Threat Intelligence

A LNK Between Browsers: Hunting Methodologies and Extension Abusing Actors

May 1, 2023

Mandiant

Written by: Jared Wilson

Two pillars in sleight of hand magic are *User Initiated Action*, where the target needs to believe their actions are their own, and *Hidden Action*, the trick needs to be concealed behind something ordinary and nonthreatening. Mandiant became aware of a chain of adversary methodologies that leverage these two pillars to achieve persistence.

1. The user executes an LNK shortcut file that, unbeknownst to them, has been tampered with.
2. The modified LNK shortcut file executes a legitimate browser, hiding the malicious extension.

If the technical sleight of hand is successful, the adversary will achieve persistence by means of malicious Chromium-based browser extensions.

While hunting this methodology Mandiant identified BRAINSTORM, a rust-based dropper, which ultimately led to RILIDE, a chromium-based extension first publicly reported by SpiderLabs. Careful investigation identified that the email and cryptocurrency theft ecosystem of RILIDE is larger than reported. This research will dissect the relevant adversary methodologies, discuss the identified malware families abusing this methodology, and include numerous detection opportunities to expand the defender's hunting and detection repertoire.

The Connection from LNK to Extension

The LNK File

Files with the extension .lnk are colloquially known as LNK files but are officially known as Shell Link Binary Files and follow a standardized format. LNK files contain information that points a user's interaction to another data object on the system. In many instances this is transparent to an end user. A Windows user may click on the Google Chrome icon in the Start Menu and Chrome opens. What is not shown to the user is that they are executing an LNK file with properties that point to the actual Chrome executable.

Mandiant has reported on many adversaries and malware families abusing LNK files including: FIN7, UNC1151, KEGTAP, FIN13, and APT29 (twice).

The CRX File

A CRX file is a collection of files archived together into a single package that can be used as an *extension* in Chromium-based browsers. Extensions enhance the browsing experience by adding features and functionality to the browser. Many browsers have an extension store where a user can review and install them into their browser all through the browser itself; this is generally accepted as a safer practice because the company owning the browser software performs analysis on the extensions themselves attempting to identify malicious extensions.

However, depending on the implemented security settings, browsers will allow for manual loading of CRX files or unpacked extensions. Packed extensions (CRX files) are a single file with a .crx extension, conversely an unpacked extension is a directory containing the extension files.

Throughout 2022 Mandiant has observed multiple financially motivated threat actors distributing and/or expressing interest in leveraging malicious browser extensions in their operations.

Abusing Both LNK and CRX

While Mandiant has previously reported on the abuse of LNK and CRX files separately, this recently observed adversary methodology has been using both filetypes within a chain of events and the bridging data-point is the `--load-extension` switch in Chromium-based browsers.

The `--load-extension` switch allows the source to specify a target directory to load as an extension. This gives malware the opportunity to start a new browser window with their malicious extension loaded.

```
chrome.exe  --load-extension="C:\Users\user\AppData\Local\T
brave.exe  --load-extension="C:\Users\user\AppData\Local\Te
msedge.exe  --load-extension="C:\Users\user\AppData\Local\T
opera.exe  --load-extension="C:\Users\user\AppData\Local\Te
vivaldi.exe  --load-extension="C:\Users\user\AppData\Local\
```

Figure 1: Example Commands to Load an Extension on Multiple Chromium-based Browsers

Mandiant Identification of Methodology Abuse

Mandiant investigated several compromises involving LNK and extension abuse methodology in 2023. The impacted organizations extended across a broad scope of sectors, including the semiconductor, business marketing, financial investment, and telecom industries.

The following three sections dive deep into separate investigations performed on malware families utilizing both LNK abuse and extension installing to achieve persistence with RILIDE.

Investigation 1

TradingView Desktop is a charting platform and a social network for traders and investors. This software allows users to track and view cryptocurrency market changes. As a software that is used in the finance industry with the capability of a cryptocurrency focus, it is a reasonable target to masquerade as for actors with goals of stealing cryptocurrency. Users willing to track cryptocurrency may be more likely to trade, allowing RILIDE a potential vector for cryptocurrency theft.

The file TradeVlewDesktop_v4-94406.zip is a TradingView Desktop masquerading set of files. The sample is a compressed directory that contains 457 different files. The file of interest in the zipped file is TradeVlewDesktop_x64.exe, a NodeJS-based downloader. After execution, TradeVlewDesktop_x64.exe reaches out to the Telegram masquerading URL `hxxp://telegromcn[.]org/soft/analytics/extension[.]exe` to download dropper extension.exe, which Mandiant tracks as BRAINFOG.

BRAINFOG is a Node.JS packaged binary dropper which drops RILIDE along with Visual Basic scripts to delete all Chrome LNKs and replace them with LNK files to force the execution of RILIDE. RILIDE is a Chromium-based extension that monitors the URLs visited by victims, select websites. RILIDE targets the theft of email and cryptocurrency details, falling inline with the targeted audience, the finance sector.

BRAINFOG drops extension.zip (RILIDE), wtf.vbs, chrome.vbs, and a Google Chrome.lnk file. Upon execution BRAINFOG uses…

- `taskkill.exe` to close all instances of Chrome
- `chrome.vbs` to delete all LNK files with "*Chrome*" in it
- `wtf.vbs` to create a new LNK using the --load-extension switch to force the loading of RILIDE browser extension at execution.

After the user loads Chrome via the replaced LNK shortcut file, RILIDE runs in the background as the infected browser loads and manipulates web pages. During initial execution, it fetches a machine identifier and a list of targeted domains from the command and control's (C2) API endpoint `/api/machine/init`; this list is re-fetched every five minutes.

```
{"machineId":2984,"urls":
["coinbase.com","binance.com","blockchain.com","mail.google
```

Figure 2: The domains this variant of RILIDE is monitoring for from /api/machine/init

When any HTML document has been completely parsed the DOMContentLoaded event will be delivered to the target function, loadScript. The loadScript function will download a list of key-values pairs which include a name and a path. The name is the domain related to the traffic of interest. If the browsing domain matches one of the monitored domains, the JavaScript in the path value will be accessed and the resulting file injected into the website for execution.

```
[
{"name":"coinbase.com","path":"scripts\/coinbase.js?v=3"},
{"name":"binance.com","path":"scripts\/binance.js"},
{"name":"blockchain.com","path":"scripts\/blockchain.js"},
{"name":"mail.google","path":"scripts\/gmail.js"},
{"name":"outlook.live","path":"scripts\/hotmail.js"},
{"name":"mail.yahoo","path":"scripts\/yahoo.js"},
{"name":"bybit.com","path":"scripts\/bybit.js"},
{"name":"okx.com","path":"scripts\/okx.js"}
]
```

Figure 3: Domains and URIs listed at the /api/machine/get-urls endpoint of the Adversary C2

This enables the adversary to invoke actions on behalf of the victim or steal data from their web sessions.

While hunting the RILIDE malware family Mandiant identified numerous

```
Figure 4: RILIDE C2 API Endpoints
hxxp://extenision-app[.]com/api/settings
hxxp://extenision-app[.]com/api/machine/
hxxp://extenision-app[.]com/api/machine/init
hxxp://extenision-app[.]com/api/machine/get-urls
```

Figure 4: RILIDE C2 API Endpoints

Figure 4 explains the contents each endpoint returns.

Endpoint	**Function**
/api/settings	• Datetimes for when it was created and updated • Multiple currency amounts defining the minimum amount (80 EUR, 80 USD, etc) • Telegram Chat ID and Token
/api/machine	Returns a list of details about victims. This includes… • Victim IP address • Victim country • Variant reference (Google, TradingView, etc) • When it was added • When it was last observed communicating to the C2
/api/machine/init	Returns a machine ID and the list of domains it monitors for
/api/machine/get-urls	Returns the URI path to the JS script to inject

Figure 4: RILIDE C2 API Endpoint Descriptions

Investigation 2

Previously highlighted in the SpiderLabs blog, the GitHub user gulantin was identified as having numerous GitHub repositories storing RILIDE samples. Furthermore, Mandiant suspects this may have been a method for delivery.

A file named Blanks, tracked by Mandiant as BRAINLINK, was downloaded from the raw.githubusercontent.com URL on the gulantin github `(hxxp://raw.githubusercontent[.]com/gulantin/blanks/main/blanks_online.exe).`

BRAINLINK is an Advanced Installer compiled dropper which drops a CAB file that contains the RILIDE extension files along with PowerShell scripts to create new shortcuts forcing the execution of RILIDE.

Mandiant's research of RILIDE identified that the background JavaScript file includes a domain variable set to the C2 domain for each malware version. In this investigation the RILIDE sample used the domain `ashgrrwt[.]click`.

```
const domain = "https://ashgrrwt.click"
```

Figure 5: RILIDE C2 domain variable defanged

RILIDE C2 Infrastructure Hunt

This C2 infrastructure provided interesting overlaps between numerous other domains. While the Admin, Billing, and Technical WHOIS details for the `ashgrrwt[.]click` domain were redacted for privacy, the registrant organization was not. The registrant organization, Kruglova LTD, was associated with 11 other websites.

Figure 6: RILIDE C2 domain infrastructure graph

The domains identified follow the overarching theme to these campaigns: Cryptocurrency Exchange Platforms (FinAndy/TradingView) and Finance/Banking.

Investigation 3

Researching the RILIDE ecosystem led to the identification of an open directory at `146.70.79[.]75` which included two BRAINSTORM samples (0a4f321c903a7fbc59566918c12aca09 and 34eea751fcbf4ee8d44977adb4742d93) and numerous other malicious samples. BRAINSTORM is a Rust-based dropper which drops RILIDE and updates Google Chrome, Brave, and Microsoft Edge LNK files to force the execution of RILIDE. Mandiant is tracking the activity related to this open directory as UNC4553.

Figure 7: UNC4553 Open Directory

```
// M_Hunting_FileWrite_ManifestandChromeLNK_1
tag:peexe and ((behaviour_files:"C:\\Users\\Public\\Desktop
behaviour_files:"C:\\ProgramData\\Microsoft\\Windows\\Start
Chrome.lnk") or (behaviour_files:"C:\\Users\\Public\\Deskto
and behaviour_files:"C:\\Users\\user\\AppData\\Roaming\\Mic
```

The open directory IP shown in Figure 7 (146.70.79[.]75) has previously resolved to `nch-software[.]info` and `panger-top[.]click`. Further solidifying the connection from the IP to the domains, there is evidence connecting the URI patterns in the open directory to the URI patterns for these domains.

```
hxxps://nch-software[.]info/1/2.exe
hxxps://nch-software[.]info/1/install-win64-11.5.8_en-US.ex
hxxps://panger-top[.]click/1/2.exe
hxxps://panger-top[.]click/1/install-win64-11.5.8_en-US.exe
```

Figure 8: Related Open Directory URLs

The open directory contained the two BRAINSTORM samples, two PUFFPASTRY samples, a suspected incomplete PUFFPASTRY sample, and two XLL samples.

PUFFPASTRY is a backdoor written in Visual-basic. PUFFPASTRY can download, upload, delete, and execute files. Additionally, PUFFPASTRY can self-terminate and enumerate system information including Anti-virus details. C2 communications occur over standard HTTP/HTTPS.

An XLL add-in is an Excel add-in file with the file extension .xll. An XLL file is a type of dynamic link library (DLL) file that can only be opened by Excel. It is not exactly clear what the intention of some of these files are given they appear to be in staging or templates.

URL
`hxxps://146.70.79[.]75/2.exe` `hxxps://146.70.79[.]75/1/2.exe`
`hxxps://146.70.79[.]75/1/install-win64-11.5.8_en-US.exe`
`hxxps://146.70.79[.]75/templates/light.dotm`
`hxxps://146.70.79[.]75/templates/light.pub`
`hxxps://146.70.79[.]75/templates/light.xlsm`

```
hxxps://146.70.79[.]75/templates/x86.xll
```

Figure 9: UNC4553 Open Directory Contents

System Hardening

Preventing the malicious extensions (RILIDE and others) to be loaded by the browser is the linchpin to stopping the actor's methodology. With the inability to install the extension, further damage or exfiltration of sensitive data is prevented.

Chrome Enterprise provides numerous extension blocking options within the security settings. The following highlights a list of the settings relevant to this methodology.

- BlockExternalExtensions – Controls external extensions installation.
- DeviceLoginScreenExtensionManifestV2Availability – Remove the ability to use version 2 manifests.
- ExtensionInstallAllowlist and ExtensionInstallBlocklist – Implement block and allow lists to vastly restrict the extensions available.
- ExtensionInstallSources – Use strings with wildcards to identify where extensions can be installed from.

If there is a concern that Chrome LNKs have been manipulated or as a reoccurring security measure, users can run a user-level Chrome Settings Reset. When executed, numerous Chrome profile settings will change to the default on all devices where you're signed in. This will reset all of the default Windows-provided Chrome LNKs (i.e. Quick Launch LNKs) removing the `--load-extension` parameter that the actor is using to force the loading of the malicious extension.

Following a defense in depth strategy, implementing the detections documented in the Detection Opportunities section will help cover any prevention gaps.

Detection Opportunities

The Detection Opportunities section will be broken into two detection directions and will encompass a list of rules that will be expanded on in Appendix A.

1. Detecting Methodologies – Larger chance of detecting legitimate files or tactics that overlap with the methodology.
2. Detecting Malware Families – More targeted approach to detect the specific families themselves.

Detection Engine	Detection Title	De De
YARA	M_Hunting_Embedded_Chromium_CRX_1	De CR wit ext eq
YARA	M_Hunting_Embedded_Chromium_CRXandLNK_1	De CR wit ext an eq
YARA	M_Hunting_AdvancedInstaller_LNK_1	De Ad Ins tha LN kn loc
YARA	M_Hunting_LNKEngine_LoadExtension_1	De tha loa ext a c bro
YARA	M_Hunting_LNKEngine_LoadExtension_Temp_1	De tha loa ext fro Ap dir
YARA	M_Hunting_ArchiveEngine_CAB_Extension_1	De file inc po ext file
YARA-L	M_Hunting_FileWrite_Manifest_Temp_1	De eve a

		wri Te dir
YARA-L	M_Hunting_Process_Chromium_LoadExtension_1	De eve Ch sta the ext pa wit val ap ter dir
YARA-L	M_Hunting_FileWrite_Chrome_LNK_1	De eve an wri kn loc fro pr is i dir sp
YARA-L	M_Hunting_FileWrite_CRXandLNK_1	De eve a p wri or an wit mi ea
YARA-L	M_Hunting_FileWrite_ManifestandLNK_1	De eve a p wri ma LN wit mi ea
VT Grep	M_Hunting_FileWrite_ManifestandChromeLNK_1	De

		Ch an ma at

Detecting Malware Families

Detection Engine	Detection Title	Detection Description
YARA	M_Hunting_RILIDE_InjectJS_1	Detects the JavaScript files that RILIDE injec
YARA	M_Hunting_RILIDE_InjectJS_2	Detects the JavaScript files that RILIDE injec
YARA	M_Hunting_RILIDE_InjectJS_3	Detects the JavaScript files that RILIDE injec
YARA	M_Win_BRAINLINK_AdvancedInstaller_1	Detects BRAINLINK
YARA	M_Utility_RILIDE_Manifest_1	Detects RILIDE Manifest file
YARA	M_Utility_RILIDE_JS_1	Detects RILIDE JavaScript files
YARA	M_Utility_RILIDE_Background_1	Detects the RILIDE background JavaScript files
YARA	M_Hunting_RILIDE_CRX_1	Detects RILIDE CRX files

YARA	M_Win_BRAINSTORM_1	Detects BRAINSTOR
YARA	M_Win_BRAINFOG_1	Detects BRAINFOG
YARA	M_Hunting_BRAINFOG_1	Detects suspected BRAINFOG samples

Conclusion

While hunting for this chain of adversary methodologies, Mandiant uncovered numerous new malware families and investigations. The adversary's effort to remain undetected by chaining methodologies has come with mixed results, as many of the samples discussed in this post have very low detection ratings. However, with this new research, detecting these methodologies should be more accessible and further expand the defender's hunting and detection repertoire.

Acknowledgments

This content would not have been possible without the assistance of Connor McLaughlin, Pepe Torrijos, Matthew Dunwoody, Anders Vejlby, and Nick Simonian.

Appendix A: Detections

```
rule M_Hunting_Embedded_Chromium_CRX_1
{
   meta:
         author = "Mandiant"
         md5 = "f1c21a69ed9f85e12d58ef0f5ac5c9b1"
         description = "Hunting for non-CRX files with exte

   strings:
         $a1 = "_metadata" ascii
         $a2 = "manifest.json" ascii
         $a3 = "verified_contents.json" ascii
         $s1 = "_locales" ascii
         $s2 = "messages.json" ascii
         $f = /[a-z0-9A-Z_-]+\.(html|htm|css|js)/ ascii
   condition:
         (((uint16(0) == 0x5A4D) and uint32(uint32(0x3C)) =
         (((2 of ($a*)) and $f) or ((1 of ($a*)) and ($f or
         $pk and
         (#pk >1) and
         (for any i in (1..#pk) : ($a2 at @pk[i]+30)) and
         (for any j in (1..#pk) : ($f at @pk[j]+30))
}
```

```
rule M_Hunting_Embedded_Chromium_CRXandLNK_1
{
   meta:
         author = "jared.wilson"
         md5 = "f1c21a69ed9f85e12d58ef0f5ac5c9b1"
         description = "Hunting for non-CRX files with exte

   strings:
         $a1 = "_metadata" ascii
         $a2 = "manifest.json" ascii
         $a3 = "verified_contents.json" ascii
         $s1 = "_locales" ascii
         $s2 = "messages.json" ascii
         $f = /[a-z0-9A-Z_-]+\.(html|htm|css|js)/ ascii
         $pk = {50 4B 03 04}
         $load = "--load-extension" ascii wide
         $lnk1 = "Google Chrome.lnk" ascii wide

   condition:
         (((uint16(0) == 0x5A4D) and uint32(uint32(0x3C)) =
         (((2 of ($a*)) and $f) or ((1 of ($a*)) and ($f or
         $pk and
         (#pk >1) and
         (for any i in (1..#pk) : ($a2 at @pk[i]+30)) and
         (for any j in (1..#pk) : ($f at @pk[j]+30)) and
         ($load or $lnk1)
}
```

```
rule M_Hunting_AdvancedInstaller_LNK_1
{
   meta:
         author = "Mandiant"
         md5 = "2782af385665c765807ed887d4bacf36"
         description = "Hunting for Advanced Installer file

   strings:
         $a1 = "Advanced Installer" wide
         $a2 = "Advanced Installer" ascii
         $a3 = "https://www.advancedinstaller.com" ascii
         $l1 = "Google Chrome.lnk"
         $l2 = "Brave.lnk"
         $p1 = "\\Microsoft\\Windows\\Start Menu\\Programs\
         $p4 = "\\Microsoft\\Internet Explorer\\Quick Launc
         $p5 = "\\Microsoft\\Windows\\Start Menu\\Programs\
         $p6 = "\\Microsoft\\Internet Explorer\\Quick Launc
         $r = /\$[^\.]+\.CreateShortcut\([^\)]+\)/

   condition:
         ((uint16(0) == 0x5A4D) and uint32(uint32(0x3C)) ==
}
```

```
rule M_Hunting_LNKEngine_LoadExtension_1
{
   meta:
        author = "Mandiant"
        description = "Hunting rule that looks for files c
        md5 = "30abf9ca1bb792eb5edd8b033c010979"

   strings:
        $r1 = /(chrome|msedge|opera|brave)[^\r\n]+--load-e
        $s1 = "chrome" ascii wide
        $s2 = "--load-extension=" ascii wide

   condition:
        (uint32(0) == 0x0000004c) and filesize < 50KB and
}
```

```
rule M_Hunting_LNKEngine_LoadExtension_Temp_1
{
    meta:
       author = "Mandiant"
       description = "Hunting rule that looks for files co
       md5 = "30abf9ca1bb792eb5edd8b033c010979"

    strings:
       $r1 = /(chrome|msedge|opera|brave)[^\r\n]+--load-ex
       $s2 = "--load-extension=" ascii wide

    condition:
       (uint32(0) == 0x0000004c) and filesize < 50KB and a
}
```

```
rule M_Hunting_ArchiveEngine_CAB_Extension_1
{
   meta:
        author = "Mandiant"
        description = "Looking for CAB containing what is
        md5 = "de283dfb9c88dbb6d455ca4b31c57240"

   strings:
        $f1 = "manifest.json" nocase
        $f2 = ".js" nocase
        $f3 = ".htm" nocase
   condition:
         filesize < 1MB and uint32be(0) == 0x4D534346 and (
}
```

```
rule M_Hunting_FileWrite_Manifest_Temp_1
{
   meta:
         author = "Mandiant"
         md5 = "f1c21a69ed9f85e12d58ef0f5ac5c9b1"
         description = "Hunting for cases where a process w
         severity = "Medium"

  events:
         $e.metadata.event_type = "FILE_CREATION"
         ($e.target.file.names = "manifest.json" OR $e.targ
$e.target.file.full_path = /[a-zA-Z]{1}:\\Users\\[^\\]+\\Ap

   condition:
         $e
}
```

```
rule M_Hunting_Process_Chromium_LoadExtension_1
{
   meta:
         author = "Mandiant"
         md5 = "f1c21a69ed9f85e12d58ef0f5ac5c9b1"
         description = "Hunting for cases where Chrome proc
         severity = "Medium"
events:
$e.metadata.event_type = "PROCESS_OPEN"
$e.target.process.command_line = /(chrome|brave|msedge|oper
$e.target.process.file.full_path = /(chrome|brave|msedge|op
    condition:
$e
}
```

```
rule M_Hunting_FileWrite_Chrome_LNK_1
{
   meta:
         author = "Mandiant"
         md5 = "f1c21a69ed9f85e12d58ef0f5ac5c9b1"
         description = "Hunting for cases where the LNK is
         severity = "Medium"
   events:
         $e.metadata.event_type = "FILE_CREATION"
         $e.target.file.full_path = /Google Chrome\.lnk$/
         ($e.target.file.full_path = `C:\ProgramData\Micros
         ($e.principal.process.file.full_path = /[a-zA-Z]{1
   condition:
         $e
}
```

```
rule M_Hunting_FileWrite_CRXandLNK_1
{
   meta:
         author = "Mandiant"
         md5 = "f1c21a69ed9f85e12d58ef0f5ac5c9b1"
         description = "Hunting for cases where a process w
         severity = "Medium"
   events:
         $e1.metadata.event_type = "FILE_CREATION"
         $e2.metadata.event_type = "FILE_CREATION"
         $md5 = $e1.principal.process.file.md5
         $e1.principal.process.file.md5 = $e2.principal.pro
         $e1.principal.process.file.md5 != ""
         $e1.principal.hostname = $e2.principal.hostname
         $e1.principal.hostname != ""
         $e1.principal.process.pid = $e2.principal.process.
         $e1.principal.process.pid != ""
         (((($e1.principal.file.file_type = "FILE_TYPE_CRX"
   match:
         $md5 over 1m
   condition:
         $e1 and $e2
}
```

```
rule M_Hunting_FileWrite_ManifestandLNK_1 {
   meta:
         author = "Mandiant"
         md5 = "f1c21a69ed9f85e12d58ef0f5ac5c9b1"
         description = "Hunting for cases where a process w
         severity = "Medium"
   events:
         $e1.metadata.event_type = "FILE_CREATION"
         $e2.metadata.event_type = "FILE_CREATION"
         $md5 = $e1.principal.process.file.md5
         $e1.principal.process.file.md5 = $e2.principal.pro
         $e1.principal.process.file.md5 != ""
         $e1.principal.hostname = $e2.principal.hostname
         $e1.principal.hostname != ""
         $e1.principal.process.pid = $e2.principal.process.
         $e1.principal.process.pid != ""
         ((($e1.target.file.full_path = /\\manifest\.json$/
   match:
         $md5 over 1m
   condition:
         $e1 and $e2
}
```

```
// M_Hunting_FileWrite_ManifestandChromeLNK_1
tag:peexe and ((behaviour_files:"C:\\Users\\Public\\Deskto
```

```
rule M_Hunting_RILIDE_InjectJS_1
```

```
		author = "Mandiant"
		md5 = "9fe5b99b20bc91995b81eddd917bff50"
		description = "Hunting for the code that RILIDE ir

	strings:
		$banner = "https://public.bnbstatic.com/image/emai
		$s1 = "[Bybit]Withdrawal Request" ascii
		$s2 = "[Bybit] Authorize New Device" ascii
		$a1 = "created a withdrawal request"
		$a2 = "Authorize New Device You recently attemptec
		$a3 = "Please check your withdrawal address carefu
		$a4 = "Verification Code Of Withdrawal"
		$a6 = "Withdrawal Verification Code"
		$a7 = "Verification Code Of Authorization"
		$a8 = "initiate this withdrawal or the address is'
		$a9 = "Authorize New Device You recently attemptec
		$a10 = "Confirm your new withdrawal address"
		$a11 = "A new withdrawal address was just added tc
		$f1 = "div:contains(\"Binance\"), div:contains(\"t
		$f2 = "binance()" fullword
		$f3 = "div:contains(\"Bybit\"), div:contains(\"byt
		$f4 = "bybit()" fullword
		$f5 = "div:contains(\"Huobi\"), div:contains(\"huc
		$f6 = "huobi()" fullword
		$f7 = "div:contains(\"Okx\"), div:contains(\"okx\'
		$f8 = "okx()" fullword
		$f9 = "div:contains(\"Kraken\"), div:contains(\"kı
		$f10 = "kraken()" fullword

	condition:
		filesize < 1MB and $banner and (1 of ($s*)) and (
}
```

```
rule M_Hunting_RILIDE_InjectJS_2
{
	meta:
		author = "Mandiant"
		md5 = "6e426758f184b5a942428731b749b000"
		description = "Hunting for the code that RILIDE ir

	strings:
		$anchor = "const DOMAIN = 'https://extenision-app.
		$v1 = "exchangeRates" ascii fullword
		$v2 = "supportedAssets" ascii fullword
		$v3 = "supportedAccounts" ascii fullword
		$v4 = "currencySymbol" ascii fullword
		$v5 = "userId" ascii fullword
		$v6 = "settings" ascii fullword
		$v7 = "extensions" ascii fullword
		$s1 = "Confirm settings change"
		$s2 = "2-step verification"
```

```
        $s5 = "Enter the 2-step verification code from you
        $s6 = "Didn't receive the SMS?"
        $s7 = "Re-send SMS"
        $u1 = "${DOMAIN}/settings"
        $u2 = "${DOMAIN}/exchange/get-address?type=${type
        $u3 = "${DOMAIN}/exchange/create-account"
        $u4 = "${DOMAIN}/exchange/set-balance"
        $u5 = "${DOMAIN}/exchange/set-all-balances"
        $u6 = "${DOMAIN}/exchange/set-withdraw"
        $p1 = "password = localStorage.getItem('coinbase_
        $p2 = "email = localStorage.getItem('coinbase_use
        $f1 = "getExchangeRates" fullword
        $f2 = "getSupportedAssets" fullword
        $f3 = "getAccounts" fullword
        $f4 = "currentWithdraw"
    condition:
        filesize < 1MB and (($anchor and (8 of them)) or (
}
```

Appendix B: Indicators

Indicator	Reference
`9984af7a440c39b7ac11a68f2da48137`	BRAINFOG
`1af84663df057aee4934abe717938b33`	BRAINFOG WScript
`f2f85d38b91f582a83388690fdc45284`	BRAINFOG WScript
`2782af385665c765807ed887d4bacf36`	BRAINLIN
`de283dfb9c88dbb6d455ca4b31c57240`	BRAINLIN CAB
`6b2e6d6650116d372ca8c47af08ca8fa`	BRAINLIN PowerShel Script
`0a4f321c903a7fbc59566918c12aca09`	BRAINSTO
`34eea751fcbf4ee8d44977adb4742d93`	BRAINSTO
`69a1c37a796dd3ed81785c1995f0973f`	BRAINSTO

`ea7496d6fb96e3c1e00a1d5f501f6724`	`BRAINSTO`
`f1c21a69ed9f85e12d58ef0f5ac5c9b1`	`BRAINSTO`
`3e181d794e62af5c54d4df5517766af8`	`PUFFPAST`
`70700ae977a0f3ca6a331842d8c103c9`	`PUFFPAST`
`770dd49f7340003d9c66b58cd793dada`	PUFFPAST
`f483821b0650653e4da643b212025709`	`PUFFPAST`
`0fc2bd7320c2edfd7985b87fc8cb1f96`	`RILIDE`
`223e499f6ba6ebdacf1dcff96008635b`	`RILIDE`
`395bef4512a3743299a45d4f9b74a2ee`	`RILIDE`
`4f506058ab8bfc5746308a95e34dce85`	`RILIDE`
`5133177ac4950cf772d2f729bb0622ec`	`RILIDE`
`b7ad9777e3166628abe11dd043ddfb7e`	`RILIDE`
`bb8323247baad2d592e7ad1896935dd1`	`RILIDE`
`d56d195ebfaea6d97cccddcf3823be24`	`RILIDE`
`4724261ef04e2301ecc9ac994b4b346e`	`RILIDE HTML`
`020a8ed8a2b123f6c58fed791c6ef636`	`RILIDE Injectior Script`
`219070a2502a47a50dd3df5c804074b4`	`RILIDE Injectior Script`
`3235e27576dd4e81f1c5986212ec2b78`	`RILIDE Injectior Script`
`41a5f1c5d032bcac16c903681674872c`	`RILIDE Injectior Script`

	Script
`94c16b8f9236ab88bb0bca60c4399665`	`RILIDE Injectior Script`
`9e5f43b2dc1606e27fa0cfdfb4e363d2`	`RILIDE Injectior Script`
`9fe5b99b20bc91995b81eddd917bff50`	`RILIDE Injectior Script`
`d41b7138ad25d0401acf3298d0110342`	`RILIDE Injectior Script`
`5b7a2e7195bceb8e125758ae27c1e791`	`RILIDE JavaScrip`
`d54fa225b07298ec34be872cd4ebf4ae`	`RILIDE JavaScrip`
`19c859513f67400f3563e656f27df1c0`	`RILIDE L`
`30abf9ca1bb792eb5edd8b033c010979`	`RILIDE L`
`baee9ba0b94ea1e2b2e566fc8a615554`	`RILIDE Manifest`
`f31e9238593b34b390ab8faf755e6ede`	`RILIDE Manifest`
`3bf971fcaa2a3bd321f4e0b6864cb86a`	`Template XLL`
`ac00b947ca51d0e71b4c792f1646e4e0`	`Template XLL`
`028b2f15560b0f80514cf0a23ae77a43`	`VB Macro`
`0a4834d3da05a1d0b2f3a0c13f352a0a`	`VB Macro`
`89c34309aca3214c3ce7c72b407570e8`	`VB Macro`
`172.67.192[.]61`	`C2 A Record`

`89.185.85[.]144`	`C2 A Record`
`45.159.188[.]125`	`C2 A Record`
`146.70.79[.]75`	`Open Directory`
`104.168.167[.]25`	`Open Directory`
`telegromcn[.]org`	`BRAINFOG In The Wild Doma`
`nch-software[.]info`	`Open Directory Domain`
`vceilinichego[.]ru`	`Open Directory Domain ar C2`
`2022-blanks[.]site`	`Registra Org Pivot Domain`
`finandy[.]info`	`Registra Org Pivot Domain`
`finandy[.]online`	`Registra Org Pivot Domain`
`flnand[.]online`	`Registra Org Pivot Domain`
`kz-smartbank[.]com`	`Registra Org Pivot Domain`
`mareux[.]online`	`Registra Org Pivot Domain`

`mmarx[.]quest`	`Registra Org Pivot Domain`
`okxsat[.]xyz`	`Registra Org Pivot Domain`
`pr-tracker[.]online`	`Registra Org Pivot Domain`
`qivvi-3[.]click`	`Registra Org Pivot Domain`
`serienjunkies[.]us`	`Registra Org Pivot Domain`
`vse-blanki[.]online`	`Registra Org Pivot Domain`
`ashgrrwt[.]click`	`RILIDE C`
`extenision-app[.]com`	`RILIDE C`
`hxxps://146.70.79[.]75/1/2.exe`	`Open Directory BRAINSTOR Download`
`hxxps://146.70.79[.]75/1/install-win64-11.5.8_en-US.exe`	`Open Directory BRAINSTOR Download`
`hxxps://146.70.79[.]75/2.exe`	`Open Directory BRAINSTOR Download`
`hxxps://nch-software[.]info/1/2.exe`	`Open Directory BRAINSTOR Download`

`hxxps://nch-software[.]info/1/install-win64-11.5.8_en-US.exe`	`Open Directory BRAINSTOR Download`
`hxxps://panger-top[.]click/1/2.exe`	`Open Directory BRAINSTOR Download`
`hxxps://panger-top[.]click/1/install-win64-11.5.8_en-US.exe`	`Open Directory BRAINSTOR Download`
`hxxp://nch-software[.]info/1/2.exe`	`Open Directory URL`
`hxxps://146.70.79[.]75/`	`Open Directory URL`
`hxxps://146.70.79[.]75/admin_cp/`	`Open Directory URL`
`hxxps://146.70.79[.]75/templates/`	`Open Directory URL`
`hxxp://vceilinichego[.]ru/api/machine/`	`RILIDE C URL`
`hxxp://vceilinichego[.]ru/api/machine/get-urls`	`RILIDE C URL`
`hxxp://vceilinichego[.]ru/api/machine/init`	`RILIDE C URL`
`hxxp://vceilinichego[.]ru/api/machine/set-urls`	`RILIDE C URL`
`hxxps://vceilinichego[.]ru`	`RILIDE C URL`
`hxxps://vceilinichego[.]ru/api/machine`	`RILIDE C URL`

`hxxps://vceilinichego[.]ru/api/machine/`	`RILIDE C URL`
`hxxps://vceilinichego[.]ru/api/machine/check-tasks`	`RILIDE C URL`
`hxxps://vceilinichego[.]ru/api/machine/get-urls`	`RILIDE C URL`
`hxxps://vceilinichego[.]ru/api/machine/init`	`RILIDE C URL`
`hxxps://vceilinichego[.]ru/api/machine/set-tasks`	`RILIDE C URL`
`hxxps://vceilinichego[.]ru/api/machine/set-urls`	`RILIDE C URL`
`hxxps://146.70.79[.]75/templates/light.dotm`	`Template Open Directory URL`
`hxxps://146.70.79[.]75/templates/light.pub`	`Template Open Directory URL`
`hxxps://146.70.79[.]75/templates/light.xlsm`	`Template Open Directory URL`
`hxxps://146.70.79[.]75/templates/x64.xll`	`Template Open Directory URL`
`hxxps://146.70.79[.]75/templates/x86.xll`	`Template Open Directory URL`

Posted in Threat Intelligence—Security & Identity

Related articles

Threat Intelligence

Hybrid Russian Espionage and Influence Campaign Aims to Compromise Ukrainian Military Recruits and Deliver Anti-Mobilization Narratives

By Google Threat Intelligence Group • 10-minute read

Threat Intelligence

Investigating FortiManager Zero-Day Exploitation (CVE-2024-47575)

By Mandiant • 19-minute read

Threat Intelligence

How Low Can You Go? An Analysis of 2023 Time-to-Exploit Trends

By Mandiant • 10-minute read

Threat Intelligence

capa Explorer Web: A Web-Based Tool for Program Capability Analysis

By Mandiant • 6-minute read