**NAME: MURITHI BRONCH MUKAMI & EMMANUEL KOECH**

DATE:26/8/2022

Cyber Security Technologies &Research Systems

**REFLECTED FILE DOWNLOAD(RFD)**

Reflected File Download (RFD) is a web attack vector that enables attackers to gain complete control over a victim’s machine. In an RFD attack, the user follows a malicious link to a trusted domain resulting in a file download from that domain. Once executed, the attacker can execute commands on the Operating System level of the client’s computer.

* 1. **RFD Attack Flow**

RFD, like many other Web attacks, begins by sending a malicious link to a victim. But like no others, RFD ends outside of the browser context:

1) The user follows a malicious link to a trusted Web site.

2) An executable file is downloaded and saved on the user’s machine. All security indicators show that the file was “hosted” on the trusted Web site.

3) The user executes the file which contains shell commands that gain complete control over the computer.



**Figure 1 – The three steps attack flow of reflected file download**

**1.2. Implications**

Attackers can use reflected file download in order to launch various attacks on users:

1) Gain complete control over the user's machine - steal data and perform actions by executing windows operating system commands and scripts. Such commands can install various types of malwares as well as take immediate and complete control over the compromised machine.

2) Gain complete control over the Chrome browser including encrypted connections – the ability to execute operating system commands enables the attacker to abuse command line arguments which are not accessible otherwise. By doing so, attackers can disable the browser's security, steal all of the information from existing sessions (including session cookies and stored passwords), access any website and impersonate the user on it.

3) Exploit vulnerabilities on installed software – attackers might choose to attack an installed software by downloading a file associated with the vulnerable software.

**1.3. RFD Requirements**

For an RFD attack to be successful, there are three simple requirements:

1) **Reflected** – some user input is being “reflected” to the response content. This is used to inject shell commands.

2) **Filename** – the URL of the vulnerable site or API is permissive, and accepts additional input. This is often the case, and is used by attackers to set the extension of the file to an executable extension.

3) **Download** – the response is being downloaded and a file is created “on-the-fly” by the Web browser. The browser then sets the filename from (2).

For each of the above requirements, I have dedicated a special section in this white paper in order to help detect and exploit RFD issues with high proficiency.

**2. Detecting RFD**

As described in the RFD Requirements section, a Web site or an API should meet three simple requirements in order to be vulnerable to Reflected File Download. Detecting RFD issues, simply means to check if these requirements can be met.

**2.1. Looking for Reflected Input**

Having the ability to control some of the content that is returned by the server in the response body is crucial for an RFD exploit to be successful. This is where the attack payload is inserted - the actual content or commands that inflict damage to the client's machine. The most common locations of user-controlled input that are reflected into the response content:

1) Request Parameters – if the page or API accepts user input, there is always a good chance that this input is going to be reflected back into the response.

2) Errors – a common bad practice is to return and display inputs that caused an error. In other words, specifying the reason for the error. Tampering with request parameters, and even the request URL and API paths resulting in reflection of the erroneous input is sufficient for a successful RFD exploit.

3) Persistent Storage – in some cases the attacker-controlled input is fetched from the application's persistent storage (Database, Files, etc.). In the majority of such cases, an "id" parameter or URL path is assigned to the stored content. Assembling a URL that fetches this information by providing the correct "id", is mostly sufficient for an RFD exploit.

4) JSONP Callbacks – by definition a JSONP callback is reflected back into the response. While this is often a more limited injection point that forbids special characters, it remains usable for some RFD payloads.



**2.2. Controlling the Filename**

In an RFD attack, we are changing the context in which the response is processed, giving the injected input a new and malicious meaning. The favorite context for all sane attackers, is one that allows executing commands on the target's machine.

However, all warnings are dismissed if one of the following strings appear in the filename:

* Install
* Setup
* Update
* Uninst

**2.3 File Extensions**

In case a scripting language interpreter is installed on the victim’s machine, the associated extensions could be used. The following table shows extensions that are used when creating the file:

|  |  |  |
| --- | --- | --- |
| **Name** | **Extensions** | **Associated Program** |
| Windows Batch Files | .bat, .cmd | cmd.exe |
| Windows Script Host | .js, .vbs, .jse, .vbe | wscript.exe (cscript.exe) |
| Windows Script Host | .wsf, .wsh | wscript.exe\* XML/INI |
| HTML Application | .hta | mshta.exe |

**SET UP**

The prerequisites for RFD attack:

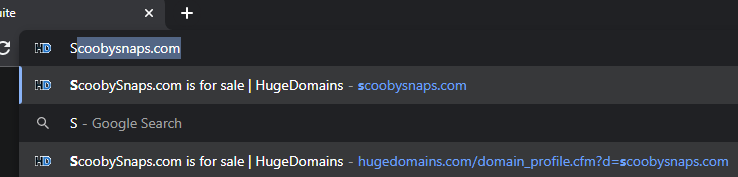
* Parrot OS 2022 (Attacker)
* Burp Suite (pre-installed) in Parrot OS
* Bettercap tool-It is used for [WiFi](https://www.bettercap.org/modules/wifi/), [Bluetooth Low Energy](https://www.bettercap.org/modules/ble/), wireless [HID hijacking](https://www.bettercap.org/modules/hid/) and [IPv4 and IPv6](https://www.bettercap.org/modules/ethernet) networks reconnaissance and MITM attacks.
* Attacker and Victim need to be in the same WIFI network connection
* Windows 7 OS (Victim)
* Chrome, Firefox, Opera version 14 and before and Internet Explorer version (8-9).
* A website that uses HTTP

**OBJECTIVES OF BURP SUITE**

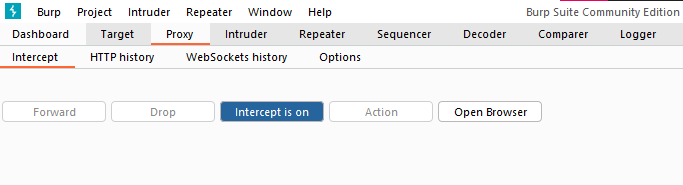
1. Intercept HTTP request
2. Modify HTTP request
3. File download and installation

**STEPS TAKEN IN BURP SUITE**

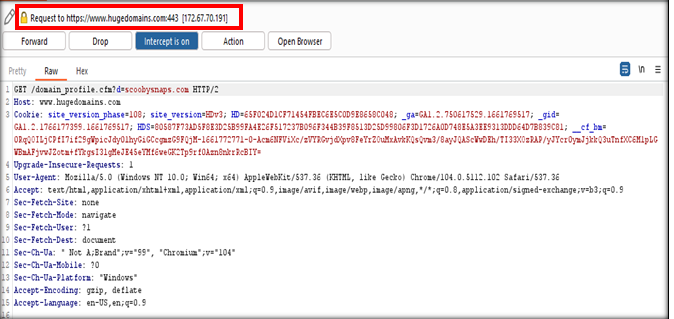
* The user (victim) searches for a website in this case its “Scoobysnaps.com”



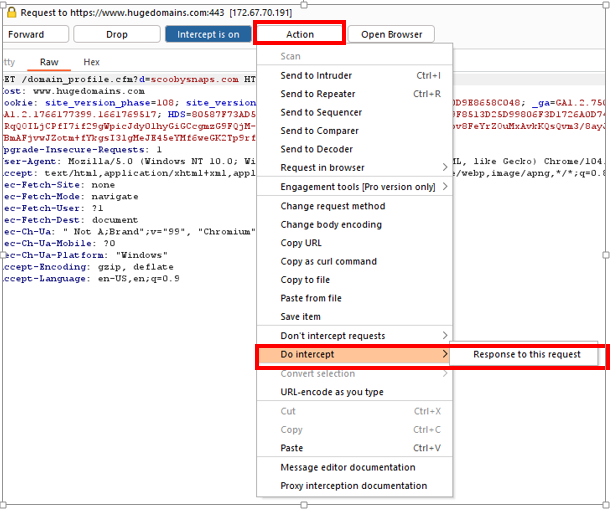
* The attacker lauches Burpe Suite to intercept the users (victims) request. Switches the intercept on



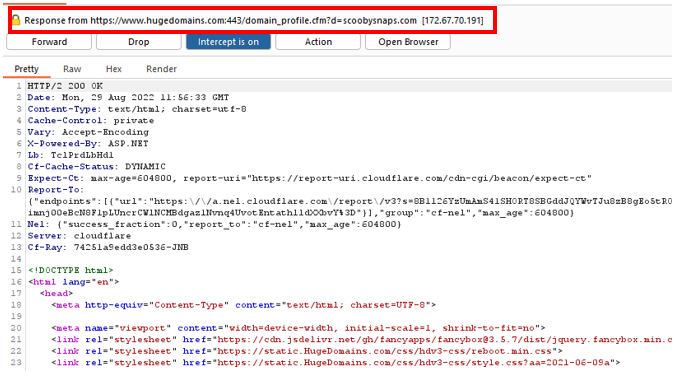
* Immediately the user (victim) sends the request, it is intercepted by the attacker



* The attacker intercepts the response of the request since that is where the payload will be placed
* Accesses the option in the drop down menu after clicking “Action” button
* Then forwards the request by clicking the “Forward” button



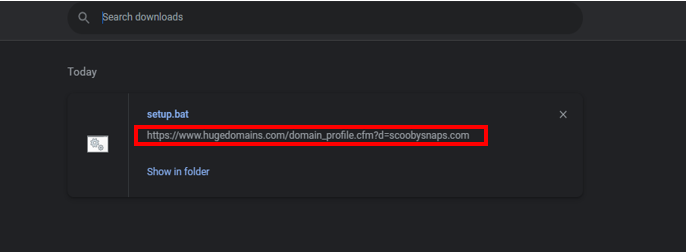
* The response comes to the attacker first. This enables the attacker to insert the payload in the response



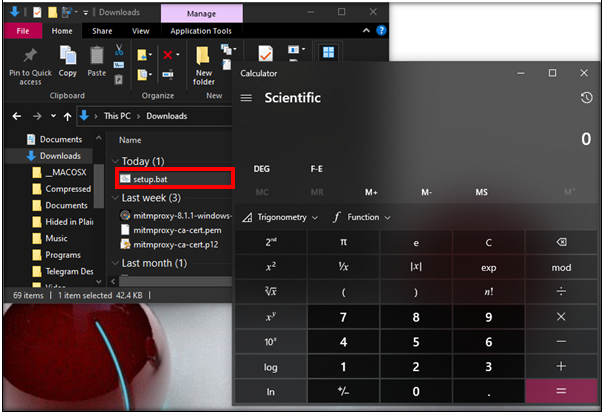
* The attacker places the payload and forwards the response the the user (victim)



* The file automatically downloads with the name given in the payload and displays the trusted domain expected by the user(victim).



* When the user (victim) clicks on the file, it executes the commands that the attacker placed in the payload. In this case, it opens the calculator.



**OBJECTIVES OF BETTERCAP**

1. Detect user’s web traffic
2. Redirect user to a different domain.

**STEPS TAKEN IN BETTERCAP**

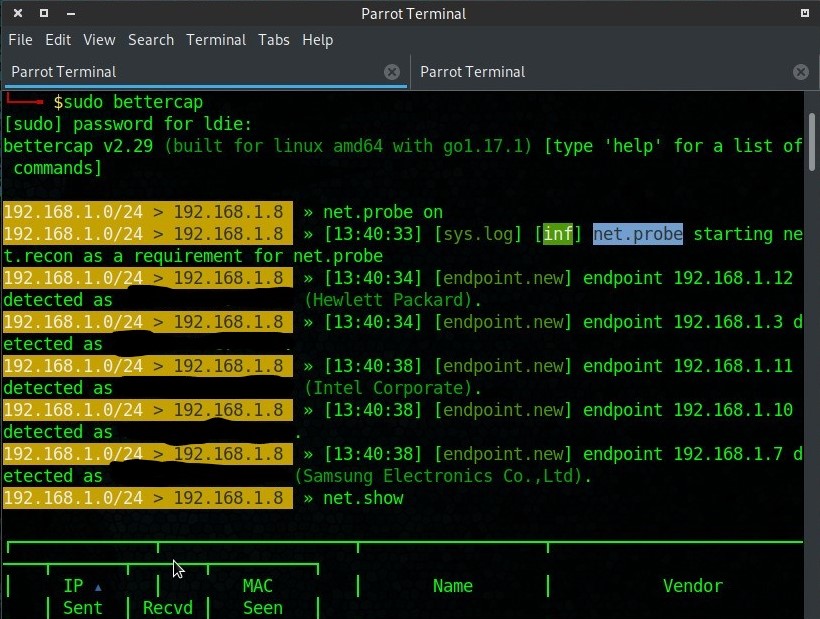
1. Install Bettercap .
2. Run the command to start bettercap:

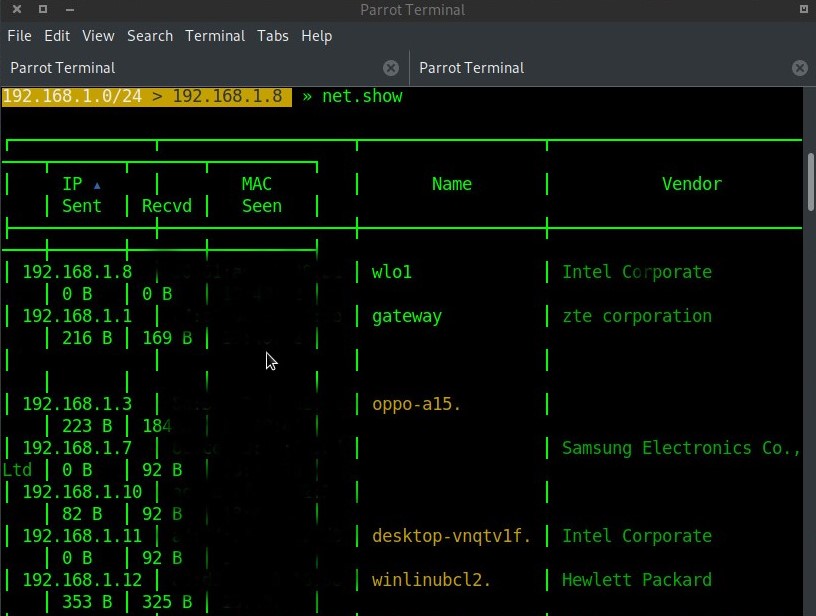
*sudo bettercap*

1. To view devices that are connected to your network and identify their name, IP and MAC addresses, run:

*net.probe on*

*net.show*





1. To execute the attack to one particular device, select the device’s IP address and set it as the target using the command:

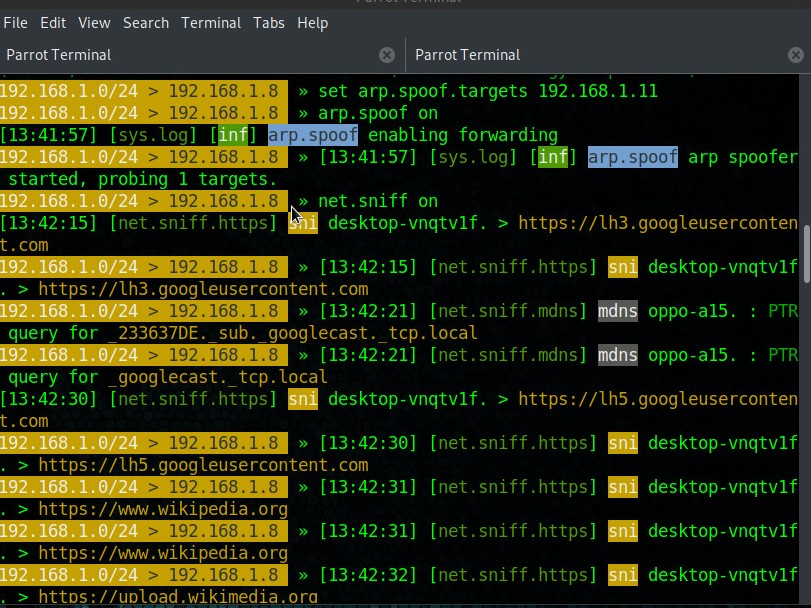
*set.arp.spoof.targets IP address*

1. Turn on the arp poisoning:

*arp.spoof on*

1. To start spying on the user’s web traffic and background activity, start sniffing:

*net.sniff on*



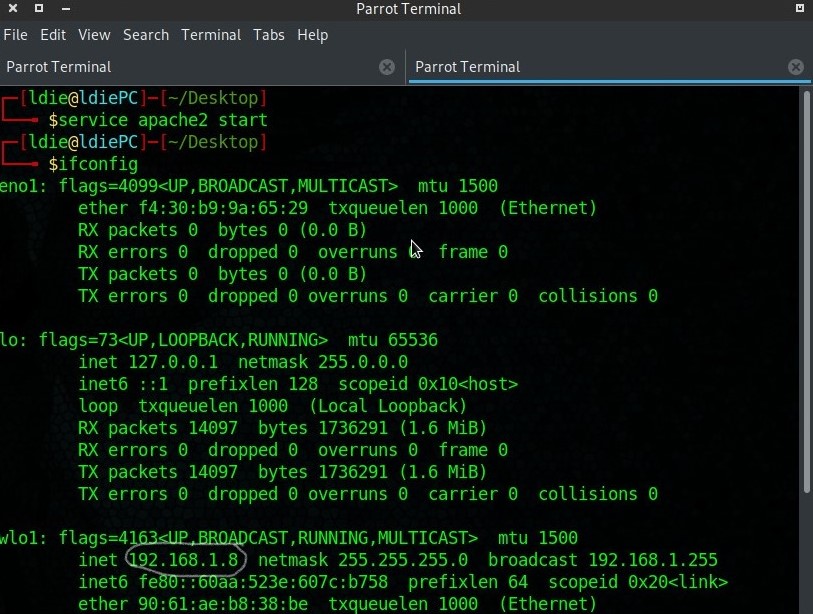
1. To redirect the victim to the attacker’s website:

*set dns.spoof.domains domain name*

1. To start the attacker’s website (default apache2 website), start apache2 and identify the attacker’s IP address

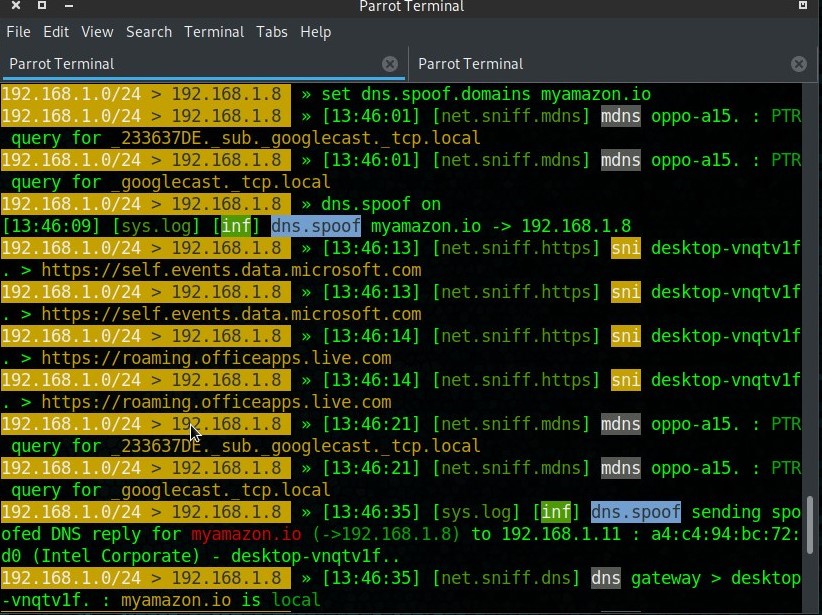
*service apache2 start*

*ifconfig*

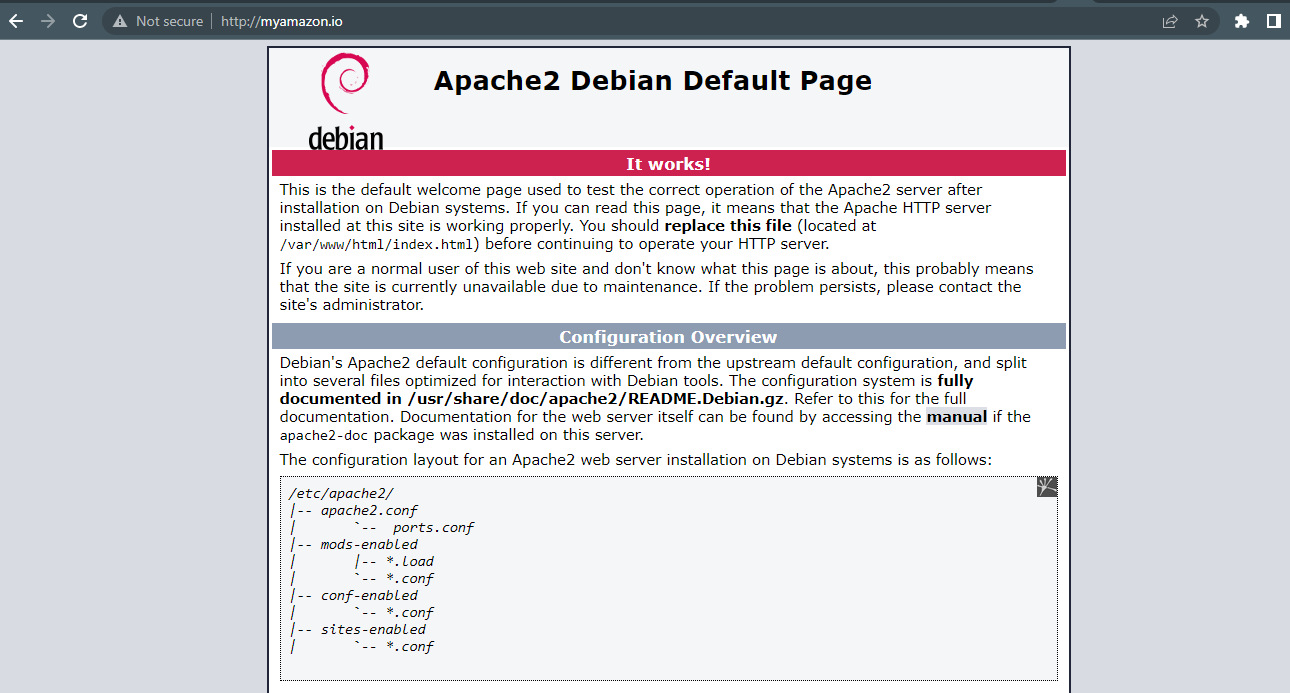


1. Then turn on dns spoofing:

*dns.spoof on*



The victim is redirected to the attacker’s website where the file is downloaded.



**HOW TO MITIGATE RFD**

There are several ways to mitigate RFD so that the user is not susceptible to any more RFD attacks. They are:

1. Use exact URL mapping – when mapping APIs, Servlet and web pages and when writing rewrite rules, make sure that hackers cannot enter additional characters after the resource name. Any additional characters in the URL should result in a 404 error.
2. Do not escape! Encode! – Escaping such as Backslash escaping always contains the problematic character. If escaping is used double quotes (“) turn into (\”). Prefer encoding of user-controlled input. In JavaScript for example, double quotes (“) turn into (\x22) or (\u0022) which is a lot safer.
3. Whitelist Callbacks – there has been numerous attacks that abuse JSONP Callbacks. If you think about it, you might not really need the Callback to be completely dynamic.
4. Require custom headers – as mentioned before, there is no reason for a user to access APIs directly. By requiring a Custom HTTP Header for all API calls, one can harness the power of Same-Origin-Policy on the client side. This makes RFD unexploitable unless another vulnerability is involved.
5. If possible, require CSRF tokens – by doing so hackers won’t be able to build a working RFD link and sent it to their victims.
6. Never include user input in API usage errors – when it comes to JSON/JSONP, code is accessing code. Usage errors are rare, and when occur should be logged. The response should not include the erroneous input but rather a reference number that can be tracked down for troubleshooting.
7. Remove support for Path parameters – If you don’t really use it then lose it. Most developers are not even aware of the existence of Path Parameters, and I found this section of the URL extremely vulnerable to various attacks (including XSS).
8. Add X-Content-Type-Options headers – if the resource is responding with a text/plain or unknown content-type, attackers can make browser “guess” that the file is binary and required download (meeting the third RFD requirement). The following header can prevent this from happening in some browsers:

*X-Content-Type-Options: nosniff*

1. Use a secure VPN (Virtual Private Network) that encrypts your internet traffic on unsecured networks to protect your online identity, hide your IP address, and shield your online data from third parties. It sends user’s internet data through a secure virtual tunnel to minimize the possibility of anyone tracking what they do online. When the user connects to a VPN service, it authenticates the client with a VPN server and applies an encryption protocol to all your internet data. The VPN service then creates an encrypted “tunnel” over the internet. That secures the data traveling between them and their destination.

**References**

Oren Hafif. (2014, November 6). *Reflected File Download a New Web Attack Vector.*

<https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/reflected-file-download-a-new-web-attack-vector/>

<http://dl.packetstormsecurity.net/papers/attack/Aspect_File_Download_Injection.pdf>

Nour's tech talk. (2022, Aug 3). *Intercept and modify network traffic - man in the middle attack*

<https://www.youtube.com/watch?v=sDTy32H9Hww>

Ubuntu Maniac. (Feb 5, 2022). *Man in the Middle Attack MITM Using Wireshark and Ettercap | Full Tutorial for Beginner 2022.*

<https://www.youtube.com/watch?v=DElzWHWDG9Q>