"Hacking Web Apps"

– Brent White / @brentwdesign

Abstract:

Understanding how to exploit vulnerabilities within a web application is beneficial to both breakers and fixers. It is a core skill for penetration testers, and it provides significant insight into secure coding practices for developers.

This talk covers all aspects of executing a web application penetration test. We will start with the discovery phase utilizing OSINT sources; such as, search engines, sub-domain brute-forcing and other methods to help you get a good idea of target's "footprint", all the way to fuzzing parameters to find potential SQL injection vulnerabilities. I will also discuss several of the tools and techniques that I commonly use. After this talk, you should have a good understanding of what is needed to break into web applications like a professional.

Detailed Outline:

Before you can start tearing into a web application for a customer, there are a few things that need to happen. There are rules and goals that need to be established for the assessment. This generally happens during the scoping and kick-off discussions. Since scoping and contracts make me sleepy, I will omit the details. The kick-off call is where the rubber starts hitting the road, and exists to bridge the gap between the legal contract and the final report using the Rules of Engagement document, also known as the RoE. The RoE answers the questions of who, when and what and identifies the boundaries between what is and is not allowed during the actual assessment. After the contents of this document have been agreed upon, you are good to go on executing the assessment.

=== Evidence Gathering ===
At the end of every assessment, a report is expected. The more evidence you collect and the better it's organized can really help shorten the time it takes to write the report as well as improve upon the value of it.
Before I start anything, I set up documents to keep evidence, notes and progress.

Here are a few tips:

- KeepNote is my program of choice. ("Dradis" is another popular choice in Kali.)
  - Available for Linux, Windows and Mac.
  - Allows you to paste screenshots, create multiple embedded pages and folders.
  - I'll group things based on vulnerability: Folder named "Default Server Pages" and each page will be the host & port.
  - I'll also color-code the folder based on risk-rating. (personal preference)
  - Allows for easy exporting of the KeepNote notebook as .html files & easy copy/paste into your program of choice while report writing.
- Document the HTTP GET Request and Response for each vulnerability.
- Document any unscheduled downtime.
- Document changes in test data.

    - Creation of additional accounts, passwords, or client-provided data.
    - Intentional or inadvertent modification of any static data resource.
    - Log all monetary transactions.
- Do not share screenshots or data/specifics of a great "hack" or "idiotic" vulnerability online.
- Make sure to get relevant, legible screenshots of the vulnerability.
- Call out or highlight the specific payload during the write-up, so the client can see exactly what was sent. Don't just leave it as URL or Base_64 encoded, etc...
- List all known affected pages and parameters for a vulnerability. Not just the one sample that shows the vuln, but everything else you found.
- Have a methodology and checklist to go by so that you're not forgetting or missing something during each assessment. OWASP has a good methodology and checklist.

=== Discovery / OSINT ===
I like to start my assessments by utilizing the power of "Open-Source Intelligence" (OSINT) to see what information I can find about the web application that others might already know.
Here are just a few OSINT resources and tools:

- Search engines, Pastebin, ShodanHQ, etc... for anything related to the application.
    - This is a manual and time-consuming process, but not one to be ignored!
    - You're looking for anything related to that application // emails, logins, dev help forums.
    - Any leaked creds, known previous hacks?
    - Sites like urlvoid.com to check for known active malware/threats for the domain.
    - I have found DB-type, schemes and test credentials through old development-help forums that were STILL valid!
- "Discover" by Lee Baird
    - Active/Passive scanning for domains, people, domain-squatting, black-listed DNS reporting.
    - Integrates many tools such as dnsrecon, theharvester, goofile, goog-mail, goohost, can start a Metasploit listener, parse XML and more.
- "The Harvester" by Christian Martorella / Edge-Security Research
    - Scrapes data from Bing, Google, Google Profiles, Ask, Jigsaw, LinkedIn, and Yahoo.

== Automated Scanning (Low-hanging fruit) ==
"Why do you run automated tools? You're supposed to be a hacker." Good question. Automated tools are a God-send as consultants usually have a very limited time to complete the project. They are very helpful at covering a wide range of tests and content discovery in a very short amount of time. However, You can't just run automated scanners and spit out a report.
That's just a vulnerability scan, and a piss poor one at that. Even though the scanners help, they don't contain the human element of manual testing and can't find everything. You can't just rely on the scanner, but it's a great addition to the assessment.
Here is a list of automated scanners that I like to use:

- Nessus
    - Looks at the host and web app. Covers everything from SSL/TLS layer, content discovery, basic CGI vulns and more.
- IBM App Scan
    - More web app focused for things such as SQL/LDAP injections, CSRF and XSS, etc...

- BurpSuite Pro
    - Built-in "Active" scanner, content spider and content discovery brute-forcer tools.
    - Tools for fuzzing parameters, crafting requests and much more.
    - Extensions available such as CSRF Scanner, XSS Validator and more.
- Nikto
    - Great for finding default pages, logins, known vulnerable scripts, CGI testing and more. Nikto is actually built in to Nessus, but can add a great deal of time to the Nessus scan if enabled, especially if there are multiple applications and hosts being scanned at the same time.
    - I prefer to un this separately, outside of Nessus. It gives me more control.
- WPScan (for WordPress sites)
    - Identifies known vulnerabilities in WordPress, enumerates themes and plug-ins and can also enumerate usernames.
    - Also other content-management system scanners out there for Joomla, Drupal and more.
- "DirBuster" by OWASP for directory and file discovery
    - I also load lists into BurpSuite Pro for discovery.
    - BurpSuite Pro also has a "Discover Content" and "Content Spidering" options to find linked content as well as brute-force content discovery.
- There are many more pre-installed options in Kali. Find them at: Kali Linux > Web Applications > Web Vulnerability Scanners
- Other scanners available are Saint and Nexpose.

Automated Scanner Pro Tips:

- You want to verify the settings of the automated scanner. Don't just blindly click "scan" after entering a URL.
- Make sure that you're not using something that's checking for Denial-of-Service (DoS) unless specifically requested by the client.
- Number of threads/connections at a time? Don't flood the host.
- Add any pages/functions the client has asked you to avoid. (Password reset page, sign-up page, "Contact" form, etc...)
- Might need to specify a specific "Page not found" or Error 404 page to help weed out false-positives for file discovery.
- Configure the log-in process and credentials for authenticated scans.
- You must take the time to verify the results and weed out any false-positives.

== Manual testing ==

- Automated Scanner results
    - Document the vulnerabilities that are legitimate. Can you take it further than the scanner?
        - For example, if the scanner suspects SQL injection, see if you can exploit that with something like SQLMap.
- Explore the application through a proxy program like BurpSuite Pro.
    - Manually explore the app while the "Spider" and "Content Discovery" tools are running.
    - OWASP's "DirBuster" is a brute force tool to find files and directories.
- Review the server response to help verify what the server is running (IIS, Apache)
- Look for parameters to fuzz
    - How do they handle characters and commands outside of the normal actions?

    - Parameters can be directly in the URL. They are also found in HTTP GET and POST requests
- Try Cross-site Scripting (XSS) payloads, Cross-site Request Forgery (CSRF), SQL/LDAP injection, Local File Inclusion (LFI) and Remote File Inclusion (RFI)
    - BurpSuite Pro has lists available that you can load for file discovery, XSS, SQLi, usernames and more. There are also several places to download your own lists, or make your own.
    - "Xenotix" by OWASP is a XSS tester.
    - Save the POST or GET request and execute with SQLMap to search for SQL injection
- Is sensitive info being passed through the URL in a GET request?
    - Usernames, Passwords, Session ID, etc...
- Look for valuable comments in the source code of the HTTP Responses
    - I've found internal IPs, database names, usernames, "hidden" admin URLs, database calls in JavaScript and more!
    - This is certainly a manual process and is time-consuming. However, a thorough and consistent review will pay off.
- Authentication - can it by bypassed or broken?
    - Can you access URLs and functions as an unauthenticated user that you could while logged in?
    - Can you re-use the session token after logging off? Is there a "log off" feature?
    - Can you have multiple sessions as the same user at the same time?
    - What are the password requirements? Can you set your password to "password"?
    - Can you re-use a previous password?
- Look at the host, not just the web app
    - Identify the web server platform. Apache, IIS?
        - If it's an old/outdated install, look for exploits
    - Is there an admin portals available?
        - cPanel, Apache Tomcat Manager, etc...
    - Are there test and default credentials available?
    - Search for backup, default and obsolete files.
    - Is directory browsing enabled? What about directory rights?
    - Look for dangerous HTTP methods
        - ie...PUT, DELETE, TRACE
    - Are they vulnerable to directory traversal, Shellshock, Heartbleed, etc...?
    - Use Nmap to see what ports/services are open to the public. Is it just 80 & 443? Or, are their other things available too? Check them out.
    - Look at SSL/TLS settings for known and weak cipher vulnerabilities and expired or untrusted certificates.
        - Use SSLScan in Kali or "TestSSLServer.jar". Qualys SSL Labs is a good online resource too but there could be a potential disclosure issue.
    - SSL Enforcement - Can you access HTTPS areas by HTTP?

There are many different areas of a web application that need to be looked at in order to conduct a thorough penetration assessment. Check out the methodology from OWASP as well as several checklists that are available to ensure that you're covering all aspects.

For questions, please contact me on Twitter at @brentwdesign or email me at BrentWhite@Solutionary.com