RSA Conference 2016

Learning Labs

Web Security Smackdown: Put Your Offensive Skills to the Test

Post-Conference Summary

Nora Sandler, Security Engineer

Max Chauhan, Senior Security Engineer

Security Innovation®

+1.978.694.1008 | www.securityinnovation.com | info@securityinnovation.com

Scenario Summary – Web Security Smackdown

Scenario Summary

- Players were tasked with hacking into Shadow Bank, a fictious online banking site.
- Shadow Bank contained more than 50 vulnerabilities ranging from SQL Injection and Cross-Site Scripting to cryptanalysis and cipher-cracking challenges. These are vulnerabilities commonly found in commercial and public applications.
- Players competed to earn the most points by attacking Shadow Bank; more challenging vulnerabilities were worth more points.
- Players needed to be creative and efficient to exploit as many vulnerabilities as possible to defeat their competitors.
- Players received a cheat sheet with tips to help get them started. They could also exchange points for hints or ask the lab facilitators for help.
- Players can return to https://cmdnctrl.net for up to a week after the event to keep playing.

Simulation Objectives

The Lab was designed to help participants gain the following skills:

- Learn How to Think Like a Hacker
  - Identify strengths and weaknesses in real-time
  - Understand how applications are attacked and exploited (in a safe sandbox)
- Discover Effective Attack Techniques
  - Cross-Site Scripting
  - SQL Injection
  - Business Logic Bypasses
  - Parameter Tampering
  - Unsafe File Upload
  - Authorization Bypasses
- Understand the Impact of Web Application Security
  - See how one code mistake can propagate throughout an application

Simulation Insight

Assume all input is malicious. Players tampered with all sorts of information sent to the server, including form fields, URL parameters, and uploaded files. They found that even values users don't normally modify, like hidden form fields and session cookies, are potential attack vectors if they contain data sent from the browser. Players also found that application security fails when developers assume that user input will make sense or adhere to the expected format. When the application expected an account number or dollar value, players instead sent negative numbers, SQL queries, and JavaScript to try to take control of the application.

System information disclosure provides valuable information to attackers. Using application stack traces and HTTP header information, players crafted sophisticated SQL injection attacks, identified the underlying web framework, and found known vulnerabilities in that framework. Although developers often consider verbose error messages and similar issues a low priority, this information is vital to an attacker crafting an exploit against your system.

Automated tools are useful, but not a silver bullet. Several players used automated security tools like sqlmap and Metasploit. Sometimes, automated tools allowed players to exploit vulnerabilities quickly and effectively. Other times, they distracted players from more important vulnerabilities that earned more points but were difficult to find automatically. Automated tools should be part of any assessment effort as they can find common and known vulnerabilities faster than humans, but they can't detect compound, business logic, or un discovered vulnerabilities. They should complement, not replace, manual testing.

Feedback from Players

"I didn't realize how simple some of the vulnerabilities are to exploit."

"An attacker can bypass all of my JavaScript validation? So that's why we need to do it server-side."

"Can I run an event like this for my students?"

Top Scores

1st Place – ek0sec (3820 points)

2nd Place – pwnsilver (2970 points)

3rd Place – OOTS (2770 points)