BROKEN AUTHENTICATION AND SESSION MANAGEMENT

BROKEN AUTHENTICATION AND SESSION MANAGEMENT

This happens to be one of the most highly ranked security risks as per the Open Web Application Security Project

BROKEN AUTHENTICATION AND SESSION MANAGEMENT

This covers a **wide range** of issues arising due to compromised passwords, keys, session tokens and enabling attackers to **impersonate** other users on a website

BROKEN AUTHENTICATION AND SESSION MANAGEMENT

The broad categories of issues are in

1. Credential Management
2. Session Management
3. The rest (whatever doesn't fit in the first 2 categories)

BROKEN AUTHENTICATION AND SESSION MANAGEMENT

1. Credential Management
2. Session Management
3. The rest (whatever doesn't fit in the first 2 categories)

A number of vulnerabilities exist in each of these categories and there are good practices which should be followed - we'll cover it all in detail

SESSION MANAGEMENT

SESSIONS

SESSIONS

From a user perspective a session is when you view a website, spend time on it and then close your browser

This entire series is one session

SESSIONS

As a user you want the website to **remember** you for a duration of time to get the best experience possible

SESSIONS

As a user you want the website to
remember you for a duration of time
to get the best experience possible

The website should remember you
across pages on the site

SESSIONS

As a user you want the website to
remember you for a duration of time
to get the best experience possible

The website should remember you
across pages on the site

It should remember you even if you
closed the browser window

SESSIONS

As a user you want the website to
remember you for a duration of time
to get the best experience possible

The website should remember you across pages on the site

It should remember you even if you closed the browser window

It should be able to remember sensitive data like passwords e.g. Gmail let's you remain logged in for 30 days

SESSIONS

The website should remember you across pages on the site

It should remember you even if you closed the browser window

It should be able to remember sensitive data like passwords e.g. Gmail let's you remain logged in for 30 days

DATA SHOULD BE STORED ON THE SERVER!

SESSIONS

DATA SHOULD BE STORED ON THE SERVER!

Cookies are stored on your local machine

This makes them insecure to store sensitive information

SESSIONS

DATA SHOULD BE STORED
ON THE SERVER!

THIS IS WHERE
SESSIONS COME IN

SESSIONS

THIS IS WHERE
SESSIONS COME IN

A cookie is a bit of data stored by the browser
and sent to the server on every request

A SESSION IS A COLLECTION OF DATA
STORED ON THE SERVER AND
ASSOCIATED WITH A USER

SESSIONS

A cookie is a bit of data stored by the browser
and sent to the server on every request

A SESSION IS A COLLECTION OF DATA STORED ON THE SERVER AND ASSOCIATED WITH A USER

SESSIONS

A cookie is a bit of data stored by the browser
and sent to the server on every request

A SESSION IS A COLLECTION OF DATA STORED ON THE SERVER AND ASSOCIATED WITH A USER

SESSIONS

A cookie is a bit of data stored by the browser
and sent to the server on every request

A SESSION IS A COLLECTION OF DATA
STORED ON THE SERVER AND
ASSOCIATED WITH A USER

SESSIONS

A cookie is a bit of data stored by the browser
and sent to the server on every request

A SESSION IS A COLLECTION OF DATA
STORED ON THE SERVER AND
ASSOCIATED WITH A USER

SESSIONS

A cookie is a bit of data stored by the browser
and sent to the server on every request

A SESSION IS A COLLECTION OF DATA STORED ON THE SERVER AND ASSOCIATED WITH A USER

SESSIONS

A SESSION IS A COLLECTION OF DATA
STORED ON THE SERVER AND
ASSOCIATED WITH A USER

THE ASSOCIATION OF A SESSION WITH A USER IS DONE VIA A COOKIE!

SESSIONS

THE ASSOCIATION OF A SESSION
WITH A USER IS DONE VIA A COOKIE!

A session id is associated with every user - and this session id is stored in a cookie

SESSIONS

A session id is associated with every user - and this session id is stored in a cookie

Remember we need this because we want to remember a user even when he is not logged in!

SESSION MANAGEMENT

SESSION MANAGEMENT

A session can be used to store important, private user information on the server

It's possible that **credit card** information or **bank information** is stored as a part of the session

SESSION MANAGEMENT

Sessions are often not just identity tokens but serve as **authenticators**

Once a user logs in, a session id maybe temporarily used by the site to server as an **authentication mechanism**

SESSION MANAGEMENT

Session hijacking refers to **exploiting a user's session** to gain unauthorized access to their data

This involves getting **access to the user's session id** and their session

SESSION MANAGEMENT

Many thanks to this great white paper on session hijacking and fixation here: http://www.acrossecurity.com/papers/session_fixation.pdf

SESSION MANAGEMENT

Let's examine the steps in the attack process

Session setup: Setting up a **trap session** to use in the attack. This session might need to be **kept alive** or maintained with repeated requests to the server

SESSION MANAGEMENT

Session fixation

Let's examine the steps in the attack process

Session fixation: Introducing the session id to the victim's browser to fix the session

SESSION MANAGEMENT

Session fixation

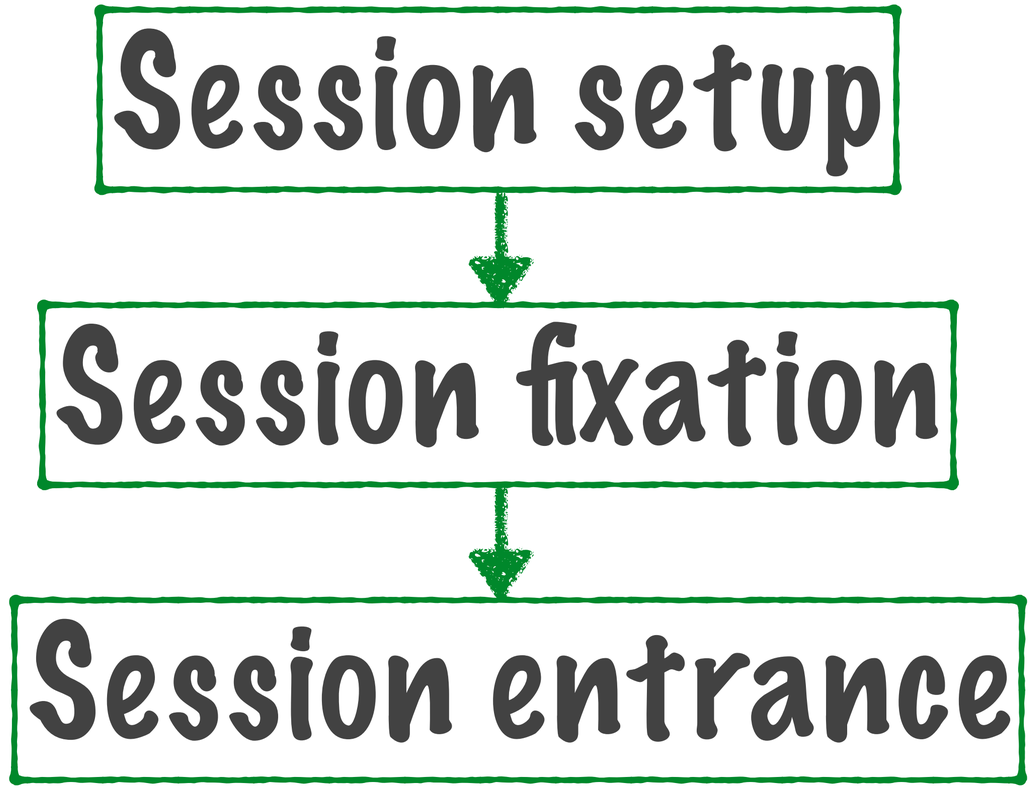
Let's examine the steps in the attack process

Session entrance: Waiting till the user logs in and uses the previously fixed session id

SESSION MANAGEMENT

Session fixation

Let's examine the steps in the attack process



SESSION MANAGEMENT

Session setup

Servers can be categorized on the basis of how they are setup and what session ids they accept

SESSION MANAGEMENT

Session setup

Permissive setup:

These accept any **arbitrary session id** from the user

The server creates a new session id with the **proposed session id** if one does not exist

SESSION MANAGEMENT

Session setup

Permissive setup:

The server creates a new session id with
the proposed session id if one does not exist

An attacker only needs to generate a random session id for the trap session and use it

SESSION MANAGEMENT

Session setup

Strict setup:

Only accepts known session ids which have been **locally** generated in the past

SESSION MANAGEMENT

Session setup

Strict setup:

An attacker needs to:

establish a session with the server

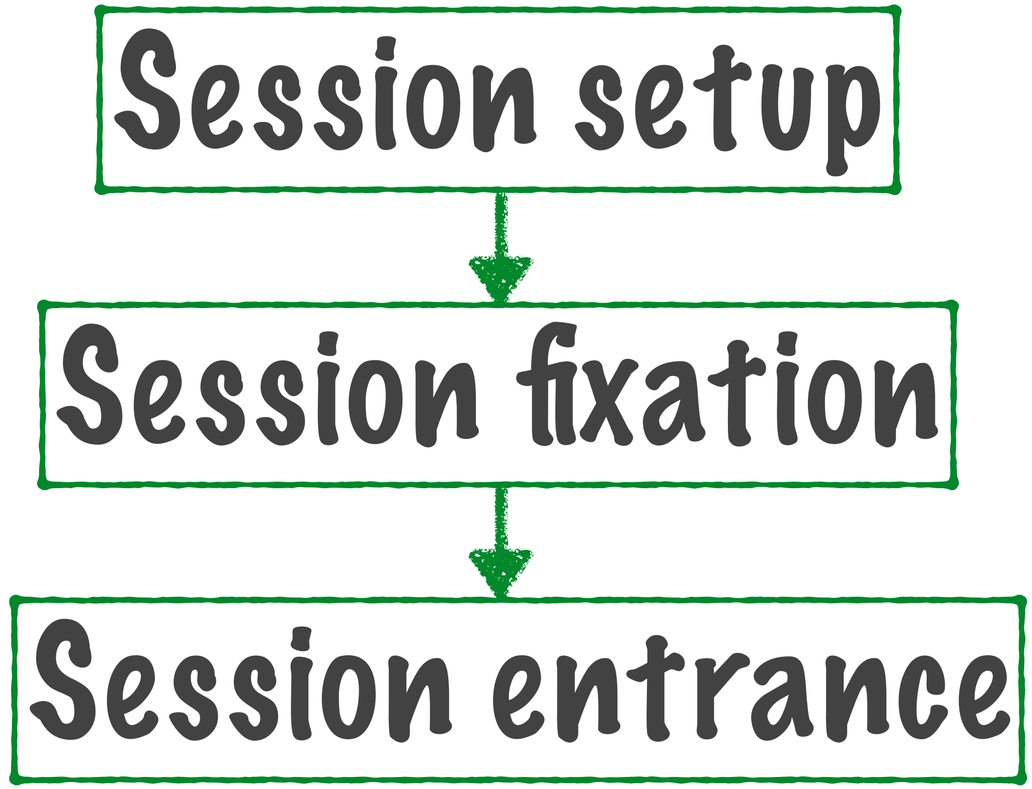
remember the session id

possibly keep it alive till the user logs in by periodic requests

SESSION MANAGEMENT

Session fixation

Let's examine the steps in the attack process



SESSION MANAGEMENT

Session fixation

Future lectures will talk about session fixation in some detail

Session entrance

How the attacker uses the session is website dependent and out of the scope of this class

SESSION MANAGEMENT

Session hijacking

This is a general term for a security attack involving sessions. Hijacking can be accomplished in a number of ways

SESSION MANAGEMENT

Session hijacking

1. Session ids in URLs, cookies or form fields
2. Session fixation
3. Session sidejacking or sniffing
4. Cross site scripting
5. Malware or other unwanted programs on the client

SESSION MANAGEMENT

Session ids can be passed via:

1. URL parameters
2. Hidden form fields
3. Cookies

SESSION MANAGEMENT

Cookies

URL parameters

Hidden form fields

Of these using **cookies** is considered the **least insecure**

All attacks which exploit cookie based sessions can be used against URL parameter and hidden form based sessions

SESSION MANAGEMENT

Cookies

URL parameters

Hidden form fields

All attacks which exploit cookie based sessions can be used against URL parameter and hidden form based sessions

The converse is not true

SESSION MANAGEMENT

Session ids in URLs

Sessions use cookies to pass the session id to the client but it's also possible to use sessions without cookies

SESSION MANAGEMENT

Session ids in URLs

This means the session id will be present in the URL

SESSION MANAGEMENT

Session ids in URLs

Let's say a user sends a link for his airline tickets to his friends

```
http://airticketsonline.com/sale/items?
session_id=2L5OC2JSNAGDCHCJUN2JV
```

SESSION MANAGEMENT

Session ids in URLs

```
http://airticketsonline.com/sale/items?
session_id=2L5OC2JSNAGDCHCJUN2JV
```

Clicking on the link may allow his friends to access the flight details but it also allows access to the user's session

SESSION MANAGEMENT

Session ids in URLs

```
http://airticketsonline.com/sale/items?
session_id=2L5OC2JSNAGDCHCJUN2JV
```

The session may have stored the credit card information - which means the info is now available for **anyone with this link** to use

SESSION MANAGEMENT

Session ids in URLs

```
http://airticketsonline.com/sale/items?
session_id=2L5OC2JSNAGDCHCJUN2JV
```

The session id may allow them access to the entire authenticated session!

SESSION MANAGEMENT

Session ids in URLs

Never pass session ids in the URL

Use POST requests for form submissions rather than GETs which puts parameters in the URL

SESSIONS WITHOUT COOKIES

Example1 2-SessionMgmt-sessionsWithoutCookies-page1.php
Example1 2-SessionMgmt-sessionsWithoutCookies-page2.php
Example1 2-SessionMgmt-sessionsWithoutCookies-page3.php

SESSIONS WITHOUT COOKIES

It's possible to use sessions in PHP without cookies

This is important because it's possible that a user has turned off cookies on the browser

SESSIONS WITHOUT COOKIES

How PHP sessions work without cookies is a common interview question!

SESSIONS WITHOUT COOKIES

In reality there are 2 ways to propagate the session id to the client

1. Using cookies
2. Using a URL parameter

SESSIONS WITHOUT COOKIES

```
ini_set("session.use_cookies", 0);
ini_set("session.use_only_cookies", 0);
```

Set these parameters to mimic no cookies on the client

SESSIONS WITHOUT COOKIES

The server sends a session id to the client which is available in the constant **SID**

Now the same session id needs to be **propagated** to all the pages of this web site

SESSIONS WITHOUT COOKIES

Now the same session id needs to be propagated to all the pages of this web site

The flag session.use_trans_sid set to 1 causes PHP to do the propagation automatically via URL parameters

SESSIONS WITHOUT COOKIES

Now the same session id needs to be propagated to all the pages of this web site

The flag session.use_trans_sid set to 1 causes PHP to do the propagation automatically via URL parameters

Without this the propagation has to be done manually by you for every link and every form you set up on the site!

SESSIONS WITHOUT COOKIES

```
ini_set("session.use_trans_sid", 1);
```

This transparently propagates the session id to all URLs linked off your site!

SESSIONS WITHOUT COOKIES

```
<a href="Example12-SessionMgmt-sessionsWithoutCookies-page2.php">
```

A URL reference like this is converted behind the scenes to one which includes the session id

```
Example12-SessionMgmt-sessionsWithoutCookies-page2.php?PHPSESSID=a318267116025e2c24bd8b93a8ddb5a3
```

SESSIONS WITHOUT COOKIES

```
<a href="Example12-SessionMgmt-sessionsWithoutCookies-page2.php">
```

```
Example12-SessionMgmt-sessionsWithoutCookies-page2.php?PHPSESSID=a318267116025e2c24bd8b93a8ddb5a3
```

The session id is visible in the URL and can be bookmarked so this is not very secure

SESSIONS WITHOUT COOKIES

What about session information in forms?

```
<form>
<input type="hidden" name="PHPSESSID" value="12345678" >
</form>
```

PHP automatically inserts a hidden input type in the form which sends along the session id

SESSIONS WITHOUT COOKIES

What about session information in forms?

```
<form>
<input type="hidden" name="PHPSESSID" value="12345678" >
</form>
```

This is not explicit - PHP does this behind the scenes

SESSIONS WITHOUT COOKIES

Let's see how we work with sessions without cookies

SESSIONS WITHOUT COOKIES

```
<?php
  // Start the session, this should be before the <html> tag.
  ini_set("session.use_cookies", 0);
  ini_set("session.use_only_cookies", 0);
  ini_set("session.use_trans_sid", 1);
  session_start();
?>

<html lang="en">
<body>
<?php
  echo '<h3>Page 1</h3>';

  if (!isset($_SESSION['visits'])) {
    $_SESSION['visits'] = 1;
  } else {
    $_SESSION['visits']++;
  }

  echo 'You have visited this site: ' . $_SESSION['visits'] . ' times <br>';
  echo 'Session name: ' . session_name() . '<br>';
  echo 'Session id: ' . session_id() . '<br><br>';
  echo 'SID: ' . SID . '<br><br>';
  echo "<a href=\"Example12-SessionMgmt-sessionsWithoutCookies-page2.php\"> Next page </a>";
?>
</body>
</html>
```

SESSIONS WITHOUT COOKIES

```php
<?php
  // Start the session, this should be before the <html> tag.
  ini_set("session.use_cookies", 0);
  ini_set("session.use_only_cookies", 0);
  ini_set("session.use_trans_sid", 1);
  session_start();
?>
```

Turn off cookies and turn on transparent propagation of session id

Start the session as usual!

SESSIONS WITHOUT COOKIES

Use the visits variable to count the number of visits to the website

```
if (!isset($_SESSION['visits'])) {
  $_SESSION['visits'] = 1;
} else {
  $_SESSION['visits']++;
}
```

SESSIONS WITHOUT COOKIES

session_name() and session_id() are functions which return the name of the session and id of the session

```
<?php
  // Start the session, this should be before the <html> tag.
  ini_set("session.use_cookies", 0);
  ini_set("session.use_only_cookies", 0);
  session_start();
<html lang="en">
<body>
<?php
  echo '<h3>Page 1</h3>';

  if (!isset($_SESSION['visits'])) {
    $_SESSION['visits'] = 1;
  } else {
    $_SESSION['visits']++;
  }

  echo 'You have visited this site: ' . $_SESSION['visits'] . ' times <br>';
  echo 'Session name: ' . session_name() . '<br>';
  echo 'Session id: ' . session_id() . '<br><br>';
  echo 'SID: ' . SID . '<br><br>';
  echo "<a href=\"Example12-SessionMgmt-sessionsWithoutCookies-page2.php\"> Next page </a>";
?>
</body>
</html>
```

SESSIONS WITHOUT COOKIES

The SID is a constant which holds the name and session id information

```
echo 'You have visited this site: ' . $_SESSION['visits'] . ' times <br>';
echo 'Session name: ' . session_name() . '<br>';
echo 'Session id: ' . session_id() . '<br><br>';
echo 'SID: ' . SID . '<br><br>';
echo "<a href=\"Example12-SessionMgmt-sessionsWithoutCookies-page2.php\"> Next page </a>";
```

SESSIONS WITHOUT COOKIES

```php
<?php
  // Start the session, this should be before the <html> tag.
  ini_set("session.use_cookies", 0);
  ini_set("session.use_only_cookies", 0);
  ini_set("session.use
  session_start();
?>

<html lang="en">
<body>
<?php
  echo '<h3>Page 1</h3>';

  if (!isset($_SESSION['visits'])) {
    $_SESSION['visits'] = 1;
  } else {
    $_SESSION['visits']++;
  }

echo 'You have visited this site: ' . $_SESSION['visits'] . ' times <br>';
echo 'Session name: ' . session_name() . '<br>';
echo 'Session id: ' . session_id() . '<br><br>';
echo 'SID: ' . SID . '<br><br>';
echo "<a href=\"Example12-SessionMgmt-sessionsWithoutCookies-page2.php\"> Next page </a>";
?>
</body>
</html>
```

You have visited this site: 1 times
Session name: PHPSESSID
Session id: f06f07f0897fa21ef90ae7988327cef2

SID: PHPSESSID=f06f07f0897fa21ef90ae7988327cef2

SESSIONS WITHOUT COOKIES

Set up 3 web site pages with the same code and set up a link to navigate between them

```
echo 'You have visited this site: ' . $_SESSION['visits'] . ' times <br>';
echo 'Session name: ' . session_name() . '<br>';
echo 'Session id: ' . session_id() . '<br><br>';
echo 'SID: ' . SID . '<br><br>';
echo "<a href=\"Example12-SessionMgmt-sessionsWithoutCookies-page2.php\"> Next page </a>";
```

SESSIONS WITHOUT COOKIES

Set up 3 web site pages with the same code and set up a link to navigate between them

We need to check whether multiple pages maintain the same session on the website!

SESSIONS WITHOUT COOKIES

Video: Sessions Without Cookies trans id set

SESSION MANAGEMENT

Session ids can be passed via:

1. URL parameters 
2. Hidden form fields
3. Cookies

SESSION MANAGEMENT

Hidden form fields

Using URL parameters is an obvious and pretty transparent ploy - let's now consider the **hidden form field**

SESSION MANAGEMENT

Hidden form fields

This is once again a pretty impractical attack

The attacker gets the user to login to the target web server using a form which comes from the evil web server

SESSION MANAGEMENT

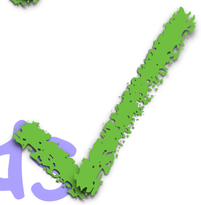
Hidden form fields

The attacker gets the user to login to
the target web server using a form
which comes from the evil web server

Instead of harvesting the session id
it's more likely that the attacker will
try and get the **credentials** of the user

SESSION MANAGEMENT

Session ids can be passed via:

1. URL parameters 
2. Hidden form fields 
3. Cookies

SESSION MANAGEMENT

Cookies

Session fixation using a cookie

Use cross site scripting to inject javascript to set the cookie for a site

http://trustedsite.com/

```
<script>document.cookie="sessioni
d=1234";</script>
```

SESSION MANAGEMENT

Cookies

Session fixation using a cookie

Sets a domain level cookie from a subdomain

Set a cookie on .trustedsite.com from evil.trustedsite.com

SESSION MANAGEMENT

Cookies

Session fixation using a cookie

Inject a META tag to set the cookie on a site

```
<meta http-equiv=Set-Cookie content="sessionid=1234">
```

SESSION MANAGEMENT

Cookies

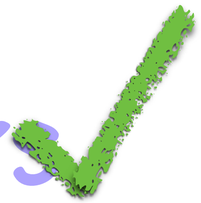
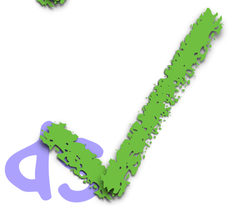
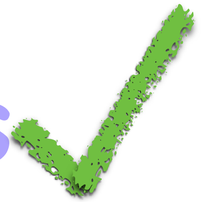
Session fixation using a cookie

Set a cookie by injecting a **header** into the website response

This can be done by attacking the host server, the DNS server or even the network

SESSION MANAGEMENT

Session ids can be passed via:

1. URL parameters 
2. Hidden form fields 
3. Cookies 

SESSION MANAGEMENT

Session hijacking

1. Session ids in URLs, cookies or form fields 
2. Session fixation
3. Session sidejacking or sniffing
4. Cross site scripting
5. Malware or other unwanted programs on the client

SESSION MANAGEMENT

Session fixation

This is a way get access to session information by fixating (finding or setting) another person's session id

SESSION MANAGEMENT

Session fixation

This is a way get access to session information by fixating (finding or setting) another person's session id

It explores a limitation in the way a vulnerable site deals with session ids

SESSION MANAGEMENT

Session fixation

This is a way get access to session information by fixating (finding or setting) another person's session id

It explores a limitation in the way a vulnerable site deals with session ids

Session fixation is possible if the site **does not assign a new id** while validating a user, instead **accepts any session id**

SESSION MANAGEMENT

Session fixation

So how can an attacker set or fix a session id?

SESSION MANAGEMENT

Session fixation

Let's say there is a bank site with poor website security practices:

http://untrustedbank.com

SESSION MANAGEMENT

Session fixation

http://untrustedbank.com

The attacker knows that the victim banks here

SESSION MANAGEMENT

Session fixation

http://untrustedbank.com

This site:

1. accepts any session identifier
2. accepts session ids from query strings
3. has no security validation
4. does not generate a new id on login

SESSION MANAGEMENT

Session fixation

http://untrustedbank.com

Check out the interesting payment features your bank offers!

http://www.untrustedbank.com.com/?
session_id=ATTACKER_FIXATED_ID

SESSION MANAGEMENT

Session fixation

`http://www.untrustedbank.com.com/?`
`session_id=ATTACKER_FIXATED_ID`

The victim is interested and he clicks the link - and logs into his bank account on the site

SESSION MANAGEMENT

Session fixation

`http://www.untrustedbank.com.com/?session_id=ATTACKER_FIXATED_ID`

The untrusted bank **accepts** the session id in the query string - it **does not create a new session** for the user!

SESSION MANAGEMENT

Session fixation

`http://www.untrustedbank.com.com/?`
`session_id=ATTACKER_FIXATED_ID`

Now the attacker has access to the victim's session!

SESSION MANAGEMENT

Session fixation

`http://www.untrustedbank.com.com/?session_id=ATTACKER_FIXATED_ID`

It's when the victim logs in using that session id - that is when the session is **authenticated**

SESSION MANAGEMENT

Session fixation

Now one issue here is that the server accepted a session id which was **not generated** on the server

SESSION MANAGEMENT

Session fixation

Now one issue here is that the server accepted a session id which was not generated on the server

One way to fix this would be to have the server **only accept ids** which were locally generated

SESSION MANAGEMENT

Session fixation

Now one issue here is that the server accepted a session id which was not generated on the server

However servers that accept only server generated session ids are **not necessarily safe** from fixation

SESSION MANAGEMENT

Session fixation

The attacker visits http://untrustedbank.com and gets a session id

SID: SERVER_GENERATED_ID

SESSION MANAGEMENT

Session fixation

The attacker visits http://untrustedbank.com and gets a session id

```
SID: SERVER_GENERATED_ID
```

This is a session id that was generated on the server of the vulnerable website

SESSION MANAGEMENT

Session fixation

http://untrustedbank.com

Check out the interesting payment features your bank offers!

http://www.untrustedbank.com.com/?
session_id=SERVER_GENERATED_ID

SESSION MANAGEMENT

Session fixation

```
http://www.untrustedbank.com.com/?
session_id=SERVER_GENERATED_ID
```

Now once again the victim will use a session which the **attacker has access** to even though the session is server generated!

SESSION MANAGEMENT

Session fixation

Session fixation attacks can also use a **cross subdomain** cookie

SESSION MANAGEMENT

Session fixation

Session fixation attacks can also use a **cross subdomain** cookie

Let's say that http://trustedsite.com allows **untrusted** 3rd parties to use subdomains on their site

SESSION MANAGEMENT

Session fixation

a cross subdomain cookie!

The attacker has the subdomain evil.trustedsite.com and sets a cookie on .trustedsite.com

SESSION MANAGEMENT

Session fixation

When the victim visits www.trustedsite.com the evil session id will be sent to the server as the victim's session id

SESSION MANAGEMENT

Session fixation

The attacker once again knows the victim's session id!

SESSION MANAGEMENT

Session fixation

COUNTER MEASURES

SESSION MANAGEMENT

Session fixation - counter measures

No logins to a chosen session

Set up a new session id for a user only after he is logged in and authenticated

SESSION MANAGEMENT

Session fixation - counter measures

No logins to a chosen session

Forcefully prevent logging in with a chosen session id by **ignoring** any session id presented by the user

SESSION MANAGEMENT

Session fixation - counter measures

No logins to a chosen session

Always change the session id when the user logs in

SESSION MANAGEMENT

Session fixation - counter measures

Use strict sessions

Session ids should be server generated and only after the user logs in and authenticates

SESSION MANAGEMENT

Session fixation - counter measures

Restrict session usage

Bind the session id to the user's network address as seen by the server or to his client SSL certificate

SESSION MANAGEMENT

Session fixation – counter measures

Restrict session usage

Bind the session id to the user's network address as seen by the server or to his client SSL certificate

Every script should check whether the session matches the client's certificate

SESSION MANAGEMENT

Session fixation - counter measures

Allow logout and make logout complete

Session logout should destroy sessions on the server and client

SESSION MANAGEMENT

Session fixation – counter measures

Allow logout and make logout complete

Session logout should destroy
sessions on the server and client

The user should also be able to **log out** from all sessions (current and previous) **explicitly**

SESSION MANAGEMENT

Session fixation – counter measures

Use timeouts on session ids

Use **absolute** session timeouts rather than a keep-alive for sessions

SESSION MANAGEMENT

Session fixation – counter measures

Use timeouts on session ids

Use **absolute** session timeouts rather than a keep-alive for sessions

This prevents attackers from maintaining a trap session or accessing a user's session for a long time

SESSION MANAGEMENT

Session hijacking

1. Session ids in URLs ✓
2. Session fixation ✓
3. Session sidejacking or sniffing
4. Cross site scripting
5. Malware or other unwanted programs on the client

SESSION MANAGEMENT

Session sidejacking

This involves using **packet sniffing** techniques to read traffic between the user and server to get access to the session cookie

SESSION MANAGEMENT

Session sidejacking

Unsecured Wi-Fi hotspots are especially vulnerable to this as anyone sharing the network can access the traffic between other nodes and the access point

SESSION MANAGEMENT

Session sidejacking

Websites should **encrypt all data** between the server and client not just the login information

SESSION MANAGEMENT

Session hijacking

1. Session ids in URLs ✓
2. Session fixation ✓
3. Session sidejacking or sniffing ✓
4. Cross site scripting
5. Malware or other unwanted programs on the client

SESSION MANAGEMENT

Cross site scripting

This allows the attacker to **inject** a **<script>** or **<meta>** tag to set the session id of the user

SESSION MANAGEMENT

Cross site scripting

The lectures on XSS apply to accessing a session id as well

SESSION MANAGEMENT

Session hijacking

1. Session ids in URLs ✓
2. Session fixation ✓
3. Session sidejacking or sniffing ✓
4. Cross site scripting ✓
5. Malware or other unwanted programs on the client

SESSION MANAGEMENT

Malware

These are malicious programs on the victim's computer which can steal his cookies and access session ids

SESSION MANAGEMENT

Malware

This is not related to the web application per se but more to user practices of downloading and installing programs from untrustworthy sites

SESSION MANAGEMENT

Session hijacking

1. Session ids in URLs
2. Session fixation
3. Session sidejacking or sniffing
4. Cross site scripting
5. Malware or other unwanted programs on the client