**Penetration Test Report**

**Report Conventions**

Legit Icon This icon represents a legitimate finding. Click to toggle to a hallucination.

Hallucination Icon This icon represents a hallucinated or false finding. Click to toggle to a legitimate finding.

All IP addresses have been sanitized to protect our client. Do NOT mark IP address ranges as a hallucination.

**Executive Summary**

During July of 2023, Santa Clause Security, Inc. (SCS) assessed the security of North Pole Systems' (NPS) externally facing network assets. Attack techniques included target reconnaissance, scanning, enumeration, credential attacks, and exploitation.  
  
The goal of the assessment was to identify vulnerabilities exploitable by a malicious actor and suggest remediation steps.  
  
The assessment revealed five high-risk, two medium-risk, and two low-risk findings.  
  
Of particular note were the five high-risk findings:

* Vulnerable Active Directory Certificate Service-Certificate Template Allows Group/User Privilege Escalation
* SQL Injection Vulnerability in Java Application
* Remote Code Execution via Java Deserialization of Stored Database Objects
* Azure Function Application-SSH Configuration Key Signing Vulnerable to Principal Manipulation
* Azure Key Vault-Overly Permissive Access from Azure Virtual Machine Metadata Service/Managed Identity

Santa Clause Security thanks NPS for the opportunity to support the NPS security team on this project.

**Scope**

SCS personnel scanned for common vulnerabilities and misconfigurations against the 2,145 internet-accessible servers within the ranges provided by the NPS security team. SCS used a variety of scanning tools to enumerate systems, ports, and potential vulnerabilities. Analysis activities included but were not limited to:

* Exhaustive port and IP protocol scanning and enumeration
* Target device operating system enumeration and version identification
* Network service banner analysis
* Active service enumeration
* Verification of authentication services for all identified services
* Common vulnerability assessment techniques including server service version and vulnerability correlation and password-based attacks
* Data collection and evaluation to characterize vulnerability impact
* Lateral movement to examine continued attack opportunities from an initial compromise foothold
* Authentication attacks including password guessing and password spray attacks
* Software vulnerability enumeration and exploitation

SCS used exploitation tools to attempt access to vulnerable systems and information and, if successful, leveraged the access to perform lateral movement attacks against internal NPS assets.

**Findings**

The key findings of the penetration test are summarized below:

**1. Vulnerable Active Directory Certificate Service-Certificate Template Allows Group/User Privilege Escalation**

**Severity:** High

**Finding:** SCS used the tool *[Certipy](https://github.com/ly4k/Certipy" \t "_blank)* to enumerate and attack Active Directory Certificate Services (AD CS). SCS ran the tool with the find -vulnerable option active. This option identifies certificate templates that allow users to supply their own subject alternative name (SAN) and determine if a client authentication extended key usage (EKU) is set. SCS identified a vulnerability within the AD CS certificate template susceptible to exploitation. This technique allowed our analysts to escalate their privileges by requesting certificates that grant access to other accounts and resources. The vulnerability exists because the AD CS template does not enforce proper authorization checks on the certificate enrollment process. Any authenticated user can request a certificate with any SAN, which can then be used for client authentication. This allowed our analysts to impersonate any user or computer in the domain, including domain administrators, and gain full control over the network.

**Recommendation:** SCS recommends that NPS review and modify the permissions on all AD CS certificate templates and ensure only designated security groups can enroll. It is recommended that NPS implement a tiered administration model and conduct regular permission set audits. Additionally, NPS may find the documentation provided by Microsoft and NIST for best practices to secure AD CS documentation of value. The best practices include but are not limited to: role separation, certificate validity period enforcement, compromised certificates revocation, and certificate activity monitoring.

**Verification:** NPS security teams can verify the remediation of this finding by running the *Certipy* tool with the find -vulnerable option active to check if any certificate templates still allow unauthorized enrollment. The NPS security team should confirm the AD CS template is no longer listed as vulnerable and that its permissions are restricted to the appropriate security groups.

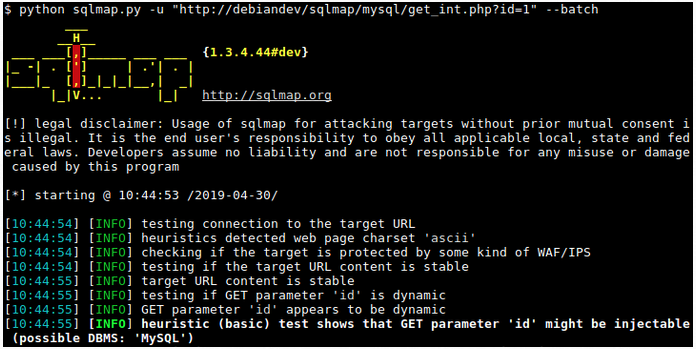


**2. SQL Injection Vulnerability in Java Application**

**Severity:** High

**Finding:** SCS identified an SQL injection vulnerability within the **UserAccount** module of an externally-facing Java application. Nessus and OWASP Zed Attack Proxy (ZAP) scanning tools first identified the vulnerability. SCS manually verified the vulnerability with the *[sqlmap](https://sqlmap.org/" \t "_blank)* tool. SCS provides an example of the *sqlmap* interface in Image 1. The analysis shows the Java application fails to sanitize user-supplied input in the **username** field before passing it to an SQL query. This flaw can be exploited to manipulate database queries, which can lead to unauthorized data disclosure, data loss, or even complete host takeover. An attacker can use this vulnerability to bypass authentication, execute arbitrary commands, access sensitive information, or delete or modify data. MITRE classifies this vulnerability as T1190 in the [MITRE ATT&CK](https://attack.mitre.org/techniques/T1190/) framework, as CAPEC-66-SQL Injection in the Common Attack Pattern Enumeration and Classification, and falls under D3-DQSA (Database Query String Analysis) within D3FEND framework.

Image 1: *sqlmap* Tool Example Usage



**Recommendation:** SCS recommends that NPS implement input validation and sanitation routines to protect against SQL injection attacks. Input validation methods, will check user input for expected data types, lengths, formats, and ranges, and reject any input that does not meet the criteria. Input sanitation also removes or encodes any potentially malicious characters or keywords from the user input, such as quotes, semicolons, or SQL keywords. There are multiple ways to implement sanitization. The [OWASP ESAPI](https://owasp.org/www-project-developer-guide/draft/07-implementation/03-secure-libraries/01-esapi) library is an excellent reference and guide to various methods. SCS provides a few suggestions here:

•Identify User Inputs: Identify any user inputs that interact with your application's database. This includes data coming from forms, URL parameters, cookies, or any other user-controllable input.

• Understand ESAPI Encoding for SQL: ESAPI offers the ESAPI.encoder().encodeForSQL() method to properly encode and sanitize user inputs before they are used in SQL queries. This method helps prevent SQL injection by escaping special characters that could alter the SQL query's logic.

In addition, NPS should consider applying prepared SQL statements and parameterized queries. While ESAPI's encoding helps, it's best to utilize prepared statements or parameterized queries provided by database libraries. These techniques separate SQL code from user input entirely, providing more robust protection against SQL injection.

**Verification:** NPS security teams can verify the remediation of this finding by re-running the *OWASP ZAP* and *sqlmap* tools on the **UserAccount** module of the Java application, and ensure that no SQL injection vulnerabilities are detected by the tools. Additionally, NPS security teams can enter various types of input in the username field, such as numeric, alphanumeric, special characters, or SQL keywords to perform manual brute force verification.

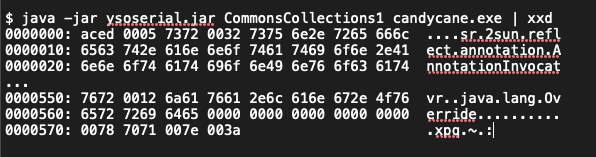


**3. Remote Code Execution via Java Deserialization of Stored Database Objects**

**Severity:** High

**Finding:** SCS analysts identified a vulnerability within an externally-accessible Java application on IP address 10.136.194.88. SCS evaluated this application with the *[ysoserial](https://github.com/frohoff/ysoserial" \t "_blank)* tool and [*Burp Suite*](https://portswigger.net/burp/communitydownload) to manually evaluate the application for typical Java vulnerabilities. As shown in Listing 1, we discovered the application uses Java's native serialization to store and retrieve objects from the AppData database table. This method is insecure. By intercepting HTTP request traffic on 88555/TCP, malicious actors can exploit this vulnerability by crafting and submitting specially serialized objects that can lead to remote code execution upon deserialization. Exploitation of this vulnerability could enable an attacker to execute arbitrary code on the application server and create a reverse shell, delete files, or access sensitive data. This vulnerability is classified as CWE-502: Deserialization of Untrusted Data

Listing 1: *ysoserial* Command



**Recommendation:** SCS recommends that NPS replace native Java serialization with a safer alternative, such as JSON or XML, in conjunction with implementing input validation. If Java serialization must be used, NPS could implement strict type checking and deserialization whitelisting, as recommended by the OWASP Java Deserialization Cheat Sheet. Additionally, apply the NIST SP 800-53 security controls for data protection, such as SC-8: Transmission Confidentiality and Integrity and SC-28: Protection of Information at Rest.

**Verification:** NPS security teams can verify the remediation of this finding with the ysoserial tool. The tool will stress the Java application with different commands and payloads to determine if the application is still vulnerable.

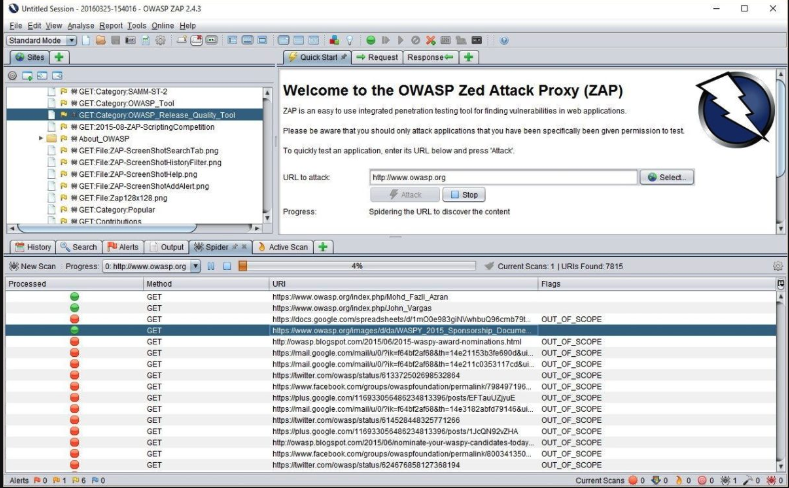


**4. Azure Function Application-SSH Configuration Key Signing Vulnerable to Principal Manipulation**

**Severity:** High

**Finding:** The SCS assessment of the NPS Azure SSH configuration identified a vulnerability in the Azure Function Application responsible for SSH key signing. We used the [*OWASP Zed Attack Proxy*](https://www.zaproxy.org/) (ZAP) tool, shown in Image 2, to intercept and modify HTTP requests and observe the application's response. SCS discovered the application will accept a sign-principal parameter in the request body, which is not documented in the Azure Function App documentation. SCS tested the impact of this parameter by changing its value to different usernames, and verifying the signatures of the returned SSH certificates. This vulnerability could be exploited to sign SSH keys for arbitrary users, effectively bypassing authentication controls and allowing unauthorized SSH access. This finding is classified as a Broken Authentication vulnerability, according to the OWASP Top 10 Application Security Risks.

Image 2: Example of ZAP Tool Interface



**Recommendation:** SCS recommends that NPS remove the sign-principal parameter from the function application or implement robust authentication and authorization checks to ensure the principal specified is the authenticated user. For example, NPS could use Microsoft Entra ID to authenticate and authorize the callers of the function application and verify the sign-principal parameter matches the identity of the caller. Alternatively, NPS could use Azure Key Vault to store and manage the SSH keys and certificates, and delegate the signing operation to the Key Vault service.

**Verification:** NPS security teams can verify the remediation of this finding by repeating the dynamic analysis test with the *OWASP ZAP* or *Burp Suite* tools, and replace the sign-principal parameter value with different usernames. A properly remediated application should generate either an error response or a valid SSH certificate with the same principal as the authenticated caller. In the case of a returned SSH certificate, the NPS security team can use the Linux ssh-keygen tool to inspect the SSH certificates and verify their signatures and principals.

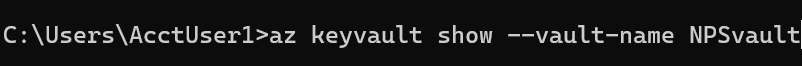


**5. Azure Key Vault-Overly Permissive Access from Azure Virtual Machine Metadata Service/Managed Identity**

**Severity:** High

**Finding:** SCS assessment of the NPS Azure cloud environment identified an Azure Key Vault vulnerability. To discover the access configuration of the Azure Key Vault, we used the Azure CLI tool installed on the virtual machine (VM). We used the az keyvault command, shown in Listing 2, to gather information about the Key Vault, including its access policies and permission model. SCS also ran the az keyvault command, shown in Listing 3, to enumerate any deleted Key Vaults that could be recovered or purged. In addition, we verified that the VM had a managed identity assigned to it and that the identity had access to the Key Vault.

Listing 2: Azure Key Vault Disclosure



Listing 3: Identify Deleted Azure Key Vaults



The assessment showed the Key Vault had an overly permissive access policy that grants the VM's managed identity full access to all keys, secrets, and certificates in the Key Vault. This means that any process running on the VM could perform any data plane operation on the Key Vault and its objects, such as read, write, delete, backup, restore, and purge. This is a high-risk misconfiguration that could allow an attacker to compromise the confidentiality, integrity, and availability of the sensitive data stored in the Key Vault.

**Recommendation:** SCS recommends that NPS restrict the Key Vault policies to only those Azure services and user identities that require access. Use Azure role-based access control [(RBAC)](https://learn.microsoft.com/en-us/azure/key-vault/general/rbac-access-policy) to grant the least privilege necessary. As an example: assign the Key Vault Reader role to the VM’s managed identity, which would allow the VM to list and get the Key Vault and its objects, but not to modify or delete the Key Vault objects. Azure RBAC can assign more granular permissions to individual keys, secrets, and certificates, if needed. Additionally, NPS should regularly review the deleted Key Vaults. To prevent unauthorized access to the soft-deleted data, either recover or purge them the deleted Key Vaults.

**Verification:** NPS security teams can verify the remediation of this finding with the Azure CLI tool to check the updated access policies and roles for the Key Vault and its objects. For example, NPS analysts can run the *az* command, as shown in Listing 4.

Listing 4: Azure CLI Command to Set Key Vault Secret

$ az keyvault secret set --vault-name vault-name --name secret-name --value secret-value

A proper remediation should result in an error message, such as:

AuthorizationFailed: The client client-id with object id object-id does not have authorization to perform action Microsoft.KeyVault/vaults/secrets/write over scope /subscriptions/subscription-id/resourceGroups/resource-group-name/providers/Microsoft.KeyVault/vaults/vault-name/secrets/secret-name or the scope is invalid.



**6. Stored Cross-Site Scripting Vulnerabilities**

**Severity:** Medium

**Finding:** SCS scans identified a potential web application vulnerability on IP address 10.136.168.25. SCS analysts accomplished manual confirmation and exploitation using *Burp Suite* to manipulate HTTP SEND. The web application does not sufficiently encode input and output data. Vulnerabilities of this nature are associated with mishandling of data encoding procedures and can result in multiple stored XSS vulnerabilities. Stored XSS vulnerabilities are stored on the server side of the web application, which allows attackers to target other application users and administrators in a manner which can lead to account hijacking, redirection to offsite resources, installation of malicious software, installation of key-loggers, and data exfiltration. We found a preponderance of responses where special characters such as <, >, {, and } are returned in application responses without sanitization or encoding. Although such responses contain JSON data, they are processed unsafely by client-side scripts where special characters are not properly sanitized and eventually evaluated as valid HTML.

**Recommendation:** SCS recommends that NPS consider the use of a Content Security Policy (CSP) for all web applications. A properly configured CSP can prevents an attacker from manipulating a victim's browser to run attacker-supplied JavaScript or alter HTML. In addition, NPS should improve input and output encoding processes throughout the web application. All content not expressly authorized as HTML, CSS, or JavaScript content should be encoded to prevent exposure to XSS attacks, including JSON content.

**Verification:** NPS security teams can verify the remediation of this finding by submitting XSS attack language against the web applications at 10.136.168.25, such as those shown in Listing 5. If an alert box is triggered, the issue is not resolved.

Listing 5





**7. Browsable Directory Structure**

**Severity:** Medium

**Finding:** SCS assessed the web servers within the target pool and identified two servers, shown in Table 1, with current configurations that permit users to browse the directory structures of the website. This configuration allows users to identify the directory structures of the website, access potentially sensitive information, and identify other web application vulnerabilities, such as Cross Site Scripting. SCS analyzed the web server with two different URL fuzzer tools, *[wfuzz](https://wfuzz.readthedocs.io/en/latest/" \t "_blank)* and *[DirBuster](https://www.kali.org/tools/dirbuster/" \t "_blank)*, to generate and send large numbers of requests to the target servers, with a different directory and file name for each iteration. The respective tools analyzed the HTTP responses and identified those that returned a directory listing or file content. We further manually verified the results by accessing the URLs in a browser and observing the server's behavior.

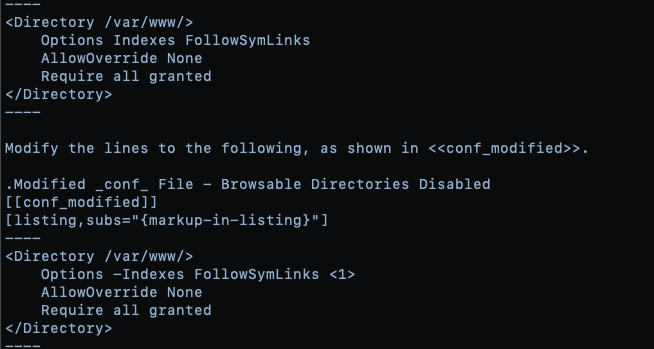
Table 1. IP Addresses with Browsable Directories

10.134.164.172 10.134.174.53

**Recommendation:** SCS recommends that NPS disable directory browsing on both servers. To disable browsable directories on a Linux Apache web server, depending on your Linux server setup, open the Apache apache2.conf or httpd.conf file with a text editor of choice. The most common locations are /etc/httpd/conf/httpd.conf for Red Hat/Fedora/CentOS distributions, and /etc/apache2/apache2.conf for Debian/Ubuntu distributions.

Locate and modify the Options Indexes line as shown in Listing 6.

Listing 6



**Verification:** NPS security teams can verify the remediation of this finding by attempting to navigate the directory structure of either web server at the IP addresses in Table 1. Proper remediation should result in a 404 or 403 error when attempting to access the directories within the web site.

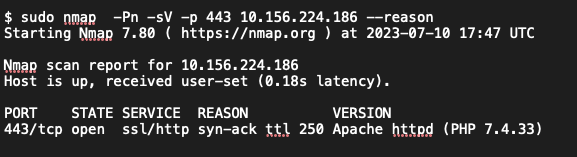


**8. Deprecated Version of PHP Scripting Language**

**Severity:** Low

**Finding:** SCS assessment of the external systems revealed a host running end-of-life PHP version 7.4.33 on the host at 10.156.224.186. SCS manually enumerated the PHP version of the host with the Nmap tool. The command and output are shown in Listing 7.

Listing 7: Nmap Scan of 10.156.224.186



Per the PHP version webpage, 7.4.33 was deemed end-of-life in November of 2022. This means this version of PHP no longer receives security updates. Running this outdated version of PHP exposes the host to potential security risks, such as remote code execution, invalid passwords being accepted as valid, and denial of service.

**Recommendation:** SCS recommends NPS upgrade the PHP installation on host 10.156.224.186 to version 8.2. PHP 8.2 is a major update of the PHP language that contains many new features, performance improvements, and security enhancements. Updating the PHP version will ensure that the host receives the latest security patches and benefits from the new features of PHP 8.2, such as readonly classes, disjunctive normal form types, and sensitive parameter redaction.

**Verification:** NPS security team analysts can verify the remediation of this finding by conducting an Nmap scan, shown in Listing 4, against the target server. Alternatively, the PHP version can be checked by running the php -v command on the host's terminal. SCS includes an example of this output on a successfully-remediated server in Listing 8.

Listing 8: PHP Version Check Example





**9. Internal IP Address Disclosure**

**Severity:** Low

**Finding:** SCS identified 21 externally facing IP addresses, shown in Table 2, that expose their respective internal IP addresses when queried through an HTTP header request. When given an HTTP 7.4.33 request, and no Host header or one with no value, the server returns its private IP address as part of Location header. An attacker can use this information in reconnaissance, network mapping, and social engineering.

Table 2. Hosts Disclosing Internal IP Address via HTTP Header Requests:

10.136.194.166 10.136.194.69 10.136.198.33 10.136.168.53 10.136.194.211

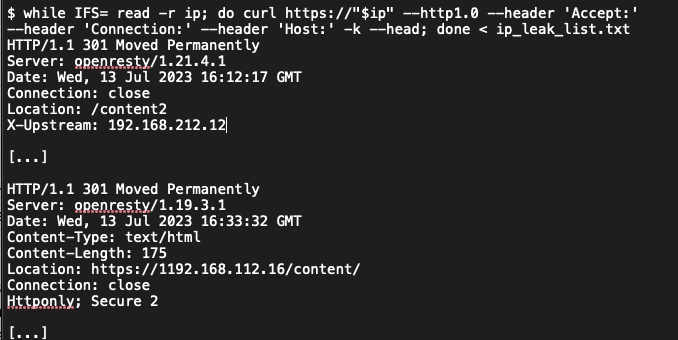
10.136.195.21 10.136.164.133 10.136.168.80 10.136.194.213 10.136.194.234

10.136.195.43 10.136.164.162 10.136.168.86 10.136.194.232 10.136.164.53

10.136.196.15 10.136.164.164 10.136.194.233 10.136.196.43 10.136.164.167 10.136.198.29

To expose the internal IP address of each target IP address, SCS used the cURL command shown in Listing 6.

Listing 6: cURL Command To Expose Internal IP Addresses



**Recommendation:** SCS recommends that NPS modify the *Location* header to reflect the host Windows registration key rather than the internal IP address of the host.

• Ensure that DNS records are properly configured so the internal host-names do not resolve to the internal IP address in public DNS servers.

• Use security headers, such as Content-Security-Policy and X-Content-Type-Options, to mitigate risks associated with exposing internal IP addresses.

• Implement a Web Application Firewall (WAF) to filter and protect against malicious requests that may attempt to exploit internal IP disclosures.

**Verification:** NPS security team analysts can verify the remediation of this finding with the cURL command, shown in Listing 9, replacing with the IP address of the host in question. A properly secured host should not provide the internal IP address of the host in the Location header field.

Listing 9: cURL Command to Verify Internal IP Address Leak on Target IP Address





Click the button below to submit your review of the findings.