
# FEDERAL VOTING ASSISTANCE PROGRAM VOTING PENETRATION TEST

## CONTRACT NUMBER: HHS CASU WII-0037-CALIBRE

**August 15, 2011**

**Submitted to:**
CALIBRE Systems

**Submitted by:**
RedPhone, LLC

[redacted]

POINT OF CONTACT: L. Jay Aceto, CISSP, ISSAP/MP, CISM, NSA-IAM/IEM

Telephone: 571-334-9225 • E-mail Address: jay.aceto@redphonecorporation.com

# Table of Contents

| Document Properties |
| --- |
| |
| Title: Multi-Vendor Mock Voting Exercise - Operation Orange Black Box Penetration Testing Report |
| Version V1.0 |
| Author L. Jay Aceto CISSP, CISM, ISSAP/MP, NSA-IAM/IEM |
| Technical Review: TC McFall |
| Peer Review: Josha Richards, Aaron Bossert, Michael Carter |
| RedPhone Penetration testers: TC McFall, L. Jay Aceto |

| Version control |
| --- |
| Version : 1.0 |
| Date : August 15, 2011 |
| |

## Executive Summary

The democratic process rests on a fair, universally accessible, anonymous voting system through which all citizens can easily and accurately cast their vote. At present, over 6,000,000 voters reside outside the United States and rely on traditional paper-based registration and voting processes that are inadequate at meeting their needs, and fraught with inherent delays. The main issues revolve around the inherent latency with the registration, receipt, and delivery of ballots by traditional mail. The Federal Voting Assistance Program (FVAP), a United States Department of Defense (DoD) controlled program, has been systematically gathering, analyzing, and reporting on the voter's experience, and exploring new technologies to improve the delivery of registration and ballot materials.

RedPhone, LLC., a Virginia-based information assurance and security consultancy to the U.S. DoD, civilian, and state governments, as well as commercial enterprises, was contracted to provide penetration testing services to CALIBRE Systems in support of the FVAP to test and evaluate the security of three Internet voting systems. The penetration test team was led by CALIBRE Management, however, the primary responsibility for the testing and analysis resided with RedPhone, LLC. Additionally, RedPhone, LLC. prepared the testing scenario and the rules of engagement that the Air Force Institute of Technology (AFIT) and other outside penetration testing teams would use to determine the scope and boundaries of the engagement. The fictitious *Operation Orange* exercise and the rules of engagement are listed within the appendices.

Beginning in May of 2011, and culminating in the actual penetration testing and mock election exercise that spanned 72 hours from August 2-4, 2011, all three participating vendors' systems were carefully evaluated for their security posture, defensive capabilities, critical logging and security architecture limitations. Historically, the application development processes associated with these critical applications have not followed industry best practices. This flawed state is the result of undisciplined software development, and a process that failed to encourage developers to anticipate or fix security holes. The closed-source approach to software development, which shielded the source code from public review and comment, only served to delay the necessary scrutiny. However, all three vendors have been highly supportive of these tests, and it is obvious that they have made great strides to improve the security posture of their respective products. Six independent technical security experts with an extensive background in web application security and information assurance were charged with attempting to breach the security of each of the three participating vendors. Two AFIT cyber security teams were also participating in the penetration testing process. This

report is the culmination of the penetration test team's findings, potential mitigations, and recommendations.

Penetration testing typically falls into the following three categories: "White box" testing is performed with the full knowledge and support of the vendor, and the vendor provides unlimited access to the software, supporting documentation and staff. "Grey box" testing is a partial knowledge test scenario where the test team has only limited knowledge of the vendor's products and services, and the rest must be obtained via research. In "black box" testing, the test teams are given very little if any advanced knowledge of the vendor's products, and therefore, must gain as much knowledge as possible independently in a discovery and reconnaissance effort. The penetration test team for this exercise used a "black box" approach, wherein little information is provided from the vendors, and only a brief window is available to research each vendor to prepare an attack strategy.

Although the penetration test teams designed various attacks, they generally fell into one of five categories:

1. vote manipulation at the client work station PC or server databases,
2. attacks aimed at breaking the authentication mechanism for PIN's or administrative access,
3. attacks directed at defeating voter anonymity,
4. analysis of data in transit that could have been altered, or
5. denial-of-service that prevents voters from being able to reach or cast votes.

Most attack vectors fell into the first category.

The RedPhone penetration test team applied the Open Web Application Security Project (OWASP) evaluation methodology of attack mapping, threat modeling, and poor trust relationship failure analysis to assess where to focus their attention, and then used standard pen-testing tools including attacking physical security, network scanning to locate and exploit vulnerabilities in each of the vendor system. This approach does not look at possible vulnerabilities that may be inherent in the system architecture or data handling procedures at the precinct level. Because of the very limited time and resources available, RedPhone, LLC. adopted an almost entirely ad hoc approach, focusing our attention on those parts of the system that we believed might provide the best attack vector to less secure devices within the DMZ. While we used some source code analysis tools-and several widely used "hacking" tools like Nessus, NMAP and Metasploit-we applied them only selectively, and instead adopted a more "curious" strategy most often used by an

attacker that seeks out weaknesses in the places where he would most likely find vulnerabilities, and then moving on to the next place of potential weakness. This is a very common approach used when limited time and information is available, and when known security is in place, such as out-sourced managed firewalls, routers, or intrusion detection and prevention devices. Our overall impression of the security posture for all three participating vendors was good. We did not find any significant technical security concerns, only minor correctable issues that can easily be mitigated. While time constraints were the biggest limitation, we did find at least one issues involving SSH installed on a server, presumably for remote management purposes. This was the most serious findings, as given more time, we could have likely cracked the password and gained access to the server. We found obvious places where SQL-injection exists, and were tested, but not to the extent that any were successful. Cross-site scripting (reflected) is another case wherein proper coding procedure isn't being followed; however, other mitigating security controls were in place that did not allow for successful penetration. We've documented a good number of informational findings that should be used to improve overall UOCAVA best practice security guidelines.

RedPhone wishes to emphasize that our results do not extend beyond the scope of our investigation of the technical security of the application as seen from the outside. Our scope was limited to that which is defined in our contract with CALIBBRE Systems, and do not contend that these systems are correct or secure beyond the specific findings we've addressed here. Unless otherwise noted, the results of this investigation should be assumed to be relevant only to these three vendor systems and the software version used for this test.

## GLOBAL OBJECTIVES

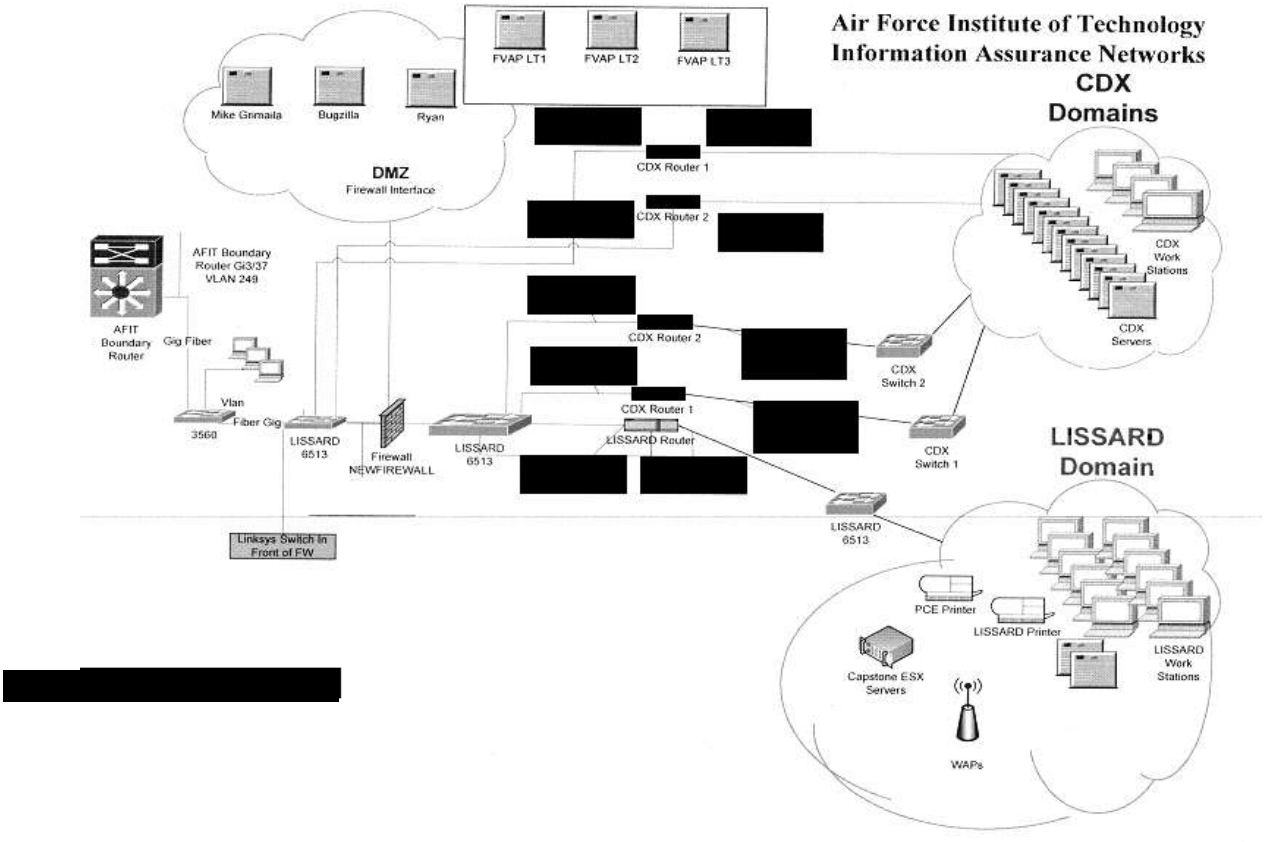
- Breach the security of each vendor's voting systems and gain access to sensitive information on the DMZ Network where a tangential attack vector could be made into the more secure voting systems.
- To emulate a realistic technical threat to the ATF computer networks from persons having no prior access or knowledge other than information that is openly available on the Internet;
- To discover and exploit any vulnerability or combination of vulnerabilities found on the system in order to meet the stated objective of the penetration test; and
- To test the extent an organization's security incident response capability is alerted and to gauge the response to such suspicious activity.

- Recommend best security practices and guidelines that would mitigate these attacks.

## PENETRATION TESTING ARCHITECTURE

The AFIT network architecture used by the two internal penetration testing teams is a traditional network architecture that includes a test lab environment, routers, firewalls, a DMZ, and unfiltered access out to the Internet where the penetration test teams used MicroSoft Internet Explorer and Mozilla Firefox browsers to connect to the target servers and local workstations used as voting stations. The AFIT penetration testing team used multiple tools that included, Nessus, NMAP, Metasploit and other tools found on the BackTrack 5 live CD. A complete list of tools used by the AFIT test teams will be provided with their documentation. The RedPhone penetration team performed all their tests remotely, but was on site daily to assist with AFIT testing coordination and support. The laptops used by the AFIT teams were located with the lab environment and provided with unfiltered access to the Internet; the voting station laptops were located within the AFIT's Doolittle lounge where other Air Force personnel could use them for simulated voting. There were no physical security controls placed upon the voting work stations. Below is a high-level representation of the AFIT information assurance network used for the testing. IP addresses have been removed or blacked out.

Figure 1. AFIT Network Architecture



# Findings

Each of the vendor's systems provided a level of security that was consistent with most business and technical security best practices. Each vendor's automated security systems detected our attempts to breach the security of the applications at the server side, and response and notification times were well within service level agreement time frames. Also, each vendor was able to quickly identify the attacking IP addresses, shut down the attack, and provide log verification. Therefore, we are confident that each vendor's security systems could detect and respond to most attempts to breach the security and gain access to the system. Specific technical findings are listed below:

## FINDING NO. 1: SSH
## SEVERITY: HIGH

Brute-force authentication attacks against one vendor's Secure Shell (SSH) service was not successful, but this service should never be made available to a production server, as penetration is almost assured given ample time.

### Issue Background

US-CERT issues SSH concerns frequently and should be heeded. The SSH is a network protocol that creates a secure channel between two networked devices in order to allow data to be exchanged. SSH can create this secure channel by using Cipher Block Chaining (CBC) mode encryption. This mode adds a feedback mechanism to a block cipher that operates in a way that ensures that each block is used to modify the encryption of the next block.

SSH contains a vulnerability in the way certain types of errors are handled. Attacks leveraging this vulnerability would lead to the loss of the SSH session. According to CPNI Vulnerability Advisory SSH:

> *If exploited, this attack can potentially allow an attacker to recover up to 32 bits of plaintext from an arbitrary block of ciphertext from a connection secured using the SSH protocol in the standard configuration. If OpenSSH is used in the standard configuration, then the attacker's success probability for recovering 32 bits of plaintext is 2^{-18}. A variant of the attack against OpenSSH in the standard configuration can verifiably recover 14 bits of plaintext with probability 2^{-14}. The success probability of the attack for other implementations of SSH is not known.*

### Impact

An attacker may be able to recover up to 32 bits of plaintext from an arbitrary

block of ciphertext.

### Issue Mitigation

We are currently unaware of a practical solution to this problem. CERT recommends the use of CTR Mode. This mode generates the keystream by encrypting successive values of a "counter" function. For more information see the Block Cipher Modes article on wikipedia.

In order to mitigate this vulnerability, SSH can be setup to use CTR mode rather CBC mode. According to CPNI Vulnerability Advisory SSH:

> *The most straightforward solution is to use CTR mode instead of CBC mode, since this renders SSH resistant to the attack. An RFC already exists to standardise counter mode for use in SSH (RFC 4344)...*

### Systems Affected

| Vendor | Status | Date Notified | Date Updated |
|---|---|---|---|
| Bitvise | Vulnerable | 2008-11-07 | 2008-11-24 |
| FiSSH | Vulnerable | 2008-11-07 | 2008-11-24 |
| Icon Labs | Vulnerable | 2008-11-07 | 2008-11-24 |
| OpenSSH | Vulnerable | 2008-11-07 | 2008-11-24 |
| OSSH | Vulnerable | 2008-11-07 | 2008-11-24 |
| PuTTY | Vulnerable | 2008-11-07 | 2009-01-05 |
| Redback Networks, Inc. | Vulnerable | 2008-11-07 | 2008-11-24 |
| SSH Communications Security Corp | Vulnerable | 2008-11-07 | 2008-11-24 |
| TTSSH | Vulnerable | 2008-11-07 | 2008-11-24 |
| VanDyke Software | Vulnerable | 2008-11-07 | 2009-01-12 |
| Wind River Systems, Inc. | Vulnerable | 2008-11-07 | 2008-11-24 |

### References


http://www.cpni.gov.uk/Docs/Vulnerability_Advisory_SSH.txt
http://isc.sans.org/diary.html?storyid=5366
http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation


## FINDING NO. 2: SQL INJECTION
## SEVERITY: MODERATE

The findings listed below are generic and do not reflect any specific vendor's environment. We have kept them generic so that FVAP can assess the overall security posture of these voting systems and make determination about the high-level

guidance and policy recommendations that may be required.

There are five instances of this issue:

### Issue background

SQL injection vulnerabilities arise when user-controllable data are 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.

Various attacks can be delivered via SQL injection, including reading or modifying critical application data, interfering with application logic, escalating privileges within the database, and executing operating system commands.

### Issue remediation

The most effective way to prevent SQL injection attacks is to use parameterised queries (also known as prepared statements) for all database access. This method uses two steps to incorporate potentially tainted data into SQL queries: first, the application specifies the structure of the query, leaving placeholders for each item of user input; second, the application specifies the contents of each placeholder. Because the structure of the query has already been defined in the first step, it is not possible for malformed data in the second step to interfere with the query structure. Documentation should be reviewed for the database and application platform to determine the appropriate APIs, which can be used to perform parameterised queries. It is strongly recommended that *every* variable data item that is incorporated into database queries is parameterised, even if it is not obviously tainted, to prevent oversights occurring and avoid vulnerabilities being introduced by changes elsewhere within the code base of the application.

FVAP should be aware that some commonly employed and recommended mitigations for SQL injection vulnerabilities are not always effective:

- One common defense is to double up any single quotation marks appearing within user input before incorporating that input into a SQL query. This defense is designed to prevent malformed data from terminating the string in which they are inserted. However, if the data being incorporated into queries are numeric, then the defense may fail, because numeric data may not be encapsulated within quotes, in which case only a space is required to break out of the data context and interfere with the query. Further, in second-order SQL injection attacks, data that has been safely escaped ("escaping" is a technique used to ensure that characters are treated as data, not as characters) when initially inserted into the database is subsequently read

from the database and then passed back to it again. Quotation marks that have been doubled up initially will return to their original form when the data are reused, allowing the defense to be bypassed.

- Another often cited defense is to use stored procedures for database access. While stored procedures can provide security benefits, they are not guaranteed to prevent SQL injection attacks. The same kinds of vulnerabilities that arise within standard dynamic SQL queries can arise if any SQL is dynamically constructed within stored procedures. Further, even if the procedure is sound, SQL injection can arise if the procedure is invoked in an unsafe manner using user-controllable data.

## FINDING NO.3: CROSS-SITE SCRIPTING (REFLECTED)
## SEVERITY: MODERATE

### Issue detail

The value of the parenturl request parameter is copied into a JavaScript string, which is encapsulated in single quotation marks. The payload bb8cf'%3b6b50cb864d6 was submitted in the parenturl parameter. This input was echoed as bb8cf';6b50cb864d6 in the application's response.

This behavior demonstrates that it is possible to terminate the JavaScript string into which data are being copied. An attempt was made to identify a full proof-of-concept attack for injecting arbitrary JavaScript, but this was not successful. The application's behavior should be manually examined and any unusual input validation or other obstacles that may be in place should be identified.

### Remediation detail

Echoing user-controllable data within a script context is inherently dangerous, and can make XSS attacks difficult to prevent. If at all possible, the application should avoid echoing user data within this context.

### Issue background

Reflected cross-site scripting vulnerabilities arise when data are 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 that, 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, and logging their keystrokes.

Users can be induced to issue the attacker's crafted request in various ways. For example, the attacker can send a victim a link containing a malicious URL in an email or instant message. They can submit the link to popular websites that allow content authoring, for example, in blog comments. And they can create an innocuous looking website which causes anyone viewing it to make arbitrary cross-domain requests to the vulnerable application (using either the GET or the POST method).

The security impact of cross-site scripting vulnerabilities is dependent upon the nature of the vulnerable application, the kinds of data and functionality that it contains, and the other applications that belong to the same domain and organization. If the application is used only to display non-sensitive public content, with no authentication or access control functionality, then a cross-site scripting flaw may be considered low risk. However, if the same application resides on a domain that can access cookies for other more security-critical applications, then the vulnerability could be used to attack those other applications, and so may be considered high risk. Similarly, if the organization that owns the application is a likely target for phishing attacks, then the vulnerability could be leveraged to lend credibility to such attacks by injecting Trojan functionality into the vulnerable application and exploiting users' trust in the organization in order to capture credentials for other applications that it owns. In many kinds of application, such as those providing online banking functionality, cross-site scripting should always be considered high risk.

### Remediation background

In most situations where user-controllable data are copied into application responses, cross-site scripting attacks can be prevented using two layers of defenses:

- Input should be validated as strictly as possible on arrival, given the kind of content that it is expected to contain. For example, personal names should consist of alphabetical and a small range of typographical characters, and be relatively short; a year of birth should consist of exactly four numerals; email addresses should match a well-defined regular expression. Input which fails the validation should be rejected, not sanitized.

- User input should be HTML-encoded at any point where it is copied into application responses. All HTML metacharacters, including < > " ' and =, should be replaced with the corresponding HTML entities (< > etc).

In cases where the application's functionality allows users to author content

using a restricted subset of HTML tags and attributes (for example, blog comments that allow limited formatting and linking), it is necessary to parse the supplied HTML to validate that it does not use any dangerous syntax; this is a non-trivial task.

## FINDING NO.4: SSL COOKIE
### SEVERITY: LOW

#### Issue detail

The following cookie was issued by the application and does not have the secure flag set:

- ASP.NET_SessionId=51dw1odzrv11hdjzl5ztmosw; path=/; HttpOnly

The cookie appears to contain a session token, which may increase the risk associated with this issue. The contents of the cookie should be reviewed to determine its function.

#### Issue background

If the secure flag is set on a cookie, then browsers will not submit the cookie in any requests that use an unencrypted HTTP connection, thereby preventing the cookie from being trivially intercepted by an attacker monitoring network traffic. If the secure flag is not set, then the cookie will be transmitted in clear-text if the user visits any HTTP URLs within the cookie's scope. An attacker may be able to induce this event by feeding a user suitable links, either directly or via another website. Even if the domain that issued the cookie does not host any content that is accessed over HTTP, an attacker may be able to use links of the form http://example.com:443/ to perform the same attack.

#### Issue remediation

The secure flag should be set on all cookies that are used for transmitting sensitive data when accessing content over HTTPS. If cookies are used to transmit session tokens, then areas of the application that are accessed over HTTPS should employ their own session handling mechanism and the session tokens used should never be transmitted over unencrypted communications.

## FINDING NO. 5: SSL CERTIFICATES
### SEVERITY: LOW

This finding is more informational than an actual vulnerability. The vendor had "self-signed" the certificate, and therefore, would not be a trusted certificate, but the vendor had brought this to our attention and explained that this would not be the norm. The other two vendors had implemented the use of certificates

properly.

### Issue background

SSL helps to protect the confidentiality and integrity of information in transit between the browser and server, and to provide authentication of the server's identity. To serve this purpose, the server must: present an SSL certificate that is valid for the server's hostname, is issued by a trusted authority and is valid for the current date. If any one of these requirements is not met, SSL connections to the server will not provide the full protection for which SSL is designed.

It should be noted that various attacks exist against SSL in general, and in the context of HTTPS web connections. It may be possible for a determined and suitably-positioned attacker to compromise SSL connections without user detection even when a valid SSL certificate is used.

## FINDING NO. 6: COOKIE WITHOUT HTTPONLY FLAG SET
## SEVERITY: LOW

This is mostly informational but does constitute a concern.

### Issue detail

The following cookie was issued by the application and does not have the HttpOnly flag set:

- JSESSIONID=AB6295DFFAFA6F01E835E88C50F597ED; Path=/portal-webapp; Secure

The cookie appears to contain a session token, which may increase the risk associated with this issue. The contents of the cookie should be reviewed to determine its function.

### Issue background

If the HttpOnly attribute is set on a cookie, then the cookie's value cannot be read or set by client-side JavaScript. This measure can prevent certain client-side attacks, such as cross-site scripting, from trivially capturing the cookie's value via an injected script.

### Issue remediation

There is usually no good reason not to set the HttpOnly flag on all cookies. Unless legitimate client-side scripts are specifically required within an application to read or set a cookie's value, the HttpOnly flag should be set by including this attribute within the relevant Set-cookie directive.

Guidance should make implementers aware that the restrictions imposed by the

HttpOnly flag can potentially be circumvented in some circumstances, and that numerous other serious attacks can be delivered by client-side script injection, aside from simple cookie stealing.

## FINDING NO. 7: REFERER-DEPENDENT RESPONSE
## SEVERITY: INFORMATIONAL

### Issue description

The application's responses appear to depend systematically on the presence or absence of the Referer header in requests. This behavior does not necessarily constitute a security vulnerability, and the nature of and reason for the differential responses should be investigated to determine whether a vulnerability is present.

Common explanations for Referer-dependent responses include:

- Referer-based access controls, where the application assumes that if the user has arrived from one privileged location then he/she is authorized to access another privileged location. These controls can be trivially defeated by supplying an accepted Referer header in requests for the vulnerable function.
- Attempts to prevent cross-site request forgery attacks by verifying that requests to perform privileged actions originated from within the application itself and not from some external location. Such defenses are not robust- methods have existed through which an attacker can forge or mask the Referer header contained within a target user's requests by leveraging client-side technologies such as Flash and other techniques.
- Delivery of Referer-tailored content, such as welcome messages to visitors from specific domains, search-engine optimisation (SEO) techniques, and other ways of tailoring the user's experience. Such behaviors often have no security impact, however, unsafe processing of the Referer header may introduce vulnerabilities such as SQL injection and cross-site scripting. If parts of the document (such as META keywords) are updated based on search engine queries contained in the Referer header, then the application may be vulnerable to persistent code injection attacks, in which search terms are manipulated to cause malicious content to appear in responses served to other application users.

### Issue remediation

The Referer header is not a robust foundation on which to build any security measures, such as access controls or defenses against cross-site request forgery. Any such measures should be replaced with more secure alternatives that are not

vulnerable to Referer spoofing.

If the contents of responses is updated based on Referer data, then the same defenses against malicious input should be employed here as for any other kinds of user-supplied data.

## FINDING NO. 8: OPEN REDIRECTION
## SEVERITY: INFORMATIONAL

### Issue detail

The value of the Referer HTTP header is used to perform an HTTP redirect. The payload //acec8732e3c7ad76d/a%3fhttp%3a//www.google.com/search%3fhl%3den%26q%3d was submitted in the Referer HTTP header. This caused a redirection to the following URL:

- //acec8732e3c7ad76d/a%3fhttp%3a//www.google.com/search%3fhl%3den%26q%3d

The application attempts to prevent redirection attacks by blocking absolute redirection targets starting with http:// or https://. However, an attacker can defeat this defense by omitting the protocol prefix from their absolute URL. If a redirection target starting with // is specified, then the browser will use the same protocol as the page that issued the redirection.

Because the data used in the redirection are submitted within a header, the application's behavior is unlikely to be directly useful in lending credibility to a phishing attack. This limitation considerably mitigates the impact of the vulnerability.

### Remediation detail

When attempting to block absolute redirection targets, the application should verify that the target begins with a single slash followed by a letter and should reject any input containing a sequence of two slash characters.

### Issue background

Open redirection vulnerabilities arise when an application incorporates user-controllable data into the target of a redirection in an unsafe way. An attacker can construct a URL within the application, which causes a redirection to an arbitrary external domain. This behavior can be leveraged to facilitate phishing attacks against users of the application. The ability to use an authentic application URL, targeting the correct domain with a valid SSL certificate (if SSL is used), lends credibility to the phishing attack because many users, even if they verify these features, will not notice the subsequent redirection to a different

domain.

### Remediation background

If possible, applications should avoid incorporating user-controllable data into redirection targets. In many cases, this behavior can be avoided in two ways:

- Remove the redirection function from the application, and replace links to it with direct links to the relevant target URLs.
- Maintain a server-side list of all URLs that are permitted for redirection. Instead of passing the target URL as a parameter to the redirector, pass an index into this list.

If it is considered unavoidable for the redirection function to receive user-controllable input and incorporate this into the redirection target. One of the following measures should be used to minimize the risk of redirection attacks:

- The application should use relative URLs in all of its redirects, and the redirection function should strictly validate that the URL received is a relative URL.
- The application should use URLs relative to the web root for all of its redirects, and the redirection function should validate that the URL received starts with a slash character. It should then prepend http://yourdomainname.com to the URL before issuing the redirect.
- The application should use absolute URLs for all of its redirects, and the redirection function should verify that the user-supplied URL begins with http://yourdomainname.com/ before issuing the redirect.

## FINDING NO. 9: CROSS-DOMAIN SCRIPT INCLUDE
## SEVERITY: INFORMATIONAL

### Issue detail

The response dynamically includes the following script from another domain:

- https://seal.verisign.com/getseal?host_name=www.intvoting.com&size=S&use_flash=NO&use_transparent=NO&lang=en

### Issue background

When an application includes a script from an external domain, this script is executed by the browser within the security context of the invoking application. The script can therefore do anything that the application's own scripts can do,

such as accessing application data and performing actions within the context of the current user.

If a script from an external domain is included, then that domain is trusted with the data and functionality of your application, and the domain's own security to prevent an attacker from modifying the script to perform malicious actions within your application.

### Issue remediation

Scripts should not be included from untrusted domains. If there is a requirement that a third-party script appears to fulfill, then ideally the contents of that script should be copied onto your own domain and include it from there. If that is not possible (e.g., for licensing reasons), then re-implementing the script's functionality within your own code should be considered.

## FINDING NO.10: EMAIL ADDRESSES DISCLOSED
## SEVERITY: INFORMATIONAL

### Issue detail

During the discovery and reconnaissance phase, we found many vendor email addresses were available. Caution should be taken to train all employees of spear phishing attacks. Spear phishing describes any highly targeted phishing attack. Spear phishers send e-mail that appears genuine to some or all the employees or members within a certain company, government agency, organization, or group. The message might look like it comes from your employer, or from a colleague sending an e-mail message to everyone in the company (such as the person who manages the computer systems) and could include requests for user names or passwords.

The truth is that the e-mail sender information has been faked or "spoofed." Whereas traditional phishing scams are designed to steal information from individuals, spear phishing scams work to gain access to a company's entire computer system. If an employee responds with a user name or password, or if click links or open attachments in a spear phishing e-mail, pop-up window, or website, he/she might become a victim of identity theft and might put his/her employer or group at risk.

Spear phishing also describes scams that target people who use a certain product or website. Scam artists use any information they can to personalize a phishing scam to as specific a group as possible.

### Issue background

The presence of email addresses within application responses does not necessarily constitute a security vulnerability. Email addresses may appear intentionally within contact information, and many applications (such as web mail) include arbitrary third-party email addresses within their core content.

However, email addresses of developers and other individuals (whether appearing on-

screen or hidden within page source) may disclose information that is useful to an attacker; for example, they may represent usernames that can be used at the application's login, and they may be used in social engineering attacks against the organization's personnel. Unnecessary or excessive disclosure of email addresses may also lead to an increase in the volume of spam email received.

### Issue remediation

FVAP should review and offer guidance concerning the email addresses being disclosed by the application, and consider removing any that are unnecessary, or replacing personal addresses with anonymous mailbox addresses (such as helpdesk@example.com).

## FINDING NO. 11: ROBOTS.TXT FILE
## SEVERITY: LOW/INFORMATIONAL

While this issue can often give away information to an attacker, this particular instance did not. Therefore, this is informational only.

### Issue detail

The web server contains a robots.txt file.

### Issue background

The file robots.txt is used to give instructions to web robots, such as search engine crawlers, about locations within the website that robots are allowed, or not allowed, to crawl and index.

The presence of the robots.txt does not in itself present any kind of security vulnerability. However, it is often used to identify restricted or private areas of a site's contents. The information in the file may, therefore, help an attacker to map out the site's contents, especially if some of the locations identified are not linked from elsewhere in the site. If the application relies on robots.txt to protect access to these areas and does not enforce proper access control over them, then this presents a serious vulnerability.

### Issue remediation

The robots.txt file is not itself a security threat, and its correct use can represent good practice for non-security reasons. You should not assume that all web robots will honor the file's instructions. Rather, assume that attackers will pay close attention to any locations identified in the file. Do not rely on robots.txt to provide any kind of protection over unauthorized access.

## FINDING NO. 12: CACHEABLE HTTPS RESPONSE
## SEVERITY: INFORMATIONAL

There are three instances of this issue. This is a minor issue, bordering on informational. These are the result of implementation errors that can be easily corrected.

### Issue 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.

### Issue 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. Alternatively, most web development platforms allow control of the server's caching directives from within individual scripts. Ideally, the web server should return the following HTTP headers in all responses containing sensitive content:

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

## FINDING NO. 13: SCRIPT FILES
## SEVERITY: MODERATE

We successfully downloaded all site scripts from every vendor, no exceptions. With more time allotted to a penetration, this would be a severe issue. Going through the script's contents (and comment sections, etc.) would allow for detailed mapping of site functionality. Hardening of application server configurations is highly recommended for each vendor, in order to mitigate this threat.

### Additional tests performed

These types of Distributed Denial-of-Service (DDoS) attacks are not new. Organizations have been battling them since they became popular in the late 1990s. While techniques to defend against DDoS attacks have become more sophisticated, they still represent a difficult challenge and major risk. Limited Denial-of-Service (DoS) attacks were performed. These were unsuccessful. However, mention should be given that no DDoS attacks were performed due to lack of

resources available for the test. It is entirely feasible for a mass denial attack to be successful, and this is an eventuality that is difficult to mitigate.

The DoS attack is focused on making unavailable a resource (site, application, server) for the purpose it was designed. There are many ways to make a service unavailable for legitimate users by manipulating network packets, programming, logical, or resources handling vulnerabilities, among others. If a service receives a very large number of requests, it may stop providing service to legitimate users. In the same way, a service may stop if a programming vulnerability is exploited.

Sometimes the attacker can inject and execute arbitrary code while performing a DoS attack in order to access critical information or execute commands on the server. DoS attacks significantly degrade service quality experienced by legitimate users. It introduces large response delays, excessive losses, and service interruptions, resulting in direct impact on availability.

### DoS & DDoS Locking Customer Accounts

The first DoS case to consider involves the authentication system of the target application. A common defense to prevent brute-force discovery of user passwords is to lock an account from use after between three to five failed attempts to login. This means that even if a legitimate user were to provide their valid password, they would be unable to login to the system until their account has been unlocked. This defense mechanism can be turned into a DoS attack against an application if there is a way to predict valid login accounts.

Note: there is a business vs. security balance that must be reached based on the specific circumstances surrounding a given application. There are pros and cons to locking accounts, to customers being able to choose their own account names, to using systems such as CAPTCHA, and the like. Each enterprise will need to balance these risks and benefits, but not all of the details of those decisions are covered here. It should be noted that one vendor does incorporate CAPTCHA as a deterrent to this form of attack. Specific controls to combat DDoS attacks can include:

1. working with the Internet Service Provider (ISP) to establish quality of service rates to limit the amount of bandwidth one customer can utilize;
2. using firewalls and filtering devices to filter all unnecessary ports and protocols;
3. incorporating redundancy and resiliency into designs of key systems; and
4. utilizing IDS/IPS to identify and block attacks in progress

### Related Attacks

- Resource Injection
- Setting Manipulation
- Regular expression Denial of Service - ReDoS

### Related Vulnerabilities

- Category: Input Validation Vulnerability
- Category: API Abuse

## Summary & Conclusions:

Internet based voting systems should be certified and recertified on a regular basis since changes to the operating systems, applications, services, protocols etc. change frequently. All defensive strategies should be risk-based and right-sized to match the risk. In a perfect world, every company could employ every defense possible to protect against every type of attack on every part of its infrastructure. In reality, however, time and resources are not unlimited. Defenses have to be selected and deployed based on a cost-benefit methodology. Voting systems face unique threats, some are at the nation-state level, and therefore, unlimited resources, and game changing technologies could be leveraged to crash services, corrupt votes via insider threats, or devise methods to social engineer perceptions causing voter disenfranchisement. The controls must be appropriate to the risks.

RedPhone suggests that the FVAP determine what department within the federal government is responsible for determining threats associated with the voting process so that an appropriate risk assessment can be done based on known threats. FVAP should use formal risk analysis and cost-benefit analysis to help ensure their control environment is appropriate for their risk profile and tolerance. The risk analysis should include several key steps.

First, the FVAP should perform a formal risk analysis to determine the actual risk to the environment. The risk assessment should consider the value of the assets being protected, likelihood of probable threats and attack vectors, impact of a successful attack, inherent risk of the condition, existing safeguards, and the residual risk as compared to current tolerance.

Next, based on the results of the risk assessment, determine what areas of the voting process are operating at unacceptable levels of risk. Identify controls that can reduce the likelihood of the threat source or lessen the impact to acceptable levels. Perform a cost-benefit analysis to determine if the suggested controls provide an appropriate risk reduction benefit.

The next step should be to implement appropriate controls based on this analysis. Test the controls and likely attack scenarios to validate the controls operate properly and provide the desired effect. Employ monitoring, metrics and measures to ensure key controls continue to perform adequately and provide the expected protections. Continually update the risk assessment as new threats emerge, the business makes changes or other factors change that would affect the risk assessment results. The risk assessment should be updated at least annually to ensure it is still appropriate for the organization and the current environment.

It should be noted that this test had several limitations that would not exist in the “real world”, and therefore additional testing is highly recommended. Also, it should be noted that all testing is a “point-in-time-analysis”, and therefore should never be considered lasting. Testing should be performed with some regularity to maintain the highest level of security posture at all times.

Operational policies for high confidentiality, integrity and availability focus on setting and establishing processes, policies, and strict configuration and patch management. They are divided into the following categories:

- Service Level Management for High Availability
- Planning Capacity to Promote High Availability
- Change Management for High Availability
- Backup and Recovery Planning for High Availability
- Disaster Recovery Planning
- Planning Scheduled Outages
- Staff Training for High Availability
- Documentation as a Means of Maintaining High Availability
- Physical Security Policies and Procedures for High Availability

In addition to the above policies, a well defined and documented software development life-cycle should be adopted. The Capability Maturity Model Integration (CMMI) is a widely followed and adopted best practice that defines practices that include eliciting and managing requirements, decision making, measuring performance, planning work, handling risks, and more. None of the vendors' voting systems are being developed using such a defined life-cycle. We recommend that voting systems vendors adopt rigorous software engineering practices based on CMMI level-3 or better to ensure that system life-cycle, documentation, and methodologies are not random, but instead meet or exceed best practices.

The single greatest risk to Internet voting from an end-users computer is the fact that election officials do not have access to the voting workstation to determine its integrity, nor the upstream Internet supporting infrastructure. However, if a kiosk approach is employed, the election officials still have some control over the environment; it is recommended that the kiosk periodically send "status votes" or "test ballots" that test the integrity and accuracy of the voting system and the end-to-end transmission of the encrypted data. Control of the client-side voting computer, the local network, or upstream Internet Service Providers (ISP's) infrastructure will always present significant challenges to Internet based voting. Therefore, it is imperative that both end-points, and the lines of communication be as secure as possible to maintain the vote integrity, confidentiality, system availability and voter anonymity.

## Appendix - C Operation Orange

Jonathan Wright is a tall, handsome, slightly exotic looking Harvard grad, who has served in the U.S. Senate for 8 years. He has recently won the appointment as a candidate for the office of the President of the United States. He has the backing of the military and firefighters of America, as well as various police districts. However, unbeknownst to most of the American public is the fact that though he was born in the U.S., Senator Wright's grandfather, still resides in this fictional nation state.

Now, this nation state is very interested in the latest election because the incumbent president of the U.S. is considering a boycott of all CFS light bulbs, a major product for this nation state. For years they have been the only manufacturers of this product; however, the light bulbs often have defects that have caused severe injuries to American consumers-leading to a public outcry against the product. American and Mexican companies are now producing a superior, if more expensive, light bulb.

Because this issue is in the fore front of the American psyche, the incumbent president wants it to be one of the issues of his platform. A boycott of this product would be a devastating financial blow to their economy. This nation state requires a president sympathetic to their cause in the oval office.
Mr. Wright will champion the product over an American or Mexican one. Primarily, because Mr. Wright still has close family that resides in this nation state; and therefore, he should honor the family name as a proud descendant. This nation state government believes that Mr. Wright would want to support his family's home nation, and maintain their status has the premier supplier of CFS light bulbs. Therefore, this nation state is confident that they will be able to hack the American electronic voting systems to ensure Mr. Wright's election to the office of president.

### Specific Objectives:

Acting as hackers, your objective is to hack into the voting system, obtain administrator level rights and access, and *change* the votes so that Senator Wright becomes the next president of the United States. You must "recon" the targeted electronic voting system(s) and thoroughly plan your plan of attack employing sophisticated penetration techniques. If the changes are detected and an audit deems hacking has altered the targeted system(s), the election will merely be deemed void or corrupt and a new one will take place using old fashioned methods beyond the control of the nation state. Furthermore, you must do your best to cover

your “tracks” such that cyber security personnel will not be able to forensically trace the hack to your IP address.

You will have a limited amount of time to perform your reconnaissance of the vendor system(s), determine what tools to use, and ultimately penetrate the system(s) and make the needed changes to ensure the desired outcome. A denial of service attack would quickly be detected and traced, therefore this method of disruption should not be considered.

Keeping in mind that these penetration tests are intended to provide the following:

- Evaluate the protection of the Vendor’s electronic voting systems with a special emphasis on the effectiveness of logical access and system software security controls

- Provide value to the Vendor’s electronic voting system by identifying opportunities to significantly strengthen applicable controls within budgetary and operational constraints

  i.e., documented mitigation strategies, or security patches and/or procedures that improve the security posture of their respective systems.

- To facilitate timely, cost-effective completion of this project, Tiger Teams will make maximum practical use of the relevant work of others where possible (i.e., internal assessments by the auditee, internal and external audits, and vulnerability testing on covered IT assets).

- In order to optimize the effectiveness of the Penetration Test team members, the Vendor’s need to provide access to systems, services, and employees. To perform the work specified in this statement of work, the Tiger Teams will require the following from the customer:

  1. Access to relevant personnel including: technical support, data center personnel, application developers and end-users and functional experts.

  2. Relevant documentation including: System Administration Guides, System Architecture diagrams that include IP addresses of target systems. Previous security threat assessments if available.

  3. A primary point of contact for emergency remediation if needed.

  4. Coordination of events with customer team members.

5. Signed NDA, Authorization to Proceed, and the below Rules of Engagement.

## Appendix - D Tools

Information Gatheringbr
Assbr
DMitrybr
DNS-Ptrbr
dnswalkbr
dns-bruteforcebr
dnsenumbr
dnsmapbr
DNSPredictbr
Finger Googlebr
Firewalkbr
Goog Mail Enumbr
Google-searchbr
Goograpebr
Gooscanbr
Hostbr
Itracebr
Netenumbr
Netmaskbr
Piranabr
Protosbr
QGooglebr
Relay Scannerbr
SMTP-Vrfybr
TCtracebr
Network Mappingbr
Amap br
Assbr
Autoscan _Rbr
Fpingbr
Hpingbr
IKE-Scanbr
IKEProbebr
Netdiscoverbr
Nmapbr
NmapFEbr
Pfbr
PSK-Crackbr
Pingbr
Protosbr
Scanrandbr
SinFPbr
Umitbr
UnicornScanbr
UnicornScan pgsql e
module version br
Analysisbr br
Servicesbr
SNORTp
SIPcrackbr
XProbebr
PBNJ br
OutputPBNJbr
ScanPBNJbr
Genlistbr
Vulnerability Identificationbr
Absinthebr
Bedbr
CIRT Fuzzerbr
Checkpwdbr
Cisco Auditing Toolbr
Cisco Enable Bruteforcerbr
Cisco Global Exploiterbr
Cisco OCS Mass Scannerbr
Cisco Scannerbr
Cisco Torchbr
Curlbr
Fuzzer br
GFI LanGuard br
GetSidsbr
HTTP PUTbr
Halberdbr
Httprintbr
Httprint GUIbr
ISR-Formbr
Jbrofuzzbr
List-Urlsbr
Lynxbr
Merge Router Configbr
Metacoretexbr
Metoscanbr
Mezcal HTTPSbr
Mibble MIB Browserbr
Mistressbr
Niktobr
OATbr
Onesixtyonebr
OpenSSL-Scannerbr
Paros Proxybr
Peachbr
RPCDumpbr
RevHostsbr
SMB Bruteforcerbr
SMB Clientbr
SMB Serverscanbr
SMB-NATbr
SMBdumpusersbr
SMBgetserverinfobr
SNMP Scannerbr
SNMP Walkbr
SQL Injectbr
SQL Scannerbr
SQLLibfbr
SQLbrutebr
Sidguessbr
SmbKbr
Snmpcheckbr
Snmp Enumbr
Spikebr
Stompybr
SuperScanbr
TNScmdbr
Taofbr
VNC_bypauthbr
Wapitibr
Yersiniabr
sqlanlzbr
sqldictbr
sqldumploginsbr
sqlquerybr
sqluploadbr
Penetrationbr
Framework-MsfCbr
Framework-MsfUpdatebr
Framework-Msfclibr
Framework-Msfwebbr
Init Pgsql (autopwn)br
Milwrm Archivebr
MsfClibr
MsfConsolebr
MsfUpdatebr
OpenSSL-To-Openbr
Update Milwrmbr
Privilege Escalationbr
Ascend attackerbr
CDP Spooferbr
Cisco Enable Bruteforcerbr
Crunch Dictgenbr
DHCPX Flooderbr
DNSspoofbr
Driftnetbr
Dsniffbr
Etherapebr
EtterCapbr
FileCablebr
HSRP Spooferbr
Hash Collisionbr
Httpcapturebr
Hydrabr
Hydra GTKbr
ICMP Redirectbr
ICMPushbr
IGRP Spooferbr
IRDP Responderbr
IRDP Spooferbr
Johnbr
Lodowepbr
Mailsnarfbr
Medusabr
Msgsnarfbr
Nemesis Spooferbr
NetSedbr
Netenumbr
Netmaskbr
Ntopbr
PHossbr
PackETHbr
Rcrackbr
SIPdumpbr
SMB Snifferbr
Singbr
TFTP-Brutebr
THC PPTPbr
TcPickbr
URLsnarfbr
VNCrackbr
WebCrackbr
Wiresharkbr
Wireshark Wifibr
WyDbr
XSpybr
chntpwbr
Maintaining Accessbr
proxybr
Backdoorsbr
CryptCatbr
HttpTunnel Clientbr
HttpTunnel Serverbr
ICMPTXbr
Iodinebr
NSTXbr
Privoxybr
ProxyTunnelbr
Rinetdbr
TinyProxybr
sbdbr
socatbr
Covering Tracksbr
Housekeepingbr
Radio Network
AFragbr
ASLeapbr
Air Crackbr
Air Decapbr
Air
Replaybr
Airmon Scriptbr
Airpwnbr
AirSnarfbr
Airodumpbr
Hexdumpbr
Airoscriptbr
Airsnortbr
CowPattybr
FakeAPbr
GenKeysbr
Genpmkbr
Hotspotterbr
Karmabr
Kismetbr
Load IPWbr
Load acxbr
MDKbr
MDK for Broadcombr
MacChangerbr
Unload Driversbr
Wep_crackbr
Wep_decryptbr
WifiTapbr
Wicrawlbr
Wlassistantbr
Bluetoothbr
Bluebuggerbr
Blueprintbr
Bluesnarferbr
Btscannerbr
Carwhispererbr
CuteCombr
Ghettotoothbr
HCIDumpbr
Ussp-Pushbr
OllyDBGbr
PcapSipDumpbr
PcapToSip_RTPbr
SIPSakbr
Hexeditbr
SIPdumpbr
SIPpbr
Smapbr
Digital Forensicsbr
Allinbr
Autopsybr
DCFLDDbr
DD_Rescuebr
Foremostbr
Magicrescuebr
Mboxgrepbr
Memfetchbr
Memfetch Findbr
Pascobr
Rootkithunterbr

Sleuthkitbr Vinettobr Reverse Engineeringbr GDB GNU Debuggerbr GDB Console GUIbr
GDB Serverbr GNU DDDbr VOIP & Telephony Analysisbr

Because of some last minute corrections to the ROE/MNDA/ATS documentation, we requested email confirmation of the acceptance. Those e-mail acceptances are below:

From Vendor-2.com
to Jay Aceto <jay.aceto@redphonecorporation.com>
date Sat, Jul 30, 2011 at 9:35 PM
subject RE: Error found. Please resign ROE's & Authorizations to Scan ASAP
Important mainly because it was sent directly to you.

Jay,

On behalf of Vendor-2 I accept the changed documents. I will bring signed copies Monday.

Vice President
Vendor-2

from @Vendor-3.com
to Jay Aceto <jay.aceto@redphonecorporation.com>
date Mon, Aug 1, 2011 at 9:51 AM
subject RE: Error found. Please resign ROE's & Authorizations to Scan ASAP
mailed-by Vendor-3.com

Jay,

I accept the corrections on behalf of Vendor-3.

Vendor-3

From: Vendor-1.com>
To: "Jay Aceto (jay.aceto@redphonecorporation.com)"
<jay.aceto@redphonecorporation.com>
Date: Fri, 22 Jul 2011 15:58:37 -0700
Subject: Student Forms

Hi Jay,

Attached are our authorization signatures and Rules of Engagements for the students…

Vendor-1, Inc.