EVALUATION OF WEB SECURITY MECHANISMS USING VULNERABILITY ANALYSIS & PATTERN MINING

BIMAL VARGHESE

Guide : Ms. SIMI STEPHEN

FISAT MOOKKANNOOR

May 17, 2015

OUTLINE

INTRODUCTION

- Usage of Web Applications are common these days
- Internet boom have made them more popular to common man.
- Usage include Social Networking, Online Banking,Online Shopping, Emails etc.
- Global accessibility of Web Applications increases its risk.

NEED FOR WEB APPLICATION SECURITY

- Primary Responsibility - Application developers
  - Need to have an understanding of magnitude and relevance of assets they handle.
- Common reasons why securing a web application becomes tricky
  1. Numerous languages and frameworks
  2. Exposure to huge number of audience
  3. Developer inexperience.
  4. Need for remote access of Organizational resources.

SECURING THE WEB APPLICATIONS

- Security Standers
- Tools for evaluating security.
- Counter measures.
- Proper Training.
- Auditing and Patching

LITERATURE SURVEY

[1] Fault Injection and Dependability Evaluation of Fault-Tolerant Systems

- Fault Injection in Traditional System.
- Utilizes fault injection to explicitly remove design or implementation faults in a complex fault tolerant system.
- Aims in reducing, by verification, the presence of faults
- Faults injected to uncover potential issues and to improve the system

[2] Xception: Software Fault Injection and Monitoring in Processor Functional Units

- Software implemented fault injection (SWIFI) - for high complex systems.
  - Difficult to control and observe the fault effects inside the processor.
  - Detection of the activated faults is very complex
- Simulation based fault injection is proposed.
- Fault Emulation
  - Application execution is interrupted
  - Specific fault injection software code is executed.

[3] Emulation of Software Faults: A Field Data Study and a Practical Approach

- Injection of representative software faults.
- Base principle - “Software faults is the root cause of computer failures”.
- Bugs in complex software have serious effect on the system.

CONTD ...

Software fault are injected according to following principle:

- Fault is injected to a component to evaluate the component in the presence of faulty component.
    - Separation between target component and system under observation.
- System behavior in presence of faulty component is observed.

Advantages.

1. Validation of fault-tolerant mechanisms.
2. Prediction of worst-case scenarios and experimental risk assessment.
3. Dependability benchmarking.

[4] Using Attack Injection to Discover New Vulnerabilities

- Existence of a vulnerability may not cause a security hazard until it is exploited.
- Intrusion can be prevented by removing vulnerability.
- Can be done at -
  1. Development phase : identify programming flaws.
  2. Operational phases : discovery of configuration errors and other similar problems.

CONTD ...

- AJECT (Attack inJECtion Tool) used for vulnerability detection and removal.

  1. Simulates the behavior of an adversary by injecting attacks against a target system.
  2. Observes execution of the system to determine if the attacks have caused a failure.
  3. If failure occur, presence of vulnerability identified and traditional debugging methods employed to fix it.

- Experiment conducted with IMAP servers.

[5] Finding Security Vulnerabilities in Java Applications with Static Analysis

- Popularity of Web Applications & hidden Vulnerability in it.
- Exposure to wider audience.
- Inability of detection using firewalls & other methods
  - Attacks utilizes *http* which is unhindered in firewalls.
- High level languages (eg.Java) provides language level security.
  - Restrict direct memory access.
  - Automatic Garbage collection etc.
- Logic errors can compromise Web Application security.
- Static code analysis detects these issues.

[6] An Empirical Analysis of Input Validation Mechanisms

- Application Security & Programing Language efficiency.
  - How bad a programing languages in term of propensity of mistakes.
- Type System (Strong / Weak) & Type checking (Static / Dynamic) in software robustness.
- A strong typed language with a static type checking can help deliver a safer application without affecting its performance

[7] Preliminary Results on Using Static Analysis Tools for Software Inspection.

- Software code inspections & Software Quality
  - Can detect as little as 20% to as much as 93% of number of defects in a software.
- Defect classification scheme was proposed.
- Vulnerability discoverys model(VDM)
  - Ability of a system to perform its required functions without software-caused violations of its explicit or implicit security policy.

CONTD ...

Two nature of software systems are considered.

1. Engineering nature:
    - Employs statistical analysis of vulnerabilities
    - Features like when was a vulnerability introduced, when was it discovered, how is the source code of a system changing, etc.
2. Economic nature:
    - Features like what is the auction-ascertained price of a previously-unreported vulnerability in a specific system.
    - First person to report vulnerability receives the reward.

[8] Semi-Automatic Security Testing of Web Applications from a Secure Model

- Non Monolithic nature and Distributed components in Web Applications.
- White-box penetration testing:
  - All applications are to develop in the same language
- Black-box penetration testing:
  - Not highly effective because of weaknesses of the crawling step which misses lots of potential interaction with the user
- Model checkers for security analysis was proposed

CONTD . . .

- For System Under Validation (SUV),formal model **M** is used.
- Vulnerability is injected by mutating the formal model of the web application.
- Model checker outputs attack traces that exploit those vulnerabilities.
- Attack traces are translated into concrete test cases.
- Tests are executed on the real system using an automatic procedure

[9] GAUGING SOFTWARE READINESS WITH DE- FECT TRACKING

- In competitive commercial market,time of release is very important for software.
- Strict Deadlines have to be met for programmers.
- Softwares with known bugs are released to meet the time.

- To judge, if a software is ready to meet the market
  - Measure defect density ie, number of defects per line of code
  - Separate defect reports into groups and track them separately

CONTD . . .

- Track the number of defects reported in each pool and the number of total defects reported.
  $Defects_{total} = \frac{Defects_A * Defects_B}{Defects_{(A+B)}}$
- The number of unique defects reported at any given time is:
  $Defects_{unique} = Defects_A + Defects_B - Defects_{(A+B)}$

where A & B two groups considered

Table: Comparison of various vulnerability analysis methods

Sl.No.	Paper Name	Method Used	Implemented on
1	Fault Injection and Dependability Evaluation of Fault-Tolerant Systems	Fault Injection	Hardware Level
2	Xception: Software Fault Injection and Monitoring in Processor Functional Units	Fault Injection	Software Simulation
3	Emulation of Software Faults: A Field Data Study and a Practical Approach	Bug Injection	Software Components
4	Using Attack Injection to Discover New Vulnerabilities	Server Software	IMAP
5	Finding Security Vulnerabilities in Java Applications with Static Analysis	Static Code Analysis	Java
6	Preliminary Results on Using Static Analysis Tools for Software Inspection	Source Code Analysis	Coding Standard
7	Semi-Automatic Security Testing of Web Applications from a Secure Model	Modal Analysis	Web Application Model
8	Gauging Software Readiness with Defect Tracking	Defect Density	Software Defects

METHODOLOGY

METHODOLOGY

- Based on injection of realistic vulnerabilities and the subsequent controlled exploit of those vulnerabilities to attack the system.
- Can be used to test counter measure mechanisms Like IDS, Firewalls etc.
- Vulnerability Attack Injection tool (VAIT) is created.
- Inspects application for input validation vulnerabilities. Like SQLi,XSS etc.

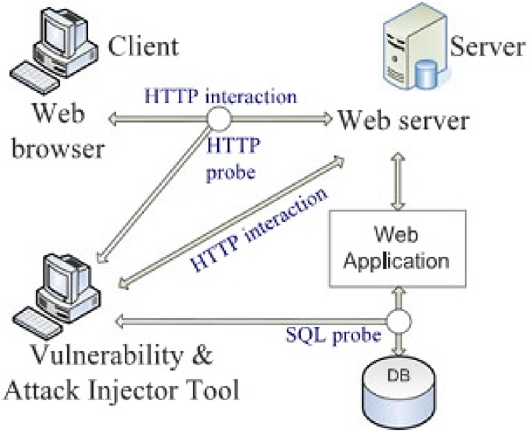


Figure: VAIT setup

ATTACK PROCEDURE

4 main stages

1. Preparation stage
   - Crawls Web Application.
   - Analyze HTTP & SQL communications.
   - Generate correlation between http input and SQL queries
2. Vulnerability injection stage
   - Analyze source code.
   - Inject vulnerability.
     - Done by removing the protection of the target variables say call to a sanitizing function
   - perform specific code mutation in order to inject one vulnerability in that particular location.

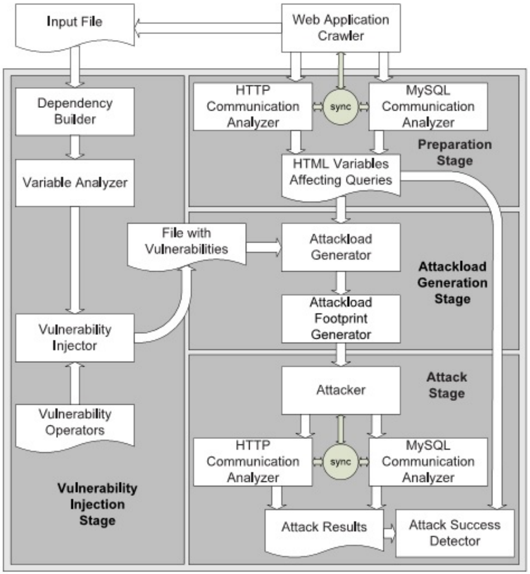
CONTD . . .

3. Attackload generation stage
   - Attackload - Malicious activity data, needed to attack a given vulnerability
   - Built around the interaction patterns derived from the preparation stage
     - Through fuzzing process.
     - prefix (>,), ‘, “, . . .)
     - suffix (<, , #, ‘, ”, . . .)
4. Attack stage
   - Malicious interaction with web application.
   - Alter SQL query or HTML data.
   - Vulnerable source code files are injected one at a time.
   - SQL & HTTP probes are again deployed.
   - Attack footprints analyzed for success.

VULNERABILITY & ATTACK INJECTOR TOOL

- VAIT - performs attack injection methodology.
- Targets Linux,Apache. MySQL, PHP (LAMP) applications.
- Process done with minimum human intervention.
- Interactions can be manual or through automating tools.
- Monitoring done using built in proxies.

VAIT architecture.


Input File
Web Application Crawler
Dependency Builder
Variable Analyzer
File with Vulnerabilities
Vulnerability Injector
Vulnerability Operators
Vulnerability Injection Stage
HTTP Communication Analyzer
sync
MySQL Communication Analyzer
Preparation Stage
HTML Variables Affecting Queries
Attackload Generator
Attackload Generation Stage
Attackload Footprint Generator
Attacker
Attack Stage
HTTP Communication Analyzer
sync
MySQL Communication Analyzer
Attack Results
Attack Success Detector

- SQLi & XSS attacks web Application.
- Remote Code Execution (RCE) & File Inclusion (FI) attacks the system on which the application runs.
- VAIT tool will be modified so as to identify other validation errors like RCE & FI.
- By utilizing Pattern mining methods, VAIT tool can identify similar vulnerability of web applications .

CONCLUSION

CONCLUSION

- Methodology can analyze, validation vulnerabilities in Web Applications
- Vulnerabilities are derived from extensive field study.
- VAIT tool will be able to identify validation issues.
- VAIT tool can be modified to identify similar vulnerability of web applications .

REFERRENCES

REFERRENCES I

[1] cgisecurity.net, www.cgisecurity.com/articles/csrf-faq.shtml# whatis, Dec. 2008.

[2] G. Alvarez and S. Petrovic, “A New Taxonomy of Web Attacks Suitable for Efficient Encoding,” Computers and Security, vol. 22, no. 5, pp. 435-449, July 2003

[3] S. Christey and R. Martin, “Vulnerability Type Distributions in CVE,” Mitre Report, May 2007.

[4] J. Duraes and H. Madeira, “Emulation of Software Faults: A Field Data Study and a Practical Approach,” IEEE Trans. Software Eng., vol. 32, no. 11, pp. 849-867, Nov. 2006.

[5] M.R. Lyu, Handbook of Software Reliability Engineering. IEEE Computer Society Press & McGraw-Hill, 1996.

[6] J. Williams and D. Wichers, “OWASP Top 10,” OWASP Foundation, Feb. 2013.

[7] N. Neves, J. Antunes, M. Correia, P. Verissimo, and R. Neves, “Using Attack Injection to Discover New Vulnerabilities,” Proc. IEEE/IFIP Intl Conf. Dependable Systems and Networks, 2006.

[8] J. Fonseca, M. Vieira, and H. Madeira, “Testing and Comparing Web Vulnerability Scanning Tools for SQLi and XSS Attacks,” Proc. IEEE Pacific Rim Intl Symp. Dependable Computing, Dec. 2007.

[9] S. Clowes, “A Study in Scarlet, Exploiting Common Vulnerabilities in PHP Applications,” http://www.securereality.com.au/ studyinscarlet.txt, 2013.

[10] B. Livshits and S. Lam, “Finding Security Vulnerabilities in Java Applications with Static Analysis,” Proc. USENIX Security Symp., pp. 18-18, 2005.

[11] M. Buchler, J. Oudinet, and A. Pretschner, “Semi-Automatic Security Testing of Web Applications from a Secure Model,” Proc. Intl Conf. Software Security and Reliability, 2012.

[12] S. McConnell, “Gauging Software Readiness with Defect Tracking”, IEEE Software, vol. 14, no. 3, May/June 1997.

[13] N. Nagappan, L. Williams, J. Hudepohl, W. Snipes, M. Vouk, “Preliminary Results on Using Static Analysis Tools for Software Inspection.” Proc. Intl Symp. Software Reliability Eng., pp. 429-439, 2004.

REFERRENCES II

[14] OSVDB, “Open Sourced Vulnerability Database,” http://osvdb. org, May 2013.

[15] D. Powell and R. Stroud, “Conceptual Model and Architecture of MAFTIA,” Project MAFTIA, Deliverable D21, 2003.

[16] J. Fonseca and M. Vieira, “Mapping Software Faults with Web Security Vulnerabilities,” Proc. IEEE/IFIP Intl. Conf. Dependable Systems and Networks, June 2008

[17] S. Fogie, J. Grossman, R. Hansen, A. Rager, and P. Pektov, XSS Attacks: Cross Site Scripting Exploits and Defense. Syngress, 2007. Intell., vol. 32, no. 6, pp. 11271133, Jun. 2010.

[18] J. Arlat, A. Costes, Y. Crouzet, J.-C. Laprie, and D. Powell, “Fault Injection and Dependability Evaluation of Fault-Tolerant Systems,” IEEE Trans. Computers, vol. 42, no. 8, pp. 913-923, Aug. 1993.

[19] T. Scholte et al., “An Empirical Analysis of Input Validation Mechanisms,” Proc. ACM Symp. Applied Computing, pp. 1419-1426, 2012.

[20] J. Carreira, H. Madeira, and J.G. Silva, “Xception: Software Fault Injection and Monitoring in Processor Functional Units,” IEEE Trans. Software Eng., vol. 24, no. 2, Feb. 1998.

[21] N. Tomatis, R. Brega, G. Rivera, and R. Siegwart, “May You Have a Strong (-Typed) Foundation Why Strong Typed Programming Languages Do Matter,” Proc. IEEE Intl Conf. Robotics and Automation, 2004.

[22] J. Christmansson and R. Chillarege, “Generation of an Error Set that Emulates Software Faults,” Proc. IEEE Fault Tolerant Computing Symp., 1996.

[23] H Madeira, M. Vieira, and D. Costa, “On the Emulation of Software Faults by Software Fault Injection,” Proc. IEEE/IFIP Intl Conf. Dependable System and Networks, 2000.

[24] M. Cukier, R. Berthier, S. Panjwani, and S. Tan, “A Statistical Analysis of Attack Data to Separate Attacks,” Proc. Intl Conf. Dependable Systems and Networks, pp. 383-392, 2006.

THANK YOU