Hands-On Web Hacking

September 2018

Introductions

Who are we?

Seth Law - @sethlaw - Old Guy - Redpoint Security

Who are we?

Justin Larson -

- @Phant0mTrav3ler
- ~~Expert~~ marginally skilled hacker

Overview

Hands-On Web Hacking - Why?

- Hands-on experience with application security tools
- Exploit specific application security issues.
- Normal Assessment Process
  - Information Gathering
  - Vulnerability Identification
  - Documentation

Hand-On Web Hacking - Format

- Module
  - Introduction/Demonstration (5-10 minutes)
  - Hands-on Activities (15-20 minutes)
  - Break (5 minutes)

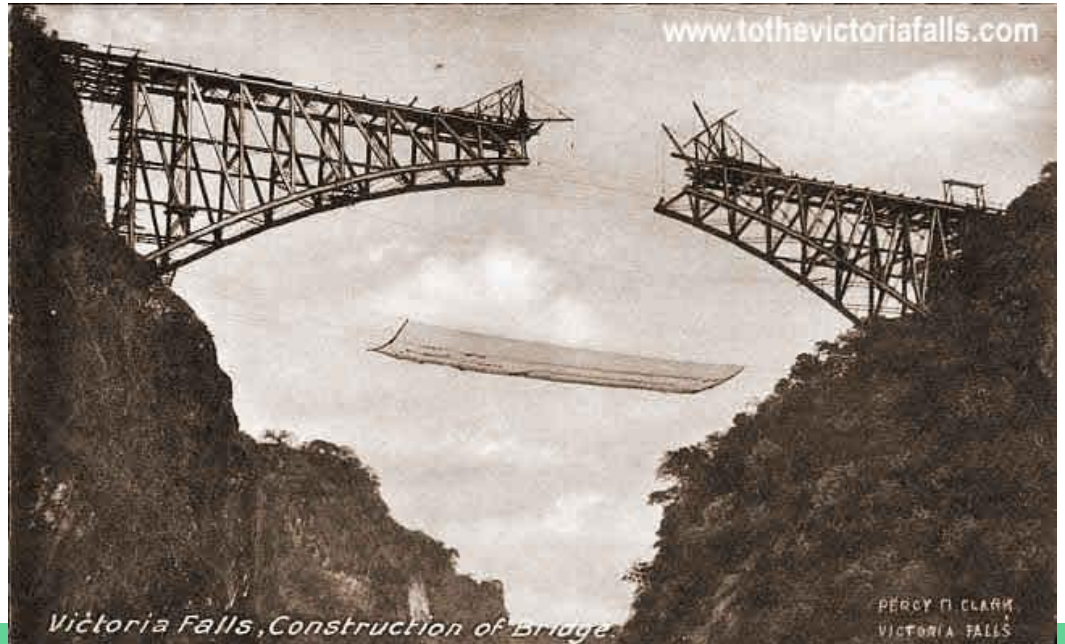
Hand-On Web Hacking - Modules

- Access Control
  - Broken Access Control
  - Insecure Direct Object Reference
- User Enumeration
- Brute Forcing
- SQL Injection
- Cross-Site Scripting
- Insecure Web Services

Setup

Requirements - Targets

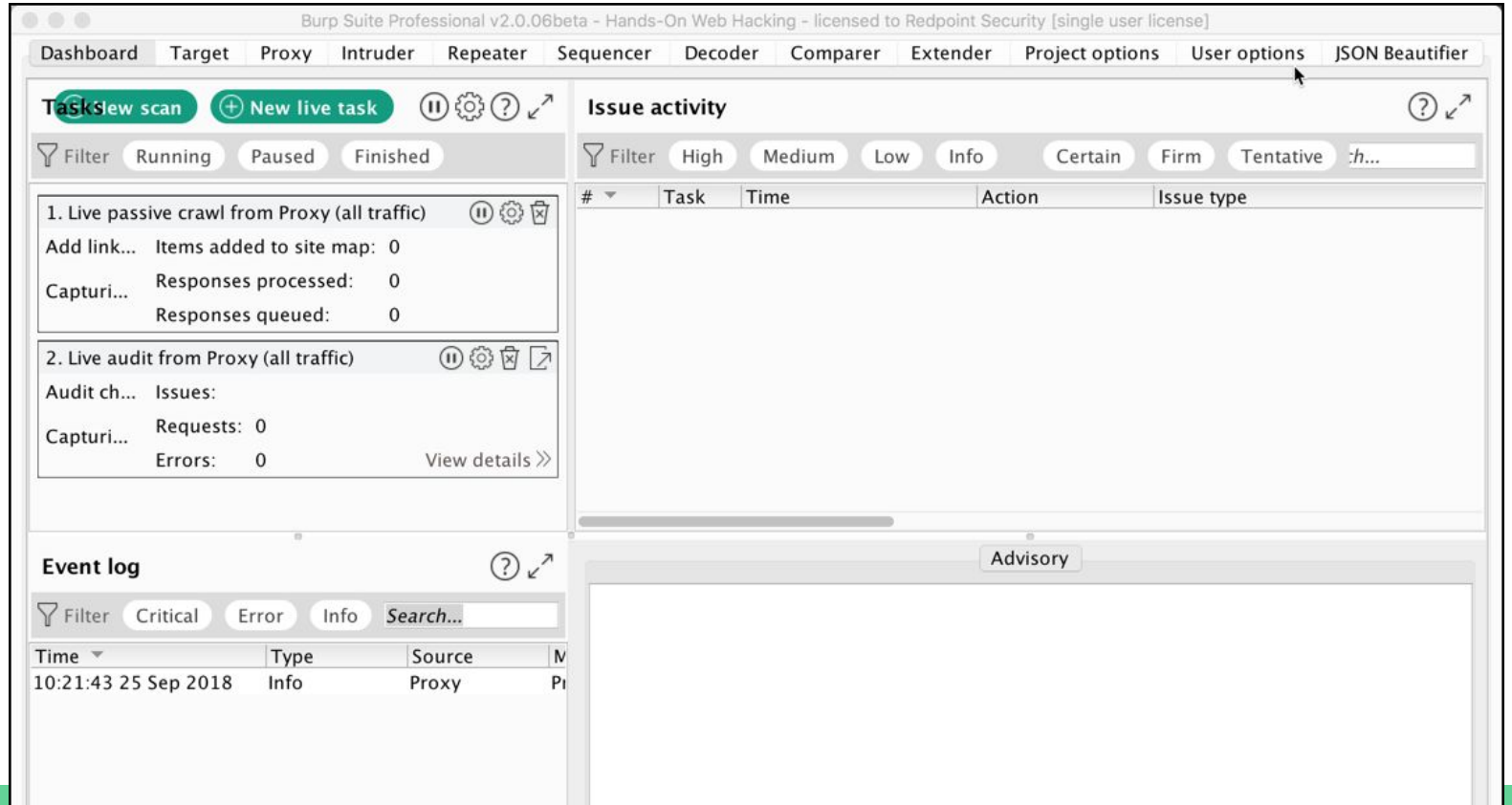
- https://github.com/justinlarson/Web-App-Hacking-Workshop
- http://vtm.cheetahbiscuits.com:8000/
-



Requirements - Tools

- Go here https://github.com/justinlarson/Web-App-Hacking-Workshop
- Kali / Mac OS X / Windows (unsupported)
  - git clone https://github.com/justinlarson/Web-App-Hacking-Workshop.git
  - ./setup_Mac.sh
  - ./setup_Kali.sh
  - http://localhost:8000(vtm)
- Docker
- Git
- Sqlmap
- Brew
- Hydra
  - brew install hydra (Mac OS X)
- Google Chrome
- Burp Suite Professional

Burp Suite Professional - Walkthrough



Access Control

Broken Access Control

- Access Control - Policy that enforces that users cannot act outside of their intended permissions.
- OWASP Top 10 2017 - A5
- Broken Access Control - Ability to access higher privilege functionality by manipulating the request for resources.
- TLDR; Improper vertical access controls
- One of the most common and easily exploited flaws.

Broken Access Control

Broken Access Control - Discovery Techniques

- Brute-forcing
  - Common directories
  - Integer values
  - Using common sense
- External resources
  - Google/Bing/Yahoo hacking
  - Shodan.io
- Unintended data leakage
  - Hidden values in the developer console
  - HTML Comments
  - Source code

Broken Access Controls - Tools

- Burp Suite Intruder
- Be smart about the wordlists you use
  - SecLists (https://github.com/danielmiessler/SecLists)
  - RAFT Wordlist (shameless plug, included in SecLists)
- Watch your status codes
  - 403 vs 404 vs 302
- Alternates:
  - Dirbuster
  - ZAP
  - Custom script


IDOR

Insecure Direct Object Reference (IDOR)

- Ability to access data through tampering of direct object references
  - http://example.com/accounts/user/2
  - http://example.com/accounts?user=2
- TLDR; Improper lateral/horizontal access controls
- Does not have to be an integer
  - GUIDs can be copied/posted as well
  - Integers are easier to discover
- One of the most common and easily exploitable flaws, yet near impossible for a scanner to find without help.

IDOR - Discovery Techniques

- Brute-Forcing
  - Where in the app do you see qualifiers that change the application?
  - Check all parameters (GET/POST/PUT), folder paths, filenames, etc
- External Resources
  - Google and search engines are good at caching these items if linked
- Data Leakage
  - Any ID parameter is a potential, look through hidden app resources

Access Control - Walkthroughs

Broken Access Control - Directory Enumeration

- Target: Vulnerable Task Manager
- URL: http://localhost:8000/
- Tool: Burp Suite Professional - Intruder

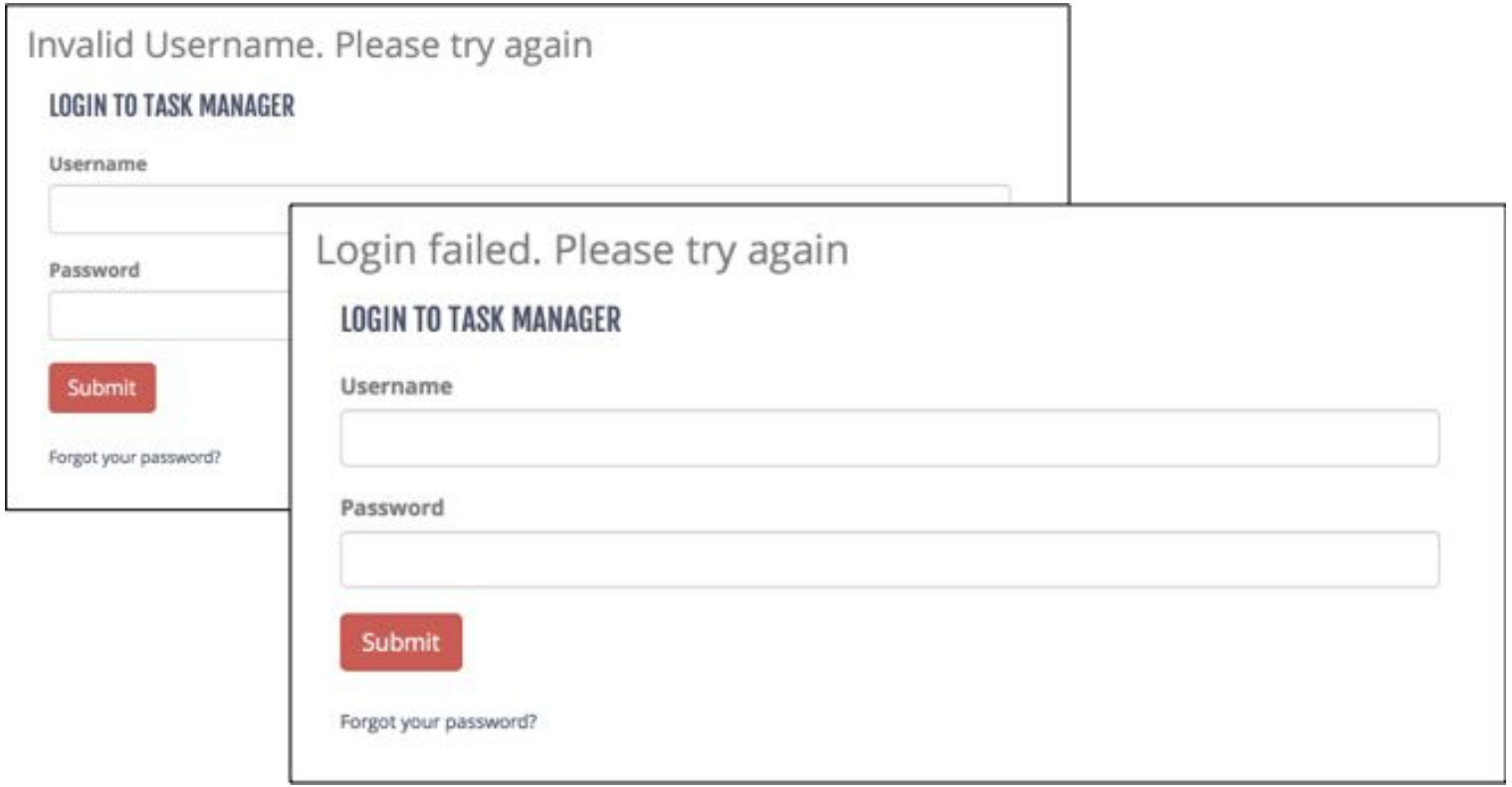
Insecure Direct Object Reference

- Target: Vulnerable Task Manager
- URL: http://localhost:8000/taskManager/profile/3
- Tool: Burp Suite Professional - Intruder

User Enumeration

User Enumeration

User Enumeration



User Enumeration

- Ability to enumerate user information through changing responses from requests using invalid and valid information.
  - “Provided username does not exist” vs. “Authentication failed. Try again”
- Response variations included error message, query string parameters, spacing, timing, etc.
- Disclosure of usernames, email addresses, phone numbers, etc.
  - This is ½ of the data required for authentication
- Can exist /anywhere/ the application responds to user-specific data
  - Login, account recovery, registration, messaging, etc.
- Devastating when paired with weak password policies, lack of anti-automation, and other lower-severity vulnerabilities.

User Enumeration - Discovery Techniques

- Brute-Forcing
  - Check the error messages for login, registration, forgot password, data flows
  - Check the timings for the above
  - Account Lockout
  - Remember Broken Access Control?
- External Resources
  - Social Media account harvesting
- Unintended Data Leakage
  - HTML Comments
  - API endpoints

User Enumeration - Tools

- Burp Suite Intruder
  - Includes some username lists
  - For emails and looking for common usernames, extend this out with gmail.com, hotmail.com, icloud.com, etc.
  - Learn the difference between positive/negative responses before starting intruder
  - Is it status code? Response length? Error message? Notification? Timing?
- Alternatives:
  - Custom script.

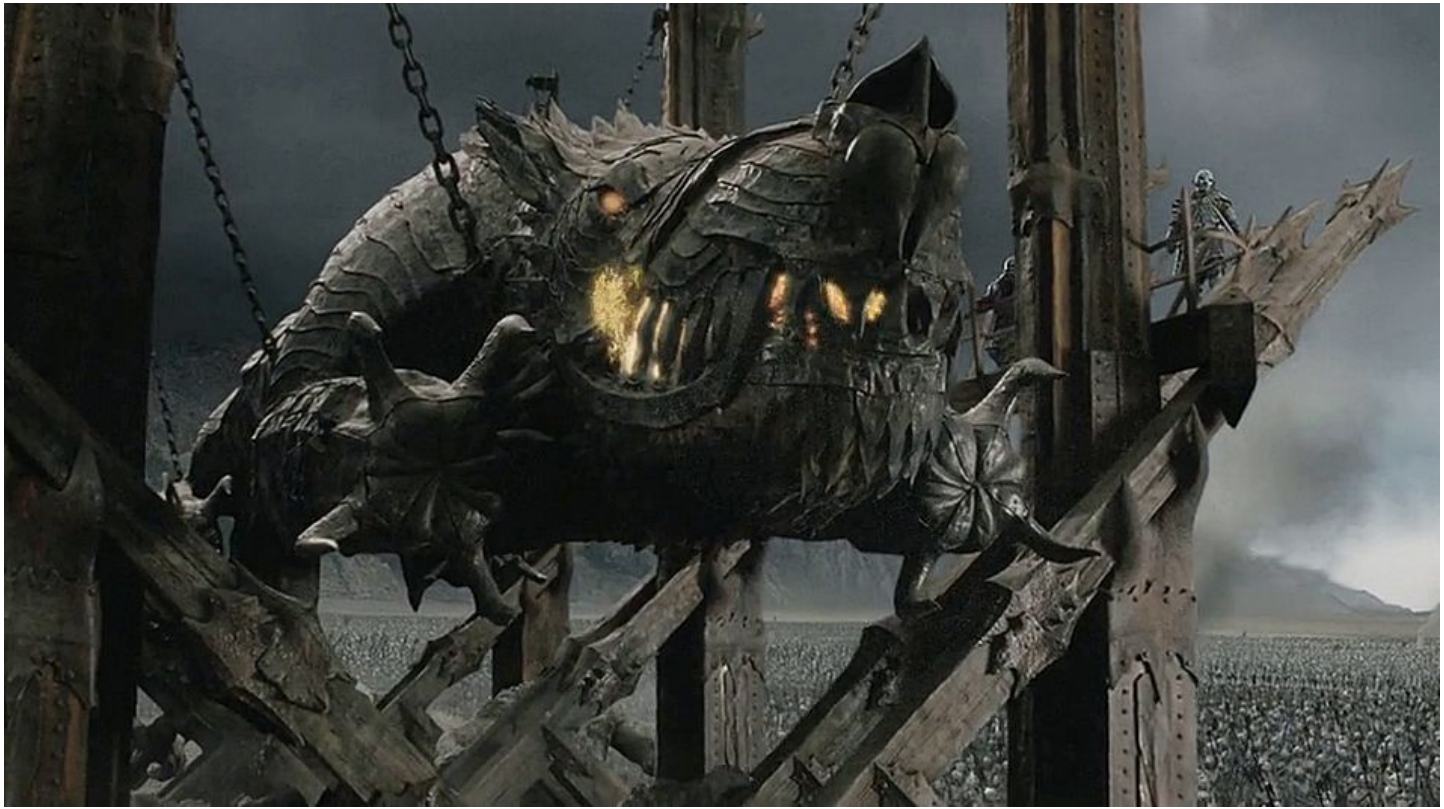
User Enumeration - Walkthrough

Username Enumeration

- Target: Vulnerable Task Manager
- URL: http://localhost:8000/taskManager/login/?next=/taskManager/
- Tool: Burp Suite Professional - Intruder

Brute Force

Brute Forcing



Brute Forcing

- Ability to send hundreds or thousands of requests without limit.
  - Eventually gaining access to unauthorized resources
- Monitor response variations including timing, error messages, status codes to determine whether access was successful.
- Try to determine full access details (username & password)
- Similar to other enumeration attacks, can exist /anywhere/ the application responds to user-specific data.

Brute Forcing - Discovery Techniques

- Account Lockout
  - How many attempts with a valid username before error message or timing changes?
- User Registration
  - Can you pre-register email addresses and accounts?
- Combination - Register an account and try to lock it out.
  - For accounts that require email addresses, use gmail and the `+` sign.

Brute Force - Tools

- Burp Suite Intruder
  - Includes both common username and password lists - can supplement
  - Learn the difference between positive/negative responses before starting intruder
  - What is behavior on successful access?
- Alternatives:
  - THC Hydra
  - Custom script

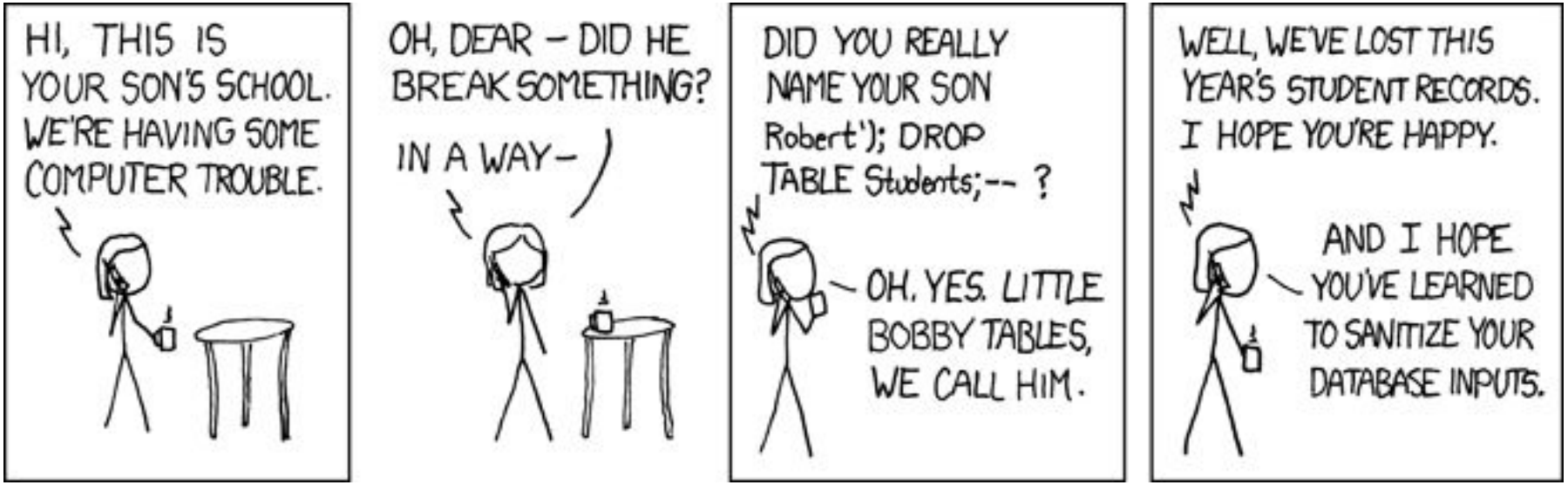
Brute Force - Walkthrough

Credentials Brute Force

- Target: Vulnerable Task Manager
- URL: http://localhost:8000/taskManager/login/?next=/taskManager/
- Tool: Burp Suite Professional - Intruder
- Tool: THC Hydra

SQL Injection

SQL Injection



SQL Injection

- Ability to influence the structure of a SQL database query through unsafe includes of user input
  - `stmt = "SELECT * FROM users WHERE username=" + INPUT[user]`
- Provides access to database at the permission level of the configured application's database user.
- Different database provide different ways to manipulate and use the database
  - Filesystem access: read and write files
  - Operating system access: run commands
- Query intent dictates the CRUD impact
  - Create = INSERT, Read = SELECT, Update = UPDATE, Delete = DELETE

SQL Injection

SQL Injection - Discovery Techniques

- Classic discovery is through in-band techniques
  - Error-based - misconfigured applications reveal SQL details.
  - UNION-based - app returns additional information on top of normal data
- Blind or Inferential
  - Boolean-based - time-consuming, but effective
  - Time-based - has to be reliable since network communications can vary in length
- Out-of-band
  - Leverages specific database functionality (e.g. DNS, HTTP, SMTP) to determine the result of boolean evaluations

SQL Injection

```
if request.method == 'POST':
    t_email = request.POST.get('email')

    try:
        result = User.objects.raw("SELECT * FROM auth_user where email = '%s'" % t_email)

        if len(list(result)) > 0:
            result_user = result[0]
            # Generate secure random 6 digit number
            res = ""
            nums = [x for x in os.urandom(6)]
            for x in nums:
```

SQL Injection - Tools

sqlmap - http://sqlmap.org

- Command Line tool for finding and exploiting SQL injection flaws
- Can point at an endpoint and it will test each parameter

Configuration Flags

- `--data=DATA` String to be sent through POST
- `--cookie=COOKIE` Session cookie for authenticated requests
- `--proxy=PROXY` Upstream proxy
- `-p TESTPARAMETER` Restrict testing to a single parameter
- `-r REQUESTFILE` Load HTTP request from file for testing

SQL Injection - Tools

Quick check flags for confirming vulnerability

- `-b, --banner` Retrieve DBMS banner
- `--current-user` Logged in DB user
- `--current-db`

Data Dump

- `--dump-all` Careful, dumps anything/everything
- `--dump` Dump targeted data
- `-D DB` Target Database
- `-T TBL` Target Table
- `-C COL` Target Column

SQL Injection - Walkthrough

Unauthenticated SQL Injection

- Target: Vulnerable Task Manager
- URL: http://localhost:8000/taskManager/forgot_password/
- Parameter: email
- Tool: sqlmap

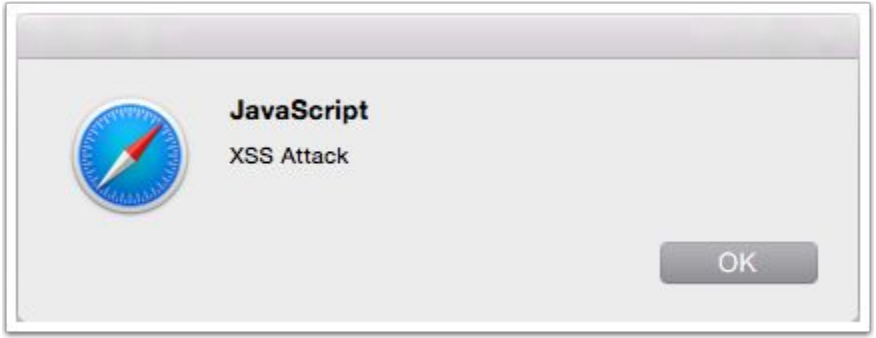
Authenticated SQL Injection

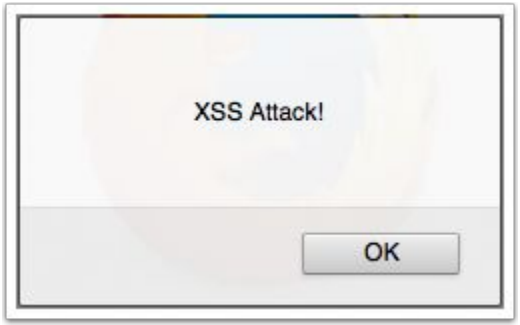
- Target: Vulnerable Task Manager
- URL: http://localhost:8000/taskManager/7/project_details/
- Parameter: URL Project ID parameter
- Tool: sqlmap

Cross-Site Scripting

Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS)





Cross-Site Scripting (XSS)

- Also known as Script or HTML Injection
- Ability to inject and execute arbitrary code within a user's browser
  - Search results for {{ query }}
- Violates the trust a user has with the application
- Allows complete control of the browser by an attacker
- Context is everything
  - Where does the user-controlled content land on the page?
  - HTML, HTML attribute, JavaScript, CSS?
- Three flavors
  - Stored
  - Reflected
  - DOM-based

Reflected XSS

- Non-persistent XSS
- Input is reflected back within the context of the immediate response
- Commonly seen in search functionality and error pages
- Most dangerous as a GET request parameter
  - Still exploitable as a POST
  - Test for Method Interchange to find instances where POST is accepted
- Request user interaction
  - Phishing email
  - Link in document/page/etc.

Stored XSS

- Persistent XSS
- User input is stored by the server and reflected back at any point following the initial request
- Commonly seen in public user profiles, messaging systems, and data tables.
- Requires no user interaction other than browsing to the page where the payload is rendered
    - Typically used in targeted attacks
    - Could be sped up by social engineering
- Enhanced by admin approval/support systems and as data is passed between applications

DOM-based XSS

- User input is processed by the browser, independent of the server, and added to the page through DOM manipulation
- Identified by tracing source to sinks in client-side code
- Commonly seen in applications that depend heavily on client-side rendering
- Difficult for scanners to find as most do not parse and execute JavaScript
- In some cases, servers never have a chance to protect the client
- Exploited similar to Reflected XSS
- Script tags won’t work for DOM XSS if the DOM is updated after the initial page load.
    - Event handlers are the preferred payload-delivery mechanism

XSS Discovery Techniques

- The (old) manual standby:

```
'"><script>alert(1)</script>
```

- Now look everywhere for an alert box or where the string could be displayed
- Hint: use something besides `1` as your alert string
- You might need more (or fewer) characters in the string to actually execute the attack.

XSS - Walkthroughs

Reflected XSS

- Target: Vulnerable Task Manager
- URL: http://localhost:8000/taskManager/search/
- Parameter: q

Stored XSS

- Target: Vulnerable Task Manager
- URL: http://localhost:8000/taskManager/7/13/
- Parameter: title

XSS - Walkthroughs

DOM-based XSS

- Target: Vulnerable Task Manager
- URL: http://localhost:8000/taskManager/login/?next=/
- Parameter: #

Insecure Web Services

Insecure Web Services

- Types
  - REST
  - SOAP
  - Websockets
  - Graphql (the new hotness)
- Generally not vulnerable to anything different than previous vulnerabilities
- There isn’t a UI to visualize the exploit working.

Web Services - Discovery Techniques

- Soap
  - Find wsdl file
  - Look for private services in the wsdl file
  - Fuzzing all parameters
  - XXE - External Entity Processing - weakly configured XML processing
    - Command injections, DOS, Server Side Request Forgery, etc
  - XPATH Injection - query XML documents
- REST
  - Look for exposed swagger UI
  - Fuzz endpoints
  - Access Controls
  - Data Leakage (too much data returned in response)

Web Services - Discovery Techniques

- Websockets
  - Identify if websocket is encrypted ws:// vs wss://
  - Authorization, Authentication, and Input Sanitization checks
- Graphql
  - Find graphql endpoint, can either be descriptive or non-descriptive
    - Descriptive (/graphql/, /graphql/console, graphql.php)
    - Non-Descriptive (/service, /data, /items)
- Introspection query
  - query IntrospectionQuery {__schema {queryType { name } mutationType { name } subscriptionType { name } types {...FullType .....
- SQLi, IDOR, Access Control

Web Services - Tools

- Burp Suite Professional
  - WSDLER
- Soap UI
- Postman
- ZAP
- Graphql-IDE

Security Misconfigurations

Security Misconfiguration

- OWASP Top 10 2017 - A6
- Server, framework, or other configuration setting that gives access to unintended functionality.
- Can exist at any layer of the application stack
- Includes:
  - Exposed administrative interfaces
  - Verbose error messages
  - Default sample applications
  - HTTP response headers
  - Framework security settings
- Usually easy for a scanner to find

Security Misconfiguration - Discovery Techniques

- Brute-Forcing
  - Can use same techniques as identification of access control issues
  - Might need to investigate framework/server if it’s not common
- External Resources
  - Google and search engines always good
  - Github/Stack Overflow of target developers
- Data Leakage
  - Application Framework server version
  - CSP headers
  - Robots.txt