

BEYOND PEN-TESTING: AUTOMATED STATIC ANALYSIS

Why Black-box Testing is Not Enough

Security Lab at Armorize Technologies, Inc.
2007/3/1

Beyond Pen-Testing: Automated Static Analysis Why Black-box Testing is Not Enough

Static Code Analysis enables customer to

• Secure the application proactively

The development team is able to find and fix bugs before reaching the customer and hacker.

• Avoid the cost of service downtime

The financial drain of post-incident remediation efforts, lost employee productivity, and lost revenue due to security breach can be effectively prevented.

• Assure corporate reputation

Even a single incident of compromised customer data could put corporate goodwill and image at risk.

A penetration test (PT or Pen-Test) is a method of evaluating the security of a website or computer application by simulating a series of attacks. This assessment is carried out from the perspective of a potential attacker and provides a level of practical assurance that “script kiddies” will not be able to penetrate the system easily. Until recently, PT tools have been adopted to assist IT professionals in collecting, tracking, and reporting the status of the IT infrastructure against known vulnerabilities. However, as most penetration testing activities are performed by outside consultants or a specialized IT team within the organization, both methods require considerable capital resources.

Furthermore, penetration tests address only a portion of a comprehensive security assessment process and are known to produce many false negatives. False negatives (overlooking or missing instances of vulnerability) severely complicate the ability of the management team to accurately identify high-risk vulnerabilities pertinent to business and operations.

To address the problems inherent in penetration testing, enterprises have begun to utilize static code analysis, which is the analysis of computer software performed without having to execute or run the programs built from that software. The analysis performed varies from those that only consider the behavior of individual statements and declarations, to those that include the complete data/control flow of a program in their analysis. The information obtained from the analysis offers a comprehensive assessment report of all code rather than the partial check-up offered by PT.

Recently, static code analysis tools that provides a cost-effective way to look at real vulnerabilities have become available. These commercial applications automate the steps of a proven software development lifecycle and security testing process.

PT customer often complains, “The thing that really used to irritate me is when I had someone come in, scan my network, and then tell me I had a list of medium-level vulnerabilities. I thought, my IT staff are loaded with updates and backups, my dev teams are struggling in between all of the patching and integrations, when are they supposed to divide-and-conquer these vulnerabilities? Then a month later, a security breach due to ineffective remediation efforts costs the company millions.”

The Need For Proof: Static Code Analysis Verifies All Possible States While PT Gives Partial Check-ups

Generally, developers and IT staff are overwhelmed with tasks. They spend their time making sure projects are committed on-time and loaded with valuable services. When it comes to thinking about their systems being vulnerable, they take a "show me" attitude. Penetration testing is capable of providing factual information about real vulnerabilities that may exist in the application and it allows people to see exactly what an attacker can do. It has typically been presented on the premise of behaving "like an attacker," however attackers and security consultants have entirely different objectives regarding the security assessment task; while an attacker needs only to find a single vulnerability, a security consultant must be able to identify all potential routes through which an attacker could exploit the code. The method an attacker uses is exactly what a penetration test provides, it involves hunting for a few known services and then trying to exploit them. Static code analysis, on the other hand, checks for all possible entry-points and program paths, back-tracing every line of code, identifying the exploit potential. Furthermore, a PT does not tell you how to eliminate an attack if a vulnerability has been found, whereas static code analysis pinpoints the lines of code on which the vulnerability lies, and traces the propagation of the tainted variable that introduces the vulnerability right back to its point of origin.

With static code analysis, organizations are able to intelligently manage risk because of the detailed information regarding actual, exploitable security bugs. Using static code analysis is like having a team of security experts reviewing code on a daily or even hourly basis. It can immediately identify which vulnerabilities are critical, which are insignificant, and which have been overlooked.

Typically, the lifecycle of a penetration test lasts around two to four weeks and for this reason, it is generally only conducted once or twice a year. The problem lies in the fact that PT operates by running a series of attack parameters or 'test cases' against the system in order to identify which points are vulnerable. It doesn't allow for immediate identification of vulnerability, in fact a whole host of test cases need to be run against the system to ensure improved accuracy. However, the more test cases that are run, the slower the entire process as more test cases generate more traffic and consume more bandwidth. In addition, when the system being tested is a production system, there is a high probability that the reliability of the system itself may be impacted. Thus another problem arises, because if the network becomes congested, packets (containing test cases) may be discarded leading to the possible generation of additional false negatives. The only solution in this case would be to reduce the volume of traffic being generated, but this requires the implementation of fewer test cases. It is therefore evident that PT suffers from some very serious trade-offs which inevitably leave developers, managers and security professionals in the dark when trying to conduct a serious, intensive security risk assessment.

Armorize's static source code verification technology begins by parsing and transforming the code into an intermediate language. The next step is to determine the data entry points, conduct program path analyses, control flow and data flow analyses which provide an incredibly accurate contextual model of the program and describes the interdependency of all its functions. This enables the ability to calculate all the behavioral aspects of the program and determine all possible outcomes. The program does not need to be running, or even entirely complete to conduct a verification scan, though a more complete code base will necessarily lead to more accurate results as more possibilities will be included in the algorithms. Nonetheless, the true potential lies in the ability of developers and project managers to keep track of code security during the development of the code, immediately verifying whether or not any changes to the code repository and/or production server have inadvertently created new vulnerabilities, and if so how to go about removing them. This automated code review and proactive approach to security frees up significant portions of the developers' time, allowing them to pursue other mission-critical assignments and improving aggregate productivity.

The Need For Efficiency

The flexibility and accuracy of automated static source code analysis improves the efficiency and overall productivity of an organization's software development and QA concerns. The improved, real-time identification of security vulnerability which illustrates the causal linkages between tainted variables and exploitable vulnerability, minimizes not only search costs in terms of isolating problems and identifying their solutions, but by integrating the traditionally separated aspects of development and security assurance, also reduces many of the transaction costs associated with having two separate departments befuddling each other with misinterpretations, incongruent schedules, separated departmental objectives and differing perspectives on whose responsibility it is to develop quality software, assuming, of course, that one regards application security as being an aspect of overall quality.

At face-value penetration testing may appear to be a simple solution to a serious problem, but upon consideration of all the elements of a comprehensive, efficient application security program, one begins to notice that the true value of a penetration still lies at the very back end of a comprehensive security program and is all too often, and all too dangerously, allowed to proxy a proven secure software development lifecycle. When a more comprehensive solution exists, one that eliminates many of the costs and frustrations associated with measures such as penetration testing, and delivers a more accurate, reliable, infinitely repeatable assessment of the security status of an application, one must begin to question the motivation for continuing to conduct penetration tests as the sole approach to securing an application.

The False Sense of Security: PT Incorrectly Positioned As An Automated Substitute For Security Assessment

Whilst there remain many problems involved with conducting a penetration test, it would be unfair to conclude that it does not have a useful place in an enterprise's overall assessment program. The problem with PT is that it has been positioned as an automated substitute for security assessment and not as an effective vulnerability scanner, which is what it is. A point that has regrettably been missed is that PT provides, at best, only a part of the vulnerability assessment process, a single move in the comprehensive stratagem that is application security management.

For those organizations that do not have the resources to conduct and maintain a thorough security assessment policy, penetration testing still has some value in that it provides at least some protection. However, organizations that genuinely wish to constantly assess their exposure and maintain high levels of security, efficiency and productivity, starting with static source code analysis is by far the better option.

	Penetration Testing	Static Code Analysis
No Performance Bottleneck	×	✓
No Side-effect	×	✓
High Precision	×	✓ (traceback support)
High Coverage	×	✓ (walk through every line)
Timeliness	×	✓
	(take weeks, twice a year)	(on-the-fly, scheduled)
Learn-as-you-go	×	✓ (fix suggestion, better coding)



About Armorize Technologies, Inc.

Armorize Technologies (www.armorize.com) is the world-leading provider of source code analysis solution for Web application security. The company's award-winning product, CodeSecure Verifier, precisely identifies Web code vulnerabilities on-the-fly and facilitates remediation at the source. Headquartered in Santa Clara, California, Armorize customers include leaders in telecom, finance, government, and high-tech industries around the world.

Armorize Headquarter: 5201 Great America Parkway, Suite 320, Santa Clara, CA 95054, USA

<http://www.armorize.com>

info@armorize.com



5201 Great America Parkway
Suite 320, Santa Clara, CA 95054

Office: 1-408-216-7893
Fax: 1-408-562-5745

A whitepaper from Armorize Technologies, Inc. <http://www.armorize.com>