25 SEPTEMBER 2017

SECURE CODE REVIEW REPORT
Version 1.4

IniTech, Inc.
Bill Lumbergh | Chief Technology Officer

PAGE INTENTIONALLY LEFT BLANK

ASSESSMENT INFORMATION

Security Consultant

Dwight Hohnstein
dwight.hohnstein@rhinosecuritylabs.com
(888) 944-8679

Assessment Prepared for

Bill Lumbergh
Chief Technology Officer
IniTech, Ltd.
(888) 584-2481

Security Engagement Manager

Christopher Lakin
chris.lakin@rhinosecuritylabs.com
(888) 944-8679

Project Number

09-25-ITL-SE

Assessment Scope Summary

Engagement Timeframe

05/02/2017 – 05/21/2017

Engagement Scope

12,000 Lines of Code

50 Files

Revision History

05-21-2017	Dwight Hohnstein	First Draft
05-22-2017	Christopher Lakin	Edits
05-22-2017	Benjamin Caudill	Edits
05-23-2017	Dwight Hohnstein	Final Draft

TABLE OF CONTENTS

ENGAGEMENT OVERVIEW

Rhino Security Labs specializes in Secure Code Review Assessment techniques and practices, identifying areas for improvement. At Rhino Security Labs we specialize in manual assessments that go beyond basic automated tests to identify real attack vectors that can be used against your application.

With decades of combined experience, Rhino Security Labs is at the forefront of application security and penetration testing. With a veteran team of subject matter experts, you can be sure every resource is an authority in their field.

Service Description

Secure Code Review Assessment

With their growing complexity and demand, applications and APIs have become the target of choice for hackers. Rhino Security Labs' Secure Code Review Assessments help protect your applications from a range of security threats.

Application security issues are not only the most common type of vulnerability, they're also growing in complexity. While the OWASP Top 10 is used as the standard for identifying application security flaws, that's just a start – many advanced vulnerabilities are not included in that list. Automated code review scanners focused on OWASP will fall behind new threats, leaving the application exposed to unknown risks.

At Rhino Security Labs, we go far beyond the OWASP Top 10 in static code analysis, continually pushing the boundaries of application security and detailing the way unique architectures can be abused.

Campaign Objectives

Vulnerability Identification

Rhino Security's consultants use the results of automated application scans, paired with their expert knowledge and experience, to conduct a manual security analysis of the client's applications. Our assessors then identify vulnerabilities in the code and provide thoughtful suggestions for remediation. The detailed results of both the vulnerability scan and the manual testing are shown in this report.

KEY PERSONNEL

Passionate and forward-thinking, our consultants bring decades of combined technical experience as top-tier researchers, penetration testers, application security experts, and more. Drawing from security experience in the US military, leading technology firms, defense contractors, and Fortune 100, we pride ourselves on both depth and breadth of information security experience.

Chris Lakin - *Cybersecurity Engagement Manager*

Chris Lakin has accumulated over eight years of project management and customer engagement experience across a multitude of industries. A proponent of constant iteration and improvement, his knowledge from time spent in business and marketing adds a valuable perspective to every cybersecurity engagement. Receiving a Masters of Science in Cybersecurity Engineering from the University of Washington, Mr. Lakin excels at connecting the technical with the goals of the business.

Dwight Hohnstein - *Security Consultant*

Dwight Hohnstein started his professional career as a web developer, but quickly found a penchant and passion for application security. His research involves enterprise level applications, firmware modification and reversing, and creating exploits for capture the flag competitions. For examples of his research, you can find the two-part blog post that revealed six high level vulnerabilities in Unitrends Enterprise Backup or Pentesting in AWS Environments on the Rhino Security research section of the website.

Benjamin Caudill - *CEO and Founder*

Benjamin Caudill is an adept cybersecurity professional, researcher, and entrepreneur. A veteran of the defense and aerospace industry, Mr. Caudill led investigations into advanced cyberattacks, coordinating with federal intelligence communities on complex engagements. As Founder and CEO of Rhino Security Labs, Mr. Caudill has built the boutique security firm and turned it into a major player in the penetration testing market. In addition to his executive role, Mr. Caudill oversees company research and development, ensuring the continued development of key offensive technologies.

SECURE CODE REVIEW ASSESSMENT METHODOLOGY

At Rhino Security Labs, our Secure code review report targets the entire range of vulnerabilities in your Secure code review. Using the same techniques as sophisticated real-world attackers, we providing unique visibility into security risks automated tools often miss. To ensure high quality, repeatable engagements, our Secure code review report methodology follows these steps:

1

Code Enumeration

This process begins with detailed scanning and research into the architecture of the application, with the assistance of automated testing for known vulnerabilities. Collecting, parsing, and correlating information on the language, codebase, and dependencies is key to identifying these risks.

2

Vulnerability Scanning

After the technology stack and dependencies have been fully identified, automated scanning of the codebase begins. Rhino Security Labs first uses a range of both commercial scanning products and proprietary tools to identify security flaws. With decades of experience integrated into these custom scripts, subtle flaws are identified many automated scanners overlook.

3

Manual Source Code Analysis

At this stage of the assessment, our consultants review all previous data to manually analyze key areas of the application. As attack vectors in these sensitive points in the code are discovered, focus turns to identifying technical risk and business impact. During each phase of the examination, we keep client stakeholders informed of critical risks that we find.

4

Assessment Reporting

Once the engagement is complete, Rhino Security Labs delivers a detailed code analysis and executive summary, including remediation steps. Our consultants set an industry standard for clear and concise reports, prioritizing the high The assessment includes the following:

- Executive Summary
- Strategic Strengths and Weaknesses
- Identified Vulnerabilities and Risk Ratings
- Affected lines of code for each security risk
- Detailed Risk Remediation Steps

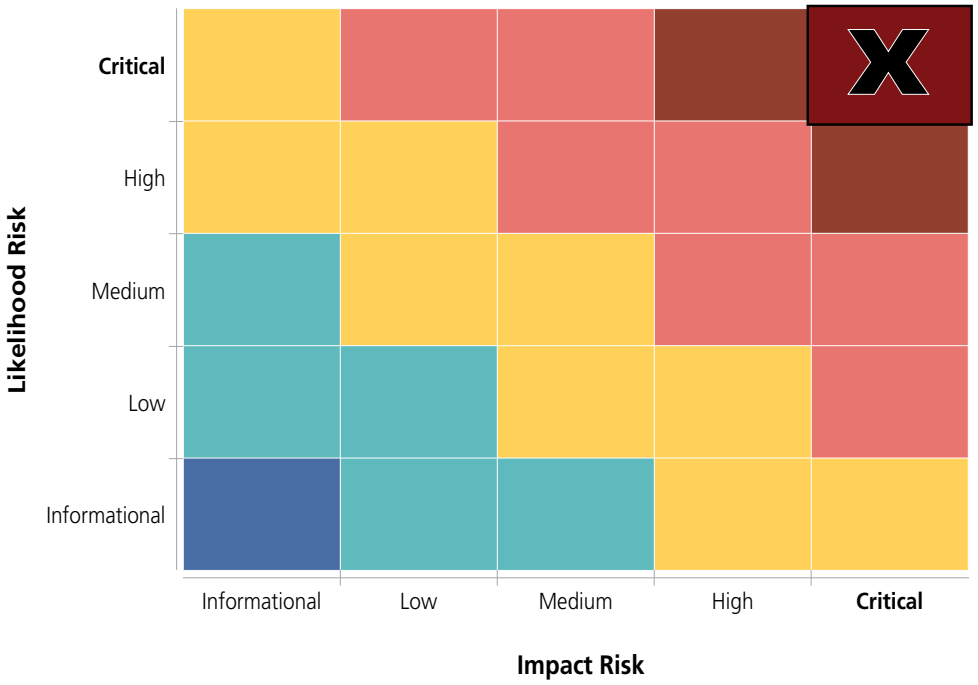
EXECUTIVE SUMMARY

Rhino Security Labs conducted a secure code assessment for IniTech, Inc. This test was performed to assess IniTech's defensive posture and provide security assistance through proactively identifying vulnerabilities, validating their severity, and providing remediation steps. Rhino Security Labs reviewed the security of IniTech's architecture and had determined a Critical risk of compromise from external attackers, as shown by the presence of the vulnerabilities detailed in this report. The detailed findings and remediation recommendations for these assessments may be found later in the report.

SECURE CODE REVIEW RISK RATING

Rhino Security Labs calculates Secure Code Review risk based on Exploitation Likelihood (Ease of exploit) and Potential Impact (Technical Controls).

Overall Risk Rating: CRITICAL



Summary of Strengths

Rhino Security Labs acknowledged a number of security and technical controls that blocked attempts to carry out malicious actions. Understanding the strengths of the application can reinforce security best practices and provide strategy and direction toward a robust defensive posture. The following traits were identified as strengths in the IniTech application.

- Client implements proper Two-Factor Authentication for login forms
- Application is using basic encryption to help protect user data storage and in transit

Summary of Weaknesses

Rhino Security Labs discovered and investigated many vulnerabilities during its assessments of IniTech. We have categorized these vulnerabilities into general weaknesses across the application, and provide direction toward remediation for a more secure code.

- Poor sanitation of data from majority of user fields in application
- Input fields are not protected against a variety of DDoS and brute forcing attacks
- Password use a cryptographically weak hashing algorithm and insecure storage practices
- Session tokens and cookies are poorly configured and insecurely managed

Strategic Recommendations

Not all security weaknesses are technical in nature, nor can they all be remediated by security personnel. Companies often focus on the root security issues and resolve them at their core. These strategic steps are changes to the operational policy of the organization. Rhino Security Labs recommends the following strategic steps for improving the company's security.

1. Ensure proper development and repository practices by in-house and 3rd party developers
2. Remove all sensitive pages and directories from publicly accessible servers; Require authentication on such pages as necessary
3. Remove legacy servers, subdomains, pages, and other web resources when no longer in use
4. Sanitize all user inputs before processing on webserver or other internal resources
5. Enhance security defenses with additional detection and response capabilities, such as a SIEM

SCOPING AND RULES OF ENGAGEMENT

While real attackers have no limits on Secure code review report engagements, we do not engage in penetration testing activities that threaten our ethics and personal privacy.

Constraints

In addition, the following limitations were put into place:

- Penetration testing was limited to the agreed upon engagement period, scope, and other additional boundaries set in the contract and service agreement.

Scope of Service

The predetermined scope for Rhino Security Labs to carry out the Secure code review report was:

IniTech Application

Lines of Code
12,000

Files Reviewed
50

Repo Branch
Master

Repo Pull Date
9/26/2017

Languages
Go, Python, ReactJS

Associated Architecture
PostgreSQL Database (containing credit cards)

SUMMARY VULNERABILITY OVERVIEW

Rhino Security Labs performed a Source Code Review Assessment for IniTech, Ltd. (IniTech) on 09-01-2017 – 09-25-2017. This assessment utilized both commercial and proprietary tools for the initial code enumeration and scanning, as well as custom tools and scripts for unique vulnerabilities.

During the manual analysis, assessors analyzed the code for vulnerabilities and determined the technical risk and business impact for each, including those listed in the OWASP Top 10 Vulnerabilities list. The following vulnerabilities were determined to be of highest risk, based on several factors including asset criticality, threat likelihood, and vulnerability severity.

Vulnerability Risk Definition and Criteria

The risk ratings assigned to each vulnerability are determined by averaging several aspects of the exploit and the environment, including reputation, difficulty, and criticality.

CRITICAL Vulnerabilities pose a very high threat to a company's data, and should be fixed on a top-priority basis. They can allow a hacker to completely compromise the application or cause other serious impacts to security.

HIGH Severity vulnerabilities should be considered a top priority regarding mitigation. These are some of the most severe issues and generally cause an immediate security concern to the enterprise.

MEDIUM Severity vulnerabilities are a lower priority, but should still be remediated promptly. These are moderate exploits that have less of an impact on the application.

LOW Severity vulnerabilities are real but trivially impactful to the environment. These should only be remediated after the HIGH and MEDIUM vulnerabilities are resolved.

INFORMATIONAL Vulnerabilities have no impact as such to the application by themselves; however, they might provide an attacker with information to exploit other vulnerabilities.

VULNERABILITIES BY RISK RATING

The following vulnerabilities were found within each risk level. It is important to know that the total number of vulnerabilities is not a factor in determining risk level. Risk level depends upon the severity of the vulnerabilities found.

1	2	2	2	1
Critical	**High**	**Medium**	**Low**	**Informational**

Total Vulnerabilities Discovered: 8

C1 - Outdated "OAuthHandler" Python Library

Risk: CRITICAL

Remediation
Update the OAuthHandler Python dependency to the most recent version

H1 - Unsecured Administrative Page Found

Risk: HIGH

Remediation
Remove the associated Admin page or enforce strong authentication controls

H2 - Sensitive GIT Configuration File Leakage

Risk: HIGH

Remediation
Remove the .git directory from being publicly accessible

M1 - Hidden Web Functions Identified

Risk: MEDIUM

Remediation
Remove sensitive files and directories from public servers

M2 - Weak Cryptography Used to Hash Passwords

Risk: MEDIUM

Remediation
Move password hash algorithm from MD5 to bcrypt

L1 - Cookie Scoped to Parent Domain

Risk: LOW

Remediation
Remove the explicit domain attribute from your Set-cookie directive

I1 - Cross Domain Script Include

Risk: LOW

Remediation
Ensure no scripts included in the site are loaded from untrusted domains

I2 - HTML Does Not Specify Charset

Risk: INFORMATIONAL

Remediation
To reduce additional attack surface, include a recognized character set

VULNERABILITY FINDINGS

The vulnerabilities below were identified and verified by Rhino Security Labs during the process of this Source Code Review Assessment for IniTech. Retesting should be planned to follow the remediation of these vulnerabilities.

C1 Outdated "OAuthHandler" Python Library

Risk Rating: CRITICAL

Exploitation Likelihood: **CRITICAL**
Potential Impact: **CRITICAL**

Description

The outdated "OAuthHandler" library contains a known Remote Control Execution (RCE) vulnerability, which can be used to compromise any web application that utilizes it for authentication. Exploitation of this vulnerability would allow an attacker to modify the web application logic, host malicious pages to users, and compromise the associated database.

The associated vulnerability in this library is in versions 0.2 - 1.6, where 1.8 is the current version (as of the date of this report). This application uses version 1.2.

File Name

`/app/oauthhandler/*`

Remediation

Remediation of this flaw is simple - update the vulnerable dependency to the latest version. Ensuring a proper process for testing and updating libraries such as this will remove these issues in the future.

H1 Insecure Administrative Page Found

Risk Rating: HIGH

Exploitation Likelihood: **HIGH**
Potential Impact: **HIGH**

Description

A page for developers to execute testing code was left insecure on the server. While the path and naming convention would make it difficult to find by directory brute forcing, a successful guess (or leak of source code) by attackers could result in a serious breach.

File Name

/admin/testdevexecute

Remediation

Removing the insecure admin page (and any similar developer admin pages from the application) would remove this vulnerability. Although not recommended as a long term solution, this could also be corrected by implementing git filtering for what is pushed to production, removing test features such as these from the live application.

H2 Sensitive GIT Configuration File Leakage

Risk Rating: HIGH

Exploitation Likelihood: **HIGH**
Potential Impact: **HIGH**

Description

The ".git" directory was found on the web-server which allowed public access to the configuration file. This file was found to contain cleartext credentials as well as the IP address of a git repository.

File Name and Code Snippet

/app/.git/config.git

```
415    http://admin:RepoPassword123@23.41.44.103/repository
```

Remediation

Remove the .git directory from being publicly accessible.

M1 Hidden Web Functions Identified

Risk Rating: MEDIUM

Exploitation Likelihood: **MEDIUM**
Potential Impact: **MEDIUM**

Description

There is a hidden web function to get the userID of any IP address. It's not particularly sensitive; however, unintended forms such as these add additional attack surface for attackers to target.

File Name and Code Snippet

/app/files/admin.py

```
18  if len(sys.argv) != 2:
19      print usage
20      sys.exit(0)
21  longurl = sys.argv[1]
22  response = b.shorten(longUrl=longurl)
23  print response['url']
```

Remediation

Removing the hidden web function (and any similar developer pages from the app) would remove this risk. Although not recommended as a long term solution, this could also be corrected by implementing git filtering for what is pushed to production, removing test features such as these from the live application.

M2 Weak Cryptography Used to Hash Passwords

Risk Rating: MEDIUM

Exploitation Likelihood: **LOW**
Potential Impact: **CRITICAL**

Description

Application passwords are hashed using MD5, which is considered weak and easily cracked with modern password cracking hardware. If the password database was compromised, the hashing algorithm provides a final line of defense against cleartext recovery of sensitive credentials. Insufficient hashing protections could enable a successful attacker to spread quickly to applications and servers they otherwise would be unable to compromise.

File Name and Code Snippet

/app/auth.py

```
4   from Crypto.Hash import MD5

28  h = MD5.new()
29  h.update(bytearray("any of your string"))
30  print h.hexdigest()
```

Remediation

Use bcrypt for all password hashing, which is more resilient to password cracking.

L1 Cookie Scoped to Parent Domain

Risk Rating: LOW

Exploitation Likelihood: **LOW**
Potential Impact: **LOW**

Description

A cookie's domain attribute determines which domains can access the cookie. If the cookie contains sensitive data (such as a session token) then this data may be accessible by less trusted or less secure applications residing on other subdomains, leading to a security compromise.

File Name and Code Snippet

/app/auth.py
/app/cookieissued.py

```
910  rootPath = '/'
911  pattern = '*'
912  for root, dirs, files in os.walk(rootPath):
913      for filename in fnmatch.filter(files, pattern):
914          print( os.path.join(root, filename))
```

Remediation

If possible, remove the explicit domain attribute from your Set-cookie directive. This will give cookies the scope of the issuing domain and all subdomains. If you'd like the cookie available to the parent domain, thoroughly consider the security implications of having a larger cookie scope.

I1 Cross-Domain Script Included

Risk Rating: INFORMATIONAL

Exploitation Likelihood: **INFORMATIONAL**
Potential Impact: **INFORMATIONAL**

Description

When scripts are loaded from another domain, they have the permissions and access of the current domain, such as access to cookies. If malicious scripts are loaded from an untrusted domain, they can compromise security of both the site and users.

File Name and Code Snippet

/app/domainloads.py

```
910  rootPath = '/'
911  pattern = '*'
912  for root, dirs, files in os.walk(rootPath):
913      for filename in fnmatch.filter(files, pattern):
914          print( os.path.join(root, filename))
```

Remediation

Ensure no scripts included in the site are loaded from untrusted domains.

I2 HTML Does Not Specify Charset

Risk Rating: INFORMATIONAL

Exploitation Likelihood: **INFORMATIONAL**
Potential Impact: **INFORMATIONAL**

Description

If a web response states that it contains HTML content but does not specify a character set, then the browser may analyze the HTML and attempt to determine which character set it appears to be using. Even if the majority of the HTML actually employs a standard character set such as UTF-8, the presence of non-standard characters anywhere in the response may cause the browser to interpret the content using a different character set. This can have unexpected results, and can lead to cross-site scripting vulnerabilities in which non-standard encodings like UTF-7 can be used to bypass the application's defensive filters. In most cases, the absence of a charset directive does not constitute a security flaw, particularly if the response contains static content

File Name and Code Snippet

/app/domainloads.py

```
910  rootPath = '/'
911  pattern = '*'
912  for root, dirs, files in os.walk(rootPath):
913      for filename in fnmatch.filter(files, pattern):
914          print( os.path.join(root, filename))
```

Remediation

To reduce attack surface in the Source Code Review, the application should include a directive specifying a standard recognized character set, such as charset=ISO-8859-1.

STRATEGIC ACTIONS AND RECOMMENDED NEXT STEPS

Regular maintenance and preventative care for the overall best security practices help to avoid catastrophic incidents from occurring. While we can never guarantee 100% security safety, we can recommend general strategic actions and next steps.

Based upon this engagement, we would like to recommend IniTech takes the following strategic actions:

1. Ensure proper development and repository practices by in-house and 3rd party developers
2. Remove all sensitive pages and directories from publicly accessible servers; Require authentication on such pages as necessary
3. Remove legacy servers, subdomains, pages, and other web resources when no longer in use
4. Sanitize all user inputs before processing on webserver or other internal resources
5. Enhance security defenses with additional detection and response capabilities, such as a SIEM

Next Steps

As IniTech is probably well aware, security is never over. As a follow up to this engagement we recommend:

Remediation Testing

Complete in 1 - 3 months - Approx. 10-01-2017

Once the vulnerabilities outlined in this engagement are patched and mitigated, Rhino Security labs is able to retest the application to verify that the vulnerability no longer exists.

Ongoing Web Vulnerability Scan

Complete in 6 months - Approx. 12-01-2017

Full penetration tests are not always needed. Web vulnerability scans act as a supplement In between full annual penetration tests. We recommend conducting at least one web vulnerability scan six months after a penetration test to ensure no new vulnerabilities have been introduced. If significant changes to the application have occurred, we will recommend moving forward with a comprehensive penetration test.

Annual Secure Code Review

Complete in 12 months - Approx. 07-01-2018

We recommend conducting a full Secure code review report every year regardless of changes made to the application. As new vulnerabilities are discovered, applications that were once considered secure could develop new vulnerabilities.

APPENDIX A: TOOLS & SCRIPTS

The software and tools used for security analysis are constantly evolving and changing. To stay at the forefront of industry trends, Rhino Security Labs regularly updates and integrates new tools into its secure code review methodology. Below is the toolset our consultants use during a secure code review assessment.

IBM Appscan

IBM Appscan is an industry-leading tool designed for in-depth security testing of web and mobile software applications. It allows for both dynamic and static analysis of application code across the entire stack, from data-layer procedures up to front-end scripts.

OWASP Dependency Check

Developers often include third-party libraries in their applications which can be susceptible to publicly-disclosed vulnerabilities. OWASP Dependency Check is a utility that identifies project dependencies and checks if there are any known vulnerabilities.

RIPS

RIPS is a popular tool that tokenizes and parses PHP source code files into a program model to automatically detect vulnerable functions that malicious users can exploit. RIPS also offers an integrated code audit framework.

CPPCheck

CPPCheck is a tool for the static analysis of C and C++ code. It checks for a large number of vulnerabilities commonly found in these languages, and aims to provide accurate reporting with no false-positives.

Burpsuite

Burpsuite is security platform created specifically for the purposes of intensive web application testing. Its capabilities cover the entire vulnerability assessment process, from mapping and analysis of an application to the exploitation identified vulnerabilities.

Additional Custom Scripts and Application

In addition to the above tools, Rhino Security Labs also makes use of its own proprietary tools and scripts to quickly adapt to new and unique environments.

APPENDIX B: OWASP TOP 10 APPLICATION SECURITY RISKS

The Open Web Application Security Project surveys security professionals across the globe collecting information on the most frequently found security vulnerabilities within Secure code reviews. The results are compiled, categorized, and ranked to determine the top-ten. Rhino Security Labs uses both automated and manual processes to identify if these vulnerabilities are present within your application. The appendix is to help explain how some of your vulnerabilities might be categorized.

A1-Injection	Injection flaws, such as SQL, OS, XXE, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
A2-Broken Authentication and Session Management	Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities (temporarily or permanently)
A3-Cross-Site Scripting (XSS)	XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user supplied data using a browser API that can create JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A4-Broken Access Control	Restrictions on what authenticated users are allowed to do are not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.
A5-Security Misconfiguration	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, platform, etc. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
A6-Sensitive Data Exposure	Many Secure code reviews and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

A7-Insufficient Attack Protection

The majority of applications and APIs lack the basic ability to detect, prevent, and respond to both manual and automated attacks. Attack protection goes far beyond basic input validation and involves automatically detecting, logging, responding, and even blocking exploit attempts. Application owners also need to be able to deploy patches quickly to protect against attacks.

A8-Cross-Site Request Forgery (CSRF)

A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. Such an attack allows the attacker to force a victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.

A9-Using Components with Known Vulnerabilities

Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

A10-Underprotected APIs

Modern applications often involve rich client applications and APIs, such as JavaScript in the browser and mobile apps, that connect to an API of some kind (SOAP/XML, REST/JSON, RPC, GWT, etc.). These APIs are often unprotected and contain numerous vulnerabilities.

888.944.8679
info@rhinosecuritylabs.com
1200 East Pike Street Suite 510 | Seattle, WA 98122