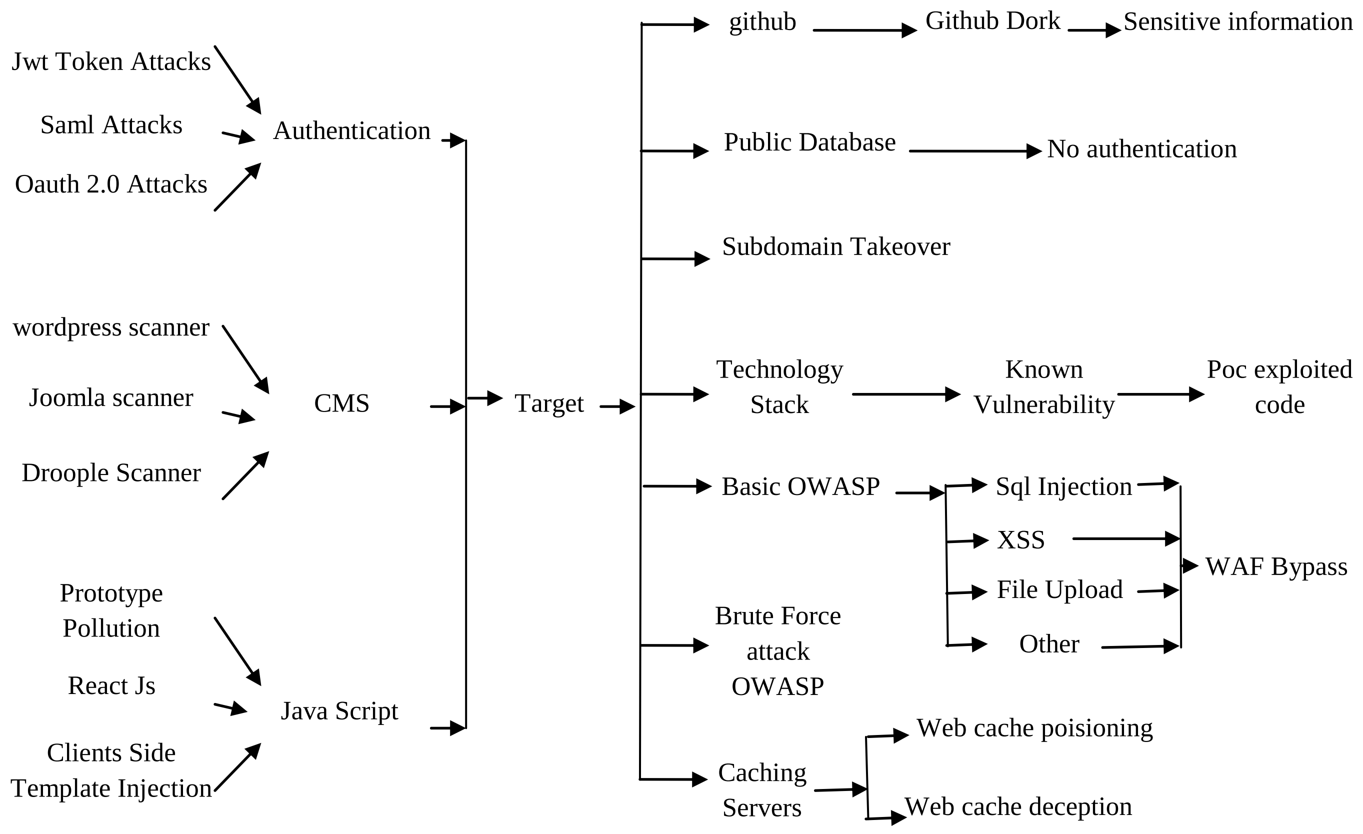
Introduction

In this version of the Bug Bounty methodology and techniques I use during the recon and fingerprinting phase of an engagement. As you probably know there are 3 main phases of a bug bounty engagement: reconnaissance, fingerprinting, and exploitation.

$$\begin{tikzcd}\text{Reconnaissance} \arrow[r] & \text{fingerprinting} \arrow[r] & \text{exploitation}\end{tikzcd}$$

The exploitation phase of a hunt is where all the true hacking occurs. Everything up until this stage is just prep work and now it's time to get busy.



Each target you go after will most l likely be utilizing different technology stacks so it's important that you know the vulnerabilities and mis-configuration impacting an array of technologies. For example having knowledge of Github is important when mining for hardcoded passwords and other sensitive information. If you don’t know what Github is how are you supposed to know the possible security failures companies can impose when using it? You need to have deep knowledge on a wide range of technologies. In addition to this you also need deep knowledge of web application vulnerabilities. The vast majority of a company's public facing assets are going to be web apps so it's vital that you know at the very l east the OWASP top 10. The more vulnerability you know how to exploit the better chances you have of finding one.

Note I won't be teaching you how to use tools, for the most part everything we do will be done manually so you can get a deep understanding of the process. Once you know how things work at a deep level you will want to replace some of your manual process with tools and automation.

Basic Hacking Known Vulnerabilities

Introduction

One of the first things you l earn i n hacker school i s 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.

$$\begin{tikzcd}
\text{Target} \arrow[r] & \begin{array}{c}\text{Discover endpoints technology}\\ \text{\& version numbers}\end{array} \arrow[r] & \begin{array}{c}\text{find proof of concept}\\ \text{AKA exploit Code}\end{array} \arrow[r] & \text{Exploit}
\end{tikzcd}$$

As shown above 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 exploited 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 i s 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 i s dropped we immediately start scanning and exploiting everything before the defenders have a chance to patch.

Identifying technologies

Introduction

When attempting to exploit a target with a known vulnerability you could just launch your exploit at every target and hope for the best or you can do things a little smarter. Identifying the target technology stack will help you find the exploits impacting that stack. Not knowing this information will l eave you blind and you will have to take random guesses at which exploits might work.

Wappalyzer

If you're attempting to discover the technologies running on a website the best place to start is wappalyzer. An alternative to wappalyzer is “https://builtwith.com/” but I personally like wappalyzer better.

I personally like to use the wappalyzer browser plugin as it makes it easy to determine an endpoints tech stack when browsing their website. As you can see in the image above this website is running “Ruby on

Rails", "Jquery 3.5.0", "and Backbone.js 1.4.0", and a few other things. Note that if you use a command line tool you can scan multiple websites at once; this is nice if you're trying to scan hundreds or thousands of sites at once.

Summary

You need to know the technology stack your target is running so you can find associated exploits. There are a few ways to determine the technologies an endpoint is running but I almost always use wappalyzer. If you can't determine this information with wappalyzer there is other techniques to find an endpoints technology stack.

Identifying the vulnerabilities

Introduction

You know what software your target is running but how do you determine what vulnerabilities it has? The whole point of l earning a target technology stack is so you can use this information to find associated vulnerabilities.

Google

When I'm looking to see what vulnerabilities a technology has the first place I go is Google. Actually, Google i s 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

There is all kinds of stuff here! I see SQL injection exploits, LFI exploits, and much more. I recommend you click on the first couple l inks to see what interesting vulnerabilities there are.

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/

You can use the website to search for things but I typically use the command line tool called searchsploit. You can download this tool from Github as shown below:

https://github.com/offensive-security/exploitdb

./searchsploit "name of technology"

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 i s no better place to search than NIST.

https://nvd.nist.gov/vuln/search

Searching for “Gila CMS” gives us 17 CVEs, the newer the CVE the better as there i s a better chance the target hasn't patched their systems yet. Note that just because you find a CVE doesn't mean you can exploit it. To exploit a CVE you need the proof of concept (POC) exploit code, without that you're stuck.

Summary

Locating the vulnerabilities impacting a technology stack i s relatively easy. All you really have to do is search for them. Between Google, ExploitDB, and NIST you should be able to find everything you're looking for.

Finding the POC

Introduction

You have identified that the target application contains vulnerabilities but to exploit them you need the proof of concept (POC) exploit code. If you don't have the exploit code you’re only other option is to make it yourself. However, this is beyond the scope of this book.

Github

One of the best places to find exploit code i s Github. Github i s an American multinational corporation that provides hosting for software development and version control using Git. It offers the distributed version control and source code management functionality of Git, plus its own features. Developers love Github and hackers do as well.

You can easily search for a CVE on Github as shown in the above image. If there is a POC you will most likely find it on here. However, BE AWARE OF FAKE POCs as these exploits are not vetted and come from untrusted third parties.

ExploitDB

I already mentioned ExploitDB earlier so I’m not going to talk about it again but this is a great resource for finding POCs.

https://www.exploit-db.com/

Summary

9 times out of 10 you are going to find the exploit code you're looking for on Github or on ExploitDB. If you can’t find it in one of those locations it probably doesn't exist and you will have to create your own POC. However, don’t be afraid to search for resources.

Exploitation

Once you have a working POC you are ready to test it against your target. I always recommend setting up a vulnerable machine to test the exploit against first so you know what to expect from a real target. Once you're ready just run the exploit on your target and review the results to see if they are vulnerable or not.

Conclusion

Exploiting known vulnerabilities is one of the oldest tricks in the book. That being said it’s still one of the best methodologies to use for quick easy wins. There are really only three steps when using this approach. First determine your targets tech stack, search for any vulnerabilities in that tech stack, and finally run the exploits.

Basic Hacking CMS

Introduction

Content management systems (CMS) such as wordpress, drupal, and joomla make up the vast majority of the internet. According to a survey performed by W3Techs 62% of the internet is run on a CMS and 39.1% percent of the internet i s run on wordpress. As an attacker this means the vast majority of the sites you are going to be going up against will be run by a CMS.

WordPress

As of right now over a quarter (25%) of the internet is built using WordPress. This is useful to know because that means a single exploit has the potential to impact a large portion of your target’s assets. There are in fact hundreds of exploits and mis-configurations impacting WordPress and its associated plug-in. One common tool to scan for these vulnerabilities is wpscan:

https://github.com/wpscanteam/wpscan

The only thing that’s annoying about this tool i s that it's written i n ruby, I prefer tools written in python or Golang. During the fingerprinting phase you should've discovered the technologies running on your target's assets so i t should be easy to search for sites running WordPress. Once you find a site scan i t with wpscan as shown below:

```
wpscan --URL <URL>
```

The vast majority of the sites you scan are going to be patched. This is because most of these WordPress sites are managed by third party vendors who perform automatic updates. However, you will run into vulnerable plugins quite frequently but many of these exploits require credentials to exploit. Another thing I find all the time i s directly listing on the uploads folder. Always make sure to check:

“/wp- content/uploads/”

You can often find sensitive information such as user emails, passwords, paid digital products, and much more.

Drupal

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

Joomla

WordPress is by far the most popular CMS with over 60% of the market share. Joomla comes in second so you can expect to run into this CMS as well. Unlike WordPress sites that seem to be fairly locked down Joomla is a mess. If you want to scan for vulnerabilities the most popular tool is Joomscan:

https://github.com/rezasp/joomscan

perl j oomscan.pl -u <URL Here>

Adobe AEM

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 aem-hacker:

https://github.com/0ang3el/aem-hacker

python aem_hacker.py -u <URL Here> --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.

Other

There are hundreds of different CMSs so it wouldn't be practical for me to mention every single one of them. The vast majority of sites are going to be running WordPress, Joomla, and Drupal but you still might run into other CMSs.

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/

For instance, if I discover a CMS named "Magento" I would perform the following search on exploit-db:
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.

As it turns out there is a Magento vulnerability scanner called magescan so we can just use that:

https://github.com/steverobbins/magescan

Make sure to use this process whenever you come across a CMS framework you don't recognize.

Conclusion

Over half of the internet is being run by a CMS framework. So, you are almost guaranteed to run into a CMS at one point or another. When you do find a CMS, you don't want to waste time manually testing the endpoint; you want to test for known CVEs and misconfigurations. The best way to do this is to find some sort of CMS specific vulnerability scanner. If you can find that you can try searching exploit-db and Google for known CVEs. If you still come up empty handed it's probably best to move on unless you're hunting for zero days.

Basic Hacking Github

Introduction

GitHub is a web-based version-control and collaboration platform for software developers and as of right now it's one of the easiest ways to compromise an organization. This is one of my go to techniques when I want an easy high impact finding.

Finding Sensitive Information

Pilliging github for sensitive information disclosures is one of the easiest ways to compromise an organization. It doesn't matter how hardened your external perimeter is if your developers are hard coding credentials and posting them online you're going to get compromised. It's fairly common for developers to hard code test accounts, API keys, or whatever when they are writing a piece of software. This makes things easy for the developer as they won't have to enter their credentials every time they go to run/test their program. However, more times than not these credentials remain in the source code when they push it to Github, if this repository is public everyone can view it. The first thing you need is a list of sensitive words to search on. This can be a file name, file extension, variable name, or anything else. A good list can be found below thanks to "@obheda12":

Once you have a list of sensitive things to search for your ready to hunt! I normally just type i n the domain of the target followed by the Github Dork as shown below:

Domain.com "password"

As you can see above, searching for the domain "hackerone.com" and the term "password" gave us 7,390 results. In a typical scenario I would end up going through 90% of these results by hand for a few hours before I find something juicy. Having to spend hours sorting through a bunch of trash i s really the only downside to this technique. However, when you do find something it typically leads to an install high or Critical finding.

Conclusion

As of right now Github i s one of the easiest ways to get a high or critical vulnerability. Almost every developer uses Github and these same developers also like hard coding passwords in their source code. As long as you're willing to spend a few hours searching through thousands of repos you’re almost guaranteed to find something good.

Basic Hacking Subdomain Takeover

Introduction

Another extremely popular vulnerability is subdomain takeover. Though this vulnerability has died down significantly it is still very common in the wild. If you are unfamiliar with this type of vulnerability according to Google “Subdomain takeover attacks are a class of security issues where an attacker is able to seize control of an organization's subdomain via cloud services like AWS or Azure”.

Subdomain Takeover

A subdomain takeover occurs when a subdomain i s pointing to another domain (CNAME) that no longer exists. If an attacker were to register the non existing domain then the target subdomain would now point to your domain effectively giving you full control over the target’s subdomain. What makes this vulnerability so interesting is that you can be safe one minute and a single DNS change can make you vulnerable the next minute.

The vulnerability here is that the target subdomain points to a domain that does not exist. An attacker can then register the non existing domain. Now the target subdomain will point to a domain the attacker controls.

If you’re planning on hunting for this vulnerability you are definitely going to be referencing the following github page as it contains a bunch of examples and walkthroughs on exploiting different providers:

https://github.com/EdOverflow/can-i-take-over-xyz

As you can see above this page contains a large list of engines that can be exploited by this vulnerability. If you click on the issue number it will give you a walk through exploiting that particular engine. Because every provider has its own way of registering domains you will need to l earn the process of registering a domain on the engine that impacts your target.

Github Takeover

One of the easiest ways to spot a subdomain takeover vulnerability is by the error message it throws. As you can see above when we visit our target site it throws a 404 status code and gives us the error message “There isn’t a Github Pages Site here”. If we go to the subdomain takeover wiki we can confirm that this error message indicates the possibility of subdomain takeover.

Now that we have an indicator this site is vulnerable we need to get the github page the vulnerable subdomain is pointing to. We need this information so we can register the domain through github. As shown above a “dig” command can be used to gather the DNS records of the vulnerable domain. We can also see that the domain points to the github page “ghostlulzvulntakeover.github.io”, if we can register this domain we win. To figure out the process of registering a domain on Github you can Google it or you can follow the tutorial in the subdomain takeover github page

Now that we know the steps to register a domain on Github we just need to do it. First I created a Github repo with the same name as the CNAME record. After that create an “index.html” file in the repo as shown. The next step is to set the repo as the main branch. Finally specify the target domain you are going after. That's it! Now when you visit the target domain you should see the page you set up.

We WIN! As you can see above we successfully exploited the subdomain takeover vulnerable and got our page to appear on the targets subdomain. Note that this is the process for Github, if your target is vulnerable to something else you will have to follow the steps for that provider. Lucky for us all this is documented on the subdomain takeover github wiki.

Conclusion

A few years ago subdomain takeover was all over the place but it has started to die down recently. However, you will still find plenty of organizations vulnerable to this type of attack. It is extremely easy to pull off and it allows attackers to completely take over the target subdomain. If you’re looking for an easy high security finding this is it.

Basic Hacking Databases

Introduction

A database is an organized collection of data, generally stored and accessed electronically from a computer system. If you’re attacking a web application a lot of the time one of the main goals is to compromise the back end database as it's where all the sensitive user data is stored.

Compromising these databases normally involves exploiting sql injection vulnerability but sometimes it can be much easier. These databases are often exposed to the internet without authentication leaving them opens to hackers for pillaging as discussed i n the following sections.

Google Firebase

Introduction

According to Google “The Firebase Real-time Database is a cloud-hosted database stored as JSON and synchronized in real-time to every connected client”. An issue can arise in firebase when developers fail to enable authentication. This vulnerability is very similar to every other database misconfiguration, there's no authentication. Leaving a database exposed to the world unauthenticated is an open invite for malicious hackers.

Misconfigured Firebase Database

When I’m hunting for this I'll try to keep an eye out for the “*.firebaseio.com” url, if you see this then you know your target is utilizing Google's firebase DB. An example domain can be found below:

Vuln-domain.firebaseio.com

If the developer forgot to enable authentication the database will be exposed to the word. You can easily view the database by appending a “/.json” to the url as shown below:

vuln-domain.firebaseio.com/.json

As you can see above we were able to dump a bunch of passwords belonging to an organization. An attacker could then leverage these credentials to perform additional attacks on the application.

Summary

Finding and exploiting this misconfiguration i s extremely easy and requires zero technical skills to pull off. All you need to do i s find an application using firebase, append “/.json” to the url, and i f there i sn't authentication you can export the entire DB!

ElasticSearch DB

Introduction

You have probably heard of the popular relational database called MySQL. Elastic search like MySQL i s a database used to hold and query information. However, elastic search is typically used to perform full text searches on very large datasets. Another thing to note is that Elastic Search is unauthenticated by default which can cause a lot of security problems as described in the following sections.

ElasticSearch Basics

According to Google “ElasticSearch i s a document- oriented database designed to store, retrieve, and manage document-oriented or semi-structured data. When you use ElasticSearch, you store data in JSON document form. Then, you query them for retrieval.” Unlike MySQL which stores its information i n tables, elastic search uses something called types. Each type can have several rows which are called documents. Documents are basically a j son blob that holds your data as shown in the example below:

{"id":1, "name":"ghostlulz", "password":"SuperSecureP@ssword"}

In MySQL we use column names but i n ElasticSearch we use field names. The field names in the above j son blob would be id, name, and password. In MySQL we would store all of our tables in a database.
In Elastic Search we store our documents i n something called an index. An index is basically a collection of documents.

Unauthenticated ElasticSearch DB

Elastic search has an http server running on port 9200 that can be used to query the database. The major issue here is that a lot of people expose this port to the public internet without any kind of authentication.

This means anyone can query the database and extract information. A quick Shodan search will produce a turn. Once you have identified that your target has port 9200 open you can easily check if it is an ElasticSearch database by hitting the root directory with a GET request. Once you know an endpoint has an exposed Elastic Search db try to find all the indexes (Databases) that are available. This can be done by hitting the "/_cat/indices?v" endpoint with a GET request. This will list out all of the indexes. This information along with other details about the service can also be found by querying the " /_stats/?pretty=1" endpoint. To perform a full text search on the database you can use the following command "/_all/_search?q=email". This will query every index for the word "email". There are a few words that I like to search for which include:

- Username
- Email
- Password
- Token
- Secret
- Key

If you want to query a specific index you can replace the word "_all" with the name of the index you want to search against.

Another useful technique i s to list all of the field names by making a GET request to the "/INDEX_NAME_HERE/_mapping?pretty=1" endpoint. I typically search for interesting field names such as

- Username
- Email
- Password
- Token
- Secret
- Key

As you can see above we have the field names addressable_type, city, and much more which isn't displayed as the output was too large. To query all values that contain a specific field name use the following command "/_all/_search?q=_exists:email&pretty=1" . This will return documents that contain a field name(column) named email. Again you can replace "_all" with the name of an i ndex to perform searches specifically against that endpoint.

Summary

ElasticSearch is just another database where you can store and query information. The major problem is that people expose the unauthenticated web service to the public. With unauthenticated access to the web service attackers can easily dump the entire database. Always be on the lookout for port 9200.

Mongo Database

Introduction

Like ElasticSearch MongoDB is a nosql database that uses JSON-like documents to store data. Also similar to the rest of the databases we have talked about Mongo DB fails to implement authentication by default. This means it's up to the user to enable this which they often forget.

MongoDB

If you're searching for MongoDB instances, be on the lookout for port 27017. As mentioned earlier MongoDB doesn't have authentication enabled by default so to test for this vulnerability just try to login. To do this I normally just use the mongo cli as shown below:

```
mongo ip-address-here
```

Once logged into the database try issuing a command, if you get an “unauthorized” error message prompting for authentication then the endpoint has authentication enabled.
However, if you can run arbitrary commands against the system then authentication has not been set up and you can do whatever you want.

Summary

If you see port 27017 open or any other MongoDB associate ports make sure to test the endpoint to see if it’s missing authentication. Exploiting this misconfiguration is as easy as connecting to the database and extracting the data. This i s as easy as it gets folks.

Conclusion

If an application needs to store data chances are its being stored i n a database. These databases hold all kinds of sensitive information such as passwords, tokens, private messages, and everything else. That's why databases are always popular targets by hackers. Since these are such popular targets you would think they would be fairly secure but they aren't. A lot of databases are missing authentication by default! This means if connected to the internet anyone could connect to these devices to extract the information they hold.

Name	Endpoint
Firebase DB	*.firebaseio.com/.json
Elasticsearch	Port:9200
MongoDB	Port:27017
CouchDB	Port:5985,6984
CassandraDB	Port:9042,9160

Basic Hacking Brute Forcing

Introduction

Brute forcing is a classic attack that has been around forever and shows no signs of being eliminated. Passwords are a weak point of security and as an attacker you should take full advantage of this. Easily guessable passwords, using default passwords, and password reuse are easy ways for an organization to get compromised. The rule of thumb is if there is a login screen i t should be brute forced.

Login Pages

There are three things you need to have if you want to launch a brute force attack. The three things you need are an endpoint with a login page, a username, and a password. First you need to find the endpoint you want to target.

Name	Endpoint
Web Application Login Page	Web application login page, Outlook mail, VPN, Router, Firewall, WordPress, admin panel, etc
SSH	Port:22
RDP	Port:3389
VNC	Port:5900
FTP	Port:21
Telnet	Port:23

Default Credentials

Now that you know which endpoints to look out for you need to get a list of usernames and passwords? This technique may be basic but you would be surprised at the number of times i v compromised an organization because they are using default credentials.

As shown above one of the best places to find default passwords i s SecList:

https://github.com/danielmiessler/SecLists/tree/master/Passwords/Default-Credentials

The above picture is an example file containing default usernames and passwords to hundreds of routers. All you have to do is look up the target vendor and try all the default passwords it uses, this technique works very well as people often forget to change the default credentials.

If you are targeting an SSH server or something other than a router the process will be slightly different. Not really, those services also come with default credentials. Depending on the service you are brute forcing you will want to find or create a list of credentials tailored toward that. You may also find that sec list does not have any default passwords impacting the target technology. If that's the case just perform a Google search or two, I normally find these things i n the first few l inks.

Brute Forcing

Once you have a good set of credentials you can start the actual process of brute forcing. You could do this by hand but I would 100% recommend using a tool for this j ob unless you are only testing 5 passwords or something small like that.

https://github.com/vanhauser-thc/thc-hydra

If you're performing a brute force attack you probably want to use the tool “hydra”. This tool supports a bunch of different protocols and has never let me down. Once you have the target endpoint and credentials you can use any tool to perform the brute force attack just pick one you like.

Conclusion

Brute force attacks are an easy way to compromise a target application. With the use of default passwords, easily guessable passwords, and password reuse finding a target vulnerable to this shouldn't be that hard. All you need is a good credential list and you're ready to go.

Basic Hacking Burp Suite

Introduction

If there is one tool that you N EED to have to be a successful Bug Bounty Hunter it would be Burp Suite. You can find plenty of bugs without ever leaving Burp, it is by far my most used and favorite tool to use, almost every web attack I pull off is in Burp. If you don’t know what Burp is it’s a tool for performing security tests against web applications. The tool acts as a proxy and allows you to inspect, modify, and replay, etc to web requests. Almost every exploit you’re going to pull off will be done with Burp.

https://portswigger.net/burp

Note that there is a free version (community) but I HIGHLY recommends purchasing a professional license. This is a must have tool

Proxy

The proxy tab is probably the most important tab in Burp. This is where you can see all of your traffic that passes by the Burp proxy. The first things you want to do when Burp loads are making sure your proxy is up and running

The next step is to force your browser to route its traffic through the Burp proxy, this can be accomplished by changing your browsers proxy setting and shown below, note this will be different depending on which browser you use. Once you have the Burp proxy listening, the browser configured to use Burp, and you imported the Burp certificate in your browser you will be good to go. Once you navigate to a web page you should see the request show up in Burp as shown below:

As you can see i n the above image the “intercept” tab is toggled on, this means that Burp will intercept each HTTP request and you will have to manually press the “forward” button for the request to continue to the server. While on this tab you can also modify the requests before forwarding it to the back-end server. However, I only use this tab when i 'm trying to isolate requests from a specific feature, I normally turn “intercept” to off and I view the traffic in the “HTTP History” tab and shown.
As you can see the “HTTP History” tab shows each HTTP request and response that was made by and sent to our browser. This is where I spend 80% of my time looking for something that peaks my interest. When looking at the traffic I'm mostly paying attention to the method, URL, and MIME type fields. Why? Because when I see a POST method being used I think of Stored XSS, Cross site request forgery, and many more vulnerabilities. When I see a URL with an email, username, or id in it I think IDOR. When I see a JSON MIME type I think back-end API. Most of this knowledge of knowing what to look for comes with experience, as you test so many apps you start to see things that look similar and you start to notice things that look interesting.

Clicking on an HTTP request will show you the clients request and the server’s response; this can be seen in the above image. Note that while in this view these values can’t be modified, you will have to send the request to the repeater if you want to modify the request and replay it, this will be discussed in more detail later.

One functionality that I use to find a lot of vulnerabilities and make my life easier is the search feature. Basically you can search for a word(s) across all of your Burp traffic.

This is extremely powerful and has directly led me to finding vulnerabilities. For example I may search for the word “url=” this should show me all requests which have the parameter URL in it, I can then test for Server Side Request Forgery (SSRF) or open redirect vulnerabilities. I might also search for the header “Access-Control-Allow-Origin” or the “callback=” GET parameter when testing for Same Origin Policy (SOP) bypasses. These are just some examples, your query will change depending on what you're looking for but you can find all kinds of interesting leads. Also don't worry if you don't know what SSRF or SOP bypass means these attacks will be discussed in the upcoming chapters.

Burps proxy tab is where you will spend most of your time so make sure you are familiar with it. Any traffic that is sent by your browser will be shown i n the HTTP history tab just make sure you have intercept turned off so that you don’t have to manually forward each request.

Target

I generally don't find myself in the target section of burp suite but I think it's still important to know what it is. The “Site Map” sub tab organizes each request seen by the proxy and builds a site map as shown.

As you can see in the above image a site map i s built which easily allows us to view requests from a specific target. This becomes fairly useful when hitting an undocumented API endpoint as this view allows you to build a picture of the possible endpoints. You can also view the HTTP requests in this tab, clicking on a folder in the sitemap will only show requests from that path. In addition to the “Site Map” tab there is a “Scope” tab. I almost never use this but if you want to define the scope of your target this will limit burps scans to only the domains in scope.

Intruder

If you're doing any fuzzing or brute forcing with Burp you're probably doing it in the “intruder” tab. When you find an interesting request right click it then click “Send to Intruder”, this will send your requests to the intruder tab as shown.

Go to the intruder tab and you should see something like this. Now click the “Clear” button to reset everything. Now from here your steps vary depending on what you're trying to do, but suppose we are trying to do some parameter fuzzing. One of the first things we need to do is select the value we are trying to modify. This can be done by highlighting the value and pressing the “Add” button as shown. As you can see above we are selecting the “cb” parameter value. Since we are attempting to do parameter fuzzing this is the value that will be replaced with our fuzzing payloads.

You may have also noticed the “Attack type” drop down menu is set to “Sniper”, there are four different attack types which are described in the table below:

Sniper	Uses a single payload list; Replaces one position at a time;
Battering ram	Uses a single payload list; Replaces all positions at the same time;
Pitchfork	Each position has a corresponding payload list; So if there are two positions to be modified they each get their own payload list.
Cluster Bomb	Uses each payload list and tires different combinations for each position.

Once you have selected your attack type and the value to be modified click on the “Payloads” sub tab. Here we want to select our payload type and the payload list. There are numerous payload types but I’m going to keep it on the default one, feel free to play around with the others. As for my payload list we want a list of fuzzing values. For this example I’m just going to use the default lists that come with Burp but there are some other good lists on SecLists:

https://github.com/danielmiessler/SecLists/tree/master/Fuzzing

Now to use Burps pre defined list just click the “Add from list” drop down menu and select one:
Now that you have your fuzzing list imported all that you have to do is press “Start attack”.

As shown above after hitting the “Start attack” button a popup will appear and you will see your payloads being launched. The next step is to inspect the HTTP responses to determine if there is anything suspicious.

Intruder is great for brute forcing, fuzzing, and other things of that nature. However, most professionals don't use intruder, they use a plugin called “Turbo Intruder”. If you don't know what “Turber Intruder” is, its intruder on steroids, it hits a whole lot harder and a whole lot faster. This plugin will be discussed more i n the plugins section.

Repeater

In my opinion this is one of the most useful tabs in Burp. If you want to modify and replay and request you do it in the repeater tab. Similar to Intruder if you right click a request and click “Send to Repeater” it will go to the repeater tab.

Once the request is sent to the Repeater tab you will see something like. One this tab you can modify the request to test for vulnerabilities and security misconfigurations. Once the request is modified you can hit the Send button to send the request. The HTTP response will be shown in the Response window. You might have noticed that at the top there are a bunch of different tabs with numbers on them. By default every request you send to the repeater will be assigned a number. Whenever I find something interesting I change this value so I can easily find it later, that's why one of the tabs is labeled SSRF, it’s a quick easy way to keep a record of things.

Conclusion

Burp Suite i s the one tool every bug bounty hunter needs i n their arsenal. If you’re doing a deep dive on a target application Burp is the only tool you need. It has a vast amount of plugins to aid in the identification and exploitation of bugs but its real power comes from allowing attackers the ability to inspect and manipulate raw HTTP requests. Once you l earn the basics of Burp you can pull off the vast majority of your hacks using the tool.

Basic Hacking OWASP

Introduction

I started off as a penetration tester specializing in web application and when I started doing bug bounties my skills carried over 100%. Legit 80% of the attacks you pull off are going to be against a web application. After all, in today's world the vast majority of a company's public facing assets are web applications. For this reason alone you MUST learn web application hacking if you want to be successful and there is no better place to start than the OWASP top 10. If all you got out of this book was l earning how to exploit these basic web vulnerabilities you will be able to find bugs all day.

SQL Injection (SQLI)

Introduction

SQL Injection (SQL) is a classic vulnerability that doesn’t seem to be going anywhere. This vulnerability can be exploited to dump the contents of an applications database. Databases typically hold sensitive information such as usernames and passwords so gaining access to this are basically game over. The most popular database is MySQL but you will run into others such as MSSQL, PostgreSQL, Oracle, and more.

The main cause of SQL injection is string concatenation as shown in the above code snippet. One line three the application is concatenating user supplied input with the sql query, if you ever see this you know you have sql injection. The reason why this is so dangerous is because we can append additional sql queries to the current query. This would allow an attacker to query anything they want from the database without restrictions.

MySql

The two most common types of sql injection are union based and error based. Union based sql injection uses the “UNION” sql operator to combine the results of two or more “SELECT” statements into a single result. Error based sql injection utilizes the errors thrown by the sql server to extract information.

Typically when I'm looking for this vulnerability I'll throw a bunch of double and single quotes everywhere until I see the famous error message.

As you can see in the first i mage appending a single quote to the “cat” variable value throws a sql error. Look at the two error messages and notice how they are different. Note that “%27” is the same as a single quote, it's just url encoded. In the following sections I'll show you how to exploit this vulnerability and no we won’t be using SqlMap, you need to know how to do this by hand.

https://github.com/sqlmapproject/sqlmap

Union Based Sql Injection

Once you know that an endpoint is vulnerable to sql injection the next step is to exploit it. First you need to figure out how many columns the endpoint i s using. This can be accomplished with the “order by” operator. Basically we are going to ask the server “do you have one column”, if it does the page will load. Then we ask “do you have two columns”, if it loads it does and if it throws an error we know it doesn't.

We can see here the page loads just fine, this means there must be at l east one column returned by the sql statement. Just keep adding one to the number until you get an error.

- Order by 1
- Order by 2
- Order by 3
- Order by 4

If you were to try “order by 4” it will fail so there must not be 4 columns which means there are 3 because “order by 3” loaded without any errors.

Now that you know how many columns the sql query is using you need to figure out which columns are being displayed to the page. We need to know this because we need a way to display the information we are extracting. To accomplish this we can use the “union all select” statement. Note that for the second select statement to show we need to make the first query return nothing, this can be accomplished by putting an invalid id.

Notice the numbers on the page. These numbers refer to the columns which are being displayed on the front end. Look at the above example. I see the numbers “2” and “3” so these are the columns we will use to display the results from our queries.

As shown above one of the first things I typically do is to display the database version, this can be accomplished with the following mysql command:
@@version
Version ()

You can see we are working with mysql version 5.1.73, it's a good idea to note this down as it might come in handy later. Extracting the database version is cool and all but what about the sensitive data.

To extract data we first need to know what database tables we want to target, we can get a list of tables with the following command:

```
Select * from i nformation_schema.tables
```

Note that “information_schema.tables” is a default table within mysql that holds a list of table names. This table has two columns we care about, table_name and table_schema. You can probably guess what the table_name column represents. The table_schema column holds the name of the database the table belongs to, so if you only want to get tables from the current database make sure to filter the results with the “where” operator.

```
union all select 1,2,group_concat(table_name) from information_schema.tables where table_schema = database()
```

As you can see above we got a list of all the tables belonging to this database. You might have noticed the function “database ()”, this function outputs the current database name and is used to filter the results via the table_schema column. You also might have noticed the “group_concat” function; this function will concatenate all the table names into a single string so they can all be displayed at once. Once you pick which tables you want to target you need to get a list of columns belonging to that table. A list of columns belonging to a table can be retrieved via the “information_schema.columns” table as shown i n the below query:

```
union all select 1, 2,group_concat(column_name) from information_schema.columns where table_name = "users"
```

As you can see above there are a few columns returned, the most interesting column names are “uname” and “pass”. The final step is to dump the contents of these two columns as shown below:

```
union all select 1,2,group_concat(uname,":",pass) from users
```

As you can see above there i s a user called “test” with the password “test”. We can then use these credentials to login to the application as that user.

Error Based Sql Injection

With union based sql injection the output is displayed by the application. Error based sql injection is a little different as the output is displayed in an error message. This is useful when there is no output except a sql error.

Xpath

If the MySql service version is 5 .1 or later we can use the “ extractvalue ()” function to exfiltrate data from the database. The ExtractValue () function generates a SQL error when it is unable to parse the XML data passed to it. Rember with error based sql injection we must extract our data via sql error messages.

First you need to understand how the ExtractValue () function works, once you understand how this function operates you can abuse it for sql injection.

As you can see in the above image the ExtractValue() function is used to parse out a value from an XML document. Here we pass in the XML string
“ <id>1</id><name>ghostlulz</name> <email>ghostlulz@offensiveai.com</email>”
and we get the value of the name tags with the second argument. So the first argument is an XML document and the second argument i s the tag we want to get the value of.

As shown above i f the second argument starts with a “;” it will cause a MySql error message to appear along with the string that caused the error. Attackers can abuse this to extract data via error messages. Looking at the above example you can see I was able to extract the database version via an error message. Armed with this knowledge you can now use this technique to perform error based sql injection.

```
AND extractvalue("blahh",concat(";",@@version))
```

As you can see above we were able to extract the MySql database version via an error message. The next step is to get a list of table names. Similar to union based sql injection we will be utilizing the information_schema.tables table to achieve this.

AND extractvalue("blahh",(select concat(";",table_name) from information_schema.tables where table_schema = database() limit 0,1))

Notice the "limit 0,1" command at the end of the query. This is used to get the first row in the table, with error based sql injection we have to query one table at a time. To get the second table you would use "limit 1,1".

As you can see above we will be targeting the "users" table. Once you have your target table you need to query the column names belonging to that table.

AND extractvalue("blahh",(select concat(";",column_name) from information_schema.columns where table_name = "users" limit 0,1))

The first column name is "uname", now we have to get the second column name as shown.

As you can see above the second column name is called "pass". The final step is to extract the data from these columns.

AND extractvalue("blahh",(select concat(";",uname,":",pass) from users limit 0,1))

As you can see above we were able to extract the username and password of the first user "test:test". To get the next user just change "limit 0,1" to "limit 1,1".

PostgreSql

If you know how to perform sql injection on a mysql server then exploiting postgres will be very similar. Just like mysql I typically throw single and double quotes every where until I see the famous error message appear.

As you can see above there is an error message displayed. The name "psycopg2" is a python library for postgres so if you see this name you know you're working with a postgres database server.

Union Based Sql Injection

Just like MySql the first step is to determine how many columns the sql query is using, this can be accomplished by using the "order by" operator. As shown below we ask the server "do you have at l east one column", and then we ask "do you have two columns", and so on until we get an error.

As you can see below once we hit 3 columns the server errors out, this tells us that there are only 2 columns being retrieved by the query.

As shown below we can use the “union all select” operator to perform the second query. Also note how the second select column is wrapped in single quotes, this is because the column types must match the original query. The first column is an integer and the second column is a string.

Note you can also use the word “null” if you don’t know the data type, so i t would look like:

Union all select null,null

If you weren't able to detect the database type from the error message you could always use the “version()” function to print the database type and version as shown.

As you can see above the application is running on PostgreSQL version 12.3. After you have the number of columns the query returns we need to find all the tables in the database. Just like MySql we can query the “information_schema.tables” table to get a list of all tables in the databases.

union all select 1,table_name from information_schema.tables where table_schema != ' pg_catalog' and table_schema != ' information_schema' offset 0

For the most part this is the same as MySql but there are a few differences. For starters PostgreSQL doesn't have a group_concat function so instead I return one table_name at a time with the “offset” operator. Offset ‘0’ get the first table name, offset ‘1’ gets the second and so on. I also filter out the default databases “pg_catalog” and “information_schema” as they tend to clog up the results.

As shown above the second table name is called “users”, this is the table we will be targeting. The next step is to extract the columns associated with the target table as shown.

union all select 1,column_name from information_schema.columns where table_name = ' users' offset 0

As shown above there are two interesting columns called username and password. These are the columns we will be extracting data from as shown i n the below query:

union all select 1,concat(username,':',password) from users offset 0

Finally the username and password of the first user is shown. An attacker could then use these credentials to log in to the application.

Oracle

MySql and PostgreSql are very similar to each other so if you know one the other will come easy. However, Oracle is different from those two and will require some additional knowledge to successfully exploit it. As always when testing for this vulnerability I usually just throw a bunch of single and double quotes around until I get an error message as shown below:

As shown above the error message starts with “ORA” and that's a good sign that you are dealing with an Oracle database. Sometimes you can’t tell the database type from the error message i f that's the case you need to return the database version from a sql query as shown below:

```
select banner from v$version
```

Note that similar to PostgreSql when you are selecting a column it must match the type of the first select statement. You can also use the word ‘null’ as well if you don't know the type. Another thing to note is that when using the select operator you must specify a table, in the above image the default table of “dual” was used.

Union Based Sql Injection

Just like MySql and PostgreSql the first step is to figure out how many columns the select statement is using. Again this can be accomplished with the “order by” operator as shown below:

As mentioned in the previous sections we increase the order by operator by one until you get an error. This will tell you how many columns there are.

As shown above an error was displayed once we got to column number 3 so there must only be 2 columns used in the select statement. The next step is to retrieve a list of tables belonging to the database as shown.

```
union all select LISTAGG(table_name,',') within group (ORDER BY table_name),null from all_tables where tablespace_name = ' USERS' –
```

If you’re used to using MySql or PostgreSql you would normally use the “information_schema.tables” table to get a list of tables but oracle uses the “all_tables” table for this. You probably want to filter on the “tablespace_name” column value “USERS” otherwise you will get hundreds of default tables which you have no use for. Also notice the “listagg()” function, this is the same as MySqls ‘ group_concat()’ function and is used to concatenate several rows into a single string. When using the listagg() function you must also use the ‘ within group()’ operator to specify the order of the listagg function results.

Once you get your target table you need to get a l ist of the column names belonging to that table as shown below:

```
union all select LISTAGG(column_name,',') within group (ORDER BY column_name),null from all_tab_columns where table_name = ' EMPLOYEES'—
```

In MySql we would have queried the “information_schema.columns” table to get a list of columns belonging to a table but with oracle we use the “all_tab_columns” table to do this. Finally once you know the table’s column names you can extract the information you want using a standard sql query as shown below:

```
Union all select email, phone_number from employees
```

As you might have noticed Oracle sql injection is a little different compared to MySql and PostgreSql but it is still very similar. The only difference is the syntax of a couple things but the process remains the same. Figure out the target table name, get the tables columns, and then finally extract the sensitive information.

Summary

SQL injection is one of the oldest tricks in the book yet it still makes the OWASP top 10 list every year. It's relatively easy to search for and exploit plus it has a high impact on the server since you are able to steal everything in the database including usernames and passwords. If you're searching for this vulnerability you are bound to come across a vulnerable endpoint, just throw single and double quotes everywhere and look for the common error messages. Unlike 90% of other hackers you should know how to exploit the vast majority of databases not just Mysql so when you do find this bug it shouldn't be too hard to exploit.