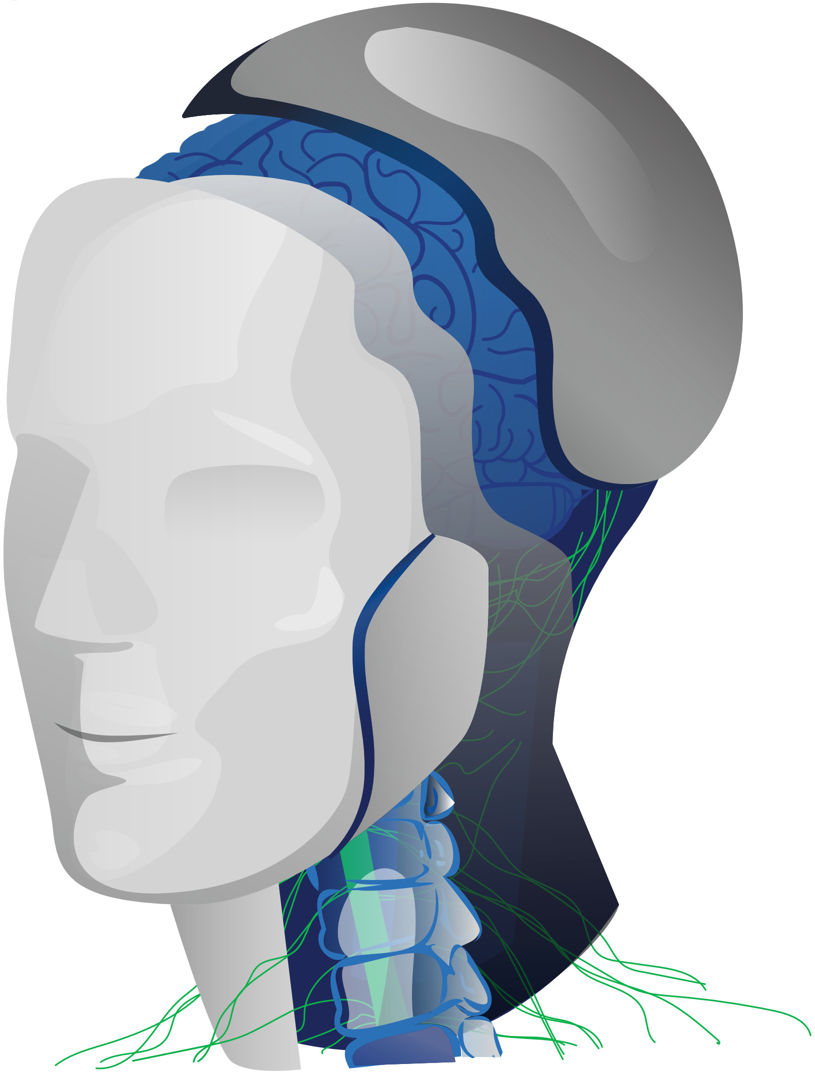
CYBER THREAT ANALYSIS

Deconstructing the Adversary Exploit Process

By Insikt Group®



CTA-2020-0603

Recorded Future's Insikt Group® conducted a study of files that test exploits as verdicted by VirusTotal from November 1, 2019 to April 1, 2020 to derive insight into how actors test exploits that they develop or modify existing exploit code. We believe this data shows that older vulnerabilities, often with easily accessible exploits or tutorials, remain popular among less sophisticated threat actors, as well as red teams and penetration testers. Because we did not have adequate data around single instances of an actor testing a network-bound vulnerability, we focused on generally file-centric vulnerabilities and exploits.

This research intends to track exploit development for actors whose actions we can observe in the data we have access to. This will inherently not net the discovery of new zero-day authors or their products, but serves to map how the average actor crafts an exploit for use in their malware. Sources include the Recorded Future® Platform, as well as VirusTotal and other open sources.

Executive Summary

Threat actors often use exploits to facilitate their intrusions without increased need to engineer or interact with victim users. As an example, these exploits may help deploy malware by making it possible to execute code on a victim system, aid in gathering normally inaccessible data, or gain access to restricted systems. However, to use an exploit, a threat actor must first identify the need an exploit should serve, find an exploit to meet that need, and then weaponize the exploit as part of a proof of concept prior to production.

To understand how this process might play out in the wild, Recorded Future identified and evaluated a series of methodologies to identify code that was being used to test exploits in VirusTotal data. As a result, we observed that Microsoft Office files represented the largest share of the potential testing files, followed by Portable Executable (Windows binary) files, and the most commonly tested vulnerabilities were CVE-2014-6352 (Sandworm) and CVE-2017-0199. These initial findings suggest that older vulnerabilities, often with easily accessible exploits or tutorials, remain popular among less sophisticated threat actors, as well as red teams and penetration testers. Documenting the activity of these groups can inform the thinking of nascent threat intelligence teams, or vulnerability management teams that are just beginning to introduce security intelligence into their calculus. These teams can use the findings outlined in this work to help prioritize patching or defensive posturing against the incursions of lower-tier threat actors.

Key Judgments

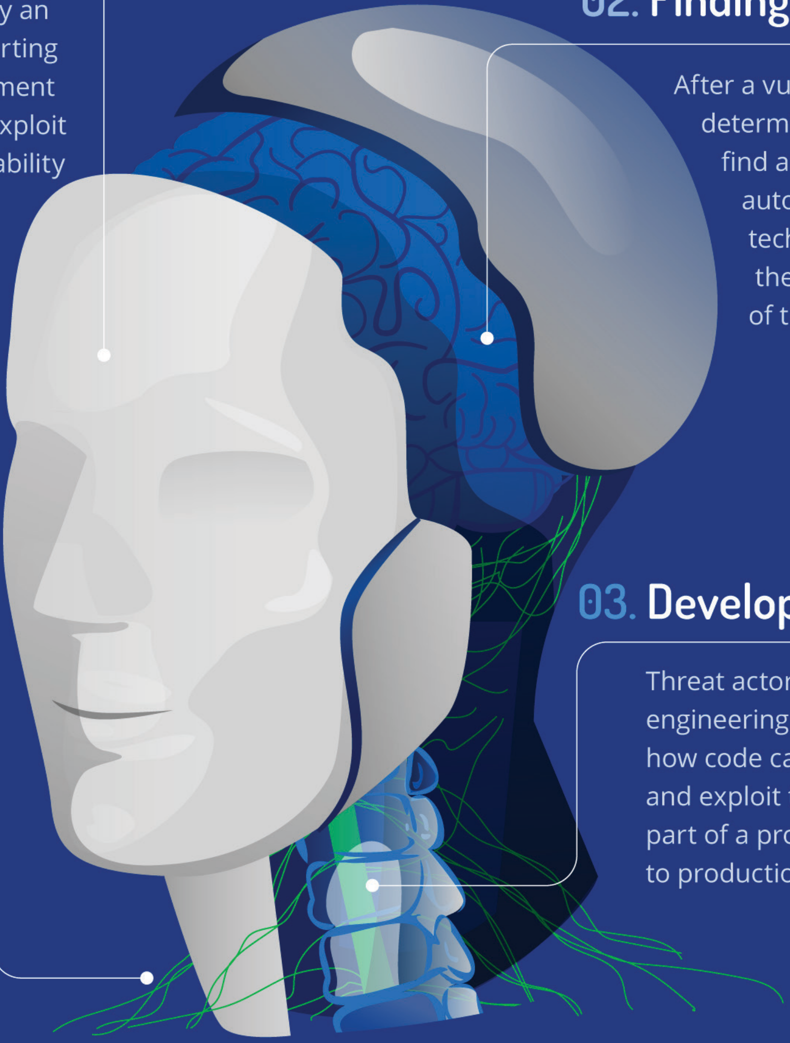
- Our findings highlight that exploits do not fall out of style, but can remain popular and reliable tools, in part because legacy systems remain in use. We do not get to stop defending against a vulnerability when the headlines go away.
- Insikt Group observed actors typically testing exploits for Microsoft Office products, very likely due to the ubiquity of the tools.
- Eight of the top 10 CVEs observed had open source exploit code available, making them easily accessible for actors to incorporate into their tool sets, both wholesale or in pieces.

01. Identification

Threat actors often identify an exploit through news reporting or a company's announcement of a patch. The use of an exploit facilitates a threat actor's ability to gain access, collect information from, or otherwise exploit a victim system.

02. Finding the Weakness

After a vulnerability is determined, threat actors find an exploit through automated or manual techniques to identify the exploitable aspect of the code.



03. Develop Usable Code

Threat actors use reverse engineering to understand how code can be weaponized and exploit the software as part of a proof of concept prior to production.

04. Code Testing

Threat actors must test their developed code to ensure it has the correct functionality as well as evaluate the code's ability to evade detections.

RECORDED FUTURE'S STUDY TO IDENTIFY THE MOST TESTED EXPLOITS AND THE ADVERSARY EXPLOIT PROCESS SHOWED THAT OLD EXPLOITS WILL CONTINUE TO BE USED IN NEAR-FUTURE INTRUSION ATTEMPTS. Documenting the activity of these groups can inform the thinking of threat intelligence teams or vulnerability management teams that are just beginning to introduce security intelligence into their processes.

Background

In a previous report titled "The Top 10 Vulnerabilities Used by Cybercriminals in 2019," Recorded Future analysts found that some of the most commonly exploited vulnerabilities are in Microsoft Office products. The continued focus of threat actors on exploiting these vulnerabilities suggests that many victim organizations continue to run outdated versions of Microsoft Office. The difference between the findings of this report and our previous research is that this data focuses on actors beginning to test their code, not on reported incidents. This data reflects the most-tested exploit in open data, rather than the most-used exploits in cyberattacks.

To understand how exploits can be productized, Recorded Future conducted research to observe whether we could detect their use in publicly available data sources. Our research focused on identifying which exploits were most readily developed into test implementations, or proofs of concept, and uploaded to VirusTotal data. We believe using this methodology gives insight into the applications of interest for threat actors who may be less sophisticated, such as script kiddies, junior penetration testers, and adversaries with imperfect operational security. While the outcome of the study may be expected, the report provides another perspective on adversarial exploit focus and intent.

Our research identified 621 files containing exploit code, based on VirusTotal's verdicts of exploits. This data was gathered from November 1, 2019 to April 1, 2020. We relied on these different measures to gather the largest but most accurate data pool we could. These files all had metadata (such as "POC" or "test" strings in the name) and compilation information that we believe showed indications of testing the exploit code or the ability of various antivirus engines to detect the exploit. The most common files (45.73%, or 284 files) were Microsoft Office files.

The actors who use VirusTotal to conduct testing are most likely of low sophistication and have minimal concern for the operational security of their work; those that create and sell zero-day exploits very likely use alternative methods of testing such as no-distribute antivirus scanners. However, documenting these actors' activity is important for threat intelligence teams and vulnerability management teams; understanding what exploits actors are testing can provide an advantage to defenders. Due to their numbers, these actors represent the largest percentage of attempted intrusions overall. Understanding how these attackers test and execute exploits will aid in actively blocking against these intrusion attempts.

Threat Analysis

1. Identification

Often, criminal threat actors will learn about vulnerabilities when the general public does — via news reporting or when a software company provides a patch. Far less common is shared reporting of exploitation by other actors. As displayed by previous Recorded Future research, the most commonly exploited vulnerabilities are found in the most popular products, such as Microsoft. Intuitively, the more ubiquitous a software or operating system, the more interest will be garnered in a published or patched vulnerability. According to Recorded Future data analyzed by Insikt Group, vulnerability announcements from Microsoft and Google are the most discussed on underground communities since the start of 2019.

2. Finding the Weakness

While this research focused on threat actors making use of known vulnerabilities as verdicted by VirusTotal, rather than developing and testing new vulnerabilities, vulnerability identification in software is critical to the success of usable exploits. The process of identifying vulnerabilities in software may require both automated and manual techniques to identify deficiencies such as privilege escalations, memory safety issues, or input validation bugs that, if exploited, can grant attackers permissions. One automated technique that can be employed is fuzzing, in which input to a software program is randomly altered to see whether a crash can be produced (dumb fuzzing), or targeted input based on the protocol or software under assessment (smart fuzzing) is created to induce the same behavior. Further, automated methods can be applied to look for known "bad" code patterns (such as the use of insecure coding functions). A more manual technique, reverse engineering, can be employed by a human to look for potential bugs within the software or to possibly exploit a bug identified in an automated way. However, it is important to remember that, even if a vulnerability is identified, this does not mean that it is exploitable in a way that helps a threat actor accomplish an end goal.

Once a vulnerability is deemed to be of interest to a threat actor, they must then identify the exploitable aspect of the code. A straightforward method can be to compare the patched software and the previously vulnerable version. The differences can reveal the exploitable aspect of the code by identifying the location and contents of a patch. Further reverse engineering is required to understand how code can exploit the software.

Another method of discovery is reverse engineering a known piece of malware that has exploited the vulnerability. This method can enable discovery of a vulnerability, without needing to fully understand the vulnerable software. Similarly, an actor can examine the code of a proof-of-concept exploit from sources such as a Metasploit module, a research blog, or on GitHub, in order to understand the vulnerability.

3. Develop Usable Code

Whether an actor develops their own exploit code or incorporates published proof-of-concept code, they must work it into a format usable for intrusions. Developing their own tooling, no matter how patchwork, becomes essential for them to weaponize the vulnerability.

What Makes a Vulnerability a Good Target for Exploitation?

A vulnerability in a system represents a weakness in that system; an exploit is something that takes advantage of that vulnerability to accomplish a goal. Productizing a vulnerability into an exploit is a consistent cost-benefit analysis that a threat actor calculates — is the payoff worth the time and resources needed to weaponize the vulnerability? Here, we discuss what a threat actor may take into account in deciding what vulnerabilities provide adequate "return on investment" (ROI).

To present a valuable target for exploitation, a vulnerability must first allow an actor to accomplish a specific goal. A few examples of an attacker's goals for which a vulnerability might be required include elevating permissions on a victim's system, gaining access to data within the system, gaining access to credentials, evading defenses on a system, or performing a specific action, such as remotely executing code.

The intentions of threat actors can be diverse — some actors, from entry level to advanced, may target any system that presents an opportunity for exploitation, while others may identify particular, high-value targets and only attack those. Once an attacker has identified a potential vulnerability, he or she is then able to evaluate whether the vulnerability can be exploited, if it has not already been.

Next, a threat actor might consider other factors in weighing the return on investment for the amount of effort that may be needed to make a vulnerability useful. To assess exploitability, the Common Vulnerability Scoring System (CVSS) uses a metric that is composed of defined values for four metrics: Attack Vector, Attack Complexity, Privileges Required, and User Interaction.

- The **Attack Vector** metric focuses on the accessibility required to exploit the vulnerability. Vulnerabilities that can be exploited more remotely (for example, over the network) are given a higher score than those that would require an attacker to have closer (physical) access to the target system.
- **Attack Complexity** quantifies the intricacies and necessary precision of conditions required to exploit the vulnerability successfully — effectively, a metric of what needs to go "right" for the exploit to work. For example, if the vulnerability requires an attacker to "time" the execution of the exploit correctly, collect a piece of system information (such as a model number, user information, or a shared key) prior to successful execution, or prepare the system ahead of using the exploit, it would receive a lower score than a vulnerability that does not require the same level of effort.
- The **Privileges Required** for a vulnerability also influences its exploitability. Higher privilege is often harder to obtain, and this value can be none (highest value), low, or high (lowest value).
- Finally, the user interaction metric describes whether this vulnerability can be exploited at the will of the attacker, or whether it requires user interaction in some way, such as the installation of a program or update to be successful. While these four metrics represent a standard way of measuring exploitability, they do not account for all aspects of the issue.

An attacker will most likely take the path of least resistance, using the "easiest" route to accomplish his or her goal. This means that if the same effect can be achieved with an exploit that requires only user-level permissions as one that requires root-level permissions, the attacker will generally select the former. Conversely, an attacker may find more appeal in exploiting a vulnerability that allows the attack to stay undetected — for example, one that avoids a User Account Control pop-up — and provides better perceived odds for success. While the CVSS metrics for exploitability provide a concrete way of assessing the ease of exploitation of vulnerabilities as a whole, there are subtle reasons for a threat actor to choose one vulnerability over another that may not directly align with these metrics.

Other factors that may also influence the appeal of a vulnerability include the number of target systems that could be affected by its use — if this fits with the threat actor's goals. If a threat actor is aiming to target as many systems as possible, as we believe is often the case with botnet or banking malware, the threat actor would most likely aim to exploit a vulnerability present on systems that have a large market share. For example, the Heartbleed Bug in the OpenSSL cryptographic library affected all code using this library, which included a number of diverse types of software, suggesting that targeting this vulnerability would offer opportunity across types of systems and software. However, if an actor is not globally targeting systems, but instead focusing on specific countries or industries, the popularity of systems and software may differ from global market share and lead to a different direction for exploit development.

Exploit Testing Study

After developing code that makes use of a vulnerability for exploitation, actors must test their code to ensure it has the correct functionality, as well as evaluate the code's ability to evade detections. This is where we focused our study of threat actors' efforts to develop exploit code.

4. Code Testing

If an actor has no access to a sandbox to test their exploit, they may rely on external parties to provide proof of exploitation. While more experienced actors advise or prohibit checking popular multi-scanning services in favor of no-distribute sites, some actors may still test their malware in VirusTotal. Recorded Future's underground forum collections showed that for the past three months, over 400 unique VirusTotal URLs were shared among forum members, implying that this method can be used to show proof of exploitation, or proof of evasion.

For this research, we relied on VirusTotal's verdicts of exploits. This data was gathered from November 1, 2019 to April 1, 2020. We relied on these different measures to gather the largest but most accurate data pool we could. Recorded Future gathered data from VirusTotal searches, which included:

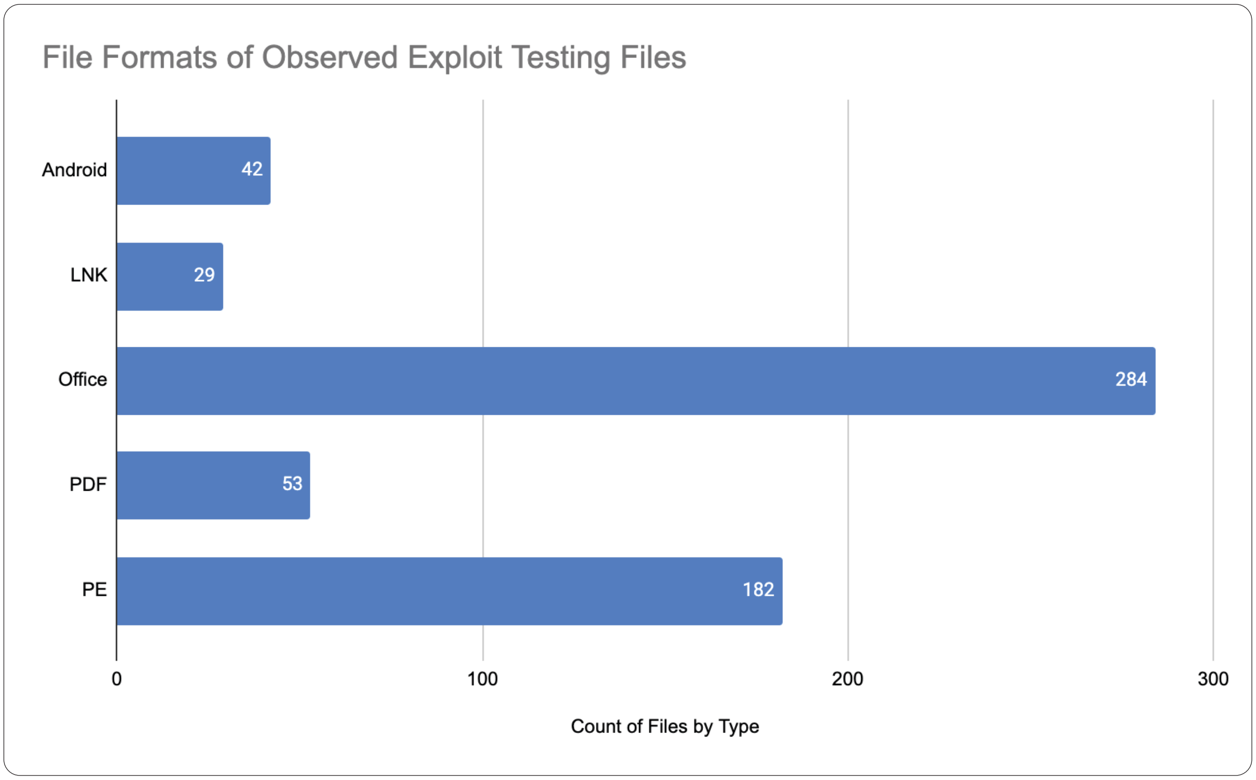
- Exploit code with a very short (under five minute) difference between compilation time and submission time, referred to as subspan
- Exploit code with the terms “test,” “exploit,” or “poc” in the filename

```
tag:exploit subspan:300- fs:2019-11-01T00:00:00+
tag:exploit (name:”test” or name:”poc” or name:”exploit”) fs:2019-11-01T00:00:00+
```

Upon examining the data, five outlier categories were immediately identified, which we removed to refine the data set. One of the measures we used, a short subspan (under five minutes between compilation and it appearing in VirusTotal) can indicate an actor rapidly testing code, or automatic uploads of worm variants. Five outliers met this criterion:

1. The worm Sality uses freshly generated LNK files exploiting CVE-2010-2568 to propagate itself, and their low subspan time appears to very likely be automatic sample generation.
2. A large number of Excel files with names from the directory “\Users\Petra\AppData\Local\Temp\” exploiting CVE-2014-6352 were found to be very likely automatically generated and submitted, which we opted to remove from the data set.
3. A dozen files detected as WannaCry, using the filename lhdfrgui.exe, were detected as exploiting CVE-2017-0147, but were found to be recompiled versions of WannaCry, or resubmitted samples. Due to this, we have elected to ignore them from our data set.
4. A large cluster of files exploiting CVE-2020-0601 were very likely compiled in the same environment (derived from identical rich header data in the executable), and shared the name wildfire-test-pe-file.exe, or some variation. Due to this very likely being an antivirus testing file (similar to EICAR test files), we can infer that they are not malicious files.
5. A series of JavaScript files exploiting CVE-2013-0422 were uploaded between January 20 and January 31, 2020. These files were all nearly identical; Recorded Future cannot readily identify whether these files were the outcome of an exploit development course or actor testing, or related to Metasploit’s capture-the-flag contest occurring around the same time.

As such, we have ignored all five of these outliers from our data set to focus on likely examples of exploit development. This netted us with 621 files containing exploit code.



Perhaps unsurprisingly, the most common file types used were Office documents, which we believe with high confidence reflects that threat actors are most interested in targeting Microsoft Office vulnerabilities. Due to the format's popularity in corporate and home environments, Office documents present an opportunity for actors to target broadly. Office Documents abusing macros without exploits are the most common phishing vector according to Cofense, but Office exploits remained high on common phishing tactics from a number of vendors. Office exploits are also prime candidates for achieving initial access onto a target victim system, along with malicious attachments and links used as part of phishing attacks.

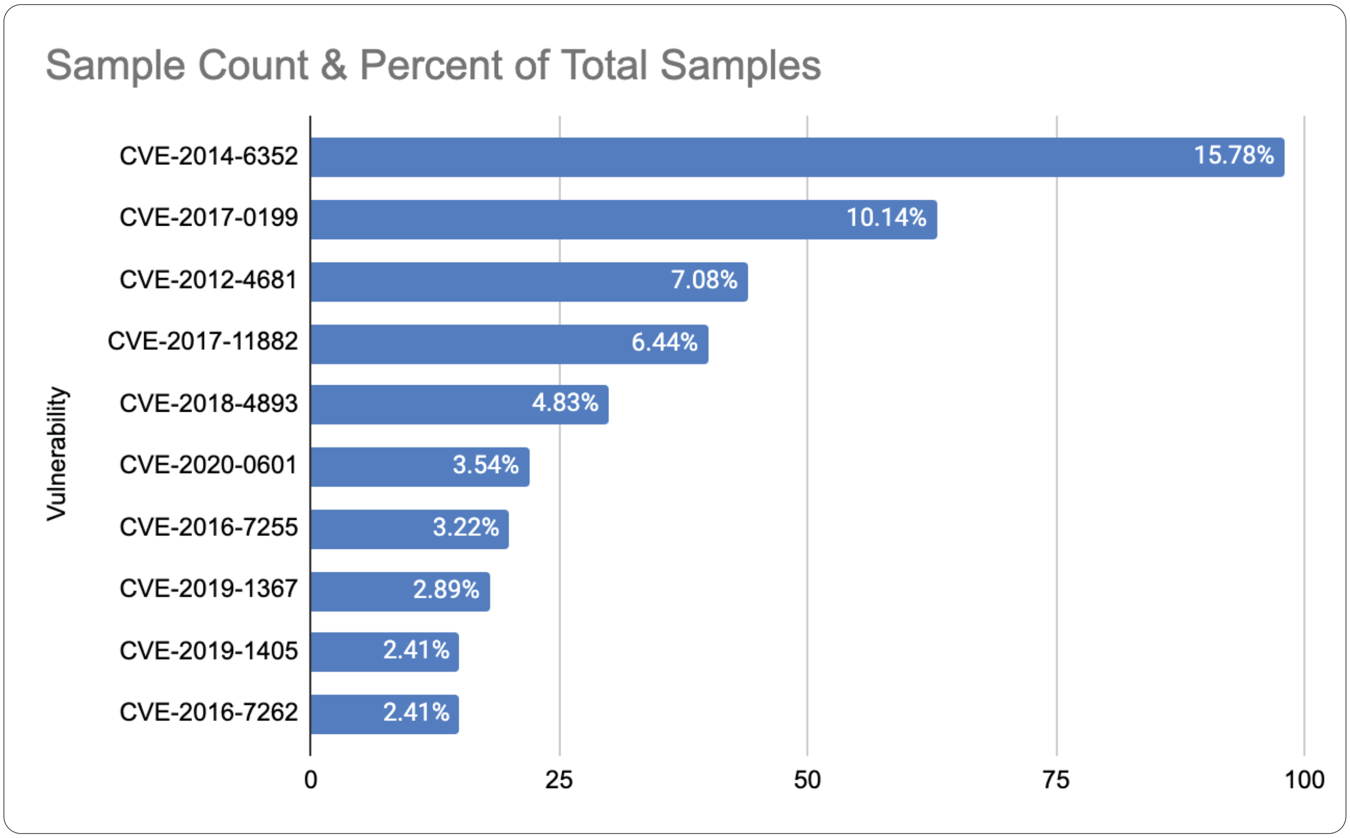
We believe with high confidence that the data reflects a prioritization of initial access for threat actors, as footholds, infection management, and lateral movement can be facilitated by open source shells and toolkits such as Metasploit and PowerShell Empire. This allows actors to focus on crafting initial access exploits, rather than creating a functional and evasive presence on the victim host.

Top 10 Most Commonly 'Tested' Vulnerabilities			
Vulnerability	Sample Count	Affected Product	Code Available
CVE-2014-6352	98 Files	Windows OS	Metasploit
CVE-2017-0199	63 Files	Microsoft Office	GitHub
CVE-2012-4681	44 Files	Oracle Java	Metasploit
CVE-2017-11882	40 Files	Microsoft Office	GitHub
CVE-2018-4893	30 Files	PDF	N/A
CVE-2020-0601	22 Files	Windows OS	GitHub
CVE-2016-7255	20 Files	Windows OS	GitHub
CVE-2019-1367	18 Files	Internet Explorer	Private Code
CVE-2019-1405	15 Files	Windows OS	Metasploit
CVE-2016-7262	15 Files	Microsoft Office	GitHub

Even after eliminating outliers from the data, files involving CVE-2014-6352 were the most commonly observed in our data. The popularity of files using CVE-2014-6352, also known as Sandworm, is perhaps surprising given the age of the vulnerability and its usability only against older Windows hosts. However, the Malaysian CERT identified APT40 (formerly known as TEMP.Periscope) using this vulnerability in a 2019 campaign, showing its continued effectiveness. We assess with high confidence that the use of this vulnerability in recent campaigns shows that legacy exploits remain relevant and important to defend, even years after the fact. Additionally, the exploit itself, which reliably provides code execution via Windows Object Linking and Embedding (OLE) on specific Windows systems running Office 2010 and 2013, is quite powerful, as reflected in its CVSS score of 9.3.

The files targeting CVE-2012-4681 were found in APKs, files used by Android devices. This is likely tied to the fact that this weakness in Java can be exploited in Android devices, and is included in the Metasploit Android framework.

The remaining popular exploits are a "who's who" of commonly exploited vulnerabilities, with an overarching theme of interest in Microsoft products, reflecting the firm's dominance in personal and enterprise computing. Microsoft Office files represented three out of the the top 10 list, far from the majority, but represented 45.73% of the total files, while Portable Executable files for Windows made up 29.3% of the total files analyzed. Others target Microsoft software such as Internet Explorer.

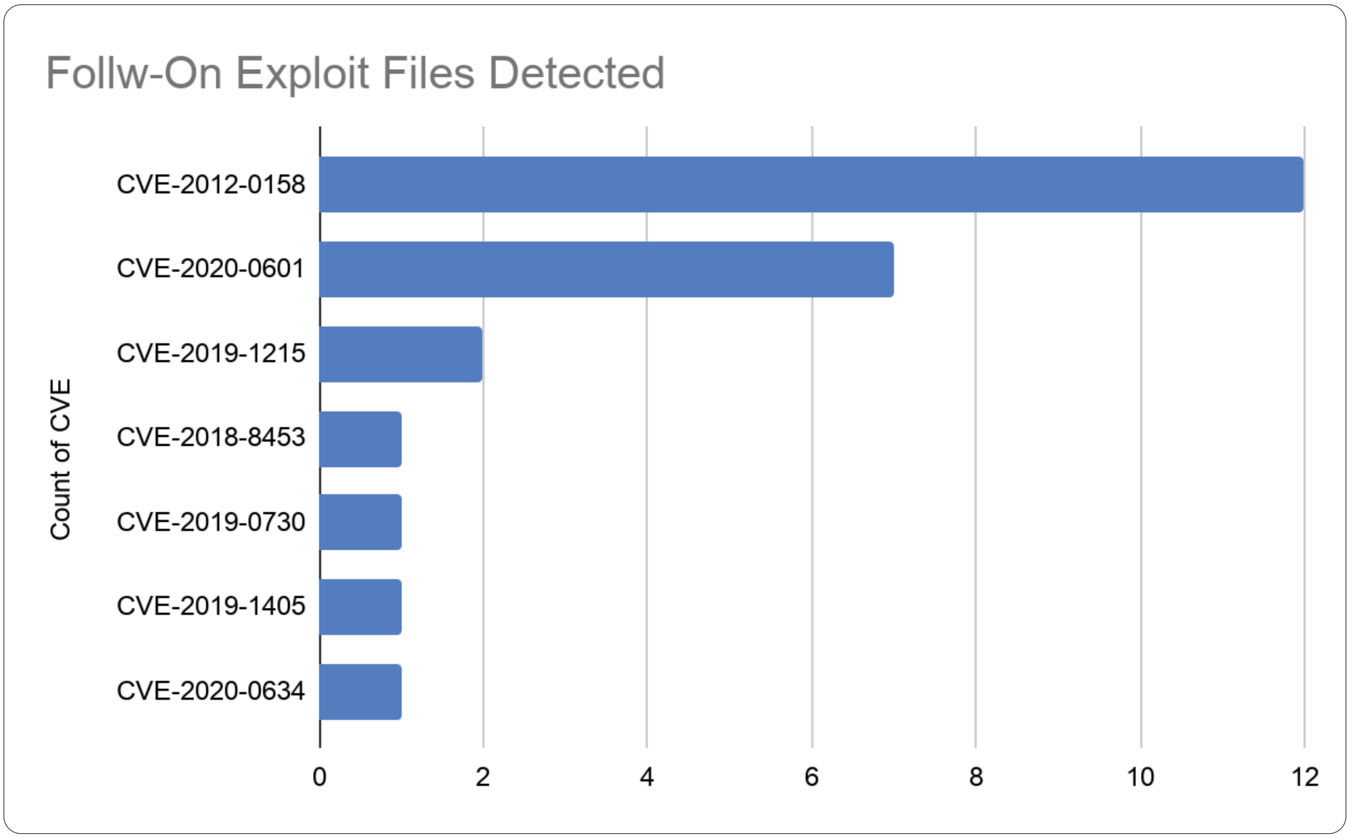


Downstream Exploit Tracking

After identifying files that were more obviously used for testing exploits, Insikt Group fingerprinted the files to track the development of unique malware. Insikt Group used various YARA file detection methods to tease out finalized versions of these exploits, introducing four different rule types; string-based, Rich Header-based, metadata-based, and program database (PDB) or C++/CPP string-based. These different filters allowed Insikt to pursue these actors beyond their initial testing attempts. Each type had its merits:

- Rich Header and PDB/CPP-based rules aimed to track malware development across a consistent compilation environment. While more advanced actors may scrub their PDB paths or spoof Rich Headers, Insikt Group assumed that these actors would not likely be testing their malware on VirusTotal, thus making such detections useful. We used PDBlaster and yararich.py to extract each from the target files.
- Rules built on metadata were helpful for monitoring for Office documents or files spoofing particular entities.
- Finally, string-based rules allowed for monitoring of the code reuse, which was used to help identify which testing files used code provided from an external POC, compared to those which appeared self-developed.

This method surfaced samples from VirusTotal, which showed that more recent exploits remained in current development, albeit at a much lower rate than the exploits found in the first survey. The exploit for CVE-2012-0158, also called VelvetSweatshop, may also be seeing a surge in this secondary detection due to trojans like LimeRAT exploiting it in recent campaigns.



However, these rulesets did not turn up any further iterations that were then found to be involved in cyberattacks or incidents; nothing in our data set had information around these samples other than their appearance in VirusTotal. We believe this is primarily due to a lack of visibility into the details of attacks, or the identified files not definitively being used in an intrusion. These rulesets did surface further iterations of previously tested files, however, suggesting that the submitters continue to evolve the code they are testing. While current data has not identified the use of these files as part of any specific cyber incident or intrusion, it is possible that they could occur in the future.

Outlook

Recorded Future conducted a study to identify the most tested exploits, which we believe shows that old exploits will continue to be used in near-future intrusion attempts. We believe the data herein does not require deep investigation if observed in an intrusion attempt, but is ripe for automation, allowing security teams to prevent intrusions using these methods, and to focus on less common incidents and threats. The most popular exploits focused on Microsoft products, with Microsoft Office files being the most common file type in this data set.

Resource constraints can challenge enterprises from keeping up to date with software patches. Identifying vulnerabilities that are being tested, or exploited in the wild, can help organizations prioritize vulnerabilities of the largest risk. We hope this reporting helps right-size the majority of the threat landscape posed to most enterprises: while new and attractive vulnerabilities are continually uncovered, they are deemed critical at a rate that is often too rapid to understand and mitigate for many organizations. Automating the blocking or detection of typically abused vulnerabilities can allow clients to become more agile to respond to newly identified vulnerabilities in their threat model.

Recommended Actions

Security teams can take action on data within this report with any of the following recommended actions:

- Defending vulnerabilities does not end after a patch is issued — actors will continue to use exploits for them well after they are identified. Recorded Future data can help clients understand the risks of vulnerabilities, and help prioritize patching or other mitigations to prevent exploitation.
- Vulnerability management teams can use Recorded Future's technical intelligence to prioritize patching based on which vulnerabilities are actively being exploited in the wild by malware.