Automating Security Testing

Mark Fallon
Senior Release Manager
Oracle

ORACLE

Some Ground Rules

- There are no silver bullets
- You can not test security into a product
- Testing however, can help discover a large percentage of implementation issues
- Automation of that testing is essential
  - It allows the process to scale
  - It reduces cost for continued maintenance
  - It helps ensure repeatability in process
- Security testing is an essential part of the overall development process

Development Process

- Ever growing number of development processes available
- Some organizations will use different processes for different projects
- Artifacts for security testing remain the same
  - Code
  - Generated objects e.g. binaries, class files
  - Running systems
- Other areas of software assurance are potentially more significantly impacted
  - Some areas of the process needs to be adapted to ensure assurance requirements are met

Code Scanning

- Code scanning is the first phase of our automated security checks
- Can start as soon as the first lines of code are committed to the project
- Helps to provide instant feedback to developers
- Can help enforce coding rules
  - Avoid poor coding styles
  - Enforce use of mandated functions
- Can reliably find a number of data flow & semantic issues e.g.:
  - Format String
  - Injection Attacks – SQL, Process, XML
  - Certain classes of buffer overflow

Rolling Out Tools

Three step program:

1. Eliminate the noise:
   - Need to present a sanitized list of issues to developers
     - Builds trust in tool
     - Makes best use of their time
2. Eliminate the backlog
   - Time needs to be added to the schedule
   - Helps to build awareness and momentum
3. Make it part of daily development
   - Provide immediate feedback to developers
   - Stops the build up of new issues

Automated Testing

- Second phase of our automated security testing
- This testing starts as soon as there is running code
- Three areas of automated tests:
  - Targeted
  - Generated
  - Leverage

Targeted Tests

- New functional/unit tests written to exercise a specific feature
  - Standard functional tests for security functionality
    - Transparent Data Encryption
  - Unit tests for feature that are used in security contexts
    - random numbers
- Destructive Protocol tests
  - Typically done with a parameterized client
  - Different attack vectors or combinations can be easily automated
  - Stateful protocols require more effort, problem may not be with initial connection
- Domain expert usually required to write these tests
- Some 3rd party test suites available in this area
  - Usually available for public protocols
    - http, ftp, LDAP etc.
  - Need to be evaluated for coverage
    - How much / which versions of protocols covered
    - Types of attacks they try.
  - Deployment issues need to be accessed
    - Fit into current testing infrastructure

Generated Tests

- Automate what others are trying against our system
  - Similar approach to application scanners
- Generated to find classes of vulnerabilities across an entire attack surface
- Most success with PL/SQL and code loaded into database
  - Data dictionary helps drive automation
    - List of PL/SQL functions – (attack surface)
    - Provides type information
      - Helps with generating parameterized attacks
- Need to be careful of false negatives
  - Certain functions will require some state to execute
  - If that state is evaluated before vulnerable parameter is, the call maybe rejected before we hit the potential issue
  - Test harness is updated with necessary code to create a valid state and this is passed in when testing the other parameters

Design To Test For Security

- Features and diagnostics that exist as part of the product that can be leveraged for security testing
- New features have been added to specifically enable better security testing
  - As these are destructive they are typically enabled by relinking a new object into the server.
- These can be used the following ways
  - In combination with the generated or targeted tests to help find issues
  - To inject errors to test for robustness
  - To turn existing functional tests into security tests

Leverage Testing

- Large body of functional regressions tests have already been written as product has matured.
  - Database product has some 250,000 nightly regressions
- These tests can be leveraged in two ways
  - Add extra checks to the results looking for specific issues
  - Enable diagnostic or specific tests features to find classes of issues.
    - This can also be done with 3rd Party tools like memory debuggers
- Some examples of extra checks are:
  - Scanning trace and log files looking for information leakage
  - Checking for file / process permissions on install tests

In Combination With Other Tests

- Done with diagnostics to highlight issues.
- Diagnostics have been added to help in support situations, typically enabled by an event.
- Example:
  - Database does it's own memory management
  - We can enables extra checking and protected pages, to detect overruns and corruption
  - Would be turned on when running the generated PL/SQL tests
- Some of this can also be achieved with 3rd party memory tools – valgrind, Purity etc.

In Combination With Other Tests

- Testing features added to the product to leverage existing tests
- Enabled through relinking of binaries
  - Literal blow-up tests
    - Parser maximizes size of all string literals
    - Turns 250,000 functional tests, into 250,000 buffer overflow tests
- Testing features added to the product to enable destructive tests
  - Robustness testing done with Iterative Kill Test

Application Scanners

- Application scanners used in final stages of automated security testing
- Language agnostic – black box testing against running applications
- Like code scanners lengthy evaluation process performed
  - Some had been used internally before
- Will only test exposed interfaces

Application Scanners

- Type of issues we see them finding
  - Parameter manipulation
  - SQL injection
  - Directory Traversal, XSS
  - Deployment issues
    - http error code checks
    - Apache configuration checks

Vulnerability Coverage

- How do you know you are done?
- Testers familiar with the concept of code coverage
- With security testing it is not that you touched a code block but how it was touched
- Vulnerability Coverage – a metric in progress
  - Enumerate the attack surface
    - API's, command line, open ports etc.
  - Enumerate the attack vectors
    - SQL injection, buffer overflow, XSS
  - Augment test scripts, tool results to log combinations
- This metric can be done to help measure completeness of the automated security tests

Acknowledgements

- Security Program Office
- Security Assurance Group
- Integration team
- QA team
- PL/SQL group
- VOS group