# Attacking the Attackers

6 minute read

The predator becomes the prey. When scanning with Metasploit Pro, your victim can counter with a XSS payload, and even take over your machine. **Never trust your victim!**

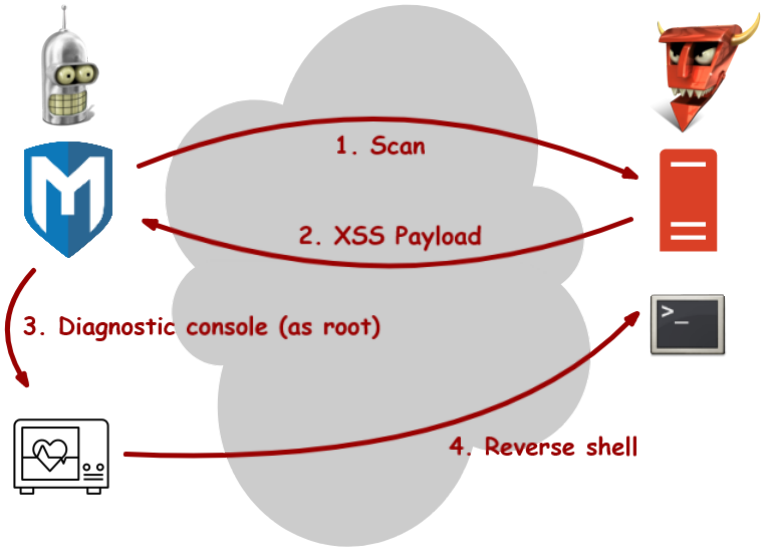
*"Clever girl!" - Robert Muldoon*

UPDATE: our paper "Never Trust Your Victim: Weaponizing Vulnerabilities in Security Scanners" has been accepted at RAID 2020 (https://raid2020.org/)! Check out the full paper here (https://www.researchgate.net/publication/344642774_Never_Trust_Your_Victim_Weaponizing_Vulnerabilities_in_Security_Scanners).

# Metasploit Pro - XSS to RCE

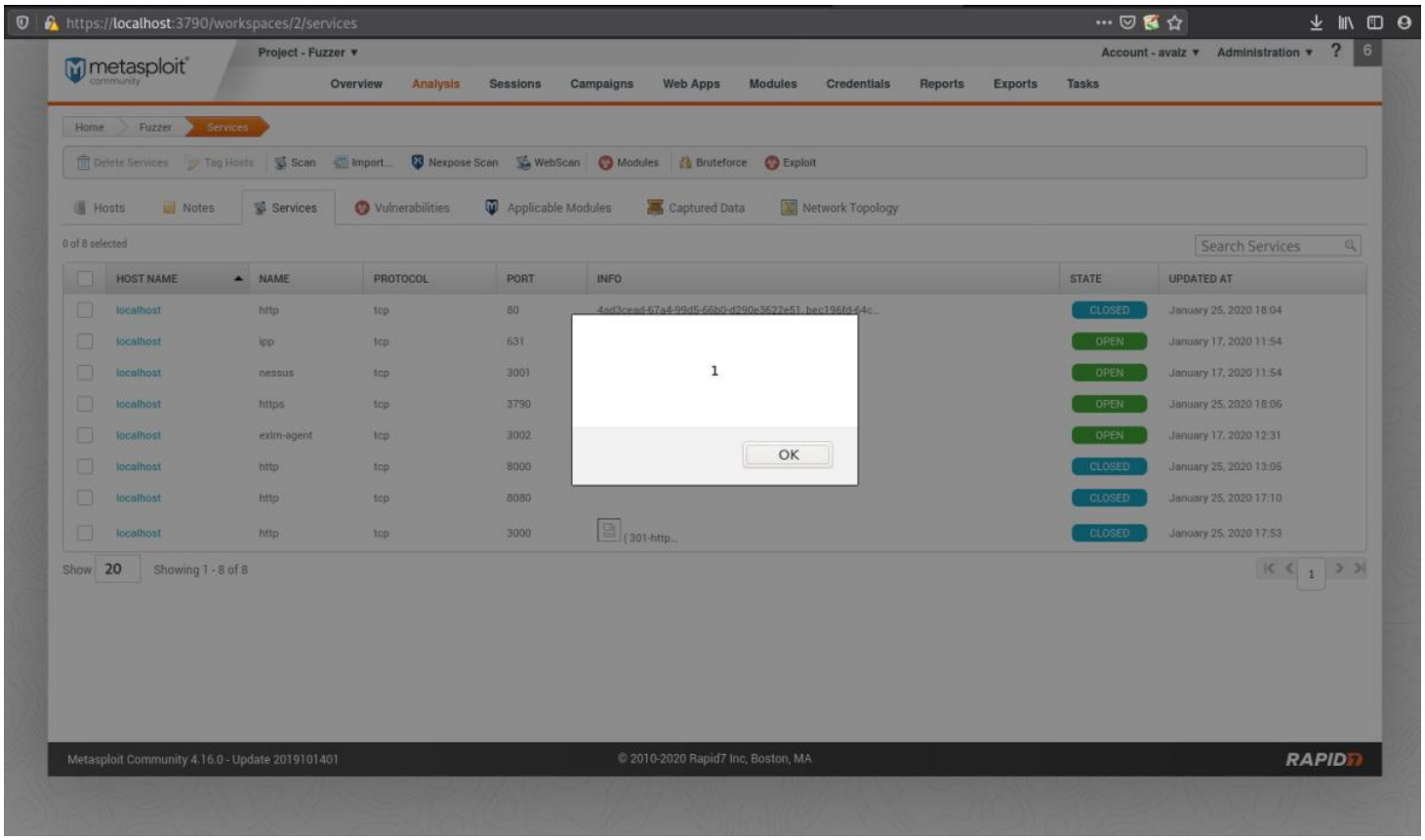
We see the targets of our scan as passive entities, and this leads to underestimating the risk of performing a network scan. However, the tools we use to scan are not immune to vulnerabilities.

A prominent example of this problem is a Metasploit Pro vulnerability [1] that I discovered . When you scan a remote host with Metasploit Pro, your target can counter-attack you with a Cross-Site Scripting (XSS) payload. Using this in conjunction with other weaknesses, such as the Metasploit Pro diagnostic console, the attacker can run commands on your machine as root.




Thanks Gabriele for this great illustration.

This vulnerability affects Metasploit Pro up to version 4.17.0. It was remediated on May 14, 2020 with patch 4.17.1 (https://help.rapid7.com/metasploit/release-notes/archive/2020/05/#20200514) [2].




PoC XSS on Metasploit Pro 4.17.0

# Metasploit Pro

Metasploit Pro is a popular, on-premise penetration testing software developed by Rapid7 (https://www.rapid7.com/).

It comes with all the standard metasploit framework (https://github.com/rapid7/metasploit-framework) features, plus a Web user interface that interacts with a local Ruby on Rails REST API server.

## Scanning a target

Metasploit Pro provides users with automatic information gathering from a target host. This step is usually considered not risky during an engagement, both for the attacker and for the target.

Under the hood, it uses Nmap (https://nmap.org/) to perform a port scan and detect what services are running on the target host. When the scan is over, Metasploit Pro displays all the information it gathered.

It mainly displays services in these pages:

- `/hosts/:id`
- `/workspaces/:id/services`

These pages contain a table of services, with columns for the host name, the service type, protocol, port, and an *INFO* column.

But how does Nmap know how to fill the *INFO* column?

## Nmap fingerprinting

One may think than Nmap has a *huge* set of rules that say "If I see this value, that means that the target is running nginx."

In a sense, that is correct.

Nmap comes with a file, *nmap-service-probes (https://svn.nmap.org/nmap/nmap-service-probes)* [3], containing a list of rules to identify the target host. It first sends a *Probe*, then reads the response using a regular expression.

Consider this minimalistic HTTP response:

```
HTTP/1.1 200 OK
Server: nginx/1.17.0

<html>...</html>
```

And this is a simple rule to detect the server version, using the *nmap-service-probes* format:

```
match http m|^HTTP/1\.[01] \d\d\d.*\r\nServer: (.*)\r\n| v/$1/
```

The *m* block contains the `(.*)`, which is a regex for "any string", in the `Server` HTTP response header. What about the parentheses? They define *capture groups*, and they allow to store what was found by the wildcard in specific registers, such as `$1`, `$2`, etc. This registers are used in the *v* field: `v/$1/`

In the case of our minimalistic HTTP response, `$1` (and hence *v*), is `nginx/1.17.0`.

You can find more detail on *nmap-service-probes* file format in the Nmap online documentation (https://nmap.org/book/vscan-fileformat.html#vscan-db-match).

# Stored XSS

Capture groups make Nmap fingerprinting leaner, without having to write a rule for each server version. However, it also means that the value of `Server` comes directly from the response of your target.

Remember that this value is printed in the Web UI of Metasploit Pro.

If your target is malicious, it can insert whatever it wants in that field, for example a XSS payload.

```
HTTP/1.1 200 OK
Server: <script>alert(1)</script>

<html>...</html>
```

This time, the detected server version using our detection rule becomes `<script>alert(1)</script>`. When this value is displayed on the analyst console, it should display a browser popup.

However, when I first injected this payload, no popup appeared. Yet, all characters were printed on the screen.

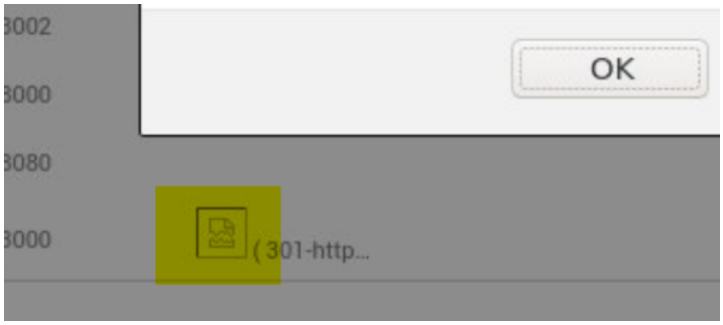
I realized that the value was not inserted on the page, but it was loaded asynchronously using jQuery. This way, the JavaScript payload is not executed because the browser does not render dynamically loaded script tags.

However, script tags are not the only way to inject JavaScript content in a page.

Since the application renders HTML elements, we can insert an `anchor` element with `<a onmouseover='alert(1)'</a>`, which registers a *mouseover* event and triggers it when the mouse passes over it. This works, but it relies on additional user interaction.

The final injection payload I chose was `<img src='x' onerror='alert(1)'/>`. It registers an *error* event on the `image` element. When the browser tries to load the `x` URL, it triggers an error that displays the content of the `onerror` attribute.

Notice the broken image on the INFO column where the server version should be.




(There are many other similar payloads, for example `<input onfocus='alert(1)' autofocus/>`.)

We have a foothold on the browser that lets us run arbitrary JavaScript code on the analyst browser, in their authenticated session. We could also deploy a BeEF (https://beefproject.com/) hook to interact with the browser.

Now what?

# Remote Command Execution

Being an on-premise software, the backend runs on the same host as the Web UI. A weakness that lets users run OS commands from the Web UI means those commands will run on the analyst machine.

One of such weaknesses is the ***diagnostic console*** *(https://www.exploit-db.com/exploits/40415)*, an embedded terminal that allows analysts to run commands directly on the host without using the Web interface.

In normal circumstances, attackers can't reach the console for the following reasons:

- The Web UI requires a login.
- The Web UI only listens to requests coming from localhost (and analysts would not expose their Metasploit instance anyway).
- The diagnostic console is disabled by default in Metasploit installations.

We automatically bypass the first two issues using Stored XSS: all requests come directly from the analyst browser as if they were issued by a normal active session.

For the third issue, although the diagnostic console is disabled by default, we can activate it directly from JavaScript. The activation is triggered by sending an Ajax request to the `/settings/update_profile` endpoint with the parameter `allow_console_access=1`.

Now we can run arbitrary commands on the system.

Also notice that Metasploit Pro documentation (https://metasploit.help.rapid7.com/docs/metasploit-web-interface-overview) clearly states that "Metasploit Pro Users Run as Root. If you log in to the Metasploit Pro Web UI, you can effectively run any command on the host machine as root." This means that commands on the diagnostic console are run as *root* by the system.

# Conclusions

Never assume that a scan target is benign and passive.

If your scanning platform is vulnerable, an attacker can counter-attack you. In this case, an attacker can run commands as root on your machine when you scan them.

When displaying any information gathered from external sources, it is important to check if an attacker can have control over them, and treat them accordingly, for example by sanitizing it.

> *If you gaze long into an abyss, the abyss will gaze back into you.*
>
> *Friedrich Nietzsche*

# Acknowledgements

This vulnerability was discovered within a research project in collaboration with Gabriele Costa (https://www.imtlucca.it/it/gabriele.costa) (IMT School for Advanced Studies Lucca (https://www.imtlucca.it)) and Alessandro Armando (https://csec.it/people/alessandro_armando/) (University of Genoa (https://unige.it)). Thanks to Giovanni Minotti (https://github.com/Giotino) who helped me complete the kill chain for this attack.

I would also like to thank the Rapid7 (https://www.rapid7.com/) support team, that fixed the vulnerability and publicly thanked us in their release notes for the patched version.

# References

1. CVE-2020-7354 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=2020-7354) and CVE-2020-7355 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=2020-7355)
2. Metasploit Pro 4.17.1 - Release Notes (https://help.rapid7.com/metasploit/release-notes/archive/2020/05/#20200514)
3. Nmap - Service and Application Version Detection (https://nmap.org/book/vscan-fileformat.html)
4. Never Trust Your Victim: Weaponizing Vulnerabilities in Security Scanners, RAID 2020 (https://www.researchgate.net/publication/344642774_Never_Trust_Your_Victim_Weaponizing_Vulnerabilities_in_Security_Scanners)

**Tags:** metasploit rce vulnerability websec writeup xss

**Categories:** Research

**Updated:** May 21, 2020