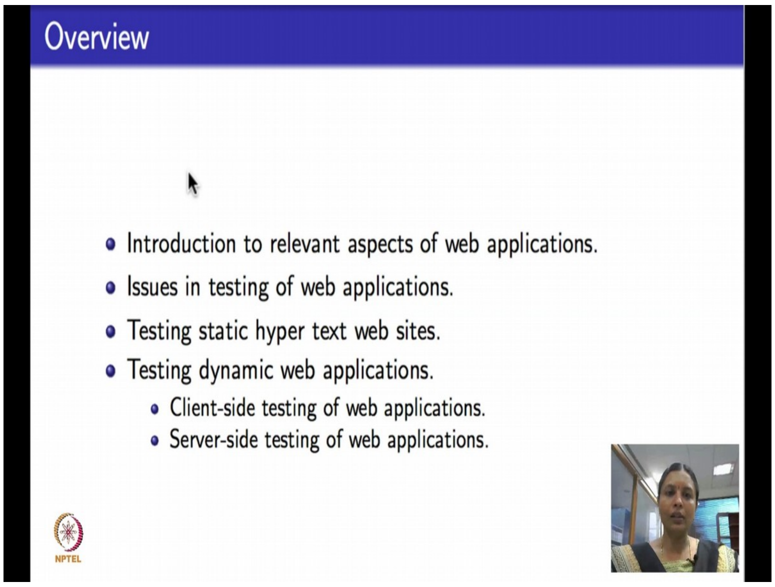
Software testing

Prof. Meenakshi D’Souza
Department of Computer Science and Engineering
International Institute of Information Technology, Bangalore

Lecture – 46
Testing of Web Applications and Web Services

Hello again, welcome to the second lecture of week 10. If you remember we had started looking at web applications testing in the last lecture.

(Refer Slide Time: 00:20)



Overview

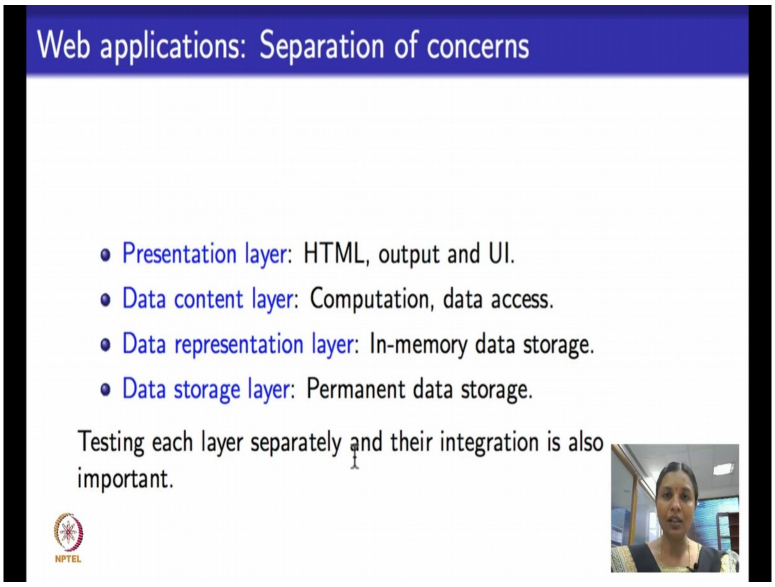
- Introduction to relevant aspects of web applications.
- Issues in testing of web applications.
- Testing static hyper text web sites.
- Testing dynamic web applications.
  - Client-side testing of web applications.
  - Server-side testing of web applications.

I had given you the following overview. So, what I did in the last lecture was to introduce you to relevant aspects of web applications it by no means has comprehensive introduction of web application, but as we would need it from the point of view or testing and criteria based testing we saw the basics. And I also talked about the issues that typically come up while testing web applications and then we ended the lecture with the brief mention of how to test static HTML, plain HTML web sites.

You basically model it as a graph where the nodes of HTML pages and the edges are hyperlinks and then that would check for typical link be working fine dead links and so on. What we going to do in the next two lectures is to look at dynamic web applications which is the last bullet her. Dynamic web applications typically have a web server as we saw in the last class where all the software of the web application reside, client access the software in the web server and the web page that the client is accessing from the server gets dynamically generated based on how the client interacts with the software residing in the server.

So, you could test this by testing on the client side of the web application or by testing on the server side of the web application. The code on the client side basically talks about the script validation that happen at the level of client. The code on the server side is fairly big large complicated web software. What we will do in this module is we will understand how to do clients side testing of web applications. In the next module I will tell you how to do server side testing of dynamic web applications.

(Refer Slide Time: 02:08)



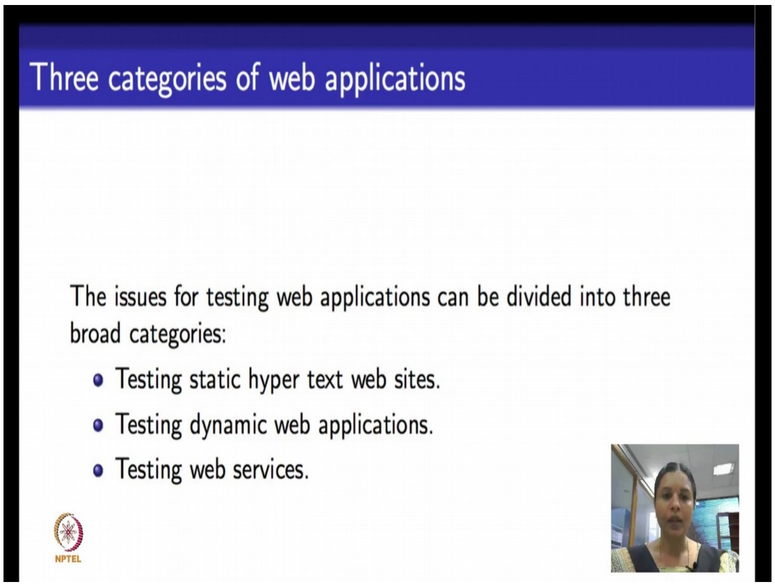
Web applications: Separation of concerns

- Presentation layer: HTML, output and UI.
- Data content layer: Computation, data access.
- Data representation layer: In-memory data storage.
- Data storage layer: Permanent data storage.

Testing each layer separately and their integration is also important.

And using the same slide deck continuing with the same slide deck that I did, so I will skip past this slides except for recapping whatever is needed. Please remember that web applications, architecture of web applications typically at a high level I have this several layers in which software is present. Each layer is typically separately tested their integration is also tested for a large fragment of the testing that we are going to see in this module and next module for web applications, we will stick to presentation layer may be I will briefly touch upon the data content layer. The software that we will be testing as a web application mainly resides only in the presentation layer. For client side and server side testing we will be consider the software as it is present in the presentation layer.

(Refer Slide Time: 02:59)



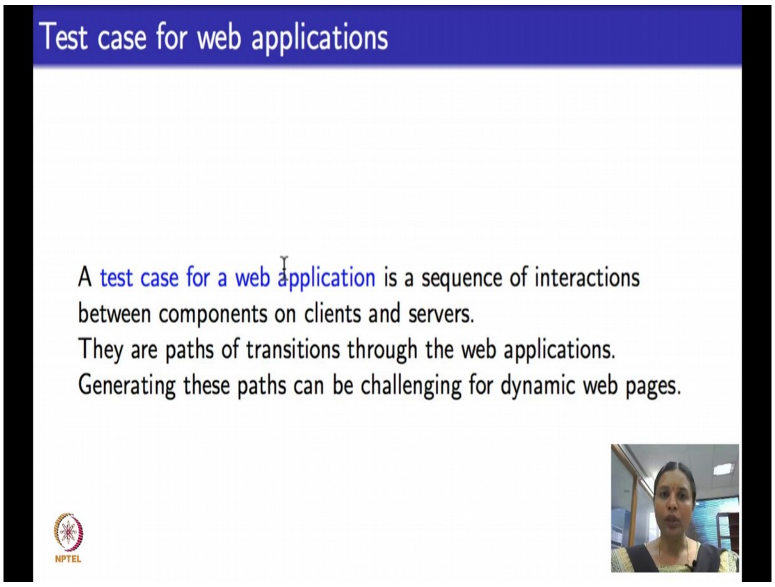
Three categories of web applications

The issues for testing web applications can be divided into three broad categories:

- Testing static hyper text web sites.
- Testing dynamic web applications.
- Testing web services.

So, I am skipping past the rest of this slide which we already discussed in the last lecture. So, as I told you that issue of testing web application is broadly divided into three categories testing static web sites, testing dynamic web applications where we will see server and client separately and then testing web services which we are going to skip.

(Refer Slide Time: 03:22)

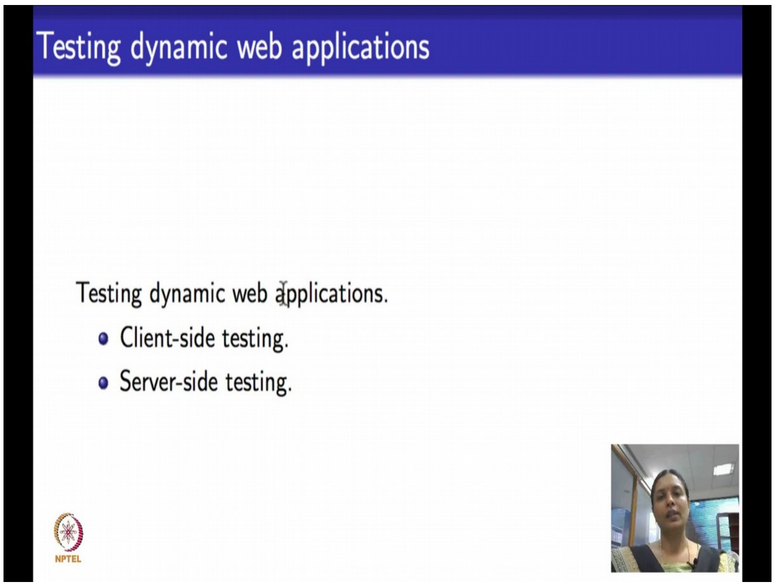


Test case for web applications

A test case for a web application is a sequence of interactions between components on clients and servers.
They are paths of transitions through the web applications.
Generating these paths can be challenging for dynamic web pages.

Last module I introduce you to a web page a static web page a dynamic web page and how a test case looks like. It is basically a sequence of interactions between components software components that lie on clients or servers.

(Refer Slide Time: 03:43)



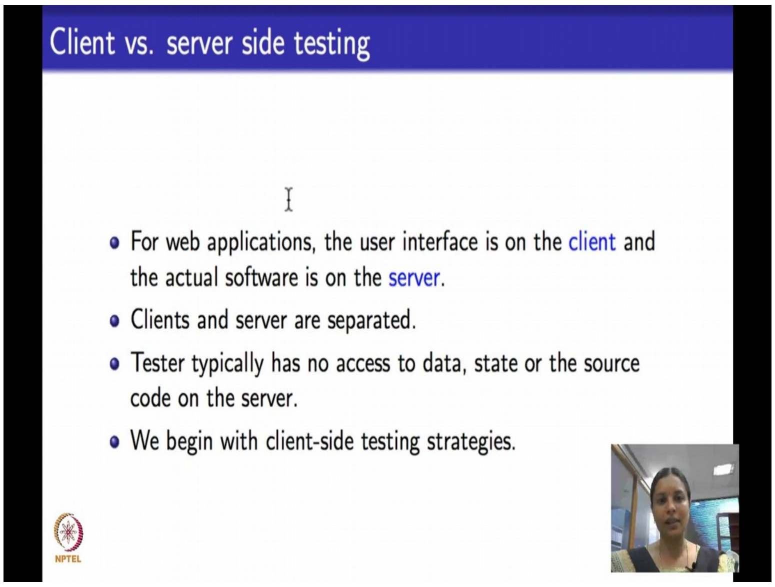
Testing dynamic web applications

Testing dynamic web applications.

- Client-side testing.
- Server-side testing.

And we also saw how to test for static web pages has graphs. Now we going to begin with this. We are going to test dynamic web applications. Separately we going to see two things: client side testing of dynamic web applications and server side testing of dynamic web applications. We begin with client side.

(Refer Slide Time: 03:56)



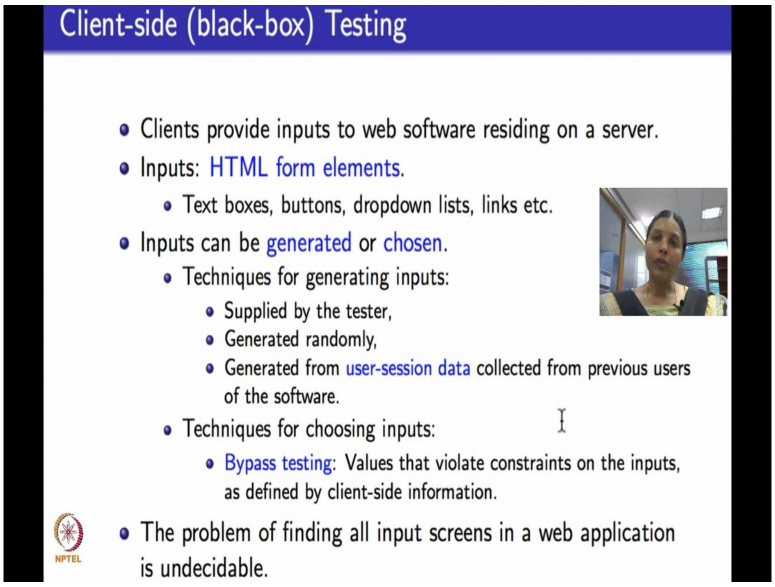
Client vs. server side testing

- For web applications, the user interface is on the client and the actual software is on the server.
- Clients and server are separated.
- Tester typically has no access to data, state or the source code on the server.
- We begin with client-side testing strategies.

So, what is the difference what is client versus what is server? Server is the main place where the web application software resides different clients from different pages could access the server. Like for example, if you take our Indian railways ticket booking system the main software will reside in a centrally located web server where the data base of the IRCTC will be made accessible. On that software various clients could access the web application based on each ones booking preferences. The clients are spread across India right. So, the user interface for web application is typically on the client and the actual software resides on the server. Clients and server are separated, geographically, the software, everything is separated.

Tester typically has no access to the data directly on the server that is a lot of data on the server, tester does not know this state of the server and typically black box tester for system level testing of web applications does not have access to the source code that is present in the server also. Now we going to see testing strategies that still manage to test in the presence of all these limitations. As I told you we will begin with client side testing in this lecture and then look at server side testing in the next lecture.

(Refer Slide Time: 05:21)



Client-side (black-box) Testing

- Clients provide inputs to web software residing on a server.
- Inputs: HTML form elements.
  - Text boxes, buttons, dropdown lists, links etc.
- Inputs can be generated or chosen.
  - Techniques for generating inputs:
    - Supplied by the tester,
    - Generated randomly,
    - Generated from user-session data collected from previous users of the software.
  - Techniques for choosing inputs:
    - Bypass testing: Values that violate constraints on the inputs, as defined by client-side information.
- The problem of finding all input screens in a web application is undecidable.

So, we what we going to do is system level black box testing of the client code. So, what does the client do? Client provides inputs to the web software. Where is the web software? The web software resides on the server.

What is a typical input that a client provides? The client could be filling up a form that is there is a web interface. So, the inputs could be HTML form elements. What are the various HTML form elements that are typically provided as a inputs to the client? They could be text boxes where you enter fields of texts. Like for example, if you go back to the same ticket booking system typically there is a text box where you enter your user name followed by password that is encrypted and sent or they could be buttons, buttons in the sense of radio buttons, options for you to choose. Again going by the same ticket reservation example suppose you entered your beginning of the journey location and the destination location then the web system comes up with the list of trains that ply from the source to the destination.

And there is one button to choose which train you want. Based on the button that you press the corresponding train gets chosen and the availability and other details about that train is displayed to you. It could also be a drop down list a classical example of a drop down list is when you filling your address, you will realize that several websites give a drop down list of all the countries that are available. They will not let you type down your country. Similarly if you are filling your address in an Indian website there is a drop down list of all the states that are available.

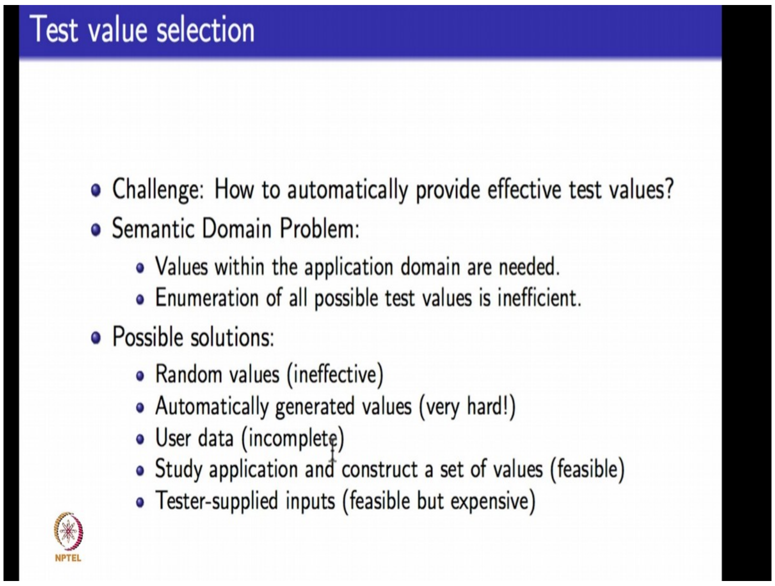
Those are various kinds of HTML form elements that we provide to create content for a HTML page. So, inputs to a client are basically HTML form elements of this kind. Now for testing these inputs can be chosen from a set or they could be generated from a set. So, what are the techniques that are used for generating inputs? Typically inputs are supplied by tester, tester could have lot of domain knowledge about the concerned applications and knows what to effectively test and supplies the test inputs or input could be randomly generated. Usually random generation of inputs is not considered very effective because you do not know what you are testing. Or, inputs could be generated from what is called a user session data which is basically collected from previous users of the software.

Each one has, each user has a software and then you store the users session data which is very easily storable if you have things like cookies and all that stuff and based on that you can manipulate that data to create test input. This is considered to be very effective and there are a couple of papers that discuss this technique of generating test inputs based on user session data and show how these techniques have been useful. I will point you to those papers towards the end of the lecture on web applications in the list of references. Now when it comes to techniques for choosing inputs to be given one popular technique that we will discuss in this lecture is what is called the technique of bypass testing.

What is the word bypass in English tell you? It tells you that you take some inputs, but you bypass, bypass in the sense you bypass by not giving inputs of right kind or not giving input values at all which means you give values that violate some constraints or the other on the inputs and the constraints where are they gotten from, they are gotten from clients side right. So, I will illustrate to you the technique of bypass testing through examples in this lecture. Now you might say now I told you inputs can be generated or chosen and we gave you high level overview of the various techniques.

But you might ask what about exhaustive testing, can I generate all possible inputs to a web application? And intuitive naturally correct answer to that would be the problem is very difficult. In fact, the problem is undecidable, you cannot find all inputs screens for a web application.

(Refer Slide Time: 09:24)



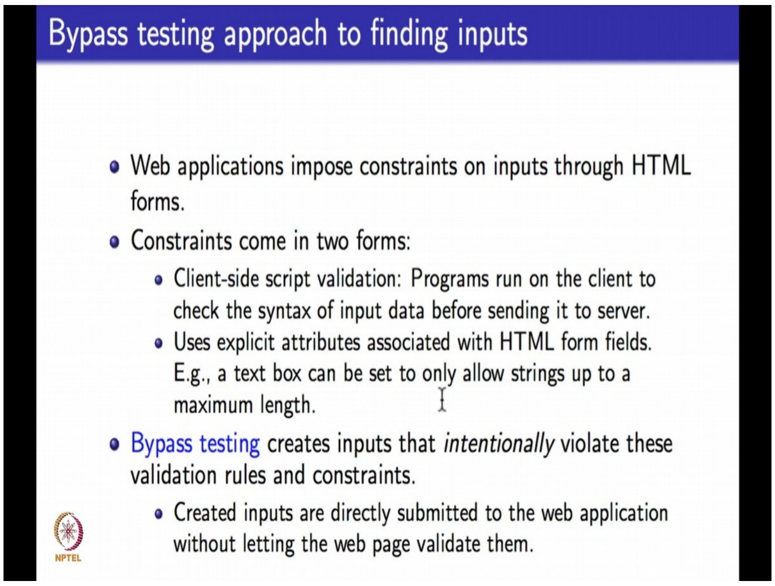
Test value selection

- Challenge: How to automatically provide effective test values?
- Semantic Domain Problem:
  - Values within the application domain are needed.
  - Enumeration of all possible test values is inefficient.
- Possible solutions:
  - Random values (ineffective)
  - Automatically generated values (very hard!)
  - User data (incomplete)
  - Study application and construct a set of values (feasible)
  - Tester-supplied inputs (feasible but expensive)

So, what we are going to do is we are going to do bypass testing before we move into bypass testing let us look at what is the test value selection problem. The challenge now is I want to be able to provide test values. How do I do? The value should be such that the values are within the application domain and I told you in the previous slide that I cannot generate all possible test values.

We discuss the random values could be possible, it is ineffective, automatically generate test values. How do you automatically generate test values? You intuitively understand that its hard problem right because this test inputs have a lot of structured. And if its user session based data you may not you might get test values that are valid, but they might be in effective from the point of view of testing to identify of loss. We study an application and construct a set of values that is a feasible technique, bypass testing that we will be seeing comes under this class.

(Refer Slide Time: 10:16)



Bypass testing approach to finding inputs

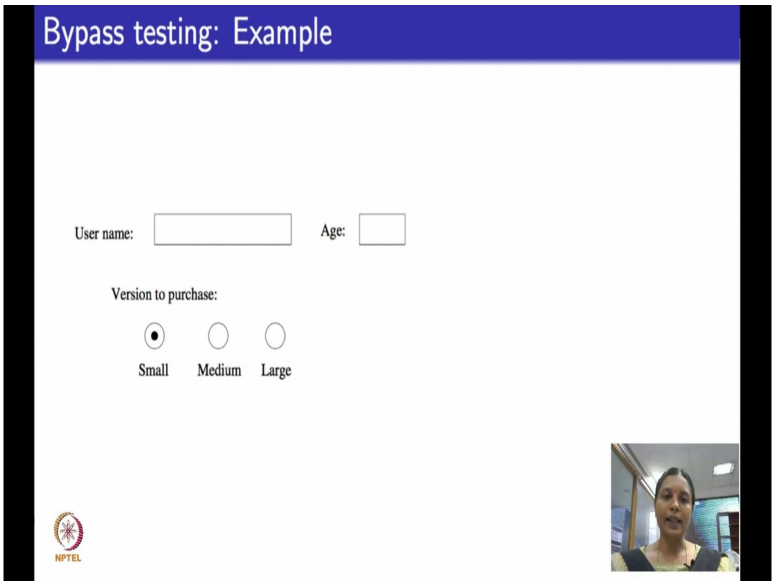
- Web applications impose constraints on inputs through HTML forms.
- Constraints come in two forms:
  - Client-side script validation: Programs run on the client to check the syntax of input data before sending it to server.
  - Uses explicit attributes associated with HTML form fields. E.g., a text box can be set to only allow strings up to a maximum length.
- Bypass testing creates inputs that *intentionally* violate these validation rules and constraints.
  - Created inputs are directly submitted to the web application without letting the web page validate them.

So, what is bypass testing? So, web applications typically impose constraints on the kind of inputs that they get through HTML forms right. What are the constraints that look like? Constraints can come in two different forms they could be constraint which is validated at the level of client itself by writing scripts for validating these constraints. What are these scripts? These scripts are basically programs software programs that run on the client to check whether the input data is of correct syntax like for example, let us say there is a field where you have to enter a name you obviously, cannot enter numbers there it has to be only alphabets.

Similarly another kind of constraints could be a explicit attributes that are associated with HTML form fields. Like for example, if you are entering age then you cannot enter a five digit number because this going to be nobody who is that old who is currently living. So, there could be constraints that could be validated by running scripts on the clients or they could be constraints that can be set by setting explicit attributes on the HTML form fields. We will see examples of both these kind of constraints. What does bypass testing do? Bypass testing now creates inputs that intentionally violate all these constraints. So, they create inputs and submit the violated inputs directly to the server by bypassing the script validation that happens on the client side.

So, the inputs that violate the validation rules and constraints are directly submitted to the web application that resides on the server and the idea is to test and see how that web application reacts to these intentionally violated constraints. The client had script validation is bypassed hence the term bypass testing.

(Refer Slide Time: 12:07)



Bypass testing: Example

User name: Age:

Version to purchase:

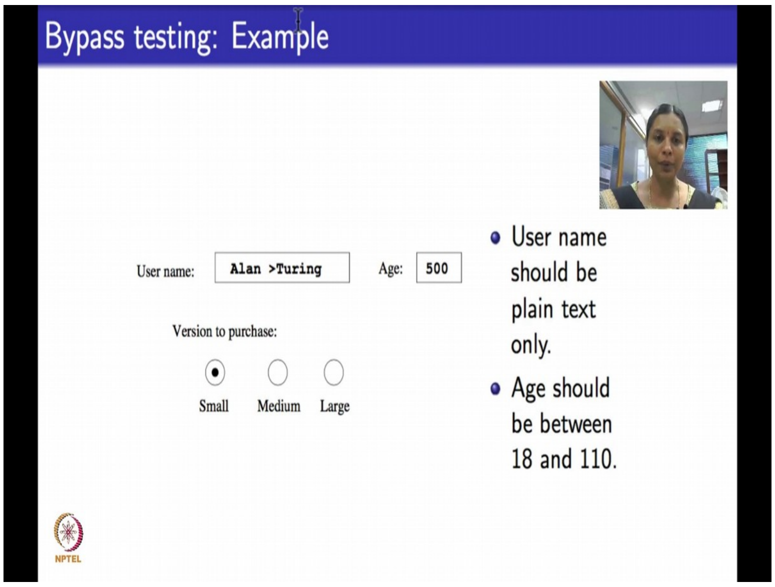
[✓ Small] Medium Large

So, for an example let us say this is just a partial form of some web page where you ask to enter a user name, you are ask to enter your age and then you are ask to choose, you are buying a product let us say you are ask to choose the version that you want to purchase. It could be a small, it could be a medium or it could be a large.

This is how the normal input that is presented on a client page of a web application looks like. What are the classical inputs that you expect to we entered in this HTML form field? User name you expect an proper name to be entered which means what it is should not have special characters it should not have numbers, age, you expect typically a two digit number to be entered you clearly do not want to enter a age which says its 700 or 1100 and so on right.

What will bypass testing do? If you enter wrong things typically at the client which is at the web page itself there will be simple validation rules like the kind that I showed you here right. Script validation rules that will tell you that the data that you have entered is wrong correct it. Bypass testing, what it does is that it bypasses the clients side script validation, enters wrong data and tries to directly send that data to the server.

(Refer Slide Time: 13:25)



Bypass testing: Example

User name: Alan >Turing Age: 500

Version to purchase:

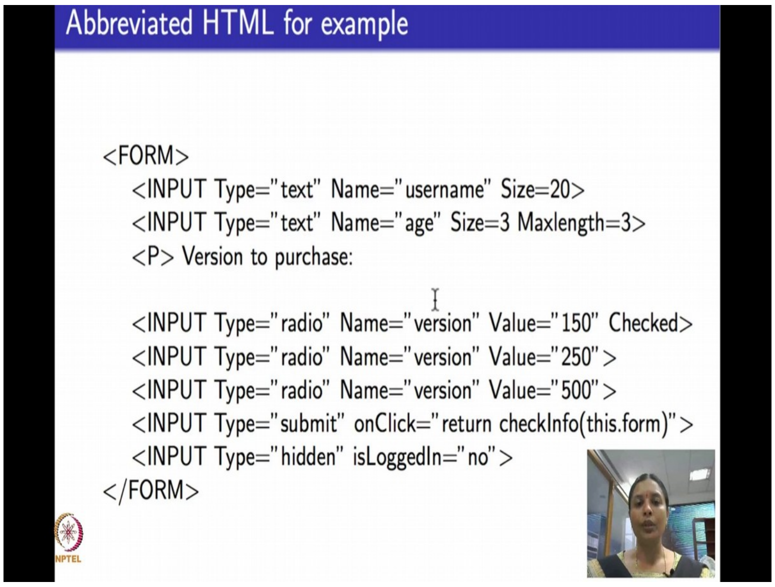
[✓ Small] Medium Large

- User name should be plain text only.
- Age should be between 18 and 110.

For example, if I am doing by pass testing for the same HTML form in the where I enter a user name I will insert a special character like this here I will say Alan greater than Turing. There should be a problem because special characters are not allowed in names, and age, I will enter some nonsensical number like 500 right.

Now, I will directly send this input by bypassing the client side validation to the software on the web server and see how the web software reacts. How do I directly send this input? How do I bypass the client side script validation? What do we do? We do the following.

(Refer Slide Time: 14:01)



Abbreviated HTML for example

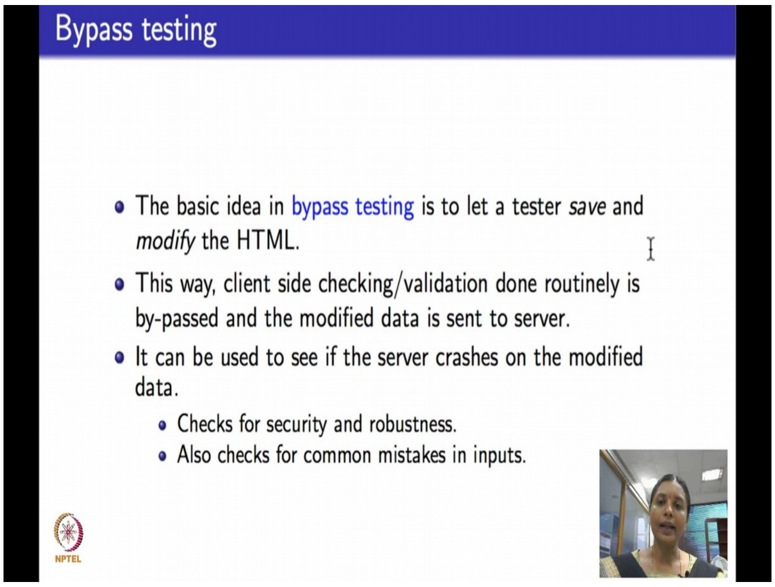
```
<FORM>
   <INPUT Type="text" Name="username" Size=20>
   <INPUT Type="text" Name="age" Size=3 Maxlength=3>
   <P> Version to purchase:

   <INPUT Type="radio" Name="version" Value="150" Checked>
   <INPUT Type="radio" Name="version" Value="250">
   <INPUT Type="radio" Name="version" Value="500">
   <INPUT Type="submit" onClick="return checkInfo(this.form)">
   <INPUT Type="hidden" isLoggedIn="no">
</FORM>
```

If the input was directly actually generated the corresponding HTML file that would be generated on the client site would look somewhat like this. So, there will be a form field, form tag in the HTML which will have so many inputs. It says the first input is of type text only text which means no special characters and numbers I used and the attribute that is entered will be user name which is passed to name and its size should be maximum 20 characters. You could increase or decrease this length based on what it is.

The next box is of type text the size should be the value that is entered is of age, size should be three and it cannot be more than length 3. Then the next box which says version to purchase could be its a radio button as we saw here these are call radio buttons small medium large, three versions to purchase small gets let us say some static value 150, medium gets another value 250, large gets an another value 500. And then typically there will be a submit button which I have not depicted here and then when you submit you on click right you return check info this form.

This is the place where the validation happens. Check info this form, check whether this form is correct. And then there is some other flag which is set which is logged in. I want to bypass this validation and directly if we wrong input to the server.

(Refer Slide Time: 15:30)



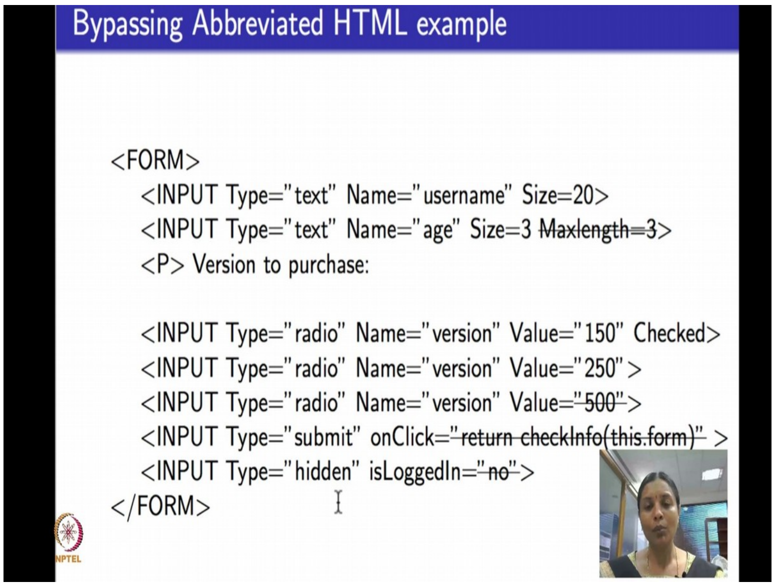
Bypass testing

- The basic idea in bypass testing is to let a tester *save* and *modify* the HTML.
- This way, client side checking/validation done routinely is by-passed and the modified data is sent to server.
- It can be used to see if the server crashes on the modified data.
    - Checks for security and robustness.
    - Also checks for common mistakes in inputs.

So, idea is to let a tester save this kind of a HTML and directly modify it. This way the client side checking or validation that is run routinely as a part of the client's script validation is bypassed and the modified data will be directly send to the server.

What can you use it from the point of view of testing? You can use it to see if the server crashes on the modified data and you can also use it for saying how the secure a server is secure how you robust it, does it a handle corrupt data well? It also checks for server better be robust for common mistakes on the inputs. So, it also checks if the server is robust enough for that. So, for the same example here, this is the HTML that corresponds to a normal form.

(Refer Slide Time: 16:20)



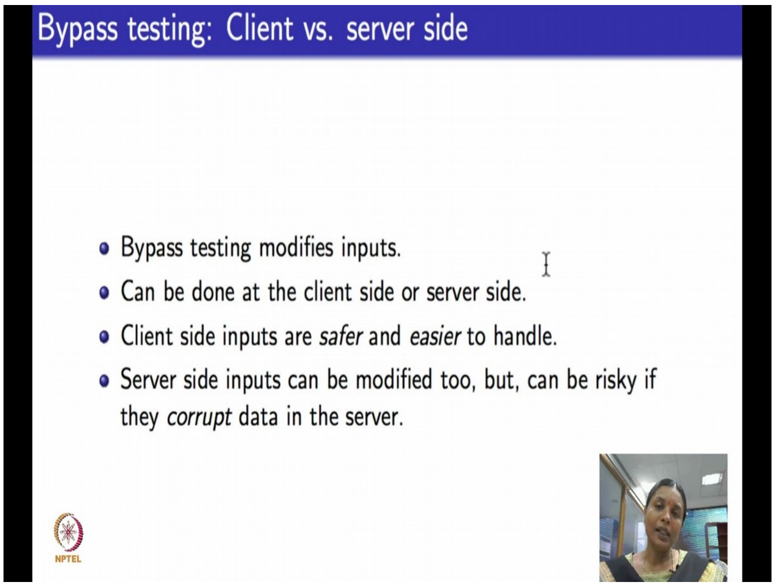
Bypassing Abbreviated HTML example

```
<FORM>
   <INPUT Type="text" Name="username" Size=20>
   <INPUT Type="text" Name="age" Size=3 ~~Maxlength=3~~>
   <P> Version to purchase:

   <INPUT Type="radio" Name="version" Value="150" Checked>
   <INPUT Type="radio" Name="version" Value="250">
   <INPUT Type="radio" Name="version" Value=~~"500"~~>
   <INPUT Type="submit" onClick=~~"return checkInfo(this.form)"~~ >
   <INPUT Type="hidden" isLoggedIn=~~"no"~~>
</FORM>
```

How do I bypass it I will modify this HTML, what are the modification that I will do to the HTML? I have depicted like this. I will remove this I will remove the part that says max length is three that let us me enter any garbage value that is a number for age.

And I will remove let us say some other modification, I could remove the value for large. And are the most important thing is I will remove this script corresponding to validation. This is the part where the script is validated, check info this form, I will remove it right. So, and now this HTML I will directly sent to the server. So, and then basically test how the server reacts to it.

(Refer Slide Time: 16:52)

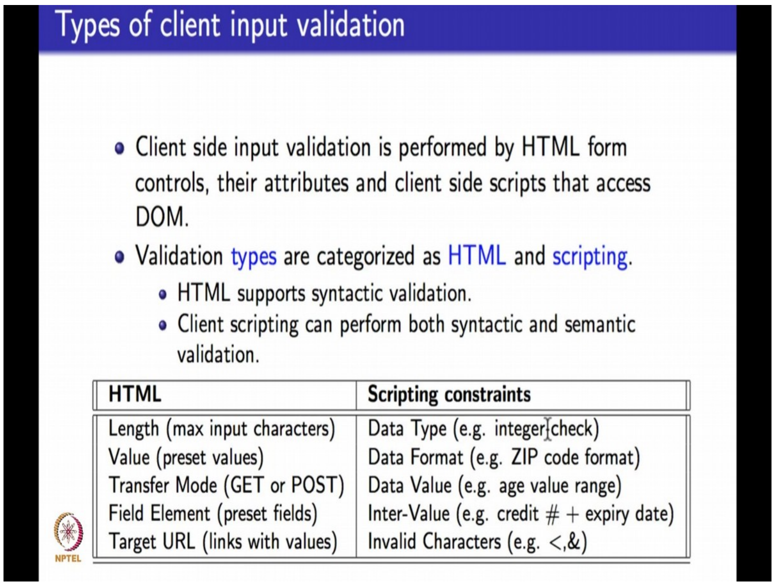


Bypass testing: Client vs. server side

- Bypass testing modifies inputs.
- Can be done at the client side or server side.
- Client side inputs are *safer* and *easier* to handle.
- Server side inputs can be modified too, but, can be risky if they *corrupt* data in the server.

So, it modifies that the inputs it can be done on the clients side or server side I showed you how to do it on the client side through an example. Client side inputs are always say for an easier to handle. Server side, unless you are a web application developer of the software that directly resides on your server. We typically do not recommend that you do bypass testing on the server because they could end up corrupting data in the server right.

(Refer Slide Time: 17:20)



Types of client input validation

- Client side input validation is performed by HTML form controls, their attributes and client side scripts that access DOM.
- Validation types are categorized as HTML and scripting.
  - HTML supports syntactic validation.
  - Client scripting can perform both syntactic and semantic validation.

HTML	Scripting constraints
Length (max input characters)	Data Type (e.g. integer check)
Value (preset values)	Data Format (e.g. ZIP code format)
Transfer Mode (GET or POST)	Data Value (e.g. age value range)
Field Element (preset fields)	Inter-Value (e.g. credit # + expiry date)
Target URL (links with values)	Invalid Characters (e.g. <,&)

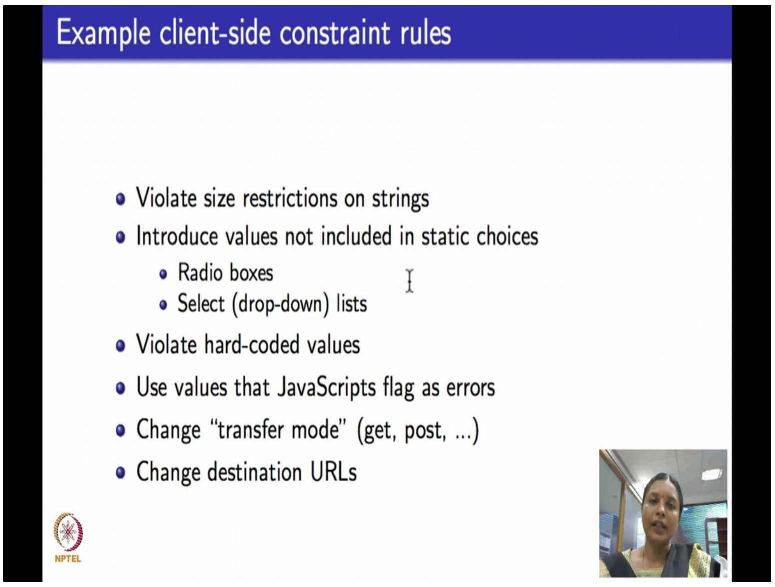
So, what are the typical types generically of the client input validation that happens? As I told you client side input validation is performed by HTML form controls, their attributes, client sides scripts that act as the data. Validation types can be characterized as plain HTML form validations or validation for which you need to write extra program or script on the client. Plain HTML form validation are of the following type, you could write as a part of the form. I will go back to the example.

Like for example, I told you right if I enter a name I can give a restriction on its size I can tell you what are the characters that I expect. So, typical HTML validation is I can set a length which gives me the maximum length of the input characters I can set of fixed value which is preset to that value, I can describe what is the transfer mode is it a get or a post, I can describe what are the various preset field elements I can describe a particular target URL. These I can describe directly with HTML. What are the scripting constraints? I could check for the type of data it could be your number and integer it could be purely alphabets it could be alpha numeric but no special characters, or when its password should always have at least one special character these kind of things I can check.

I can check for format of data typically you are allow to export or upload only a pdf file, only if the file is zipped and so on. I can check the actual value of data which means suppose I say age should be between 18 and 100. I can actually check whether the value that is entered is within that range. I can check for example, inter values like for example, I can match the credit card number and also ask for expiry date like we do in banks, I can check for invalid characters.

So, all these validations plain HTML and for those that you write scripts are done on the clients side. Ech of these validations can be done and tested by using bypass testing.

(Refer Slide Time: 19:25)

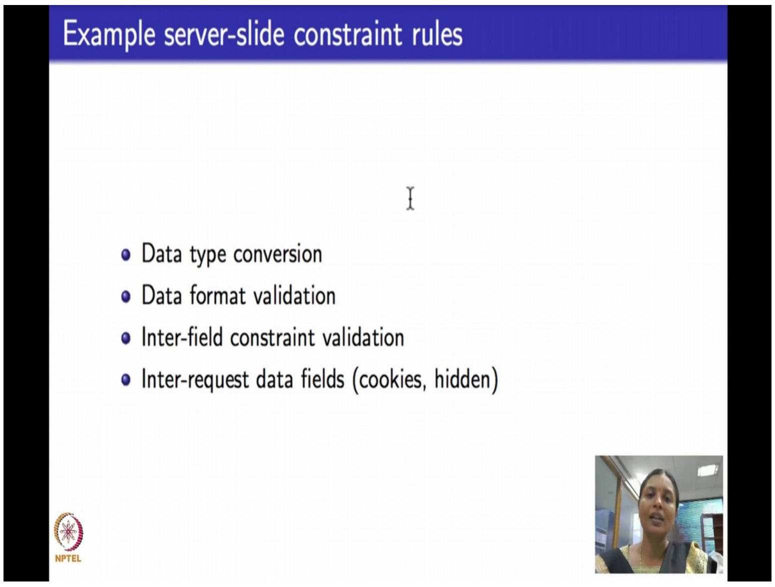


Example client-side constraint rules

- Violate size restrictions on strings
- Introduce values not included in static choices
  - Radio boxes
  - Select (drop-down) lists
- Violate hard-coded values
- Use values that JavaScripts flag as errors
- Change "transfer mode" (get, post, ...)
- Change destination URLs

For example, what can I do? I can violate size restriction on this strings, I can include values that are not in static choices like radio boxes drop down list I can violate the hard coded values, I can use values that java scripts flags as errors, I change the transfer mode to gets to post, I can change the destination URL, I can do all these things to validate client side constraints to.

(Refer Slide Time: 19:48)



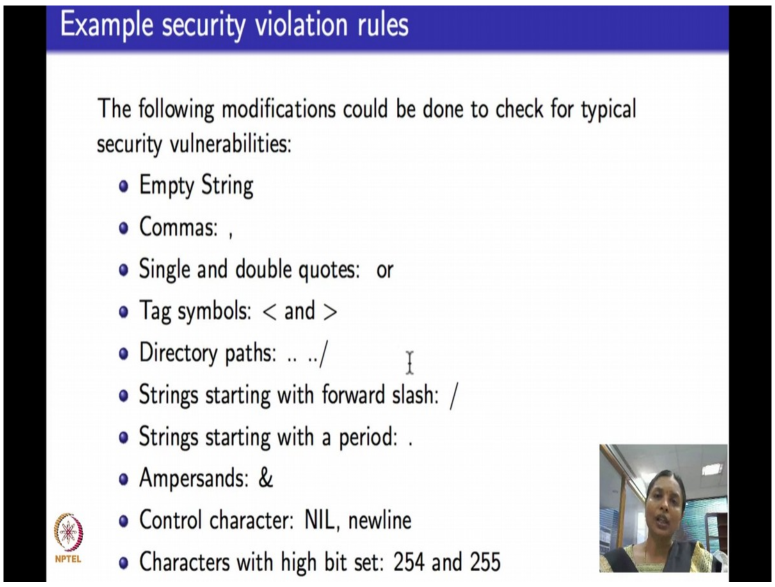
Example server-slide constraint rules

- Data type conversion
- Data format validation
- Inter-field constraint validation
- Inter-request data fields (cookies, hidden)

On the server side which I told you typically not recommended we can do bypass testing by doing data type conversion. We can change the data format of the data see how the server reacts to it.

We can change inter field constraints validation, like for example, you can say that if it is a visa card then it should be of this pattern right we can give master card with the visa pattern and see how the server reacts to it and so on. These are the various kinds of rules that can be check with bypass testing I can also check for various security violation rules.

(Refer Slide Time: 20:20)



Example security violation rules

The following modifications could be done to check for typical security vulnerabilities:

- Empty String
- Commas: ,
- Single and double quotes: or
- Tag symbols: < and >
- Directory paths: .. ../
- Strings starting with forward slash: /
- Strings starting with a period: .
- Ampersands: &
- Control character: NIL, newline
- Characters with high bit set: 254 and 255

I can give empty string, I can give a comma, I can insert special characters like single or double quotes, tag symbols, extra tag symbols that make this string syntactically invalid. I can change the path of the directory and so on. I can change the control characters by making new line paragraph and so on.

And typically all these will result in security violation in the server and if the server code or program is robust enough to handle these whatever you bypass and change server will correctly recognize it as invalid input and know how to handle it. So, this is an overview of bypass testing on the client side.

I will stop here for now. In the next lecture we will see how to do server side testing of web applications.

Thank you.