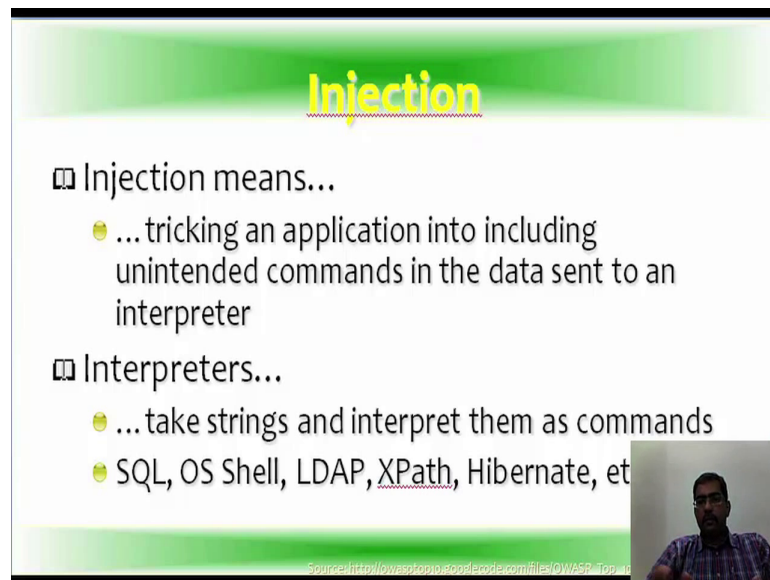


Introduction to Information Security
Prof. Dilip H. Ayyar
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 62

We will now go to OWASP top 10 2013, the first one is injection, the attack vector is easy with prevalence is common, meaning that is a wide spread of presence was there. Detectability is average and impact is severe. So, in the event a injection materializes then the impact is going to be severe.

(Refer Slide Time: 00:38)



Injection

- ☐ Injection means...
 - ...tricking an application into including unintended commands in the data sent to an interpreter
- ☐ Interpreters...
 - ...take strings and interpret them as commands
 - SQL, OS Shell, LDAP, XPath, Hibernate, et

Source: http://www.top10.googlecode.com/files/OWASP_Top_10_2013.pdf

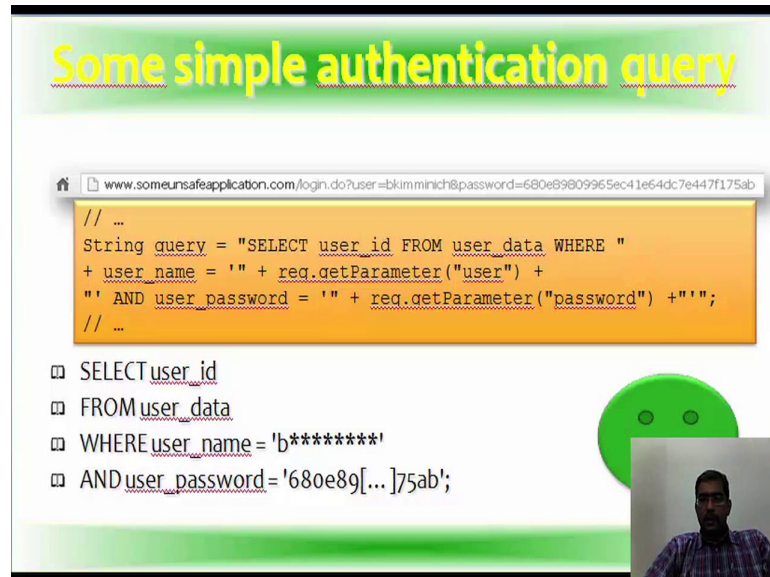
So, what is injection, injection means tricking an application into including unintended commands in the data sent to an interpreter or sent to the server. Interpreters it takes strings and interprets them as commands, like your SQL, OS Shell, LDAP, XPath, Hibernate to all these are kind of things, how at attack is going, first they find a vulnerable site. So, once a vulnerable sites gets found they type in the SQL statements into the text boxes may be to the URL or even inject them into other form fields using request manipulator, for example let us say fiddler or work suite.

Then once the SQL attack vector is gathered, now once they find out that there is an issue they give different statements say for example, for the database version for gathering the actual server name may be select at version may give you the data base version at server name is give you the actual server name. So, if it is some Microsoft they may give select at Microsoft version. So, list of other servers all user accounts or select

start from some user database. So, different SQL commands will be injected in the form fields or in the URL to cause this issue or to exploit.

(Refer Slide Time: 02:26)



Some simple authentication query



```
www.someunsafeapplication.com/login.do?user=bkimminich&password=680e8909965ec41e64dc7e447f175ab
```

```
// ...  
String query = "SELECT user_id FROM user_data WHERE "  
+ user_name = ' + req.getParameter("user") +  
"' AND user_password = ' + req.getParameter("password") + "'";  
// ...
```

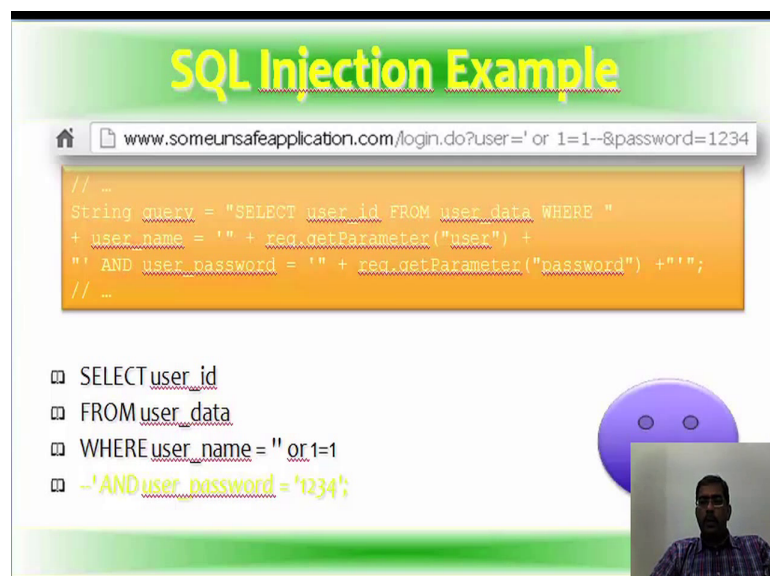
- SELECT user_id
- FROM user_data
- WHERE user_name = 'b*****'
- AND user_password = '680e89[...]75ab';



This is a simple authentication query, where SQL injection is being interpreted, select user id from user data, where user name is something and user password is something else. So, if you see the URL on top some unsafe application login dot do user equal to and password equal to some encode value you put. So, this is how a simple authentication query actually goes through from this you can actually insert certain statements to see whether any error is happening, whether the system handles the injection properly whether filtering of unnecessary characters are done.

(Refer Slide Time: 03:16)

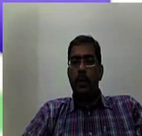
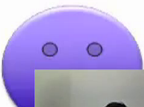
SQL Injection Example



```
www.someunsafeapplication.com/login.do?user=' or 1=1--&password=1234
```

```
// ...  
String query = "SELECT user_id FROM user_data WHERE "  
+ user_name = ' + req.getParameter("user") +  
"' AND user_password = ' + req.getParameter("password") + "'";  
// ...
```

- SELECT user_id
- FROM user_data
- WHERE user_name = " or 1=1
- AND user_password = '1234';



This is a simple SQL injection example, if you see in the URL, you will see after login dot do user equal to inverted comma or 1 equal to 1. So, basically they are trying to do a authentication bypass SQL injection and password equal to 1 2 3 4. So, select user id from user data where user name equal to the SQL or the authentication bypass thread and user password is 1 2 3 4, if you want to learn more about the commands that are to be put in SQL or for XSS, or for that matter authentication by pass, there are lot of cheat sheets available on the net hackers dot org is one.

So, you get readymade attempted scripts for you to use, so you can while doing your assessment of your own server, not with permission you can attempt those commands or those particular scripts which you readymade from the site, OWASP also has the cheat sheets where you can download owasp site to test the applications for SQL injection.

(Refer Slide Time: 04:34)

SQL Injection

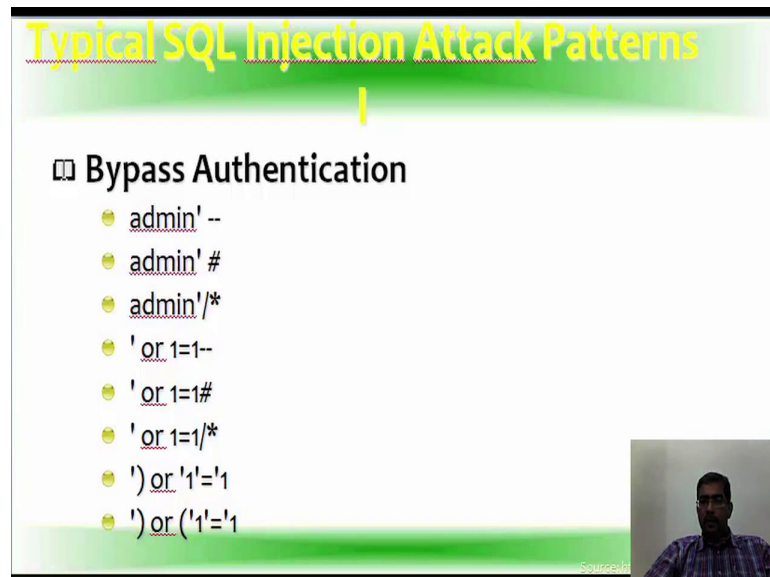
- ❑ Typical Impact
 - Spy out or manipulate data
 - Manipulate the DB server or access underlying OS
 - Bypass authentication or gain admin privileges
- ❑ Correlation with Information Leakage
 - Attackers use error messages or codes to verify the success of an attack and gather information about type and structure of the database
- ❑ Blind SQL Injection
 - If error message don't help the attacker he can still "take a stab in the dark"
 - The normal application behavior (e.g. response time) might give away clues about successful/failed Injection attempts

So, this typical impact for the SQL injection is it is spy out or manipulates data, manipulate the data base server or access the underlying operating system, by pass authentication or gain administrative privileges. So, these are the impact of us successful SQL injection attack materialized. Then, correlation with information leakage is attackers user error message or codes to verify the success of an attack and gather information about the type and structure of the data base and there is something called the blind SQL also.

If the error message do not help the attacker he can still take a stab in the dark or poke in the dark, the normal application behavior that is a response time might give away clues

on successful or failed injection attempts from many of the commercial tools detect SQL and blind SQL very fast. Now, my experience is whatever the tools gives you also whether it is open source or commercial tool try it out on that particular field or particular place where it and see whether it is actually materialized, there may be a lot of false positive also than this canvas may give. So, it is always better to try out a particular injection or a particular field to just confirm that it is actually happening.

(Refer Slide Time: 06:06)



Typical SQL Injection Attack Patterns

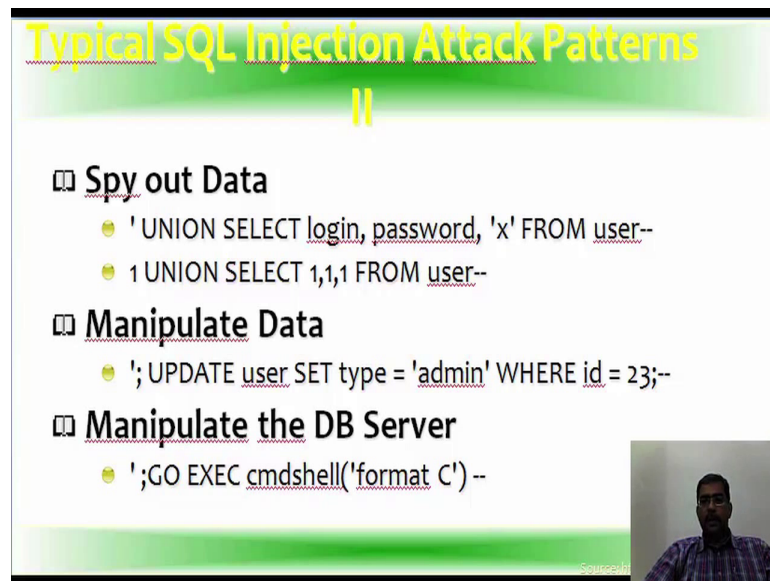
□ Bypass Authentication

- `admin' --`
- `admin' #`
- `admin'/*`
- `' or 1=1--`
- `' or 1=1#`
- `' or 1=1/*`
- `') or '1'=1`
- `') or ('1'=1`

Source: [unreadable]

This is again taken from some cheat sheet, typical SQL injection attack patterns for bypass authentication some of it is listed here, the others you can download from the internet, you can just Google the cheat sheets for a SQL injection you will get a whole lot of resources.

(Refer Slide Time: 06:27)



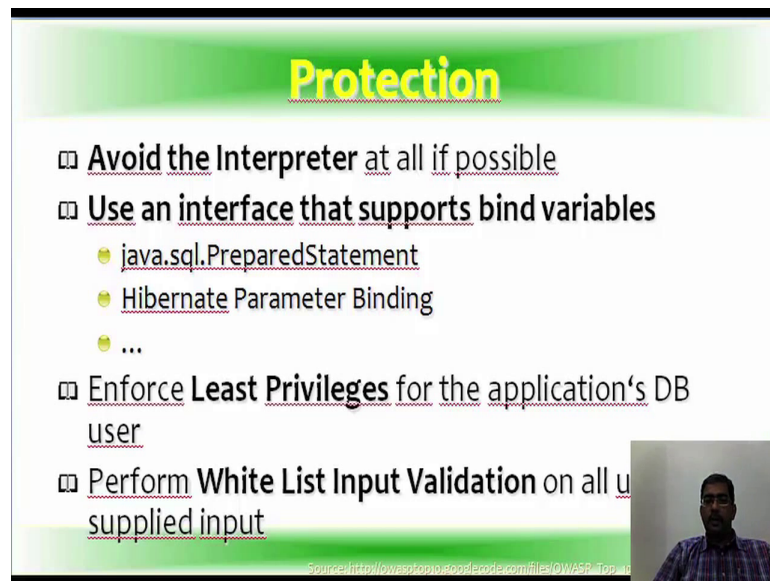
Typical SQL Injection Attack Patterns

- ❑ **Spy out Data**
 - ' UNION SELECT login, password, 'x' FROM user--
 - 1 UNION SELECT 1,1,1 FROM user--
- ❑ **Manipulate Data**
 - '; UPDATE user SET type = 'admin' WHERE id = 23;--
- ❑ **Manipulate the DB Server**
 - ';GO EXEC cmdshell('format C')--

Now, the typical SQL injection attack patterns are one is spy out data, where it is used union select command, manipulate data is updating and manipulate the data base server itself is the executing some kind of command shell or format C. So, some of the syntax and the commands the patterns of how the SQL injection attack happens is given here, you can download again you can download the cheat sheets and try it out the best ideal way to learn is to download a vulnerable application there are at least 30 or 40 vulnerable applications which are there on the net which you can download as a VM or install and try out different possibilities of world's top 10 and other vulnerabilities.

So, those applications are intentionally made vulnerable. So, that the user can learn or the candidate can learn on how actually the exploit happens in what way the commands that were used for the exploit. So, it gives a better perspective on how to secure and also do how to do a proper assessment of a web application.

(Refer Slide Time: 07:44)



Protection

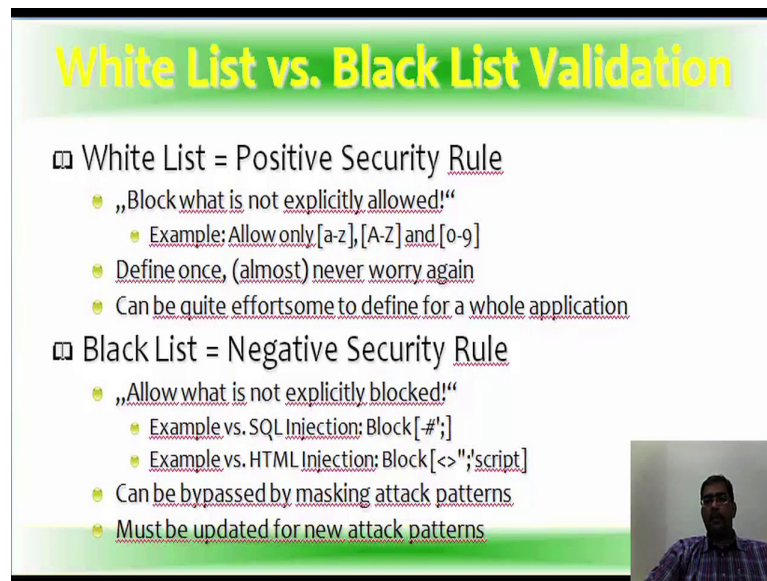
- ❑ **Avoid the Interpreter** at all if possible
- ❑ **Use an interface that supports bind variables**
 - `java.sql.PreparedStatement`
 - Hibernate Parameter Binding
 - ...
- ❑ **Enforce Least Privileges** for the application's DB user
- ❑ **Perform White List Input Validation** on all user supplied input

Source: http://www.sprono.com/code.com/files/QW57_top

What is the protection for SQL injection, avoid the interpreter at all if possible, use an interface that support bind variables like, Java, SQL prepared statement or hibernate parameter binding. Enforce the principle of least privileges for the applications data base user, perform white list input validation on all user applied input. Now, you should always implement server side validations to preventive attack, the fundamental thing you have to learn in a web application is you can never trust client site validations at anything that is installed locally on the client site can be subverted or sub commented or bypass by an attacker.


So, they can also see what you are checking for, if you are putting a client site validation which we give them even more information with what to attack. So, the input must be validated, all input must be validated at server site, this is a very wider rule since website attack originate from the client, such as a web browser and data that it is sends to a server cannot be trusted. Some, attack like your SQL injection, cross site scripting and many other vulnerabilities can be plotted or can be secured by adherent to a very simple group do not trust client site data.

(Refer Slide Time: 09:28)



White List vs. Black List Validation

- ❑ White List = Positive Security Rule
 - „Block what is not explicitly allowed!“
 - Example: Allow only [a-z],[A-Z] and [0-9]
 - Define once, (almost) never worry again
 - Can be quite effort-some to define for a whole application
- ❑ Black List = Negative Security Rule
 - „Allow what is not explicitly blocked!“
 - Example vs. SQL Injection: Block[#';]
 - Example vs. HTML Injection: Block[<>"';script]
 - Can be bypassed by masking attack patterns
 - Must be updated for new attack patterns



We will see white list versus black list validation we mentioned in the previous slide, white list is a positive security rule that is block what is not explicitly allowed. Example, allowed only small a to z capital A to Z and 0 to 9, you define once and then almost never worry again, because it is already set it can be quite effort-some to define for a whole application, but still you have to define it once. Now, there is a negative security rule on black list allow what is not explicitly blocked. So, you have to write on parameter which have block.

So, whatever is not there in that list only will be allowed example the hash, the semicolon, inverted colon, in HTML injection, greater than less than sign, semi colon, script, inverted colon. So, it can be bypassed by masking attack patterns must be updated for new attack pattern. Now, a lot of this can be encoded and actually be put in the URL also it can be masked so, that particular input is not recognized by the server, so that the validation will not happen. So, it is always better to go for a positive security rule and enforced that in the application even though it will be difficult one time you set it up and then there can be an ease of mind.

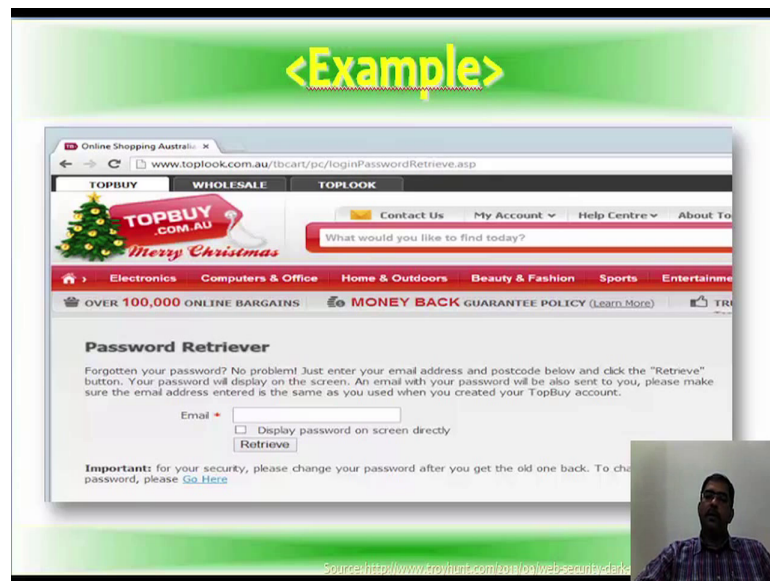
(Refer Slide Time: 11:08)



Remember, to always sanitize your data base input and this is very famous cartoon it just been going on as the SQL injection example, which is use by almost everybody who talks about SQL, now this is a scenario a mother gets a call. So, there from the other end the party says hi this is your son's school, we are having some computer trouble. So, the mother ask oh dear did you break something for this school responds saying, in a way.

Did you so, school asks her did you really name your son Robert inverted comma drop table student thats the name. Oh yes little bobby tables we call him. Well, we have lost this year's student records I hope you are happy and the mother responss yeah and I hope you have learned to sanitize your data base inputs. So, this is a very nice cartoon of how SQL injection can happen and in four little cartoons it says that you know you have to sanitize your data base inputs before it is true, then we come to A 2 which is broken authentication and session management.

(Refer Slide Time: 12:30)



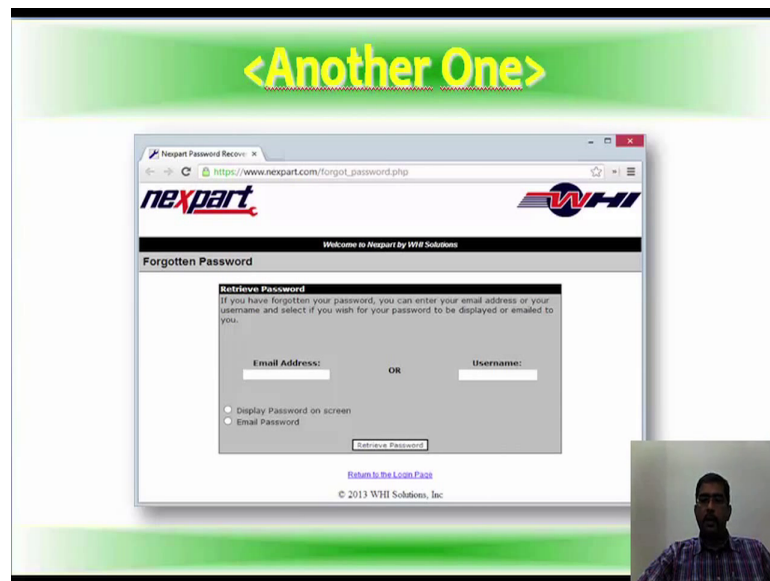
An example of this is given here, in this example you see the password retriever, they forgot your password no problem, just enter your email address and your postal code below, click retrieve button your password will be displayed on the screen and email with your password will also be sent to you, please make sure the email address entered is the same as you used when you created your top buy account.

So, for your security please change your password after you get out of the old one back to change a password please go here. Now, the developers in this case wrote out their own implementations of the session, it works nearly the same way as requiring how the cookies work, but resulting in several dollars spent out several money spent and make the application even less secure.

So, what we have to learn from this slide is session management is very important, in this case session management is not implemented properly and probably in this the largest and most vulnerable point is the session id's. But, then what are sessions it is a part of the art of session management it is storing of data on the server for later use. for now if you have a session id, where do I store it in cookies, in query strings.

So, the session management becomes a crucial factor in the development of the application and also you should do it in such a way that your session is not attacked there cannot be a manageable middle attack or session hijacking or cookie poisoning. So, those kind of issues I have to be taken care off.

(Refer Slide Time: 14:36)



So, an another example similar to the first one here enter your email address or the user name display password on screen or email password, very insecure implementation.

(Refer Slide Time: 14:50)

Broken Authentication and Session Management

- ❑ HTTP is a “stateless” protocol
 - Credentials have to be passed with every request
 - Should use SSL for everything requiring authentication
- ❑ Session Management flaws
 - SESSION ID is just as good as credentials to an attacker
 - SESSION ID is typically exposed on the network, in browser, in logs, ...
- ❑ Beware the side-doors
 - Change my password, remember my password, forgot my password, secret question, logout, email address, credentials stored in plain text in database, etc...
- ❑ Typical Impact
 - User accounts compromised or user sessions hijacked

Source: http://owasp.org/Top1000A2013/OWASP_Top_1000A2013.pdf

Then we have already discussed that http is a stateless protocol, credentials have to be passed with every request, it should be use SSL for everything requires authentication, there are session management flaws like session id is just as good as giving credential to an attacker, session id is typically exposed on the network in the browser, in logs and then beware of the side doors that is change my password, remember my password, forgot my password, secret question, logout, email address, credentials which are stored in plain text in data base. Now, those are all the side-doors for getting inside.

What are the typical impacts or what is the typical impact of this? User accounts are compromised or user sessions are hijacked. How the attacker do it? Now, the attackers or the hackers will intercept the session id either from the cookie or the request URL. So, they can replicate the session id themselves. Now, URL's are easy, they simply type it into your own browser cookies are tougher, but if they can write a cookie or inject the cookie into a HTTP requested header they can rip the server.

So, the ideal things to do is avoid a cookie session avoid using home grown authentication team, where to look into the option of IP checking then for some cases double check password for certain activities or make the user enter the password twice use SSL and the most common vulnerability or the most common loop holes that these applications have is the sessions do not expire early, there are the time of for the session is perpetual In some cases. Now even during assessment of certain application when we escalate this to the vendor he says the client does not want to have a session time out, so what can I do. Where this is actually a issue, where the session hijacking, the cookie poisoning all those things can happen, if these sessions do not expire early or if we does not expire often.

The user is just lazy to come back and re type his login and password. So, he does not want the session to expire, which is the very back perspective from the security when we said avoid cookie less sessions. say in web dot config file set cookie less equal to false these does not completely solve the problem, but it makes the task of cracking a whole lot difficult for the attacker.



(Refer Slide Time: 18:00)

Good Session ID Generator?

- h5kek4z9halrtrf
- gj7513k7hb15rtr
- 18165k45hc1rw7i
- p05jrj53hdli039
- 5urltda1he1bn46
- j51e97h9hf2yq3h
- po9531d7hg2awi9
- t6zhj2n5hh27bn0
- iu345r53hi2aw34
- o0z43411hj2njk1
- 9por42o9hk3dfrz
- ...

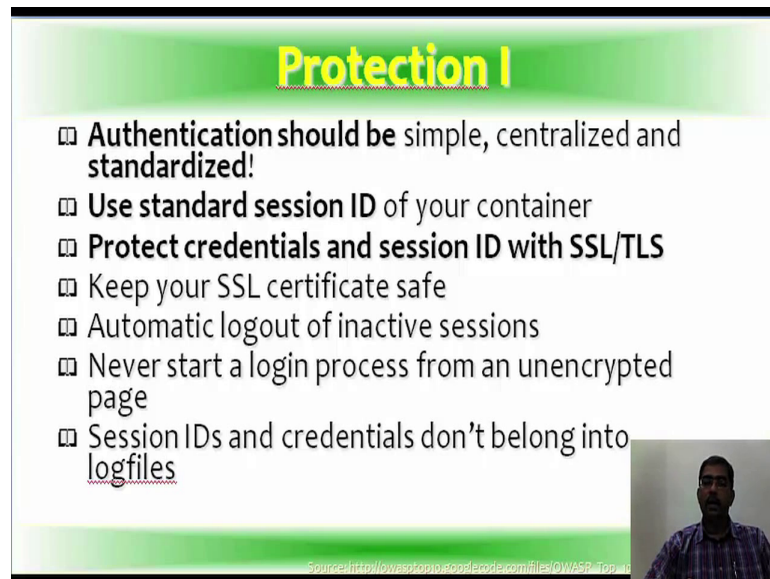
Pattern

- 9,7,5,3,1,9,7,5,3,1,9...
- h.h.h.h.h.h.h.h.h.h...
- a.b.c.d.e.f.g.h.i.j.k...
- 1,1,1,1,1,2,2,2,2,2,3,...

Is this a good session id generator. Now, if you see on the left on side these session id's are generated and there is a pattern into this if you look at it, the pattern is on the right side if you see the first one the 9, 7, 5, 3, 1 again 9, 7, 5, 3, 1 is repeated then the one next to that is fully filled with h then a, b, c, d, e, f, g, h 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3. So, is it a good session id generator it is not, because there is a pattern which is found in the session.

(Refer Slide Time: 18:39)



Protection I

- ❑ **Authentication should be simple, centralized and standardized!**
- ❑ **Use standard session ID** of your container
- ❑ **Protect credentials and session ID with SSL/TLS**
- ❑ Keep your SSL certificate safe
- ❑ Automatic logout of inactive sessions
- ❑ Never start a login process from an unencrypted page
- ❑ Session IDs and credentials don't belong into logfiles

Source: http://lowres.toni.doodlecode.com/files/QWASD_Top.jpg

What is the protection for this authentication should be simple, it should be centralized and it should be standardized. So, use standard session id of your container, protect credential and session id with SSL or TLS keep your SSL certificate safe, automatically logout inactive sessions, never start a login process from an unencrypted page and session id's and credentials that do not belong in to the log files. So, you should never log the session id and credential in the log file, these are some of the measures or protection measures.