Steve Cross

From: Julian Jacquin
Sent: Wednesday, May 5, 2021 1:45 PM
To: Steve Cross
Subject: Website Security
Attachments: [CivicPlus Help Center] Re: Wildwood, MO vulnerable to SQL injection attacks; CiviPlus - RiskSense Statement of Opinion CivicPlus Engage.pdf[5] (1).pdf; CivicPlus Engage - RiskSense SQLi testing Methodology (1).pdf; RE: [CivicPlus Help Center] Re: Wildwood, MO vulnerable to SQL injection attacks

Steve,

Per our discussion, we re-reviewed our website security measures in December 2020, following a call we received from East-West Gateway Council of Governments regarding our City website and its domain/content security. At that time, both our website vendor (CivicPlus) and our domain host (Network Solutions) assured us that the City website, its contents, and the domain (cityofwildwood.com) are all secure and not at risk of being compromised.

CivicPlus did offer additional "platinum" security tools (at an extra cost), but both CivicPlus and Network Solutions were confident that the City's website was secure. See attached emails from CivicPlus regarding the website/content, and from Kathy Arnett following her discussion with Network Solutions.

Please let me know if you have any questions. Thanks-

Julian M.D. Jacquin | Economic Development Manager
City of Wildwood
16860 Main Street
Wildwood, MO 63040-1242
P: (636) 458-0440 x113
C: (636) 399-0060
www.cityofwildwood.com

WILDWOOD
#GoWildwoodMO

RISKSENSE™

October 18th, 2020

To Whom It May Concern:

This is a statement of opinion on the application security audit and penetration testing the RiskSense team conducted to evaluate Civic Engage application in compliance with National Institute for Standards and Technology (NIST) information security guidelines.

RiskSense, an Independent Cyber Risk Management Company, was contracted to perform the penetration testing in October of 2020. Audits are conducted based on the industry best practices and in accordance with government standards.

- The Open Web Application Security Project (OWASP)Top 10
- The US Department of Homeland Security Common Weakness Enumeration (CWE) Top 25 Most Dangerous Software Weaknesses

Based on our audit procedures, we conclude that the Civic Engage application meet the requirements of NIST 53a, and industry best practices.

As a part of the audit, RiskSense conducted a comprehensive security assessment and penetration testing, which included automated testing, manual testing, configuration review, and privilege escalation. Penetration test scenarios are designed to simulate both an inside and/or an outside attack.

The Application security review and testing included OWASP Top 10 vulnerabilities, CWE/SANS Top 25 Most Dangerous Software Weaknesses, and critical database vulnerabilities.

The audit did not identify any critical or high vulnerabilities. A few medium level and low-level vulnerabilities were identified and it is recommended to remediate these findings to improve the applications overall security posture.

Based upon the results, we conclude that Civic Engage application is implemented using built-in security, secure-by-design, and best-of-breed defensive security practices. To this effect, Civic Engage application is resilient to known attack patterns and to attacks launched using known automated attack tool kits.

Therefore, it is RiskSense's opinion that a typical user of CivicPlus would require a sophisticated level of security attack skills to breach such an application. In this capacity, we are pleased to provide a positive rating on the overall security risk posture of Civic Engage application.

Sincerely,

Srinivas Mukkamala, PhD.
CEO and Co-Founder
RiskSense, Inc.

CSW Cyber SecurityWorks

Civic Engage 5 Application | Methodology to detect SQLi

CSW SQL injection detection/exploitation methodology is divided into following phases:

1. Discovery/Crawling:

- Manually crawl the entire web application with/without authentication using burp suite.
- Run Acunetix to crawl the entire web application in trained mode (Authenticated and non-authenticated) to discover all the URLs
- Additionally use burp suite to filter the JS files and pass it to relative-url-extractor to find the possible hidden/exposed endpoints
- Identify all the GET and POST parameters in the discovered URLs
- Identify all the possible injection points

2. Automated Scanning

- Perform automated scanning using Burp Suite and Acunetix for all the discovered URL and injection points
- Look for all possible SQLi's or SQL based error pattern discovered during the automated scanning

3. Manual Testing and Exploitation

- Verify all the possible SQLi's or SQL based error pattern to remove potential false positives
- Manually test for all different types of SQLi (Error based, Time based, Boolean based etc.) with Burp Suite (Intruder/Repeater) on all the injection point discovered
    - Detection - Used a single quote ('), double quote("), Braces and slashes ({,},[,],\,) or Time-based payload like ' sleep(10)-- - or Boolean based payloads like ' OR 1=1-- - etc.
    - Exploitation – Depending upon the type of SQLi detected use manual and automated approach (SQLMap) to exploit.