

Advanced Web Attacks and Exploitation (OSWE) Preparation

Table of Contents

1 Introduction

- About the AWAE Course
- 1.1.2 OSWE Exam Attempt
 - Our Approach
 - Obtaining Support
 - Offensive Security AWAE Labs
 - General Information
 - Lab Restrictions
 - Forewarning and Lab Behavior
 - Control Panel
 - Reporting
 - Backups
 - About the OSWE Exam
 - Wrapping Up

2 Tools & Methodologies

- 2.1 Web Traffic Inspection
 - 2.1.1 Burp Suite Proxy
 - 2.1.2 Using Burp Suite with Other Browsers
 - 2.1.3 Burp Suite Scope
 - 2.1.4 Burp Suite Repeater and Comparer
 - 2.1.5 Burp Suite Decoder
- 2.2 Interacting with Web Listeners using Python
- 2.3 Source Code Recovery
 - 2.3.1 Managed .NET Code
 - 2.3.2 Decompiling Java Classes
- 2.4 Source Code Analysis Methodology
 - 2.4.1 An Approach to Analysis
 - 2.4.2 Using an IDE
 - 2.4.3 Common HTTP Routing Patterns
 - 2.4.4 Analyzing Source Code for Vulnerabilities
- 2.5 Debugging
 - 2.5.1 Remote Debugging
- 2.6 Wrapping Up

3 ATutor Authentication Bypass and RCE

- 3.1 Getting Started
 - 3.1.1 Setting Up the Environment
- 3.2 Initial Vulnerability Discovery
- 3.3 A Brief Review of Blind SQL Injections
- 3.4 Digging Deeper
 - 3.4.1 When \$addslashes Are Not
 - 3.4.2 Improper Use of Parameterization
- 3.5 Data Exfiltration
 - 3.5.1 Comparing HTML Responses
 - 3.5.2 MySQL Version Extraction
- 3.6 Subverting the ATutor Authentication
- 3.7 Authentication Gone Bad

- 3.8 Bypassing File Upload Restrictions
- 3.9 Gaining Remote Code Execution
- 3.9.1 Escaping the Jail
- 3.9.2 Disclosing the Web Root
- 3.9.3 Finding Writable Directories
- 3.9.4 Bypassing File Extension Filter
- 3.10 Wrapping Up

4 ATutor LMS Type Juggling Vulnerability

- 4.1 Getting Started
- 4.2 PHP Loose and Strict Comparisons
- 4.3 PHP String Conversion to Numbers
- 4.4 Vulnerability Discovery
- 4.5 Attacking the Loose Comparison
- 4.5.1 Magic Hashes
- 4.5.2 ATutor and the Magic E-Mail address
- 4.6 Wrapping Up

5 ManageEngine Applications Manager AMUserResourcesSyncServlet SQL Injection RCE

- 5.1 Getting Started
- 5.2 Vulnerability Discovery
- 5.2.2 Servlet Mappings
- 5.2.3 Source Code Recovery
- 5.2.4 Analyzing the Source Code
- 5.2.5 Enabling Database Logging
- 5.2.6 Triggering the Vulnerability
- 5.3 How Houdini Escapes
- 5.3.2 Using CHR and String Concatenation
- 5.3.3 It Makes Lexical Sense
- 5.4 Blind Bats
- 5.5 Accessing the File System
- 5.5.2 Reverse Shell Via Copy To
- 5.6 PostgreSQL Extensions
- 5.6.1 Build Environment
- 5.6.2 Testing the Extension
- 5.6.3 Loading the Extension from a Remote Location
- 5.7 UDF Reverse Shell
- 5.8 More Shells!!!
- 5.8.1 PostgreSQL Large Objects
- 5.8.2 Large Object Reverse Shell
- 5.9 Summary

6 Bassmaster NodeJS Arbitrary JavaScript Injection Vulnerability

- 6.1 Getting Started
- 6.2 The Bassmaster Plugin
- 6.3 Vulnerability Discovery
- 6.4 Triggering the Vulnerability
- 6.5 Obtaining a Reverse Shell
- 6.6 Wrapping Up

7 DotNetNuke Cookie Deserialization RCE

- 7.1 Serialization Basics
 - 7.1.1 XmlSerializer Limitations
 - 7.1.2 Basic XmlSerializer Example
 - 7.1.3 Expanded XmlSerializer Example
 - 7.1.4 Watch your Type, Dude
- 7.2 DotNetNuke Vulnerability Analysis
 - 7.2.1 Vulnerability Overview
 - 7.2.2 Manipulation of Assembly Attributes for Debugging
 - 7.2.3 Debugging DotNetNuke Using dnSpy
 - 7.2.4 How Did We Get Here?
- 7.3 Payload Options
 - 7.3.1 FileSystemUtils PullFile Method
 - 7.3.2 ObjectDataProvider Class
 - 7.3.3 Example Use of the ObjectDataProvider Instance
 - 7.3.4 Serialization of the ObjectDataProvider
 - 7.3.5 Enter The Dragon (ExpandedWrapper Class)
- 7.4 Putting It All Together
- 7.5 Wrapping Up

8 ERPNext Authentication Bypass and Server Side Template Injection

- 8.1 Getting Started
 - 8.1.1 Configuring the SMTP Server
 - 8.1.2 Configuring Remote Debugging
 - 8.1.3 Configuring MariaDB Query Logging
- 8.2 Introduction to MVC, Metadata-Driven Architecture, and HTTP Routing
 - 8.2.1 Model-View-Controller Introduction
 - 8.2.2 Metadata-driven Design Patterns
 - 8.2.3 HTTP Routing in Frappe
- 8.3 Authentication Bypass Discovery
 - 8.3.1 Discovering the SQL Injection
- 8.4 Authentication Bypass Exploitation
 - 8.4.1 Obtaining Admin User Information
 - 8.4.2 Resetting the Admin Password
- 8.5 SSTI Vulnerability Discovery
 - 8.5.1 Introduction to Templating Engines
 - 8.5.2 Discovering The Rendering Function
 - 8.5.3 SSTI Vulnerability Filter Evasion
- 8.6 SSTI Vulnerability Exploitation
 - 8.6.1 Finding a Method for Remote Command Execution
 - 8.6.2 Gaining Remote Command Execution
- 8.7 Wrapping Up

9 openCRX Authentication Bypass and Remote Code Execution

- 9.1 Getting Started
- 9.2 Password Reset Vulnerability Discovery
 - 9.2.1 When Random Isn't
 - 9.2.2 Account Determination
 - 9.2.3 Timing the Reset Request
 - 9.2.4 Generate Token List
 - 9.2.5 Automating Resets

- 9.3 XML External Entity Vulnerability Discovery
- 9.3.2 Introduction to XML
- 9.3.3 XML Parsing
- 9.3.4 XML Entities
- 9.3.5 Understanding XML External Entity Processing Vulnerabilities
- 9.3.6 Finding the Attack Vector
- 9.3.7 CDATA
- 9.3.8 Updating the XXE Exploit
- 9.3.9 Gaining Remote Access to HSQLDB
- 9.3.10 Java Language Routines
- 9.4 Remote Code Execution
- 9.4.2 Finding the Write Location
- 9.4.3 Writing Web Shells
- 9.5 Wrapping Up

10 openITCOCKPIT XSS and OS Command Injection - Blackbox

- 10.1 Getting Started
- 10.2 Black Box Testing in openITCOCKPIT
- 10.3 Application Discovery
- 10.3.1 Building a Sitemap
- 10.3.2 Targeted Discovery
- 10.4 Intro To DOM-based XSS
- 10.5 XSS Hunting
- 10.6 Advanced XSS Exploitation
- 10.6.1 What We Can and Can't Do
- 10.6.2 Writing to DOM
- 10.6.3 Creating the Database
- 10.6.4 Creating the API
- 10.6.5 Scraping Content
- 10.6.6 Dumping the Contents
- 10.7 RCE Hunting
- 10.7.1 Discovery
- 10.7.2 Reading and Understanding the JavaScript
- 10.7.3 Interacting With the WebSocket Server
- 10.7.4 Building a Client
- 10.7.5 Attempting to Inject Commands
- 10.7.6 Digging Deeper
- 10.8 Wrapping Up

11 Concord Authentication Bypass to RCE

- 11.1 Getting Started
- 11.2 Authentication Bypass: Round One - CSRF and CORS
- 11.2.1 Same-Origin Policy (SOP)
- 11.2.2 Cross-Origin Resource Sharing (CORS)
- 11.2.3 Discovering Unsafe CORS Headers
- 11.2.4 SameSite Attribute
- 11.2.5 Exploit Permissive CORS and CSRF
- 11.3 Authentication Bypass: Round Two - Insecure Defaults
- 11.4 Wrapping Up

12 Server Side Request Forgery

- 12.1 Getting Started
- 12.2 Introduction to Microservices
- 12.2.2 Web Service URL Formats
- 12.3 API Discovery via Verb Tampering
- 12.3.1 Initial Enumeration
- 12.3.2 Advanced Enumeration with Verb Tampering
- 12.4 Introduction to Server-Side Request Forgery
- 12.4.1 Server-Side Request Forgery Discovery
- 12.4.2 Source Code Analysis
- 12.4.3 Exploiting Blind SSRF in Directus
- 12.4.4 Port Scanning via Blind SSRF
- 12.4.5 Subnet Scanning with SSRF
- 12.4.6 Host Enumeration
- 12.5 Render API Auth Bypass
- 12.6 Exploiting Headless Chrome
- 12.6.2 Using JavaScript to Exfiltrate Data
- 12.6.3 Stealing Credentials from Kong Admin API
- 12.6.4 URL to PDF Microservice Source Code Analysis
- 12.7 Remote Code Execution
- 12.7.1 RCE in Kong Admin API
- 12.8 Wrapping Up

13 Guacamole Lite Prototype Pollution

- 13.1 Getting Started
- 13.1.2 Understanding the Code
- 13.1.3 Configuring Remote Debugging
- 13.2 Introduction to JavaScript Prototype
- 13.2.2 Prototype Pollution
- 13.2.3 Blackbox Discovery
- 13.2.4 Whitebox Discovery
- 13.3 Prototype Pollution Exploitation
- 13.4 EJS
- 13.4.1 EJS - Proof of Concept
- 13.4.2 EJS - Remote Code Execution
- 13.5 Handlebars
- 13.5.1 Handlebars - Proof of Concept
- 13.5.2 Handlebars - Remote Code Execution
- 13.6 Wrapping Up

14 Conclusion

- 14.1 The Journey So Far
- 14.2 Exercises and Extra Miles
- 14.3 The Road Goes Ever On
- 14.4 Wrapping Up