 fdx-xdf. (n.d.). *GitHub*. "darkPulse." Accessed on Sep. 27, 2024, at: Link.

74 Malsearch. (n.d.). *Malsearch*. "Malsearch." Accessed on Sep. 27, 2024, at: Link.

75 Vincenzo Ciancaglini and David Sancho. (May 8, 2023). *Trend Micro*. "Back to the Hype: An Update on How Cybercriminals Are Using GenAI." Accessed on Sep. 27, 2024, at: Link.

76 Malsearch Team. (n.d.). *Malsearch*. "Malsearch." Accessed on Sep. 27, 2024, at: Link.

77 Maldev Academy. (n.d.). *Maldev Academy*. "Search Maldev Academy." Accessed on Sep. 27, 2024, at: Link.

78 VX-Underground. (n.d.). *VX-Underground*. "VX-Underground." Accessed on Sep. 27, 2024, at: Link.

79 Unprotect Project. (n.d.). *Unprotect Project*. "Unprotect Project." Accessed on Sep. 27, 2024, at: Link.

80 ytisf. (n.d.). *GitHub*. "theZoo." Accessed on Sep. 27, 2024, at: Link.