



DEFCON

HUNT

Data Driven Web Hacking & Manual Testing

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

Contribs

Motley crew at @bugcrowd

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

The Problem(z)

1. **Increasingly large and complicated Web Applications.**
Need manual testing
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**

HUNT

HUNT Burp Suite Extension

The screenshot displays the HUNT Burp Suite extension interface. The top menu bar includes 'Burp', 'Intruder', 'Repeater', 'Window', and 'Help'. Below the menu is a toolbar with tabs for 'Target', 'Proxy', 'Spider', 'Scanner', 'Intruder', 'Repeater', 'Sequencer', 'Decoder', 'Comparer', 'Extender', 'Project options', 'User options', 'Alerts', 'HUNT - Scanner', and 'HUNT - Methodology'. The 'HUNT - Scanner' tab is active.

The left sidebar shows a tree view of 'Vulnerability Classes'. The 'SQL Injection (4)' class is expanded, showing a list of parameters: id (2), select, report, role, update, query, user, name (1), sort, where, search (1), params, process, row, view, table, from, sel, results, sleep, fetch, order, keyword, column, field, delete, string, number, and filter.

The main panel displays a table of detected vulnerabilities. The table has three columns: 'Checked', 'Host', and 'Path'. A single entry is shown for 'bugcrowd.com' with the path 'https://bugcrowd.com/programs'.

Below the table, the 'Advisory' tab is selected, showing a detailed advisory for the detected SQL injection. The advisory text is as follows:

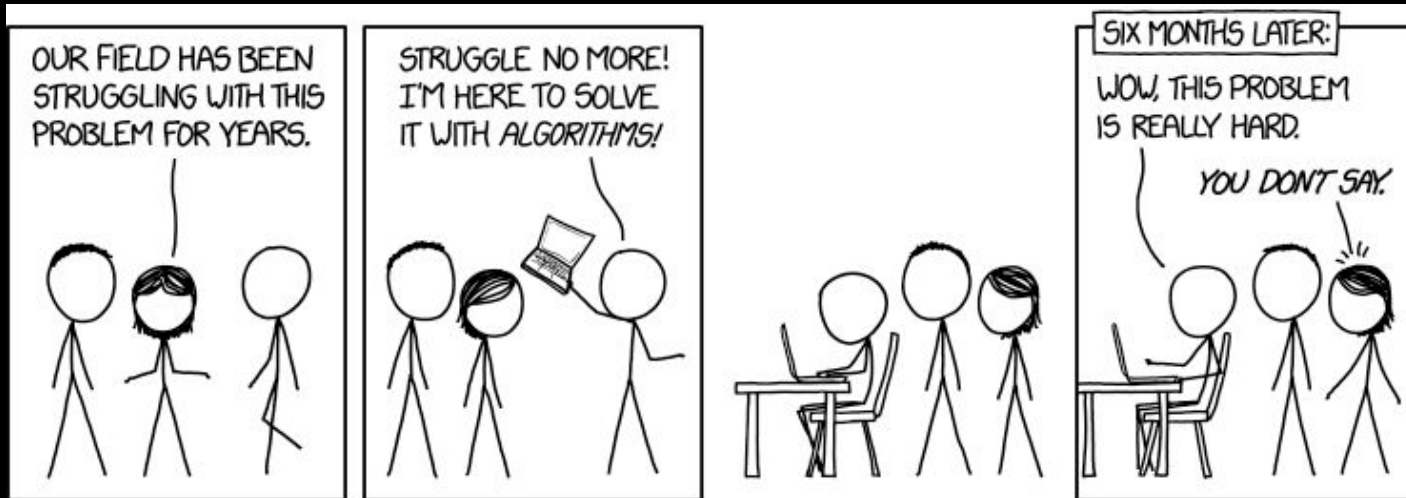
Location: https://bugcrowd.com/programs

HUNT located the **id** parameter inside of your application traffic. The **id** parameter is most often susceptible to SQL Injection. HUNT recommends further manual analysis of the parameter in question.

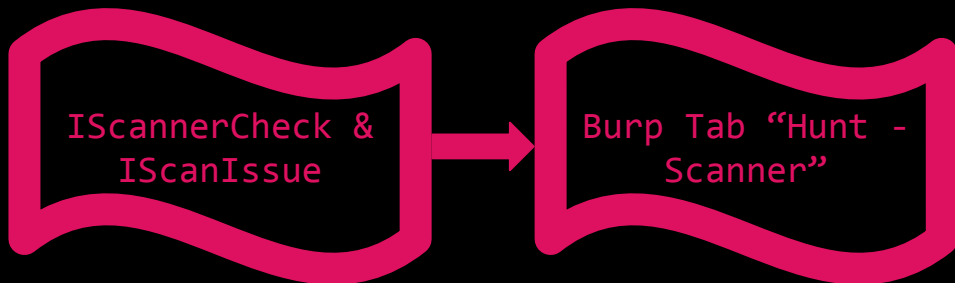
For SQL Injection HUNT references The Bug Hunters Methodology SQL Injection references table:

- [PentestMonkey's MySQL Injection Cheat Sheet](#)
- [Reiner's MySQL Injection Filter Evasion](#)
- [EvilSQL's Error/Union/Blind MSSQL Cheat Sheet](#)
- [PentestMonkey's MSSQL SQL Injection Cheat Sheet](#)
- [PentestMonkey's Oracle SQL Cheat Sheet](#)
- [PentestMonkey's PostgreSQL Cheat Sheet](#)
- [Access SQL Injection Cheat Sheet](#)
- [Access SQL Injection Cheat Sheet](#)
- [PentestMonkey's Ingres SQL Injection Cheat Sheet](#)
- [PentestMonkey's DB2 SQL Injection Cheat Sheet](#)
- [PentestMonkey's Informix SQL Injection Cheat Sheet](#)
- [SQLite3 Injection Cheat Sheet](#)
- [Ruby on Rails \(ActiveRecord\) SQL Injection Guide](#)

Tool Flow

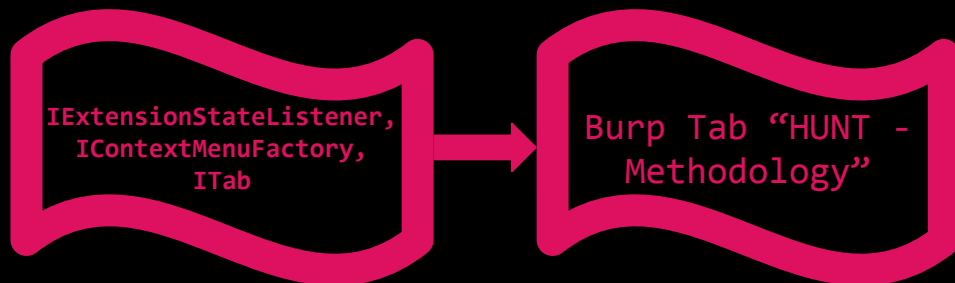


Scanner Burp Implementation (Python)



```
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 []
```

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
```

Tribal Knowledge & Bug Location



Coming up with vuln location (data)

→ 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/other data)

Bug Location by bug/vuln class



Here be dragons

SQL Injection

{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

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		

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			

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

Server Side Template Injection

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



GUI Methodology & Tester Helpers

Advisory

Burp Suite Professional v1.7.23 - Temporary Project - licensed to Bugcrowd [18 user license]

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

Vulnerability Classes

- Insecure Direct Object Reference (1)
- Remote File Inclusion (1)
- Server Side Request Forgery (2)
- Server Side Template Injection
- OS Command Injection
- SQL Injection (3)

id (1)

- select
- report
- role
- update
- query
- user
- name (1)
- sort
- where
- search (1)
- params
- process
- row
- view
- table
- from
- sel
- results
- sleep
- fetch
- order
- keyword
- column
- field
- delete
- string
- number
- filter

Checked	Host	Path
☐	bugcrowd.com	https://bugcrowd.com/programs

Advisory Request Response

https://bugcrowd.com/programs

HUNT located the **id** parameter inside of your application traffic. The **id** parameter is most often susceptible to SQL Injection. HUNT recommends further manual analysis of the parameter in question.

PentestMonkey's MySQL InjectionCheat Sheet
Reiner's MySQL Injection Filter Evasion
EviSQL's Error/Union/Blind MSSQL Cheat Sheet
PentestMonkey's MSSQL SQL Injection Cheat Sheet
PentestMonkey's Oracle SQL Cheat Sheet
PentestMonkey's PostgreSQL Cheat Sheet
Access SQL Injection Cheat Sheet
Access SQL Injection Cheat Sheet
PentestMonkey's Ingres SQL Injection Cheat Sheet
PentestMonkey's DB2 SQL Injection Cheat Sheet
PentestMonkey's Informix SQL Injection Cheat Sheet
SQLite3 Injection Cheat Sheet
Ruby on Rails (ActiveRecord) SQL Injection Guide

Alerts

Burp Suite Professional v1.7.23 - Temporary Project - licensed to Bugcrowd [18 user license]

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

Vulnerability Classes

- Insecure Direct Object Reference (2)
- Remote File Inclusion (1)
- Server Side Request Forgery (2)
- Server Side Template Injection
- OS Command Injection
- SQL Injection (3)
- id (1)**
- select
- report
- role
- update
- query
- user
- name (1)
- sort
- where
- search (1)
- params
- process
- row
- view
- table
- from
- sel
- results
- sleep
- fetch
- order
- keyword
- column
- field
- delete
- string
- number
- filter

Checked	Host	Path
false	bugcrowd.com	https://bugcrowd.com/programs

Send to Repeater
Send to Intruder

Advisory Request Response

HTTP/1.1 200 OK
Date: Tue, 11 Jul 2017 23:19:44 GMT
Content-Type: text/html; charset=utf-8
Connection: close
Cache-Control: no-cache, no-store
Expires: Fri, 01 Jan 1990 00:00:00 GMT
Pragma: no-cache
Strict-Transport-Security: max-age=15552000; includeSubDomains; preload
X-Content-Type-Options: nosniff
X-Download-Options: noopen
X-Frame-Options: SAMEORIGIN
X-Permitted-Cross-Domain-Policies: none
X-Request-Id: c8a59183-ec99-4ca7-8b57-6d2457df15dc
X-XSS-Protection: 1; mode=block
Accept-Ranges: bytes
X-Served-By: cache-lax8632-LAX
X-Cache: MISS
X-Cache-Hits: 0
X-Timer: S1499815182.452412,VS0,VE1933
CF-Cache-Status: MISS
Server: cloudflare-nginx
CF-RAY: 37cf7d3a4bdf13b9-LAX
Content-Length: 48700

<!DOCTYPE html>
<html lang='en'>
<head>
<meta content='text/html; charset=UTF-8' http-equiv='Content-Type'>
<meta charset='utf-8'>
<title>Bugcrowd 1 Your Elastic Security Team, better security testing through bug bounties and managed security programs.</title>

Right Click -> Send-To Methodology Section

The screenshot displays the Burp Suite Professional v1.7.23 interface. The main window shows a list of HTTP history items. A right-click context menu is open over the first item, which is a GET request to `https://bugcrowd.com/nrograms?id=`. The menu options include 'Send to HUNT - Methodology', which is highlighted. This option has opened a sub-menu with the following items: 'Account', 'Account Registration', 'File Download/Upload', 'Account Recovery', 'Money Transactions', 'Authentication', 'Search', 'Contact Us', 'General', and 'API'. The 'Search' item is further highlighted, and a sub-menu is visible with 'Cross Site Scripting' and 'SQL Injection' options. The 'SQL Injection' option is highlighted in blue.

Filter: Hiding out of scope items; hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment
1	https://bugcrowd.com	GET	/nrograms?id=		☒	200	49425	HTML		Bugcrowd Your Elast...	
270	https://bugcrowd.com	GET	https://bugcrowd.com/programs?id=		☒	200	28980	HTML		Bugcrowd Your Elast...	
276	https://bugcrowd.com	DELETE	/nrograms?id=		☐	200	521				
288	https://www.bugcrowd.com	GET	/		☐	200	42175	HTML		Bugcrowd Vulnerabil...	
294	https://www.bugcrowd.com	GET	/js?...		☒	200	98045	script	js		
295	https://www.bugcrowd.com	GET	/le-ana...		☒	200	8048	script	js		
296	https://www.bugcrowd.com	GET	/ery-m...		☒	200	10916	script	js		
297	https://www.bugcrowd.com	POST	/rowd/d...	%+^+I	☐	200	344776	script	js		
300	https://www.bugcrowd.com	POST	/rowd/d...	%+^+R	☐	200	1123	XML	svg		
303	https://www.bugcrowd.com	POST	/rowd/d...		☐	200	8707	XML	svg		
307	https://www.bugcrowd.com	POST	/rowd/d...		☐	200	4687	XML	svg		
308	https://www.bugcrowd.com	POST	/rowd/d...		☐	200	1217	XML	svg		
309	https://www.bugcrowd.com	POST	/rowd/d...		☐	200	35957	woff2			

Request Response

Raw Headers Hex

HTTP/1.1 200 OK
Date: Thu, 06 Jul 2023 14:00:00 GMT
Content-Type: text/html; charset=utf-8
Connection: close
Cache-Control: no-cache
Expires: Fri, 01 Jun 2023 00:00:00 GMT
Pragma: no-cache
Strict-Transport-Security: max-age=31536000
X-Content-Type-Options: nosniff
X-Download-Options: noopen
X-Frame-Options: SAMEORIGIN
X-Permitted-Cross-Domain-Policies: none
X-Request-Id: 7047a4fe-60e9-4403-a7c2-8bbabaae27e1
X-XSS-Protection: 1; mode=block
Accept-Ranges: bytes
X-Served-By: cache-lax8646-LAX
X-Cache: MISS
X-Cache-Hits: 0

Send to HUNT - Methodology

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

Cross Site Scripting

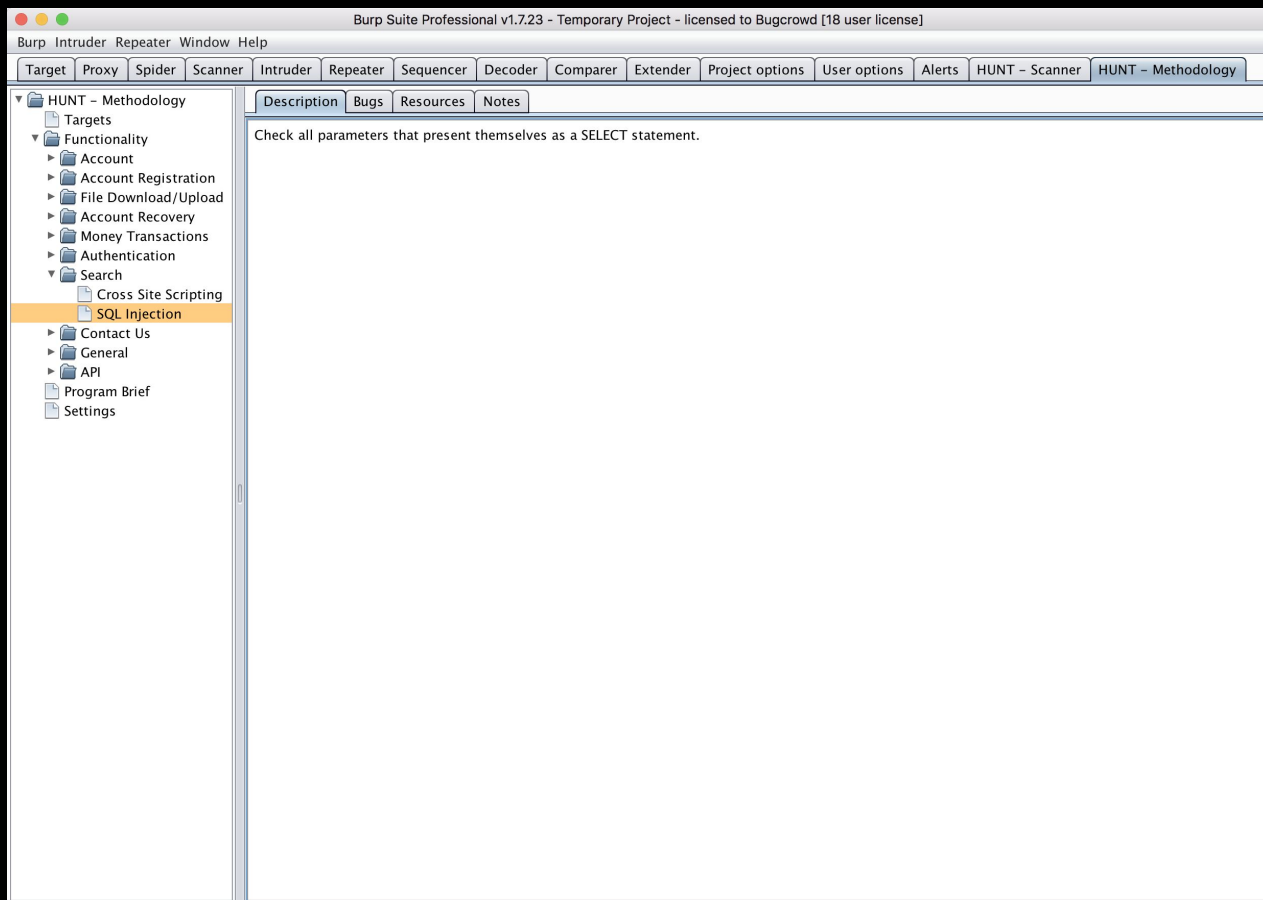
SQL Injection

Domains; preload

Type a search term

0 matches

Description



Multiple Request/Response Tracking

Burp Suite Professional v1.7.23 - Temporary Project - licensed to Bugcrowd [18 user license]

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

HUNT - Methodology

- Targets
- Functionality
 - Account
 - Account Registration
 - File Download/Upload
 - Account Recovery
 - Money Transactions
 - Authentication
 - Search
 - Cross Site Scripting
 - SQL Injection**
 - Contact Us
 - General
 - API
- Program Brief
- Settings

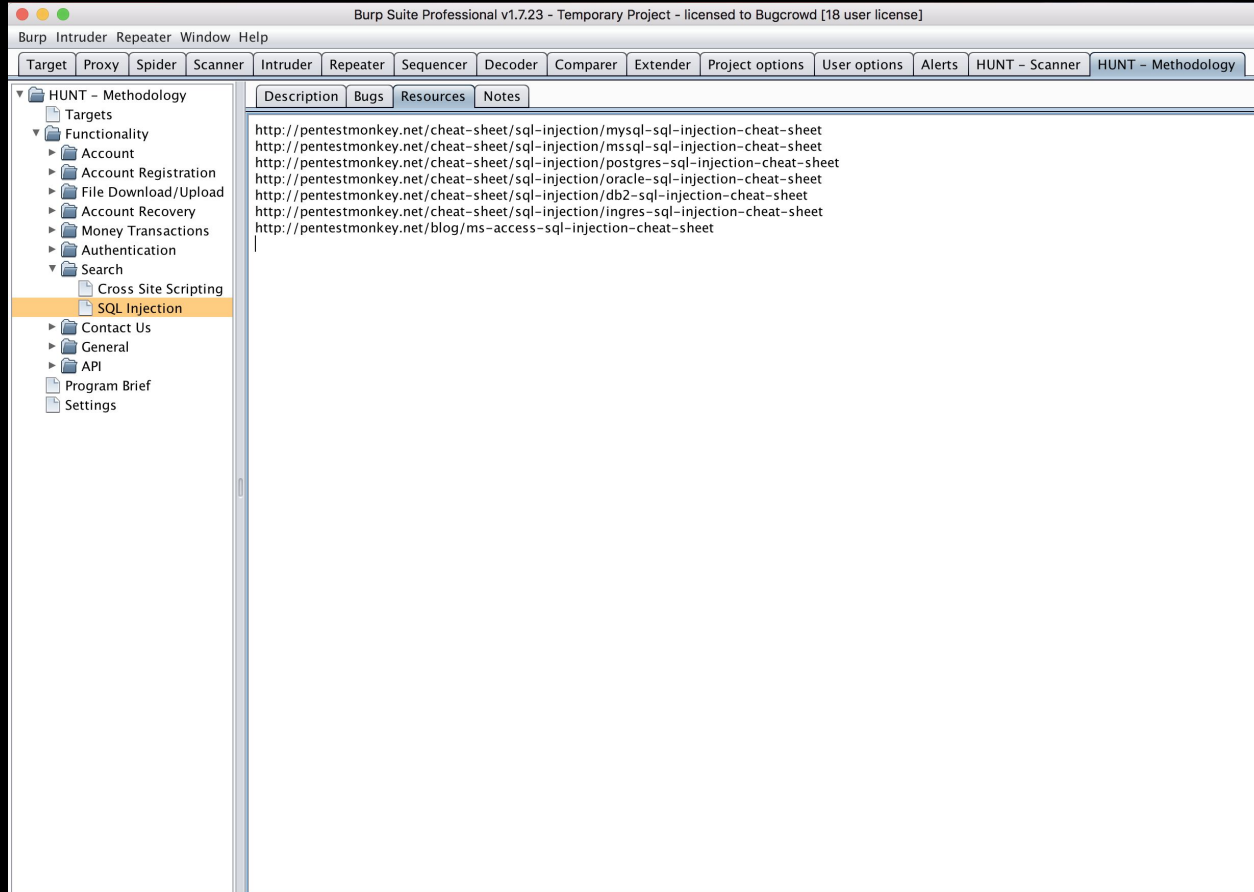
Description Bugs Resources Notes

0x 1x 2x

Request Response

GET /programs?id= HTTP/1.1
Host: bugcrowd.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
Cookie: __cfduid=d577dd611c5d9fe104e0c4c6fd5ad1c981498309999; _session_id=2467ac296701863ade9a93fc4b991314;ajs_user_id=null;ajs_group_id=null;ajs_anonymous_id=%2283dfb0fb-321b-4906-b01a-ae2ee00eee0%22;_ga=GA1.2.777501242.1498310005;_hp2_id.351079185=%7B%22userid%22%3A%220207813342702892%22%2C%22pageviewid%22%3A%220212102320056947%22%2C%22sessionid%22%3A%224335875914401339%22%2C%22identity%22%3A%22null%22%2C%22trackerVersion%22%3A%223.0%22%7D;__hstc=174498493.2b60d5da7bd21fe27d95eaf463d4313.1498310007141.1498590866972.1498603429523.9;__hssrc=1;hubspotutk=2b60d5da7bd21fe27d95eaf463d4313;intercom-id=ovg5emkk=4ec76bd5-cdd5-4e01-a87b-bf82fe8b4222
Connection: close
Upgrade-Insecure-Requests: 1

Resources



Notes

Burp Suite Professional v1.7.23 - Temporary Project - licensed to Bugcrowd [18 user license]

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

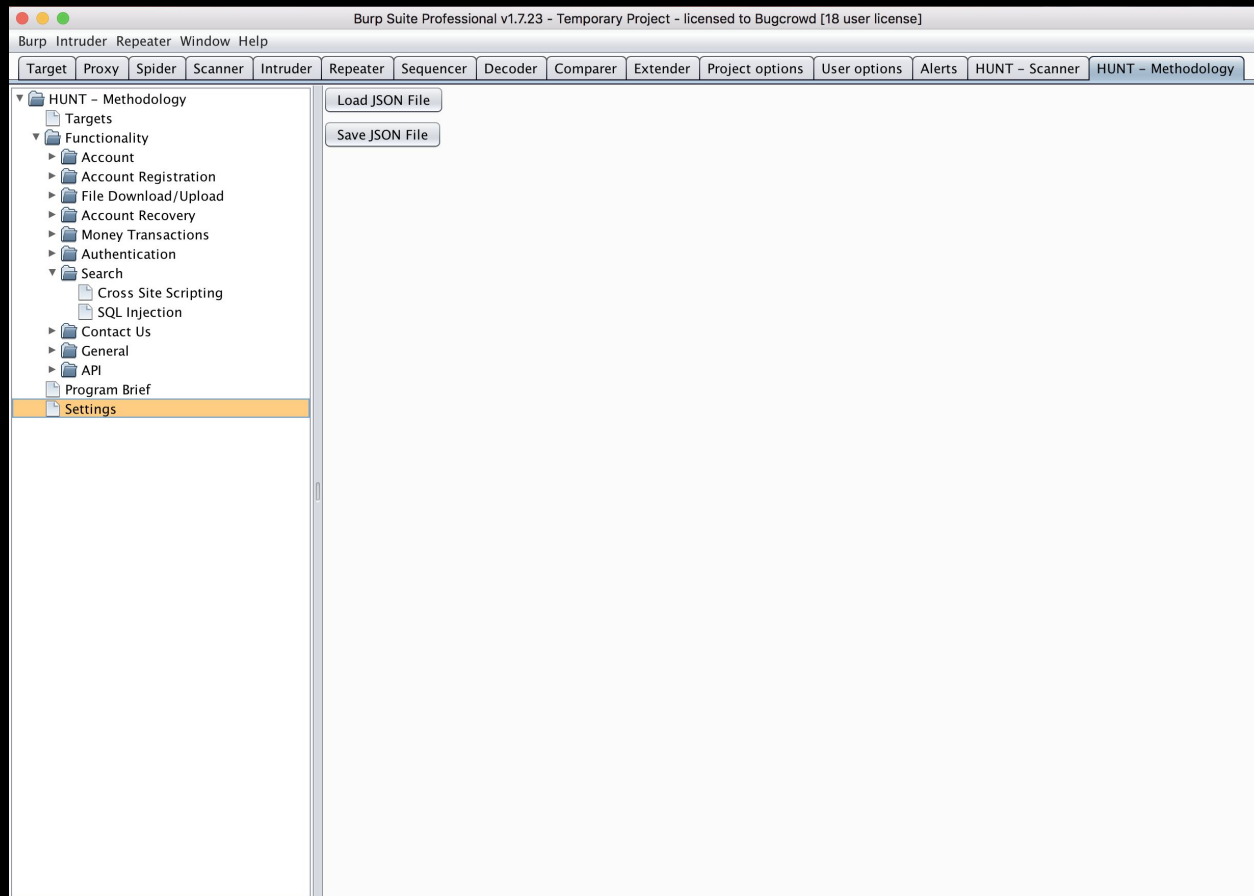
HUNT - Methodology

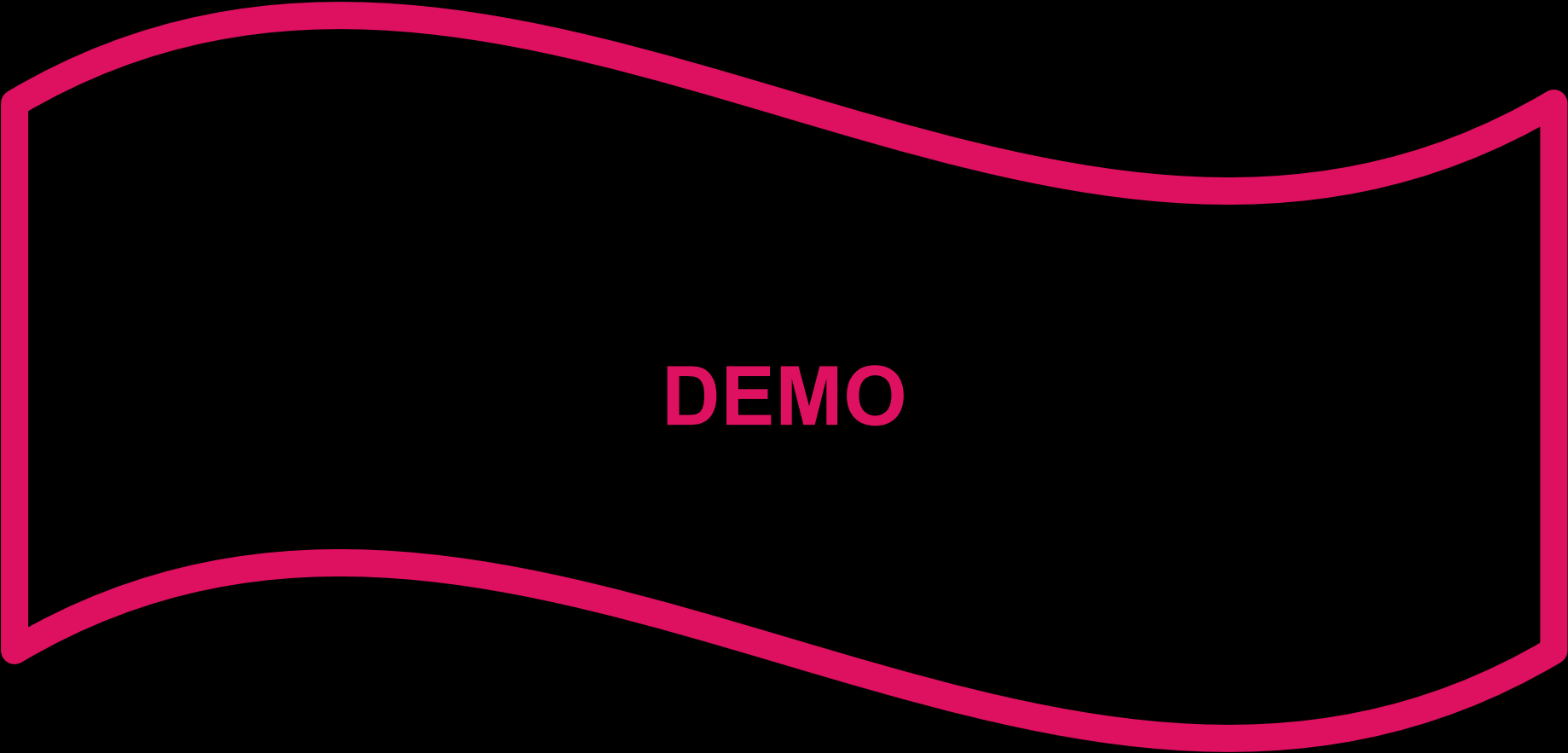
- Targets
- ▼ Functionality
 - Account
 - Account Registration
 - File Download/Upload
 - Account Recovery
 - Money Transactions
 - Authentication
- ▼ Search
 - Cross Site Scripting
 - SQL Injection
- Contact Us
- General
- API
- Program Brief
- Settings

Description Bugs Resources Notes

- Try breaking SQL statement by hand.
- SQL statement is probably a SELECT * FROM
- Use SQLmap to show as a proof of concept.

Save/Load JSON File



A thick, bright pink line forms a wavy, irregular border around the central text. The line starts at the top left, curves upwards and to the right, then downwards and to the right, then upwards and to the right, and finally downwards and to the right, ending at the bottom right. The overall shape is roughly rectangular but with wavy, organic edges.

DEMO

Extensibility



Custom Alerts



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

```
"Remote File Inclusion": {
  "check_location": {
    "request": true,
    "response": false
  },
  "detail": "There is probably a Remote File Inclusion on the
<b>$param$</b> parameter",
  "enabled": true,
  "level": "Information",
  "name": "Possible Remote File Inclusion",
  "params": [
    "file",
    "document",
    "folder",
    "root",
    "path",
    "pg",
    "style",
    "pdf",
    "template",
    "php_path",
    "doc"
  ]
}
```

Custom Methodologies

- **Web Application Hacker's Handbook**
- **PCI (future)**
- **HIPAA (future)**
- **CREST (future)**
- **OWASP (future)**
- **PTES (future)**

The Future

- **More built-in methodologies**
- **Port to ZAP?**
- **More scanner checks/vulnerability classes**
- **More resources**
- **Perfect GUI lol**

Thanks!
Questions?
(link)



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