Security testing services are based on the principles of confidentiality, integrity, availability, authentication, authorization, and access control. The alarming increase in the number and frequency of significant breach incidents from external and internal attackers has made it critical for organizations to secure all three fundamental access points to their digital data—the network, the hardware, and the software—that support business operations. Organizations spend a large chunk of their revenue on security testing programs that are often plagued by an unplanned approach. NIIT Technologies helps develop an effective, balanced line to security testing, saving time and resources, and protecting from damage to reputation.
Address Vulnerabilities at a Low Cost

Frequent and rapid changes in the application software leave it prone to several security flaws. Many organizations perform security testing after releasing the product for manufacturing. This results in huge costs and resources needed to plug the security flaws. To ensure that vulnerabilities are addressed faster, it is important to introduce security during the Software Development Lifecycle (SDLC). During each phase of the security development program, appropriate security testing activities must be performed to ensure that the software is defect-free before it is released for production.

The cost to fix security vulnerability found in production is 6.5 times higher than that found in the SDLC phase. By introducing security into the software development lifecycle, NIIT Technologies enables organizations to meet customer demands with secure products and services.

Our Solution

The NIIT Technologies solution has a three-pronged approach to ensuring complete client satisfaction:

Education: We educate the development and testing teams about the objective of security testing and common security issues.

Security Requirements: We review projects and specify security requirements based on functionality. We help analyze compliance and best-practice security guidance documents to derive additional requirements.

Secure Architecture: In the security architecture document, we help create a list of recommended software frameworks, services, and other software components, and develop a list of guiding security principles as a checklist against detailed designs.

The NIIT Technologies Security Testing services helps clients:

- Increase assurance and customer confidence
- Reduce risks before releasing in the production system
- Prioritize risk remediation effort
- Comply with industry best practices in security
- Save big on cost and resources

NIIT Technologies Security Testing services offer an effective means to handle your business challenges:
Introducing Security Testing Services in SDLC

By integrating security in the SDLC phase, organizations can ensure that secure development activities are performed as part of the standard development process. Security testing in different SDLC phases ensures that appropriate security testing services are implemented. An outline of relevant services in different phases of the SDLC process is given.

The security testing services program is divided into five stages with different kinds of security assessments performed to save costs and resources while protecting from damage to the brand.

**Define:** To have a successful testing program, organizations must first understand the testing objectives. These objectives are specified by the security requirements. It also discusses how security requirements effectively drive security testing during SDLC and how security test data can be used to effectively manage software security risks. The first step before documenting security requirements is to understand business requirements. A business requirement document can provide initial high-level information on the expected functionality of the application.

**Design:** Manual inspections and review are important activities in SDLC. In this stage, inspection of architectural diagrams and review of the system is carried out. Threat models are created as early as possible in SDLC for risk assessment of applications. This enables designers to develop mitigation strategies for potential vulnerabilities and helps them focus on limited resources and parts of the system that require it.

**Develop:** Secure code review and testing during this phase of development enables organizations to find bugs. It is also called white-box testing as static analysis of the code is performed during this phase. Bugs in the source code are found manually or with automated tools. The developers are dependent on the results of the source code analysis to verify that the developed source code does not include potential vulnerabilities and is compliant with the best practices of secure coding.

**Deploy:** When all the phases of development are completed, the application is deployed on the staging or the testing server. Penetration testing and security testing are performed on the application and the network. The tester acts like an attacker and tries to exploit the software with black box and grey box security testing techniques.

**Maintain:** The aim of security assessment is identification of gaps in security controls such as lack of basic authentication, authorization, or encryption controls. Maintaining security assessment requires controls to measure the effectiveness of the security program. Security test metrics can support security risk, cost, and defect management analysis by reducing overall vulnerabilities by up to 25% and prioritizing and fixing high and medium impact issues within the deadline.
Security Threats Demand Capabilities of a Technology Vendor

**Services Advantage:** Our Security Testing services program reduces the risk level of security flaws and insecure software to around 80% and saves the organization from monetary and brand value loss. According to Forbes.com, the cost of security flaws for an economy is estimated at $180 billion a year, and recovery cost is estimated at $216 million a year. The National Institute of Standards and Technology (NIST) reported that the cost of fixing a bug during testing is estimated at $30,000 whereas the cost of fixing the bug during coding is $5,000.

**Threat Assessment:** Build a threat model based on the documents and information received from the business team for each type of environment. This model helps to identify the criticalities of various threats and their impact on the business during the development process.

**Design Review:** Identify the entry points (attack surface/defense perimeter) in software designs. Once the entry points are identified, analyze software designs against known security risks.

**Code Review:** Create best practices of secure coding standards for the development team to help them know what kind of security mechanisms can be implemented during coding. This helps to reduce the effort of the development team for remediation of bugs by 60%.

**Penetration Testing:** Penetration testing of application and network is one of the important activities during security testing because it is performed before the product is released for production. To secure the environment, best practices of industry-specific standards (OWASP, SANS, OSTMM) of security testing are followed.

**Vulnerability Management:** Create security testing metrics with a baseline to mitigate vulnerabilities. In this process, priority is given to high and medium impact issues.

**Security Testing Deliverables**

We have performed security testing for multiple clients in various domains such as airlines, retail, banking and finance, transportation, and insurance. Following are the security testing deliverables for clients based on their requirement:

<table>
<thead>
<tr>
<th>Testing Types</th>
<th>Deliverables</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Manual inspection and review</td>
<td>• Maturity model, Web security risk assessment</td>
</tr>
<tr>
<td>• Code review</td>
<td>• Secure code review report</td>
</tr>
<tr>
<td>• Scanning</td>
<td>• Web application scanning report, network scanning report</td>
</tr>
<tr>
<td>• Penetration testing</td>
<td>• Artefact report, penetration testing report</td>
</tr>
</tbody>
</table>

**Security Testing Technique** We follow industry-specific Open Web Application Security Project (OWASP), Open Source Security Testing Methodology Manual (OSSTMM), SANS, and Web Application Security Consortium (WASC) security testing standards. Our vulnerability assessment service is designed to identify security flaws in an organization’s external and internal environment that an attacker can exploit. It also identifies vulnerabilities ranked by risks.
Our Approach to Security Testing

**Planning Phase**
- Scope and strategy of the assignment is determined
- Existing security policies, standards are used for defining the scope

**Discovery Phase**
- Collect as much information as possible about the system including data, user names, and even passwords
- Scan and probe into the ports

**Attack Phase**
- Check for vulnerabilities and follow the industry security testing standards—OWASP, SANS, WASC
- VAPT techniques involved in this phase
- Various exploits can be created

**Reporting Phase**
- Report must contain detailed findings
- Record risks of vulnerabilities found and their impact on business
- Prepare artefact document for all the known vulnerabilities
- List recommendations and solutions
Our Risk Metric Model

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Status</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify all pages and resources that require authentication except those specifically intended to be public.</td>
<td>Done</td>
<td>H</td>
</tr>
<tr>
<td>Verify enforcement of all authentication controls on the server side.</td>
<td>Done</td>
<td>M</td>
</tr>
<tr>
<td>Session Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify that the framework's default session management control implementation is used by the application.</td>
<td>Not Done</td>
<td>H</td>
</tr>
<tr>
<td>Verify that sessions are invalidated when the user logs out.</td>
<td>Done</td>
<td>L</td>
</tr>
<tr>
<td>Access Control</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify that users only access secured functions or services for which they possess specific authorization.</td>
<td>Not Done</td>
<td>H</td>
</tr>
<tr>
<td>Verify that users can only access secured URLs for which they possess specific authorization.</td>
<td>Not Done</td>
<td>H</td>
</tr>
<tr>
<td>Input Validation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify that the runtime environment is not susceptible to buffer overflows, or that security controls prevent buffer overflows.</td>
<td>Not Done</td>
<td>H</td>
</tr>
<tr>
<td>Verify that all input validation failures result in input rejection.</td>
<td>Done</td>
<td>M</td>
</tr>
<tr>
<td>Cryptography at Rest Verification Requirement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify that all cryptographic functions used to protect secrets from the application user are implemented on the server side.</td>
<td>Not Done</td>
<td>H</td>
</tr>
<tr>
<td>Verify that all cryptographic modules fail securely.</td>
<td>Not Done</td>
<td>H</td>
</tr>
<tr>
<td>Error Handling and Logging Verification Requirement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify that the application does not send error messages or stack traces containing sensitive data that could assist an attacker, including session id.</td>
<td>Not Done</td>
<td>M</td>
</tr>
<tr>
<td>Verify that all error handling is performed on trusted devices.</td>
<td>Not Done</td>
<td>M</td>
</tr>
<tr>
<td>Data Protection Verification Requirement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify that all forms containing sensitive information have disabled client side caching, including auto complete features.</td>
<td>Not Done</td>
<td>H</td>
</tr>
<tr>
<td>Verify that the list of sensitive data processed by this application is identified, and that there is an explicit policy on how to access this data.</td>
<td>Not Done</td>
<td>H</td>
</tr>
</tbody>
</table>

Comparison Graph
- H: Done 92
- M: 30
- L: 27
- Total Volume: 155
- Completion Status: 51%

Success Story: Multi-Channel Security Testing Application Security for a Client with Presence in 1000 cities

The client, with a global network and presence in 1,000 cities, is the only franchise network in the industry to do more than 1.5 million chauffeur-driven journeys every year. The client collects, stores, and transmits customer data internally utilizing heavy encryption and top-tier equipment.

Business Scenario

The client wanted security of both the Web and mobile applications with built-in security controls before the rollout. The current system presented the following challenges:

- Customer information was vulnerable to data breaches and attacks; this led to information leakage and loss of sensitive data
- Unsafe and insecure ticket booking mechanism
- Complex multi-city and multi-platform operation of the application
- Difficulty in handling third-party integration with the application

Our Solution

NIIT Technologies’ execution approach was divided into four phases:

1. Planning
   - Scope analysis of the security testing requirement performed
   - Rules of engagement, test plans, and written permission developed and signed
   - The standard time set for security testing does not affect the client production server
   - Timeline for each security testing activity set
   - Demo of the application carried out to understand its behavior and what it does
   - Test data of the application generated for security testing

2. Discovery
   - Vulnerability analysis of services and applications was carried out
   - Operating systems of scanned hosts were compared against vulnerability databases

3. Attack
   - Vulnerability scan of the Web application was performed with the security testing tool ‘Acunetix’
- Manual penetration testing of the Web application was performed using open source and free edition tools such as Burp Suite, Dirbuster, OpenSSL. It covered at least the top 10 vulnerabilities of the OWASP in all iterations.
- Mobile security testing was performed on an Android simulator, and devices with various open source tools such as Android SDK, dex2jar, agnitio, apktool.
- Static and dynamic analysis of mobile application was performed.

4. Reporting
- A separate report after performing automated and manual security testing was created; it was customized for the management and the developer.
- A step-by-step security testing report mentioning vulnerabilities was published; it included risk impact, risk severity, and recommended solutions for the issues.
- The final published report ensured that all the vulnerabilities had been mitigated by the development team.

Value Delivered
- Increased assurance and client confidence
- Saved time and money by prioritizing efforts on mitigation
- Provided proactive strategy for security testing
- Ensured strict compliance with industry best practices

The NIIT Technologies Advantage

Our security testing services not only test Web applications and software, but also deliver effective and unique services. Our proven testing approach with open source tools for manual penetration of Web applications is very effective and enhanced. It also covers industry-specific OWASP, SANS, WASC vulnerabilities without using any commercial tools.

For a balanced approach, our security testing maturity model quantifies best practices of security used by applications. The uniqueness of this model has been implemented on new and existing projects of the organization. It prioritizes the organization effort for security vulnerabilities because it shows the security risk exposure in terms of authentication, session management, authorization, and access control. Reduced costs, manpower, and faster turnaround time are some of the benefits to our clients.