Scott Conway

Information Security Researcher

Remote Code Execution in Securonix Snypr (CVE-2022-37108)

Aug 29, 2022

Over the course of the last six months or so, I've had the distinct pleasure of working with Securonix Snypr, an on-prem or cloud-hosted SIEM solution. I find SIEMs to be fairly boring, so after some time acting as a system administrator for the platform, I started pentesting it - and it didn't take long to find some issues. In this post, I'll discuss a remote code execution vulnerability that I found in the console's syslog-ng configuration settings. But first, a brief overview on how Securonix functions in a typical cloud-hosted environment.

Remote Ingesters (RINs)

RINs are hosts that are tasked with retrieving logs and shipping them to the cloud-hosted (or on-prem) application back-end for collection and processing. They may be controlled by the customer or Securonix. For example, the deployment I used had an on-prem RIN for local network log collection, and a Securonix-managed cloud RIN for capturing logs from Internet-facing cloud services. RINs can collect logs through syslog-ng, custom collectors written by Securonix, or by any user-created method that populates files on the RIN's disk.

The Console

This is the web application that you log into to manage the application. Once properly paired, the console will have root access to all RINs for the purposes of managing their syslog-ng configurations, and more. I'm sure that won't cause any issues.

The Vulnerability

In the Securonix console, any user with the "Manage Ingesters" permission is allowed to define syslog-ng source statements for any RIN under Administration -> Settings -> Manage Ingesters. The "Create/Edit Source" action for each RIN allows the user to enter pre-formatted text directly into the RIN's syslog-ng configuration file. That's right - it's not a wizard, it just copies text into the file. When I say "pre-formatted", I simply mean that the user separates the name and expression for the syslog-ng source. When submitting a name and expression, the console will simply put the raw text after a "source" statement.

eg. from the following input:

```
Source name: TCP_514
Source expression: {tcp(port(514));};
```

you get this output in your syslog-ng configuration file:

```
Source TCP_514 {tcp(port(514));};
```

Ok, simple enough. But what else can you do in syslog-ng configurations? To log something with syslog-ng, you need to define a source and destination, and join the two with a log statement. Securonix typically takes care of setting up destination and log statements, but we can write our own in the source expression field since we're just entering arbitrary text, after all.

There are quite a lot of options for source and destination statements. To get RCE, we're simply concerned with putting arbitrary text *somewhere* on disk. For a source statement, you can pick whatever you want as long as it'll be appended to at some point, so that syslog-ng will log to the destination. For example, `file("/var/log/syslog")` would likely be a fine source to use here. For the destination, we want to write attacker-controlled text to *some* file. To define the attacker-controlled text, we can simply write a string in the destination with the `template` function. eg. `file("/path/to/out_file" template("\narbitrary text\n")`. While we're here, we can override whatever options Securonix sets by default for file ownership, just so we don't mess with the output file's pre-existing permissions (if we're appending to an existing file). Here's what we have so far:

```
{file("/var/log/syslog");}; destination test_dest
{file("/path/to/out_file" template("\narbitrary text\n"));};
options { owner("root"); group("root"); perm(0600);};
```

Now, how can we get from arbitrary text to executable code? You could append to a user's .bashrc or .profile, but there are a number of considerations here. First, you don't know *when* the user will log in, if ever. Second, the shell script you're appending to (provided it exists) may have an exit (or similar) statement above it, causing appended lines to never be executed. However, since syslog-ng runs as root, we have access to all users' crontab files. A simple destination of `file("/var/spool/cron/root" template("\n* * * * * arbitrary code\n"))` will do the trick quite nicely. This destination should get the quickest results, but since we can append to any file, options for arbitrary code execution are endless. If you really wanted to, you could even write to `rc.local`, and then force a reboot by writing to `/proc/sys/kernel/sysrq` and `/proc/sysrq-trigger`.

Demo

In this demo, I specify a non-existent file (`/source.log`) in the source statement simply to control the timing of the output being written. You could easily select a high-traffic system log, such as `/var/log/syslog`, but I only wanted to get *one* instance of my arbitrary code to the output. Each new line in the source file will output the templated text to the destination file, which could get quite annoying.

In case you're curious, this is the source expression used in the demo:

```
{file("/source.log");}; destination test_dest
{file("/var/spool/cron/root" template("\n#* * * * * arbitrary
code\n"));}; log {source(vuln_test); destination (test_dest);};
options { owner("root"); group("root"); perm(0600);};
```

0:00

Potential Impact

When exploited, this vulnerability allows an attacker with the “Manage Ingesters” permission on a Securonix instance to execute arbitrary code as root on all RINs paired to the instance’s console. As noted above, Securonix typically grants access to cloud-hosted RINs in cloud deployments. With this in mind, an attacker could possibly gain a foothold in Securonix’ cloud environment. The same is true for on-prem RINs.

The Patch

As noted in the below timeline, this vulnerability was patched by Securonix sometime before July 2022, and at the latest in SNYPR version 6.4 Jun 2022 R3_[06170871]. Their patch notes don’t identify this fix, which is odd, since they seemed alright with me publishing a CVE about the vulnerability. Although I was consulted during the patching effort, I’m not privy to the “solution”. Yet, when attempting to use a malicious expression, the console UI simply states “complex source

Yet, when attempting to use a malicious expression, the console UI simply states "complex source expressions are not allowed" and does not save the attempted source expression to the RIN. This may interfere with legitimate source expressions, as I can definitely see power-users doing a *lot* in source statements.

Closing

I must clarify that this vulnerability does not allow for RCE on the console itself, as you cannot write syslog-ng configuration rules for the console.

Disclosure Timeline

- 2022-04-14 Disclosure sent to Securonix
- 2022-06-??: Patched Snypr version (specific version unknown) released
- 2022-07-25: Applied for CVE-2022-37108
- 2022-08-29: Vulnerability made public

References

https://nvd.nist.gov/vuln/detail/CVE-2022-37108

< Prev: Reverse Engineering ShopGoodwill for ... Next: Proactively Monitoring TLS Certificate D...