tracesecurity

6300 Corporate Blvd. Suite 200
Baton Rouge, LA 70809
Phone: 225-612-2122
www.tracesecurity.com

Web/Mobile Application Security Testing

Organization Name

Date

TraceSecurity

tracesecurity

Table of Contents

tracesecurity

List of Figures

tracesecurity

Document Information

Company	Organization Name
Document Title	Web/Mobile Application Test
Date	Date
Classification	Confidential
Document Type	Private

Document Recipients		
Name	Title	Organization
Client Contact Name	Client Contact Title	Organization Name
Client Contact Name	Client Contact Title	Organization Name

Document History		
Date	Author	Comments
Date	TraceSecurity	Final Document

tracesecurity

Project Definition

Definition

A Web Application Security Test identifies exploitable vulnerabilities that may impact the confidentiality, integrity, or availability of the organization's information. When appropriate to prove or demonstrate a vulnerability, the testing emulates an attack by an external entity attempting to breach the system from the outside.

Objective

To provide an analysis of the current security posture of the public web server and hosted applications implemented at ORGANIZATION NAME. This is achieved by predefining a goal (e.g. circumvent existing application behavior to obtain non-public data, modify data, or deny access to information or services) and reporting back as to whether that goal was achieved, documenting the exact process and providing proof of obtaining the defined goal.

Testing Methodology

- Security Control Testing: During testing, system and user information will be used to test and validate the security controls protecting the application, its data, and the infrastructure supporting it. Example attack scenarios in this phase include, but are not limited to: SQL injection, session spoofing, buffer overflows, application or system configuration problems, DNA attacks, address spoofing, share access, and exploitation of inherent system trust relationships. The following major areas are addressed during a web application test:
    - Authentication and access control
    - Use of cryptography
    - Session handling
    - Security of network communications
    - Server configuration and related controls
    - Application Data Storage
    - Injection vulnerabilities
    - Input handling
    - Side channel attacks and information leakage
    - Network traffic analysis
- Results Analysis: After initial testing is performed and preliminary data is generated all information will be reviewed to determine if further analysis is necessary, or if the exploit should be further tested to discover or uncover additional information.
- Final Analysis and Documentation: TraceSecurity will provide a comprehensive report outlining all discovered findings and the methods by which the information was obtained. The report will provide recommendations for correcting or mitigating any vulnerabilities found. An offsite conference call can be scheduled and performed to ensure that all information is clearly understood and that a course of action/remediation is identified.

Executive Summary

Executive Summary of Web Application Testing

ORGANIZATION NAME contracted with TraceSecurity to conduct application security testing for the organization's online applications providing marketing and member financial services. The TraceSecurity Information Security Analyst (ISA) tested a variety of real-world techniques designed to subvert the security controls applied to these applications, intended to ascertain the website's security posture.

The assessment began with mapping and vulnerability analysis of the HTML interfaces hosted on a public-facing server using proprietary and open-sourced tools. The ISA reviewed these scans to determine if any vulnerabilities could be an attack vector on the application or its host. This scan data was combined with manual testing to determine the extent any vulnerability. The ISA used these results to initiate manual testing. Manual testing included, but was not limited to: testing of application interfaces for input verification and susceptibility to code injection or buffer overflows.

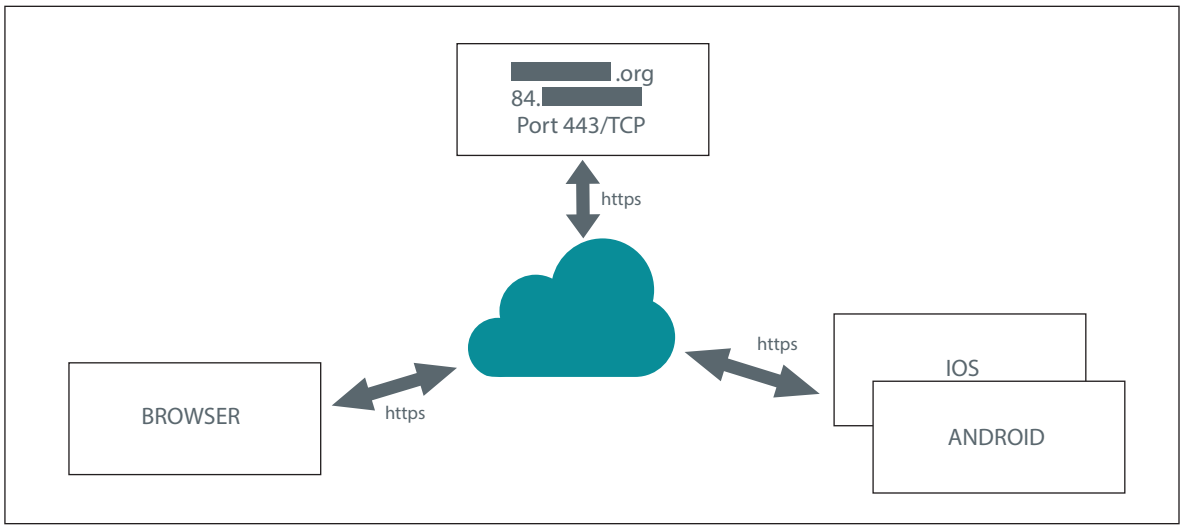
The target of evaluation was the web application interface and the mobile applications compiled for both iOS and Android mobile operating systems. A production copy of each was downloaded from the vetted app stores and installed on an iOS and Android device for testing. The client provided a testing account to provide access to all application functionality. No live member information was employed during the testing.

The ISA determined that the majority of prescribed controls were properly configured to reduce the risk of malicious exploitation or inadvertent errors. However, some areas for improvement related to session encryption and session control were identified. Recommendations are provided to update or reconfigure these services.

The web application interface host included in this assessment was https:// [redacted] ([redacted].org).

The body of this report describes details from testing, as well as an appendix presenting additional figures from the testing process.

Figure 1: Topology Overview



tracesecurity

Software Flaw Review

During this phase, the ISA conducts testing of a standard checklist of common security controls designed to defend against the Open Web Application Security Project (OWASP) Top 10 security vulnerabilities or flaws. The following graph/table indicates what type of flaws were tested and if any issues were discovered during the testing of each control group. All control domains were tested during the course of the assessment.

Name	Title	Organization
A1: Injection Vulnerabilities	No findings	
A2: Broken Authentication and Session Management	Two (2) findings	SSL cookie without secure flag set; SSL cookie without HTTPOnly flag set
A3: Cross-Site Scripting	No findings	
A4: Insecure Direct Object Access	No findings	
A5: Security Misconfiguration	One (1) finding	Slow HTTP Post Vulnerability
A6: Sensitive Data Exposure	Two (2) findings	Weak encryption protocol enabled; Strict Transport Security not enforced
A7: Missing Function Level Access Control	No findings	
A8: Cross-Site Request Forgery	No findings	
A9: Using Components with Known Vulnerabilities	No findings	
A10: Un-Validated Redirects and Forwards	No findings	

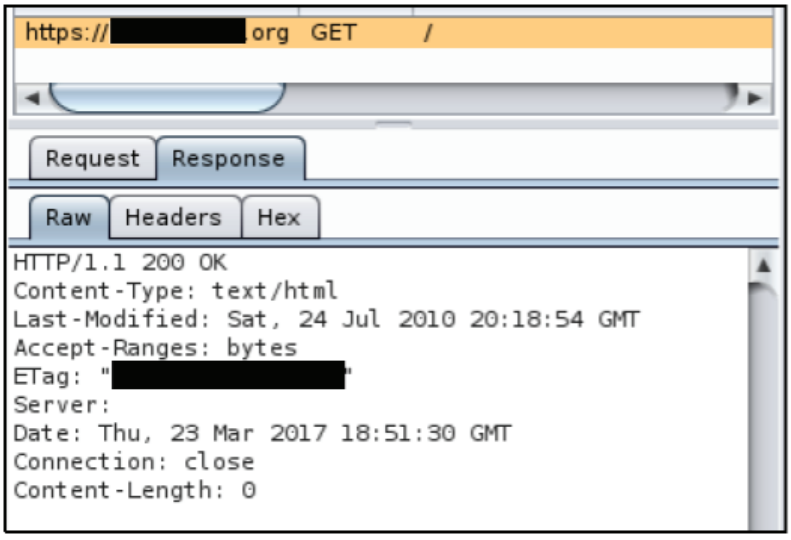
Findings

This section presents any findings resulting from testing. Each is presented with supporting documentation, along with a recommendation. TraceSecurity recommends reviewing each of the findings presented and document remediation plans in cases where the level of risk is unacceptable.

Strict Transport Security Not Enforced - Risk Level: Low

The application fails to prevent users from connecting to it over unencrypted connections. An attacker able to modify a legitimate user's network traffic could bypass the application's use of SSL/TLS encryption, and use that application as a platform for attacks against its users. This attack is performed by rewriting HTTPS links as HTTP, so that if a targeted user follows a link to the site from an HTTP page, his or her browser never attempts to use an encryption connection. The SSL strip tool automates this process.

Figure 2: Response Header Omits HTTP Strict Transport Security (HSTS)



Recommendation

TraceSecurity recommends configuring the application to instruct web browsers to only access the application when using HTTPS. To do this, enable HTTP Strict Transport Security (HSTS) by adding a response header with the name 'Strict-Transport-Security' and the value 'max-age=expireTime', where 'expireTime' is the time in seconds that browsers should remember that the site should only be accessed using HTTPS.

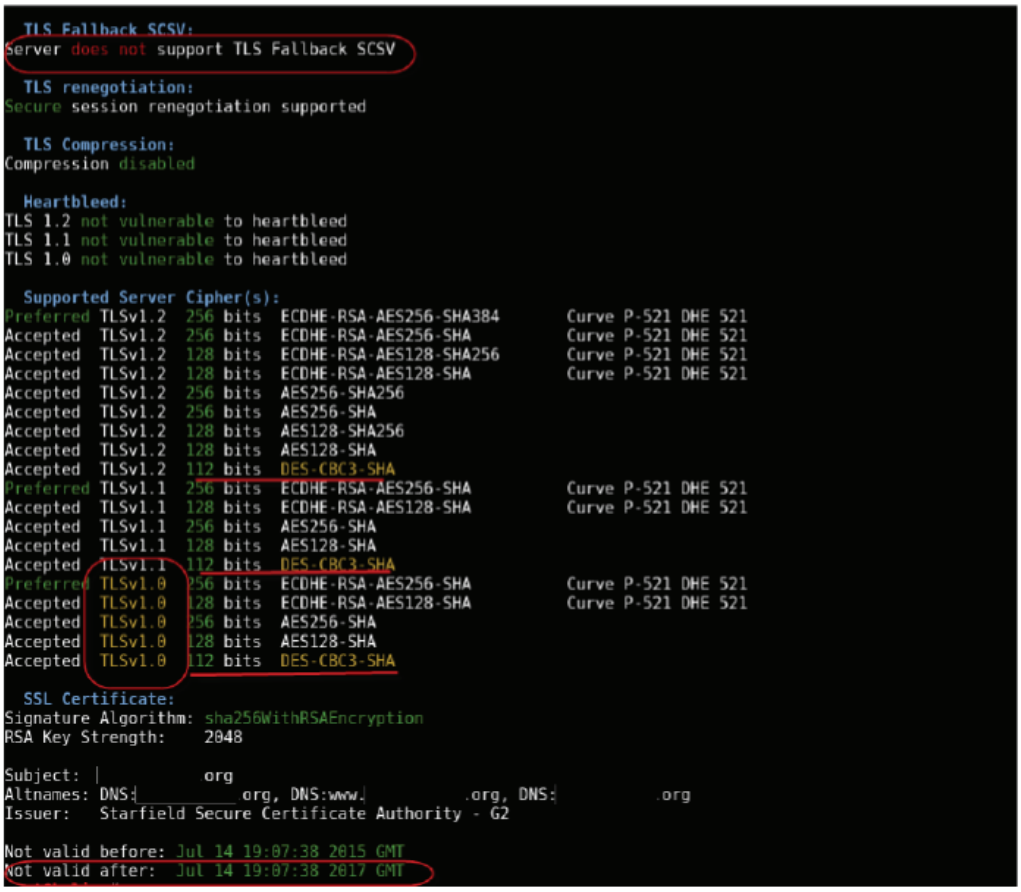
Note that, because HSTS is a "trust on first use" (TOFU) protocol, a user who has never accessed the application will never have seen the HSTS header, and will therefore still be vulnerable to SSL stripping attacks. To mitigate this risk, TraceSecurity recommends adding the optional ;preload' flag to the HSTS header, and submitting the domain for review by browser vendors

Weak Encryption Protocol/Ciphers - Risk Level: Low

The HTTP host at 84. [redacted] allows HTTPS encryption employing TLSv1.0. The protocol is used with the expectation of passing traffic over the public Internet between a client and a web server through a secure channel. But with certain weaknesses within the protocol, secure channels could be compromised with attacks, such as the man-in-the-middle attack. TLSv1.0 is vulnerable to the POODLE (Padding Oracle on Downgraded Legacy Encryption) attack for network systems. This method takes advantage of the protocol suite's ability to switch to less secure versions to maintain interoperability with older systems. The ISA noted that the server also supported ciphers less than 128 bits, increasing the risk of a successful decryption attack.

Through a man-in-the-middle attack, an attacking machine can intercept and modify the "Hello" packet, which contains a list of supported protocols and cipher suites, and modify the list to the vulnerable protocol. The list would then be presented to the server and be forced to accept a vulnerable protocol.

Figure 3: Testing for Supported Encryption Protocols on Host 67. [redacted]



Recommendation

TraceSecurity recommends supporting only secure connection protocols, such as TLS 1.2 or TLS 1.1 with strong or excellent ciphers, particularly those providing at least 128 bits of key space.

Slow HTTP Post Vulnerability - Risk Level: Low

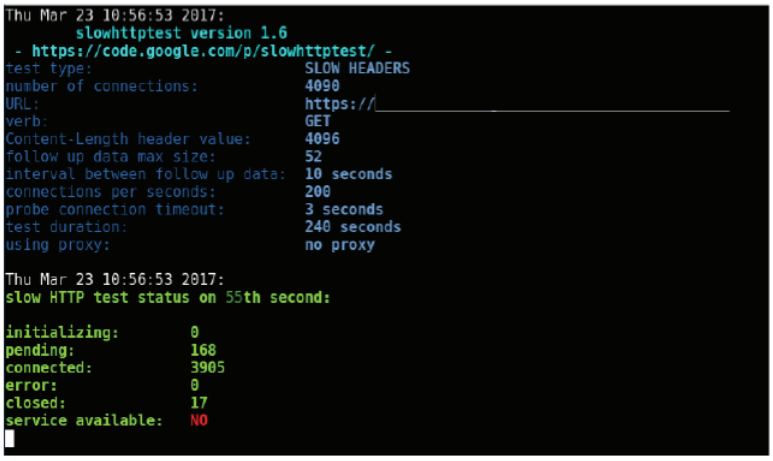
Based on automated testing, the page at [redacted].org is vulnerable to a "Slow HTTP POST" Denial of Service (DoS) attack. This is an application level DoS that consumes server resources by maintaining open connections for an extended period of time by slowly sending traffic to the server. In these cases the connections were manipulated to stay open for 123 seconds by delaying completion of a request.

If the server maintains too many connections open at once, then it may not be able to respond to new, legitimate connections. Unlike bandwidth-consumption DoS attacks, the "slow" attack does not require a large amount of traffic to be sent to the server, only that the client can maintain open connections for several minutes at a time. The attack holds server connections open by sending properly crafted HTTP POST headers that contain a Content-Length header with a large value to inform the web server how much of data to expect. After the HTTP POST headers are fully sent, the HTTP POST message body is sent at slow speeds to prolong the completion of the connection and lock up server resources. By waiting for the complete request body, the server is helping clients with slow or intermittent connections to complete requests, but is also exposing itself to abuse.

To confirm the vulnerability, the ISA briefly submitted requests to confirm that upon ~3900 connections, that service was temporarily unavailable. The test only lasted for a couple of seconds to avoid any disruption.

Figure 4: Slow Post Detected at [redacted].org

Figure 5: Controlled Test of Slow Post Response of [redacted].org



Recommendation

While solutions are server-specific, TraceSecurity recommends limiting the size of the acceptable requests to each form and establishing minimal acceptable speed rates and timeouts.

SSL Cookies Without Security Flags - Risk Level: Low

The ISA confirmed that LivePersonID cookies did not include an enabled "secure" attribute or "HTTPOnly" attribute. This was confirmed for the following cookie at the indicated URL:

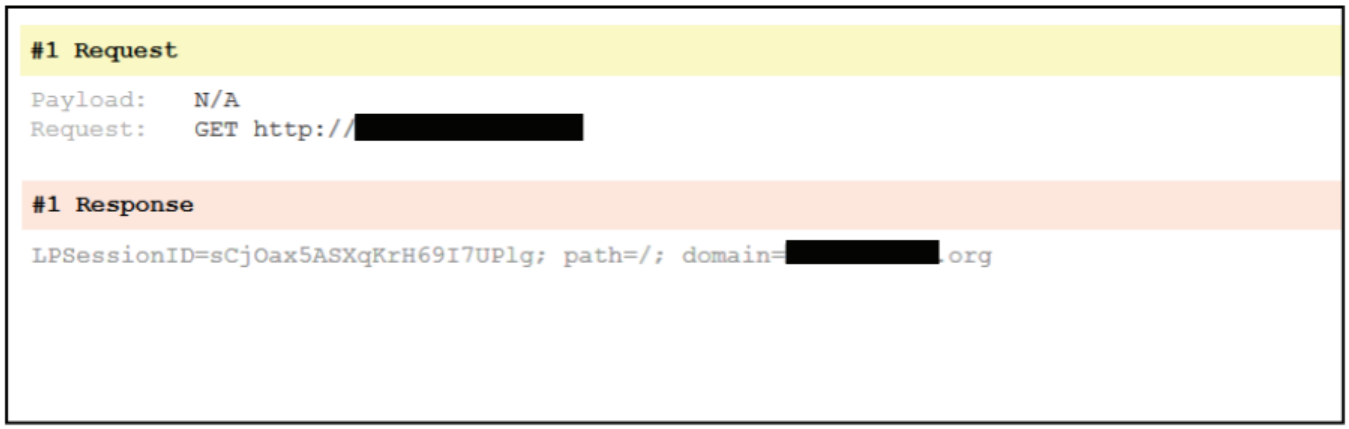
LivePersonID=LP i=xxxxxx, d=xxxxxx; path=/; domain=[redacted].org

If the secure flag is set on a cookie, then browsers will not submit it in any requests that use an unencrypted HTTP connection, preventing it from being trivially intercepted by an attacker monitoring network traffic. If the secure flag is not set, then the cookie will be transmitted in cleartext if the user visits any HTTP URLs within the cookie's scope.

An attacker could induce this event by feeding a user suitable links, either directly or through another website. Cookies without the "HTTPOnly" attribute are permitted to be accessed through JavaScript. Cross-site scripting attacks can steal cookies, which could lead to user impersonation or compromise of the application account.

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

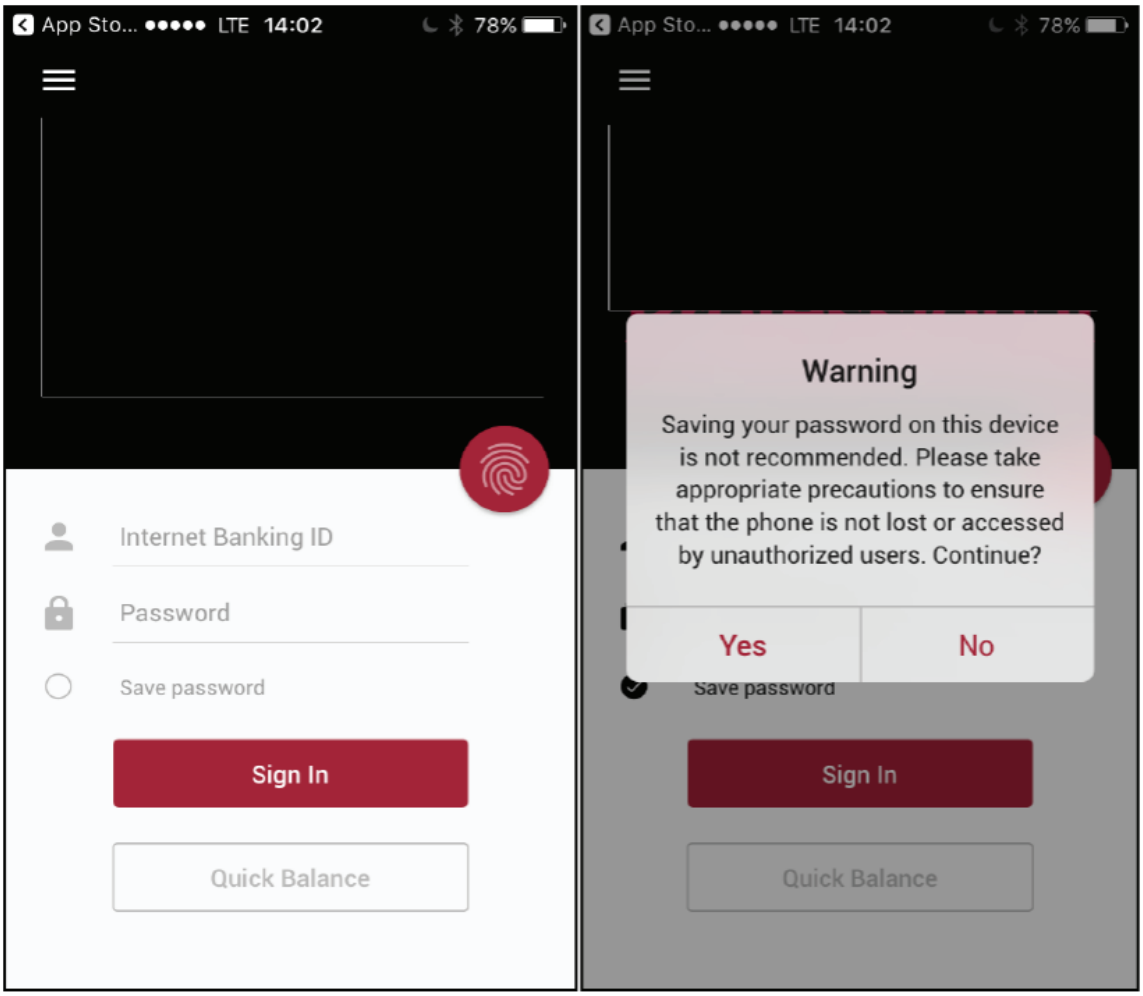
Figure 6: Cookies Without an Enabled Secure Attribute or HTTPOnly Attribute



Remember Mobile Password Feature - Risk Level: Informational

The ISA confirmed that the mobile application provided a user with the option of saving his or her online banking password on the device. While a warning acknowledging that this is not recommended was provided, it may encourage users to save these sensitive credentials on their devices. In these cases, the application cannot enforce the use of device authentication or physical security. If a device is lost, stolen, or misused, a perpetrator may gain access to the member's financial information. TraceSecurity does not recommend saving financial application credentials on mobile devices. TraceSecurity recommends omitting this feature in future upgrades.

Figure 7: Mobile Application Option to Save Password (with Warning)



Appendix - Additional Testing Information

The additional testing section is used to demonstrate some of the testing that was performed during the testing process. No findings are contained within this section of the report, rather they are included to show what types of testing was performed. It should be noted that, in consideration of report length, not every test with screenshot has been included.

Figure A1: Qualys Summaries After Web Application Scans

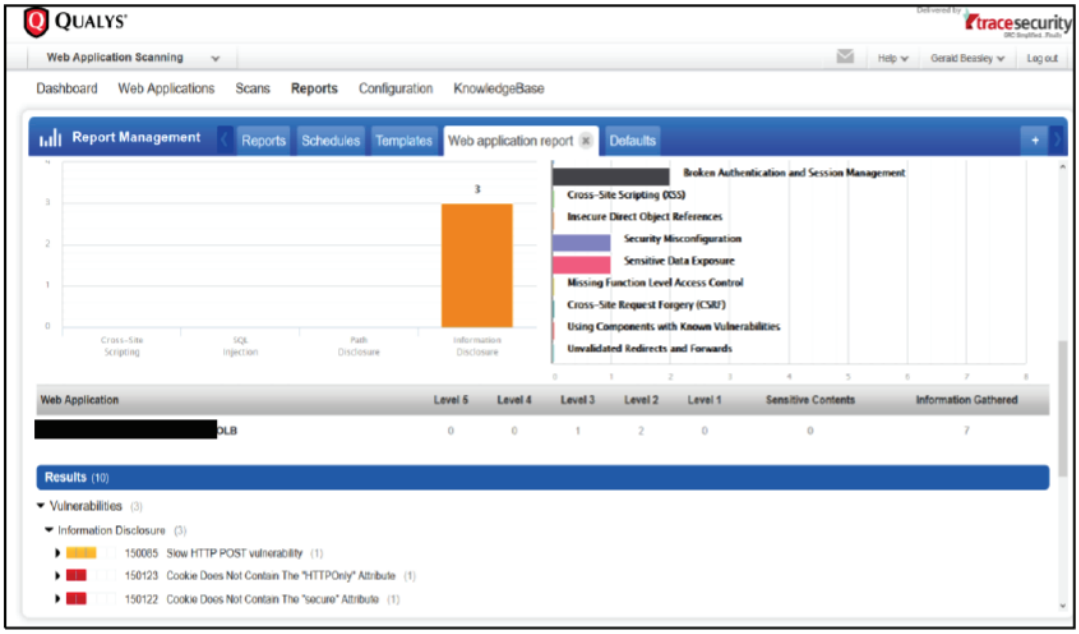


Figure A2: Sample Testing for Code Injection Vulnerability

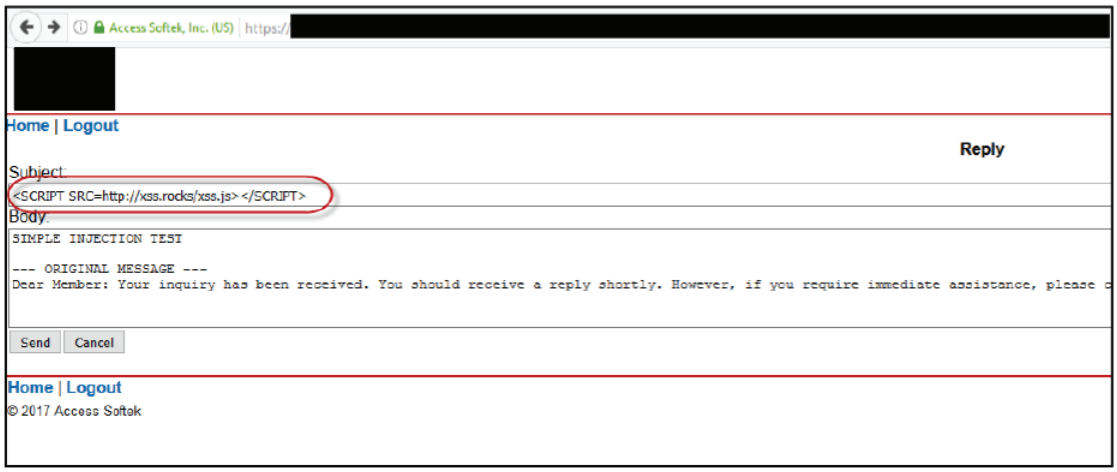


Figure A3: Unhandled Error on Code Injection Test

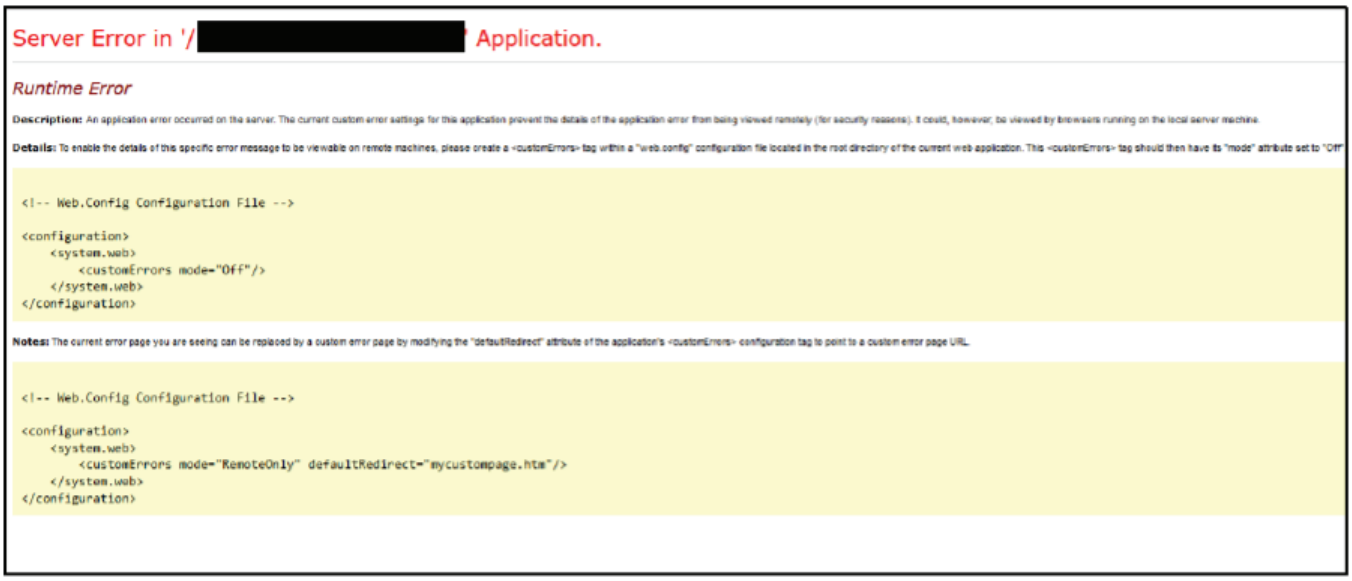


Figure A4: Verifying Input Validation (Character Type Enforced)

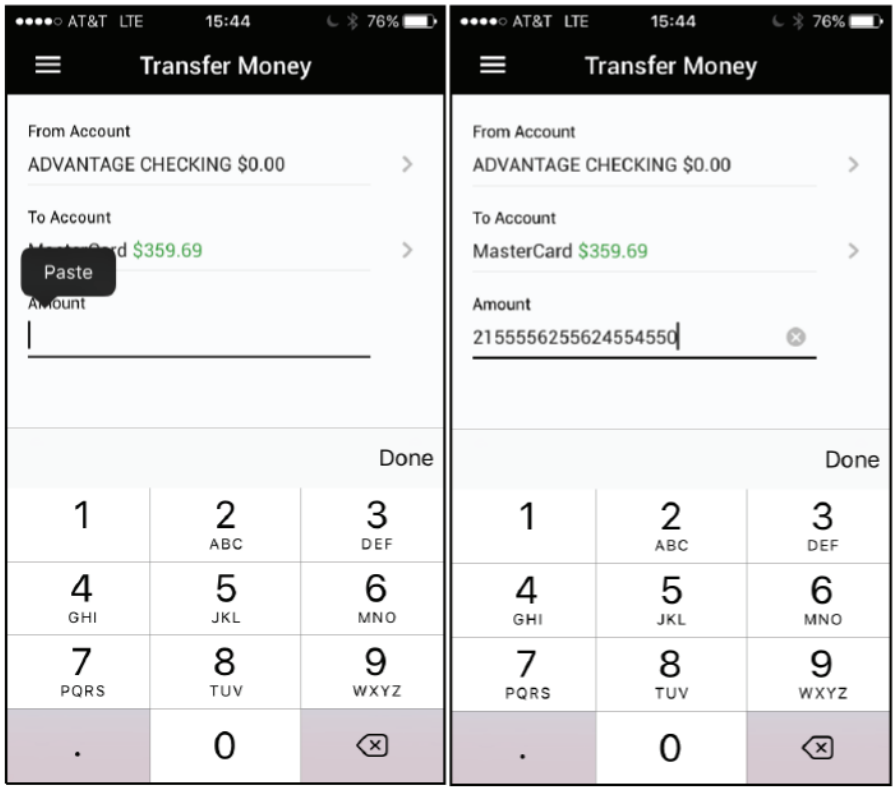


Figure A5: Sample Verifying Mobile Application Encryption (Verified HTTPS Session)

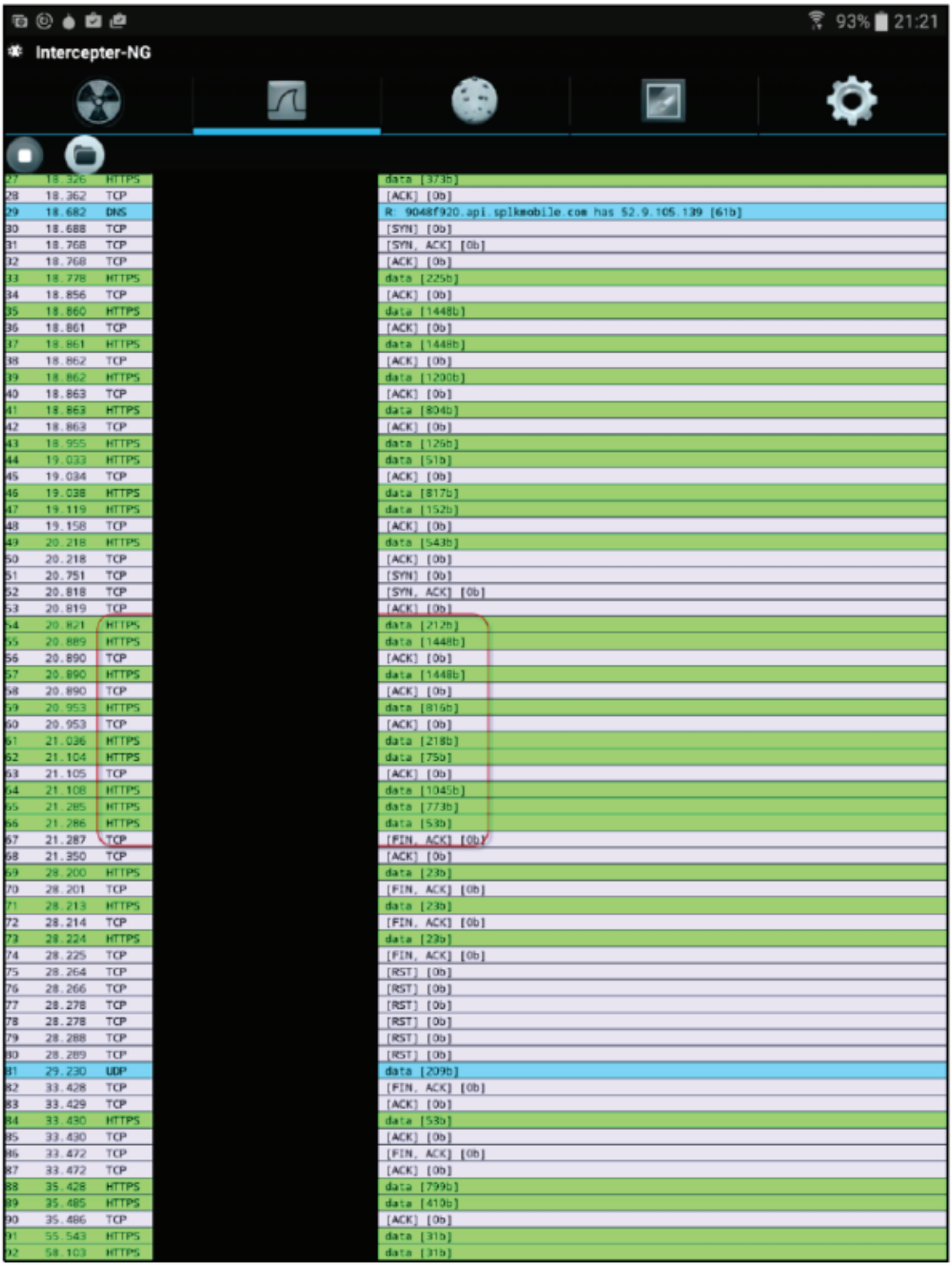


Figure A6: Mapping Web Application

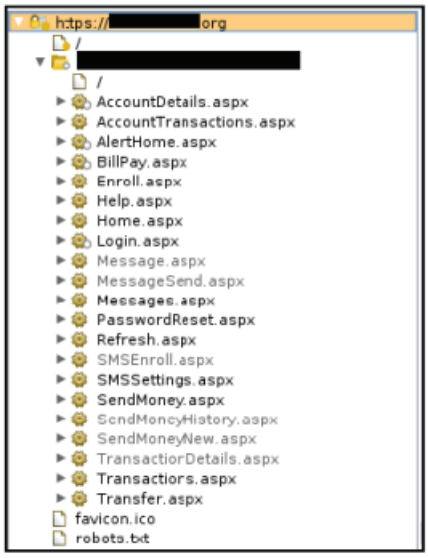


Figure A7: Testing Deposit Capture Feature on Mobile Application (Did Not Detect Persistent Storage)

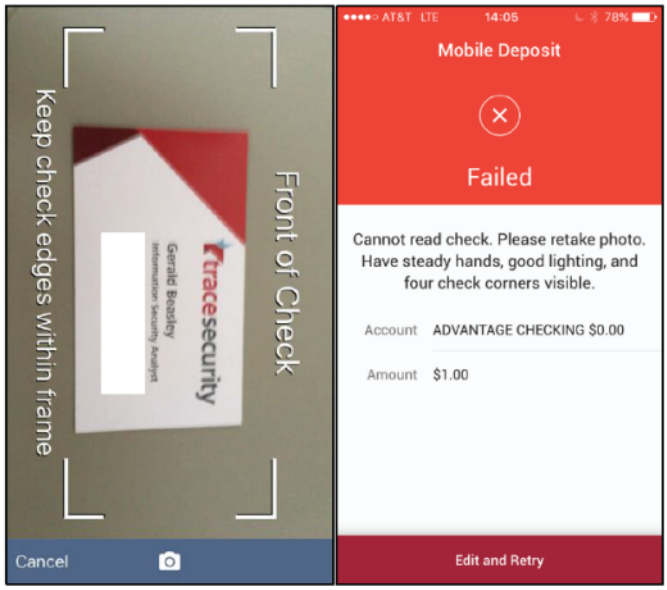


Figure A8: Confirming Autocomplete Off for Web-Based Password Field

Figure A9: Conducting Code Scan of Android Application

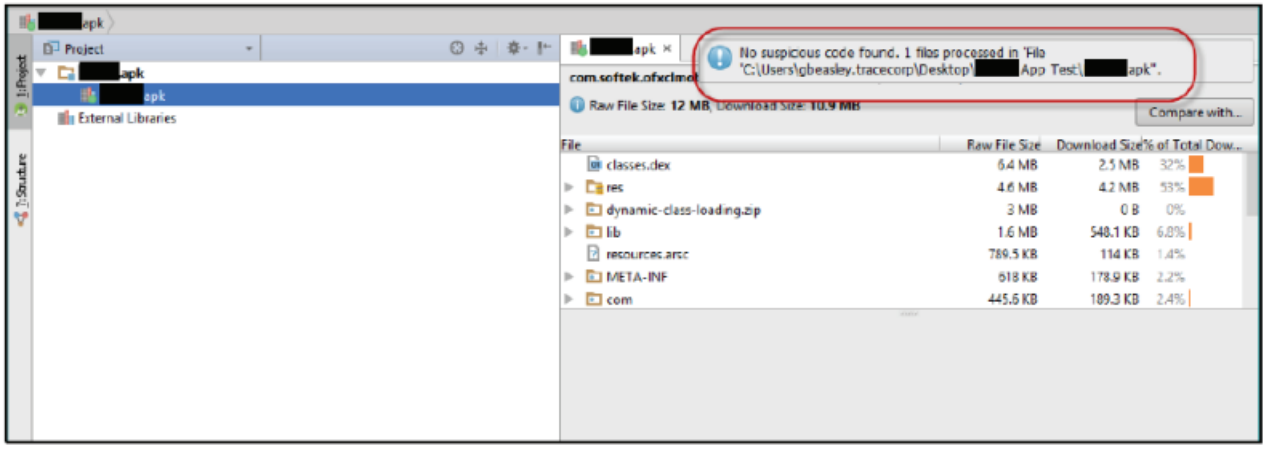


Figure A10: Examining Android Manifest for Functionality