Revisiting BatLoader C2 structure

medium.com/walmartglobaltech/revisiting-batloader-c2-structure-52f46ff9893a

Jason Reaves April 15, 2022

Jason Reaves

Apr 15, 2022

.

2 min read

By: Jason Reaves and Joshua Platt

BatLoader, named by Mandiant[7], is an interesting distribution/loading system that has been discussed previously[1,2] and leveraged by various actors. Recent media headlines show the connection to Zloader[3] after a disruption was done by multiple organizations[4] but this is not a Zloader exclusive service.

Reports of the disruption included brief mentions of BatLoader in their reporting. We assume that was not the target of their recent disruption campaign because the service is still functioning. While looking into new BatLoader campaigns, we noticed they changed the C2 structure of the loading process since our last blog[1]:

Ref: Virustotal.com
The structure looks similar to what we reported previously, but some of the data now looks like a hash value.

Previously:

```
/processingSetRequestBat1/?servername=/processingSetRequestBat2/?
servername=/processingSetRequestBat3/?servername=/processingSetRequestBat4/?
servername=/processingSetRequestBat5/?servername=/processingSetRequestBat6/?
servername=/processingSetRequestBot/?servername=/processingSetRequestCoba/?
servername=/processingSetRequestDownload/?servername=/processingSetRequestAtera/?
servername=
```

So these hash values aren't just the hash of the processing strings right?

```
>>>
hashlib.md5('processingSetRequestBat1').hexdigest()'e6a5614c379561c94004c531781ee1c5'
```

Ah, so they are. Then we can create new suricata rules based on the new patterns. However, I noticed if the hash starts with a number then the developer changes it to a character, for example:

```
>>>
hashlib.md5('processingSetRequestBat3').hexdigest()'73874ddb552a5b45cade5a2700d15587'
```

The hash used in traffic patterns however is:

```
a3874ddb552a5b45cade5a2700d15587
```

Going by the other Bat requests it appears they are going in order; a,b,c,d… So we can continue mapping hash values to new request structure:

```
/e6a5614c379561c94004c531781ee1c5/?servername=/f69af5bc8498d0ebeb37b801d450c046/?
servername=/a3874ddb552a5b45cade5a2700d15587/?
servername=/fa777fbbb8f055cb8bfcba6cb41c62e7/?
servername=/b1eeec75ef1488e2484b14c8fd46ddce/?
servername=/c003996958c731652178c7113ad768b7/?
servername=/d2ef590c0310838490561a205469713d/?
servername=/fa0a24aafe050500595b1df4153a17fb/?
servername=/i850c923db452d4556a2c46125e7b6f2/?
servername=/b5e6ec2584da24e2401f9bc14a08dedf/?
servername=/e747834ae24a1a43e044ea7b070048f0/?servername=
```

With the addition of deploying a stealer this maps like:

Most of the reporting has also shown that the templating for the msi installers has been Zoom and Teamviewer based fake installers, however as we have previously mentioned there are many affiliates involved in this service. A more exhaustive list of fake software templates for the initial MSI files can be found below:

```
zoomteamviewerAnyDeskTelegramyoutubecheatsccleanerdiscordthunderbirdluminaradobe
readerchromefirefoxbravegeminigrammarlyquickenrobinhoodamazonsmbcfidelitylogmein
```

References:

1: https://medium.com/walmartglobaltech/signed-dll-campaigns-as-a-service-7760ac676489

2: https://news.sophos.com/en-us/2022/01/19/zloader-installs-remote-access-backdoors-and-delivers-cobalt-strike/

3: https://www.microsoft.com/security/blog/2022/04/13/dismantling-zloader-how-malicious-ads-led-to-disabled-security-tools-and-ransomware/

4: https://blogs.microsoft.com/on-the-issues/2022/04/13/zloader-botnet-disrupted-malware-ukraine/

5: https://www.welivesecurity.com/2022/04/13/eset-takes-part-global-operation-disrupt-zloader-botnets/

6: https://decoded.avast.io/vladimirmartyanov/zloader-the-silent-night/

7: https://www.mandiant.com/resources/seo-poisoning-batloader-atera