

HUNT

Data Driven Web Hacking & Manual Testing

@jhaddix @swagnetow @FatihEgbatan @digitalwoot @_Sha128
@bugcrowd

Contribs

Motley crew at @bugcrowd

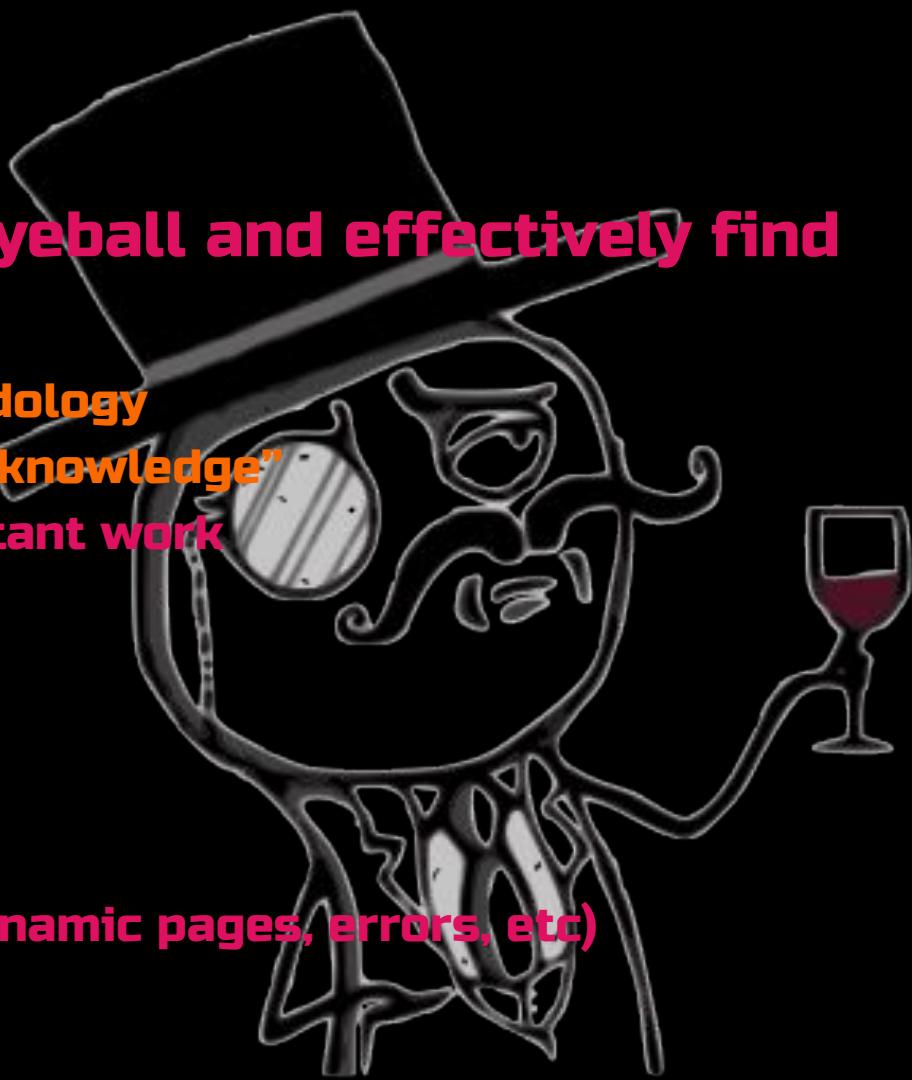
- Security Engineering & SecOps groups
- Bughunters, Pentesters, Code Analysis, ++
- Burp Suite fans

The Problem(z)

1. **Increasingly large and complicated Web Applications.**
Need manual testing. Lots of params.
2. **Applications Assessment Training** lacks “tribal knowledge” of vulnerability location
3. **No in-tool workflow for web hacking methodologies**

Current Solutions

- 1. Badass hacker who can eyeball and effectively find security bugs**
 - a. May or may not have a methodology
 - b. Definitely has accrued “tribal knowledge”
 - c. Bughunts and/or does consultant work
- 2. Dynamic Scanner**
 - a. Limited test cases (fuzzing)
 - b. Cost prohibitive
 - c. Limited in detection cases (dynamic pages, errors, etc)
 - d. Complex sites are hard (auth)



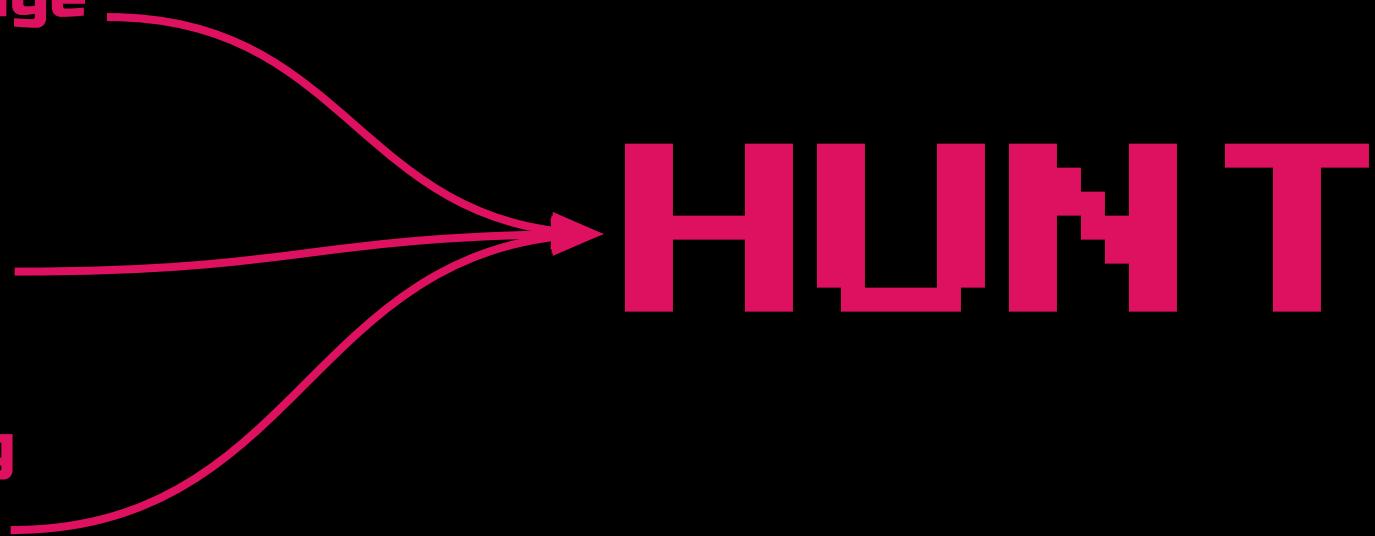
**NEW CHALLENGER
APPROACHING**



**Tribal knowledge
passive alerts**

**Methodology
in Burp**

**Manual testing
references in
Burp**



Level 1

Hunt Scanner



Tribal Knowledge & Bug Location



Coming up with bug location (tribal knowledge)

- Bugcrowd data contains over 600+ bounties and disclosure programs:
 - ◆ Programs x 2 web targets per bounty (average)
 - Ie. targets: www.defcon.org, forums.defcon.org, media.defcon.org
 - ◆ 15 (average) parameters per application
- $600 \times 2 \times 15 = \sim 18,000$ parameters seen

Coming up with vuln location (data) pt. 2

→ ~18,000 parameters:

- ◆ Reduce to params with vulns on them
- ◆ Reduce to only Critical (P1's) and High (P2's) Severity bugs/vulns
- ◆ Sort by recurring instances
- ◆ Include top 5-10 reoccurring instances per vuln/bug category
- ◆ Review top 100 for possible permutations manually and/or with regex
- ◆ Manually add ancillary data (pentest/fuzzdb/seclists/++)

Exhibit A

HTTP GET Example





Alerts

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts HUNT - Scanner HUNT - Methodology

Vulnerability Classes	Checked	Host	Path
Insecure Direct Object Reference (2)	<input type="checkbox"/>	auth.tesla.com	https://auth.tesla.com/oauth/v2/authorize

Server Side Request Forgery (5)

- dest
- dir (1)
- uri (1)**
- path
- continue
- url
- window
- next
- data
- reference
- site
- html
- val
- validate
- domain
- callback
- return
- page (1)
- feed
- host
- port
- to
- out
- view
- dir
- show
- navigation
- open (1)

Debug & Logic Parameters (1)

Server Side Template Injection (3)

OS Command Injection (2)

SQL Injection (2)

File Inclusion & Path Traversal

Advisory Request Response

```
GET /oauth/v2/authorize?client_id=tws-trusted&response_type=code&scope=openid%20email%20profile&redirect_uri=https%3A//www.tesla.com/openid-connect/generic&state=ktHKeyczXjHyneP7Ti0xPAhV-40Z9X2xnW9HmgDwUh8 HTTP/1.1
Host: auth.tesla.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:54.0) Gecko/20100101 Firefox/54.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: https://www.tesla.com/
Cookie: _ga=GA1.2.378139030.1501266153; _gid=GA1.2.1587870029.1501266153; _mkto_trk=id:929-KIG-197&token:_mch-tesla.com-1501266178398-46231; _svsid=223f5a60b44560212d2204c10e4b8798; RT=""; _gat_UA-9152935-1=1
Connection: close
Upgrade-Insecure-Requests: 1
```

Advisory

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts HUNT - Scanner HUNT - Methodology

Checked	Host	Path
<input type="checkbox"/>	auth.tesla.com	https://auth.tesla.com/oauth/v2/authorize

Vulnerability Classes

- Insecure Direct Object Reference (2)
- Server Side Request Forgery (5)
 - dest
 - dir (1)
 - uri (1)**
 - path
 - continue
 - url
 - window
 - next
 - data
 - reference
 - site
 - html
 - val
 - validate
 - domain
 - callback
 - return
 - page (1)
 - feed
 - host
 - port
 - to
 - out
 - view
 - dir
 - show
 - navigation
 - open (1)
- Debug & Logic Parameters (1)
- Server Side Template Injection (3)
- OS Command Injection (2)
- SQL Injection (2)
- File Inclusion & Path Traversal

Advisory Request Response

Location: https://auth.tesla.com/oauth/v2/authorize

HUNT located the **uri** parameter inside of your application traffic. The **uri** parameter is most often susceptible to Server Side Request Forgery (and sometimes URL redirects). HUNT recommends further manual analysis of the parameter in question.

For Server Side Request Forgery HUNT recommends the following resources to aid in manual testing:

[Server-side browsing considered harmful – Nicolas Grégoire](#)
[How To: Server-Side Request Forgery \(SSRF\) – Jobert Abma](#)
[IDOR Examples from ngalongc/bug-bounty-reference](#)
[safebuff SSRF Tips](#)
[The SSRF Bible](#)

Bug Location by bug/vuln class



Here be dragons

SQL Injection - http://acme.com/script?id=1

{regex + perm} id	{regex} select	{regex} report	{regex} role
{regex} update	{regex} query	{regex + perm} user	{regex + perm} name
{regex} sort	{regex} where	{regex + perm} search	{regex} params
{regex} process	{regex + perm} row	{regex + perm} view	{regex} table
{regex + perm} from	{regex + perm} sel	{regex} results	{regex} sleep
{regex} fetch	{regex + perm} order	{regex} keyword	{regex} count
{regex + perm} column	{regex} input	{regex + perm} key	
{regex + perm} code	{regex + perm} field	{regex} delete	{type} Custom headers
{regex} string	{regex} number	{regex + perm} filter	{type} JSON and XML services

File Includes / Dir Traversal

{regex + perm} file	{regex} location	{regex} locale	{regex + perm} path
{regex} display	{regex} load	{regex + perm} read	{regex} retrieve
{regex + perm} folder	{regex} style	{regex + perm} doc	{regex} document
{regex} root	{regex} pdf	{regex} pg	{regex} include
{regex} list	{regex} view	{regex} img	{regex} image

`http://acme.com/script?load=//file`

Server Side Request Forgery 🔥🔥🔥

Many on the File Includes / Dir Traversal table

{regex + perm} dest	{regex} redirect	{regex + perm} uri	{regex} path
{regex} continue	{regex + perm} url	{regex} window	{regex} next
{regex} data	{regex} reference	{regex + perm} site	{regex} html
{regex + perm} val	{regex} validate	{regex} domain	{regex} callback
{regex} return	{regex + perm} page	{regex} feed	{regex} host
{regex} port			

`http://acme.com/script?uri=http://site`

OS Command Injection

{regex} daemon	{regex + perm} upload	{regex + perm} dir
{regex} execute	{regex + perm} download	{regex + perm} log
{type} .cgi	{regex} ip	
{regex} cli	cmd	

http://acme.com/script?cmd=ls;%20cat%20/etc/passwd

Insecure Direct Object Reference 🔥🔥

{regex + perm} id	{regex + perm} user	
{regex + perm} account	{regex + perm} number	
{regex + perm} order	{regex + perm} no	
{regex + perm} doc	{regex + perm} key	
{regex + perm} email	{regex + perm} group	
{regex + perm} profile	{regex + perm} edit	REST numeric paths

`http://acme.com/script?user=21856`

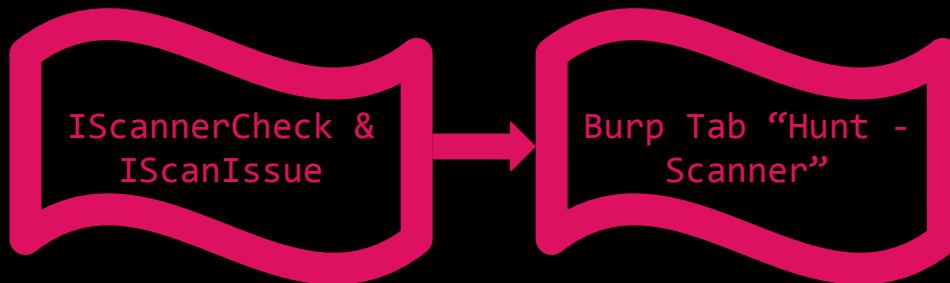
Server Side Template Injection & Logic / Debug

{regex + perm} template	content
preview	redirect
id	
view	
activity	
name	

```
"Debug & Logic Parameters": {  
    "check_location": {  
        "request": true,  
        "response": false  
    },  
    "detail": "HUNT located the <b>$param$</b> parameter",  
    "enabled": true,  
    "level": "Information",  
    "name": "Debug",  
    "params": [  
        "access",  
        "admin",  
        "dbg",  
        "debug",  
        "edit",  
        "grant",  
        "test",  
        "alter",  
        "clone",  
        "create",  
        "delete",  
        "disable",  
        "enable",  
        "exec",  
        "execute",  
        "load",  
        "make",  
        "modify",  
        "rename",  
        "reset",  
        "shell",  
        "toggle",  
        "adm",  
        "root",  
        "cfg",  
        "config"  
    ]  
}
```

http://acme.com/script?name={{2*3}}

Scanner Burp Implementation (Python)



```
def doPassiveScan(self, request_response):
    raw_request = request_response.getRequest()
    raw_response = request_response.getResponse()
    request = self.helpers.analyzeRequest(raw_request)
    response = self.helpers.analyzeResponse(raw_response)

    parameters = request.getParameters()
    url = self.helpers.analyzeRequest(request_response).getUrl()
    vuln_parameters = self.issues.check_parameters(self.helpers,
parameters)

    is_not_empty = len(vuln_parameters) > 0

    if is_not_empty:
        self.issues.create_scanner_issues(self.view, self.callbacks,
self.helpers, vuln_parameters, request_response)

    # Do not show any Bugcrowd found issues in the Scanner window
    return []
```

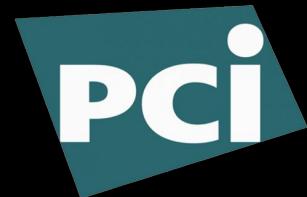
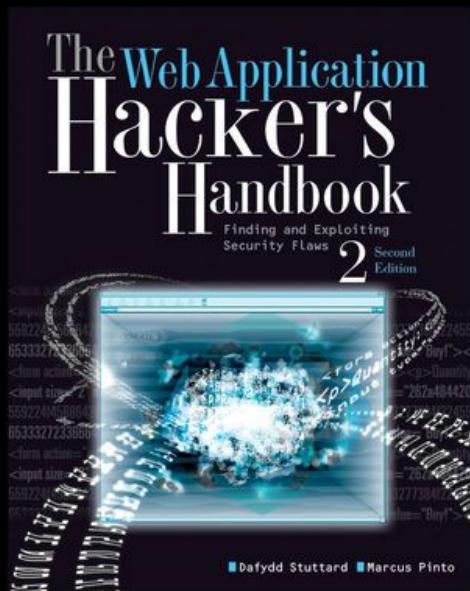
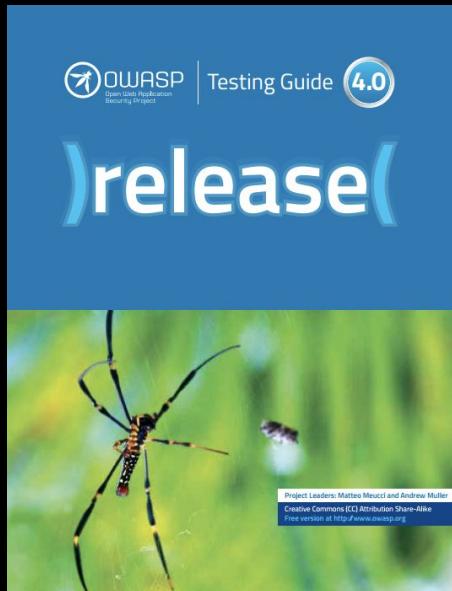
DEMO



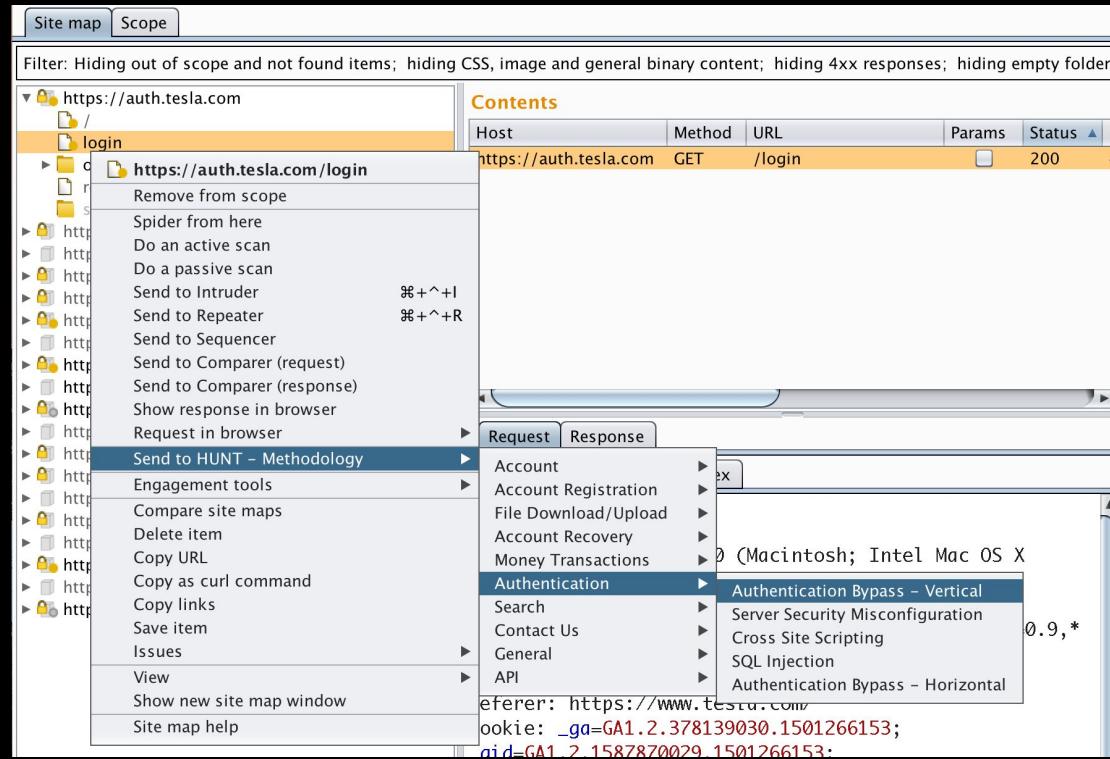
Level 2

GUI Methodology

Methodologies



Right Click -> Send-To Methodology Section



Description

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts HUNT – Scanner HUNT – Methodology

HUNT – Methodology

Functionality

Account

- Insecure Direct Object Reference
- Cross Site Request Forgery
- Authentication Bypass – Vertical
- Cross Site Scripting
- SQL Injection
- Authentication Bypass – Horizontal**

- Account Registration
- File Download/Upload
- Account Recovery
- Money Transactions
- Authentication
- Search
- Contact Us
- General
- API
- Settings

Description Bugs Resources Notes

Check to see if any kind of checks can be bypassed in any way to perform actions as a user of the same type.

Multiple Request/Response Tracking

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts HUNT – Scanner HUNT – Methodology

HUNT – Methodology

Functionality

Account

- Insecure Direct Object Reference
- Cross Site Request Forgery
- Authentication Bypass – Vertical
- Cross Site Scripting
- SQL Injection
- Authentication Bypass – Horizontal**
- Account Registration
- File Download/Upload
- Account Recovery
- Money Transactions
- Authentication
- Search
- Contact Us
- General
- API
- Settings

Description Bugs Resources Notes

0x 18 28

Request Response

```
GET /login HTTP/1.1
Host: auth.tesla.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:54.0) Gecko/20100101 Firefox/54.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: https://www.tesla.com/
Cookie: _ga=GA1.2.378139030.1501266153; _gid=GA1.2.1587870029.1501266153; _mkto_trk=id:929-KIG-197&token:_mch-tesla.com-1501266178398-46231; _svsid=223f5a60b44560212d2204c10e4b8798; RT=""; _gat_UA-9152935-1=1; sso-external.sid=eyJyZXR1cm5UbI6Ii9vYXV0aC92Mi9hdXR0b3JpeU/Y2xpZW50X2lkPXR3cy10cnVzdGVkInjlC3BvbNIX3R5cGU9Y29kZSzY29wZT1vcGVuaWQlMjBlbWFpbCUyMHByb2ZpbGUmcVkaXJIY3RfdXjpPWh0dHBzJTNLy93d3cudGVzbCEuY29tL29wZW5pZC1jb25uZWNOl2dIbmVyaWMmc3RhgdGU9a3RIS2V5Y3pYakh5bmVQN1RpMHhQQWhWLTwWjIYMnhuVzIlbWdEd1VoOCJ9; sso-external.sid.sig=ENtQ2ZOq5UztevY5c4VPTgyUEm8
Connection: close
Upgrade-Insecure-Requests: 1
```

Resources

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts HUNT - Scanner HUNT - Methodology

HUNT - Methodology

Functionality

Account

- Insecure Direct Object Reference
- Cross Site Request Forgery
- Authentication Bypass – Vertical
- Cross Site Scripting
- SQL Injection**
- Authentication Bypass – Horizontal

Account Registration

File Download/Upload

Account Recovery

Money Transactions

Authentication

Search

Contact Us

General

API

Settings

Description Bugs Resources Notes

<http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
<https://websec.wordpress.com/2010/12/04/sql-filter-evasion-cheat-sheet-mysql/>
<http://evilsq.com/main/page2.php>
<http://pentestmonkey.net/cheat-sheet/sql-injection/mssql-sql-injection-cheat-sheet>
<http://pentestmonkey.net/cheat-sheet/sql-injection/oracle-sql-injection-cheat-sheet>

Notes

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts HUNT – Scanner HUNT – Methodology

▼ HUNT – Methodology

 ▼ Functionality

 ▼ Account

 Insecure Direct Object Reference

 Cross Site Request Forgery

 Authentication Bypass – Vertical

 Cross Site Scripting

 SQL Injection

 Authentication Bypass – Horizontal

 ► Account Registration

 ► File Download/Upload

 ► Account Recovery

 ► Money Transactions

 ► Authentication

 ► Search

 ► Contact Us

 ► General

 ► API

 Settings

 Description Bugs Resources Notes

 - Try to break SQLi manually

 - Then try SQLmap

 - If all else fails, send to Bob

Save/Load JSON File

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts HUNT – Scanner HUNT – Methodology

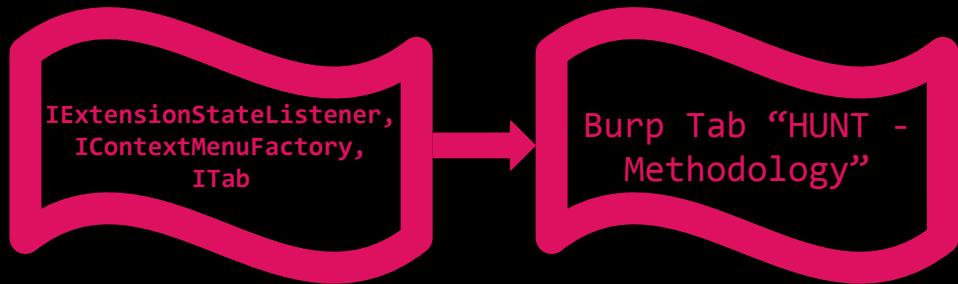
- ▼ HUNT – Methodology
- ▼ Functionality
- ▼ Account
 - Insecure Direct Object Reference
 - Cross Site Request Forgery
 - Authentication Bypass – Vertical
 - Cross Site Scripting
 - SQL Injection
 - Authentication Bypass – Horizontal
- Account Registration
- File Download/Upload
- Account Recovery
- Money Transactions
- Authentication
- Search
- Contact Us
- General
- API

Settings

Load JSON File

Save JSON File

Methodology Burp Implementation (Python)



```
def createMenuItems(self, invocation):
    # Do not create a menu item unless getting a context menu from the proxy history or
    # scanner results
    is_proxy_history = invocation.getInvocationContext() ==
    invocation.CONTEXT_PROXY_HISTORY
    is_scanner_results = invocation.getInvocationContext() ==
    invocation.CONTEXT_SCANNER_RESULTS
    is_correct_context = is_proxy_history or is_scanner_results

    if not is_correct_context:
        return

    request_response = invocation.getSelectedMessages()[0]

    functionality = self.checklist["Functionality"]

    # Create the menu item for the Burp context menu
    bugcatcher_menu = JMenu("Send to HUNT - Methodology")

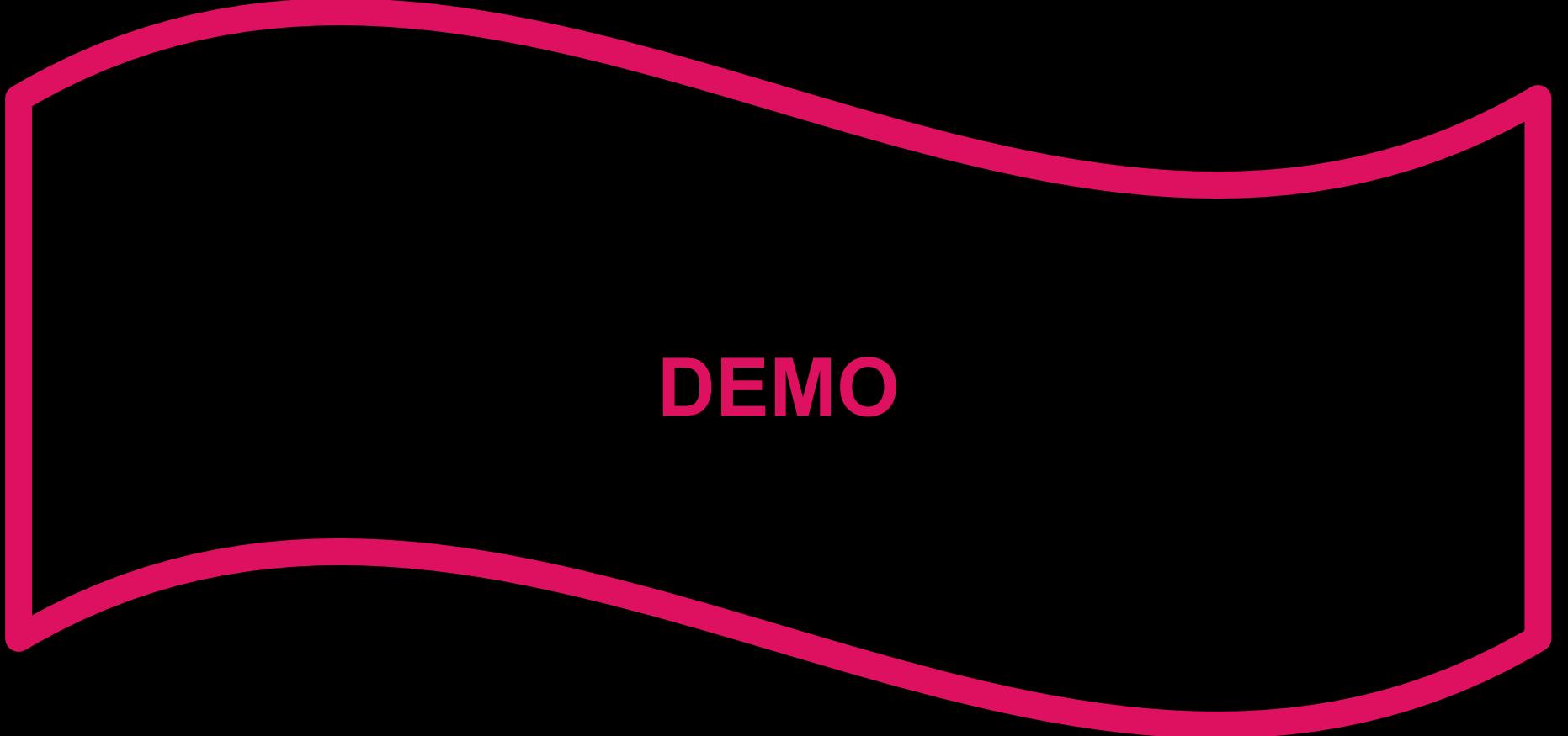
    for functionality_name in functionality:
        vulns = functionality[functionality_name]["vulns"]
        menu_vuln = JMenu(functionality_name)

        # Create a menu item and an action listener per vulnerability
        # class on each functionality
        for vuln_name in vulns:
            item_vuln = JMenuItem(vuln_name)
            menu_action_listener = MenuActionListener(self.view, self.callbacks,
                request_response, functionality_name, vuln_name)
            item_vuln.addActionListener(menu_action_listener)
            menu_vuln.add(item_vuln)

        bugcatcher_menu.add(menu_vuln)

    burp_menu = []
    burp_menu.append(bugcatcher_menu)

    return burp_menu
```



DEMO



Plugin Installation

Installation - Jython

The screenshot shows the 'User options' section of the Burp Suite interface, specifically the 'Extensions' tab. The tabs at the top include Target, Proxy, Spider, Scanner, Intruder, Repeater, Sequencer, Decoder, Comparer, Extender, Project options, and User options. The 'Extensions' tab is currently active.

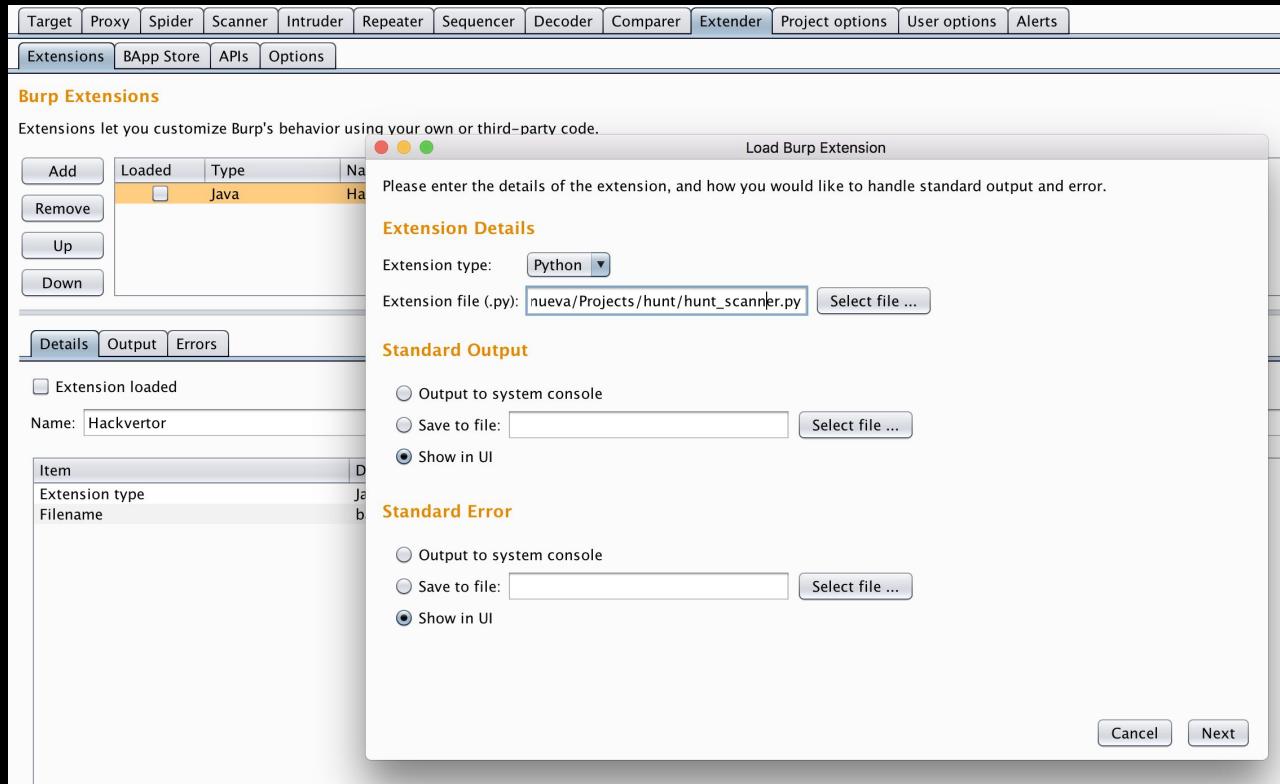
Settings
This setting controls how Burp handles extensions on startup.
 Automatically reload extensions on startup

Java Environment
These settings let you configure the environment for executing extensions that are written in Java. If your extensions use any libraries that are not included in the Java classpath, you will need to specify their location here.
Folder for loading library JAR files (optional): Select folder ...

Python Environment
These settings let you configure the environment for executing extensions that are written in Python. To use Python extensions, you will need to specify the location of the Jython standalone JAR file.
Location of Jython standalone JAR file: Select file ...
Folder for loading modules (optional): Select folder ...

Ruby Environment
These settings let you configure the environment for executing extensions that are written in Ruby. To use Ruby extensions, you will need to specify the location of the JRuby JAR file.
Location of JRuby JAR file: Select file ...

Installation - Plugin



Setting Target Scope

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

Site map Scope

Target Scope

Define the in-scope targets for your current work. This configuration affects the behavior of tools throughout the suite. All fields take use the context menus in the site map to include or exclude URL paths.

Include in scope

Add	Enabled	Protocol	Host / IP range	Port	File
	<input checked="" type="checkbox"/>	Any	tesla		

Exclude from scope

Add	Enabled	Protocol	Host / IP range	Port	File
	<input checked="" type="checkbox"/>	Any			logout
	<input checked="" type="checkbox"/>	Any			logoff
	<input checked="" type="checkbox"/>	Any			exit
	<input checked="" type="checkbox"/>	Any			signout

Setting Passive Scanner Scope

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

Issue activity Scan queue Live scanning Issue definitions Options

 **Live Active Scanning**

 Automatically scan the following targets as you browse. Active scan checks send various malicious requests designed to identify common vulnerabilities.

Don't scan
 Use suite scope [defined in Target tab]
 Use custom scope

 **Live Passive Scanning**

 Automatically scan the following targets as you browse. Passive scan checks analyze your existing traffic for evidence of vulnerabilities.

Don't scan
 Scan everything
 Use suite scope [defined in Target tab]
 Use custom scope

Running the Passive Scanner

The screenshot shows the OWASP ZAP interface with the 'Scanner' tab selected. The main pane displays a list of URLs under the heading 'Contents' with 12 items selected. The selected URLs are:

- https://auth.tesla.com
- https://issues.tesla.com
- https://location.tesla.com
- https://location.tesla.com
- https://rumcollector.tesla.com
- http://shop.teslamotors.com
- http://sjc04s1gipap.tesla.com
- https://stage.tesla.com
- http://www.tesla.cn
- https://www.tesla.cn
- https://www.tesla.com
- https://znedscsenlrq.tesla.com

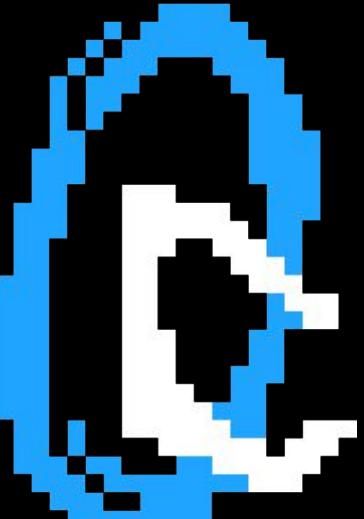
The right pane shows a table of requests made to the selected URLs. The table has columns for Method, URL, Params, and Status. All requests are GET methods with status codes 200.

	Method	URL	Params	Status
/www.tesla.com	GET	/		200
/www.tesla.com	GET	/libraries-boomerang...		200
/www.tesla.com	GET	/sites/default/files/j...		200
/www.tesla.com	GET	/sites/default/files/j...		200
/www.tesla.com	GET	/sites/default/files/j...		200
/www.tesla.com	GET	/sites/default/files/j...		200
/www.tesla.com	GET	/sites/default/files/j...		200
/www.tesla.com	GET	/sites/default/files/j...		200
/www.tesla.com	GET	/sites/default/files/j...		200
/www.tesla.com	GET	/sites/default/files/j...		200
/www.tesla.com	GET	/tesla_theme/assets/...		200

Below the table, there are tabs for 'Request' (selected), 'Response', 'Headers', and 'Hex'. The 'Response' tab shows the following content:

```
HTTP/1.1  
www.tesla.com  
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X  
10.12; rv:54.0) Gecko/20100101 Firefox/54.0  
Accept:  
text/html,application/xhtml+xml,application/xml;q=0.9,*
```

Extensibility



Scanner Extensibility



Creating new issue checks are as simple as adding to the JSON file.

```
{  
  "issues": {  
    "OS Command Injection": {  
      "check_location": {  
        "request": true,  
        "response": false  
      },  
      "detail": "HUNT located the $param$ parameter inside of your application traffic. The $param$ parameter is most often susceptible to OS Command Injection. HUNT recommends further manual analysis of the parameter in question.  
      For OS Command Injection HUNT recommends the following resources to aid in manual testing:",  
      "level": "Information",  
      "name": "Possible OS Command Injection",  
      "params": [  
        "daemon",  
        "upload",  
        "dir",  
        "execute",  
        "download",  
        "sexyparam"  
      ]  
    }  
  }  
}
```

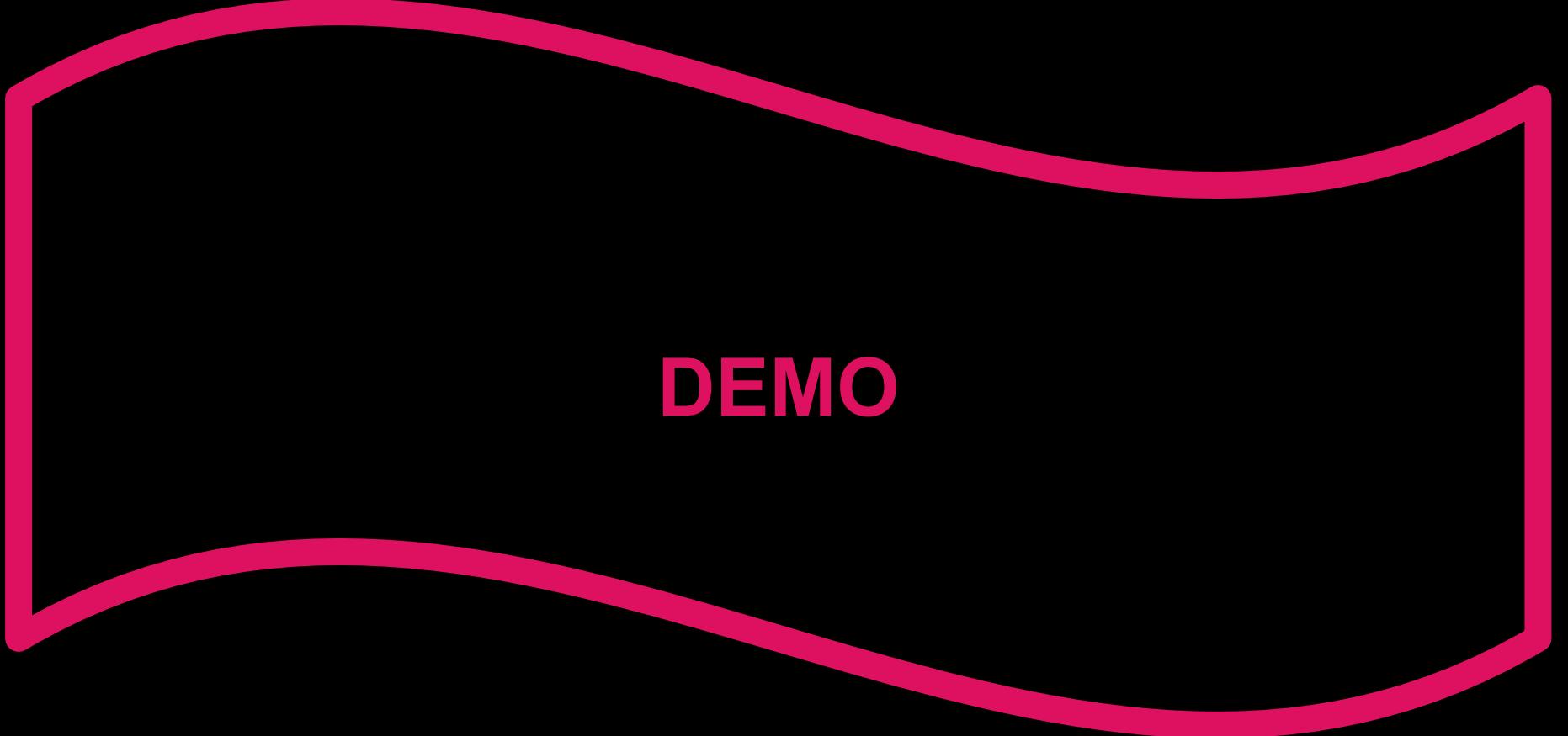
Methodology Extensibility

Choose your own

ADVENTURE

Creating new methodologies are as simple as adding to the JSON file.

```
{  
  "checklist": {  
    "Settings": "",  
    "Functionality": {  
      "SEXY METHODOLOGY SECTION": {  
        "description": "SWAG",  
        "tests": {  
          "Authentication Bypass - Vertical": {  
            "description": "Check to see if the login sequence can be bypassed in any way to get higher level permissions.",  
            "resources": [],  
            "bugs": [],  
            "notes": ""  
          }  
        }  
      }  
    }  
  }  
}
```



DEMO

The Future

- **More built-in methodologies**
 - ◆ PCI, HIPAA, CREST, OWASP, PTES
- **Port to ZAP?**
- **More scanner checks/vulnerability classes**
- **More resources**
- **Dynamic JSON structure support**
- **Perfect GUI lol**
- **REST Support**
- **Full Burp helpers (right click, search, highlight, etc)**
- **Resource/File name analysis (Instead of params)**
- **Alerts on content types (XML, JSON, Multipart-form)**
- **Response analysis alerts (errors ++)**

Thanks!

Questions?



www.github.com/bugcrowd/HUNT

@jhaddix @swagnetow @FatihEgbatan @digitalwoot @_Sha128
@bugcrowd