Security First Umbrella

Source Code Review

Prepared for:

Prepared by:

Damian Profancik — Security Engineer

Table of Contents

1 Executive Summary

Application Summary	
Application Name	Umbrella
Application Version	1.0
Application Type	Android Mobile Application
Platform	Android / Java

Engagement Summary	
Dates	June 15, 2015 – June 26, 2015
Consultants Engaged	1
Total Engagement Effort	2 person weeks
Engagement Type	Source Code Review
Testing Methodology	White Box

Vulnerability Summary	
Total High severity issues	0
Total Medium severity issues	0
Total Low severity issues	3
Total Informational severity issues	7

Total vulnerabilities identified: 10

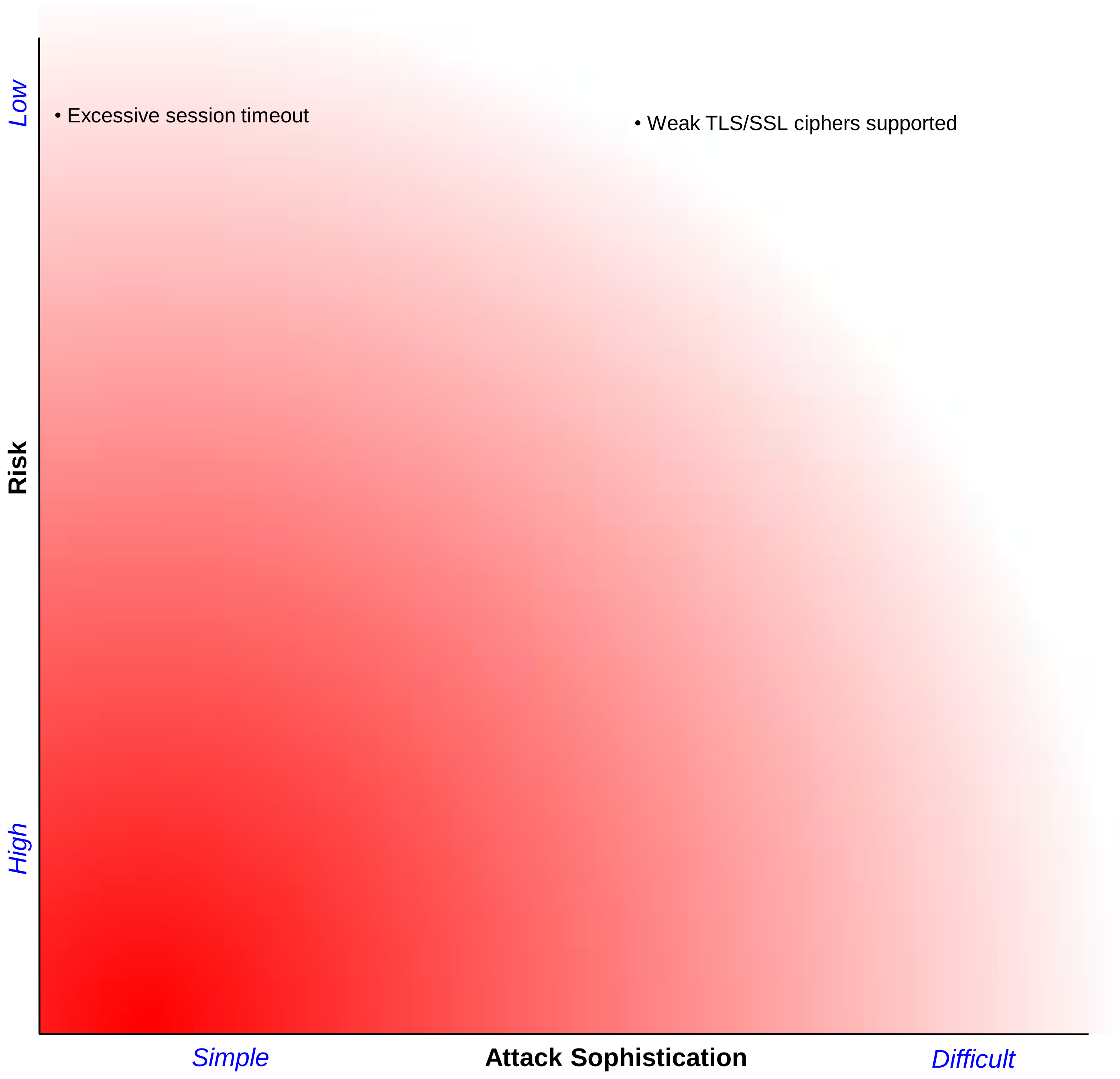
See section 3.1 on page 10 for descriptions of these classifications.

Category Breakdown:

Access Controls	1	■
Auditing and Logging	0	
Authentication	0	
Configuration	2	■■
Cryptography	2	■■
Data Exposure	1	■
Data Validation	2	■■
Denial of Service	0	
Error Reporting	1	■
Patching	0	
Session Management	1	■
Timing	0	

1.1 iSEC Risk Summary

The iSEC Partners Risk Summary chart evaluates vulnerabilities according to risk. The impact of the vulnerability increases towards the bottom of the chart. The sophistication required for an attacker to find and exploit the flaw decreases towards the left of the chart. The closer a vulnerability is to the chart origin, the greater the risk.



1.2 Project Summary

Open Technology Fund (OTF) engaged iSEC Partners (iSEC) to perform a source code assisted white box security assessment of Security First's Umbrella mobile application. One iSEC consultant performed the engagement remotely over two weeks, from June 15th, 2015 to June 26th, 2015. Security First provided iSEC access to the mobile application and the application source code during this time.

iSEC's primary focus during the assessment was on the mobile application, but also included a review of the backend API transport security. The assessment covered mobile application security best practices along with vulnerabilities that could put Umbrella users at risk. iSEC also reviewed the attack surface available prior to authentication or authorization specifically for injection flaws and Denial of Service (DoS) potential, among other classes of vulnerabilities.

The Security First team provided support and was engaged throughout the project.

1.3 Findings Summary

Due to the application's mostly read-only nature, the issues identified in this report only minimally impact the security posture of both the application, and the mobile device on which it is installed. While SQL Injection and Cross-Site Scripting vulnerabilities are both present in the application, they are only able to be triggered locally, and exploitation does not achieve anything significant because virtually no user-private data is stored in the application, nor is there any notion of an 'account' for an attacker to compromise.

Although the reported issues present a fairly low risk at this time, iSEC did highlight a number of areas, (particularly around verbose error messages, SQL Injection, Cross Site Scripting, and hardcoded encryption keys issues), in which Security First could make improvements to its security posture. Additionally, iSEC made a number of general "best practice" suggestions that Security First may choose to incorporate to further improve security posture of their application, including providing or requiring a lock mechanism for the application.

1.4 Recommendations Summary

The Umbrella application should perform more rigorous input validation, tightening of access controls, and shoring up the SSL/TLS configuration of the backend API server. Additionally, iSEC recommends hardening of the application attack surface with the below noted best practices.

Short Term

Short term recommendations are meant to be relatively easy actions to execute, such as configuration changes or file deletions that resolve security vulnerabilities. These may also include more difficult actions that should be taken immediately to resolve high-risk vulnerabilities. This area is a summary of short term recommendations; additional recommendations can be found in the vulnerabilities section.

Switch to using parameterized SQL statements. Although as currently designed, the application is not realistically vulnerable to SQL Injection, if the design of the application is changed in the future, this could enable an attacker remote SQL Injection through malformed Intents, or bypass local trust boundaries. Best practice is to use parameterized queries as a standard development practice.

Break out operationally sensitive secrets into configuration files. Depending on the secrets involved and nature of the application, the configuration files may be managed separately from the source code, to allow some degree of operational separation of the secrets from other team members or the public.

Disable debug logging in production releases. Debug information discloses sensitive internal information about the application. Do not directly log exception stack traces, especially in production releases.

Long Term

Long term recommendations are more complex and systematic changes that should be taken to secure the system. These may include significant changes to the architecture or code and may therefore require in-depth planning, complex testing, significant development time, or changes to the user experience that require retraining.

Consider proxying data retrieved from third parties. When news is retrieved from third parties that may have weak or nonexistent TLS deployment, the user's activity leaks to the network and may be modified in transit (or outright blocked). Consider proxying this data through the `api.secfirst.org` domain. While access to this domain could also be blocked, retrieving this data on the server should allow a more trustworthy connection than a user's potentially hostile mobile or wireless network. It also allows Umbrella to pin the TLS certificate to `api.secfirst.org`.

Enable session timeouts based on user inactivity. Set a reasonable expiration time limit with shorter windows available to be chosen by users with an increased risk profile. This will reduce the time an attacker will be able to retain access to the application should the device become compromised.

Perform regular checks using a TLS testing tool. Best practices in TLS server configuration have changed rapidly over the past few years. Use a tool such as SSLyze[1] or `https://www.ssllabs.com/` to regularly check that Security First's servers are maintaining best practices.

[1]`https://github.com/nabla-c0d3/sslyze`

2 Engagement Structure

2.1 Internal and External Teams

The iSEC team has the following primary members:

- Damian Profancik — Security Engineer
  dprofancik@isecpartners.com
- Tom Ritter — Account Manager
  tritter@isecpartners.com

The Security First team has the following primary members:

- Rory Byrne — Security First
  rory@secfirst.org
- Rok Biderman — Security First
  rok@secfirst.org

2.2 Project Goals and Scope

The goal of this engagement was to discover vulnerabilities in the application's source code that an attacker could leverage to violate the privacy and anonymity of Security First Umbrella users.

By reviewing the source code, this included:

- Looking for injection style vulnerabilities
- Looking for issues with authorization and authentication
- Looking for information disclosure points
- Reviewing mobile application best practices
- Reviewing the transport security of backend services

3 Detailed Findings

3.1 Classifications

The following section describes the classes, severities, and exploitation difficulty rating assigned to each issue that iSEC identified.

Vulnerability Classes

Class	**Description**
Access Controls	Related to authorization of users, and assessment of rights
Auditing and Logging	Related to auditing of actions, or logging of problems
Authentication	Related to the identification of users
Configuration	Related to security configurations of servers, devices, or software
Cryptography	Related to mathematical protections for data
Data Exposure	Related to unintended exposure of sensitive information
Data Validation	Related to improper reliance on the structure or values of data
Denial of Service	Related to causing system failure
Error Reporting	Related to the reporting of error conditions in a secure fashion
Patching	Related to keeping software up to date
Session Management	Related to the identification of authenticated users
Timing	Related to the race conditions, locking, or order of operations

Severity Categories

Severity	**Description**
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or Defense in Depth
Undetermined	The extent of the risk was not determined during this engagement
Low	The risk is relatively small, or is not a risk the customer has indicated is important
Medium	Individual user's information is at risk, exploitation would be bad for client's reputation, of moderate financial impact, possible legal implications for client
High	Large numbers of users, very bad for client's reputation or serious legal implications.

Difficulty Levels

Difficulty	Description
Undetermined	The difficulty of exploit was not determined during this engagement
Low	Commonly exploited, public tools exist or can be scripted that exploit this flaw
Medium	Attackers must write an exploit, or need an in depth knowledge of a complex system
High	The attacker must have privileged insider access to the system, may need to know extremely complex technical details or must discover other weaknesses in order to exploit this issue

3.2 Vulnerabilities

The following table is a summary of vulnerabilities identified by iSEC. Subsequent pages of this report detail each of the vulnerabilities, along with short and long term remediation advice.

Vulnerability	Class	Severity
1. Insufficient TLS/SSL certificate validation	Cryptography	Low
2. Excessive session timeout	Session Management	Low
3. Weak TLS/SSL ciphers supported	Configuration	Low
4. SQL Injection in search and password functionality	Data Validation	Informational
5. SSLv3 enabled and vulnerable to POODLE attack	Configuration	Informational
6. Hard-coded encryption key in source code	Data Exposure	Informational
7. Verbose debug error messages logged	Error Reporting	Informational
8. Reflected Cross-Site Scripting in search functionality	Data Validation	Informational
9. Certificate pinning not implemented in mobile application	Cryptography	Informational
10. Application does not lock when focus is lost	Access Controls	Informational

3.3 Detailed Vulnerability List

1. Insufficient TLS/SSL certificate validation

Class: Cryptography **Severity:** Low **Difficulty:** Low

Finding ID: iSEC-UMBRELLA-3

Targets: The `getTolerantClient` function of the `UmbrellaRestClient` class (app/src/main/java/org/secfirst/umbrella/util/UmbrellaRestClient.java; line 29).

Description: The Security First Umbrella application does not sufficiently validate the web server's SSL certificate. Additionally, the code uses the `X509HostnameVerifier` API, which is deprecated as of API level 22.[2] `X509HostnameVerifier` should be replaced with the `openConnection()`[3] function instead. The `openConnection()` functionality in the Android framework provides automatic SSL certificate validation.

This lack of certificate validation was done intentionally to make the application tolerant of SSL issues, allowing opportunistic security when retrieving data from third parties. While the Umbrella application relies on external content and cannot ensure the validity of the SSL certificates presented by third parties, the application should still warn the user if it encounters an invalid certificate. The user can then decide whether or not to accept that risk for each site.

```
25      public static AsyncHttpClient getTolerantClient() {
26          AsyncHttpClient client = new AsyncHttpClient(true, 80, 443);
27          SSLSocketFactory sslSocketFactory = (SSLSocketFactory) client.getHttpClient()
                .getConnectionManager().getSchemeRegistry().getScheme("https")
28                  .getSocketFactory();
29          final X509HostnameVerifier delegate = sslSocketFactory.getHostnameVerifier();
30          if(!(delegate instanceof WildCardSSLVerifier)) {
31              sslSocketFactory.setHostnameVerifier(new WildCardSSLVerifier(delegate));
32          }
33          return client;
34      }
```

Listing 1: Vulnerable code from `UmbrellaRestClient.java`, enabling bypass of certificate errors

For more information regarding Android security with SSL, please refer to `https://developer.android.com/training/articles/security-ssl.html`.

Exploit Scenario: A local network attacker performs a MitM attack while sharing a public network with a user using Security First Umbrella. Due to the lack of SSL certificate validation, the user is not aware that the attacker is intercepting their traffic. The attacker is then able to view the user's traffic and obtain the user's notification interests.

Short Term Solution: Warn the user if a third-party site presents an invalid SSL/TLS certificate and let the user decide whether or not to accept the risk.

Long Term Solution: In addition to warning the user of invalid SSL certificates, let them add exceptions per site that would trust a presented certificate until it changes.

[2] `http://developer.android.com/reference/org/apache/http/conn/ssl/X509HostnameVerifier.html`
[3] `http://developer.android.com/reference/java/net/URL.html#openConnection()`

2. Excessive session timeout

Class: Session Management	**Severity:** Low	**Difficulty:** High

Finding ID: iSEC-UMBRELLA-4

Targets: The Umbrella application session expiration times.

Description: The Umbrella application appears to have an excessively long or nonexistent session timeout. If an attacker were to steal a user's device, they would have an excessively long time window in which to perform malicious actions and gather information.

Exploit Scenario: A user leaves their device unattended at a coffee shop while looking at recent news alerts from the United Nations. An attacker who is watching the user steals the device and has full access to the user's security information as well as custom checklist items they had added.

Short Term Solution: The application should be configured to expire the user's session after a period of inactivity. While the application could allow the user to configure their desired timeout, it may also be appropriate to match it to the phone lockscreen, or half of that value.

Long Term Solution: Internal guidelines should dictate that all applications use proper session timeout values.

3. Weak TLS/SSL ciphers supported

Class: Configuration **Severity:** Low **Difficulty:** High

Finding ID: iSEC-UMBRELLA-5

Targets: The API server at `https://api.secfirst.org`

Description: The TLS and SSL protocols support various encryption ciphers. Some of the available ciphers have been shown to have weaknesses that may expose sensitive information to an attack in a Man-in-the-Middle (MitM) scenario. The SSL and TLS protocols configured at `https://api.secfirst.org` supports the following weak ciphers.

- `ECDHE-RSA-RC4-SHA` – weak encryption algorithm (RC4)
- `RC4-SHA` – weak encryption algorithm (RC4)

Exploit Scenario: An attacker gains man-in-the-middle (MitM) access on a public network where a user is using Security First Umbrella. The attacker then leverages weaknesses in the encryption algorithm or cipher configuration to expose the unencrypted session traffic and gain access to the user's news feeds.

Short Term Solution: Disable SSL/TLS ciphers with known weaknesses or of insufficient key strengths (less than 128 bits). Since there are no known safe SSL protocols, the application should negotiate to the strongest TLS cipher available.

Long Term Solution: Perform routine scanning of the application to ensure that only secure cipher suites and key strengths are used. This scanning can be performed with SSLyze,[4] originally developed by iSEC Partners, or `https://www.ssllabs.com/`. Modern TLS/SSL configurations can be found on the Mozilla Wiki.[5]

[4] `https://github.com/nabla-c0d3/sslyze`
[5] `https://wiki.mozilla.org/Security/Server_Side_TLS#Modern_compatibility`

4. SQL Injection in search and password functionality

Class: Data Validation **Severity:** Informational

Finding ID: iSEC-UMBRELLA-1

Targets: SQL Injection points exist in the following locations:

- app/src/main/java/org/secfirst/umbrella/SearchActivity.java; line 50
- app/src/main/java/org/secfirst/umbrella/util/Global.java; line 143

Description: The Umbrella application is vulnerable to SQL Injection (SQLi) in the locations noted above. SQLi occurs when user-supplied input is used to construct SQL queries via string concatenation, and ordinarily allows attackers the ability to retrieve data from a database they are not authorized to view, or inject malicious values. See the OWASP SQLi guide[6] for additional information.

As the application is currently designed, this vulnerability is classified as Informational, as these locations are only reachable if an attacker has physical access to the device and performs the SQL Injection manually. In these instances, the attacker is able to retrieve the same data or operate on the database in the same way without using the injection vulnerability.

```
44  String query = intent.getStringExtra(SearchManager.QUERY);
45  if (query!=null) {
46    List<Segment> mSegments = null;
47    try {
48      QueryBuilder<Segment, String> queryBuilder = global.getDaoSegment().
            queryBuilder();
49      Where<Segment, String> where = queryBuilder.where();
50      where.like(Segment.FIELD_BODY, "%"+query+"%");
```

Listing 2: Vulnerable code from `SearchActivity.java`

```
138  String pw = pwInput.getText().toString();
139  String checkError = UmbrellaUtil.checkPasswordStrength(pw);
140  if (!pw.equals(confirmInput.getText().toString())) {
141    Toast.makeText(activity, "Passwords do not match.", Toast.LENGTH_LONG).show();
142  } else if (checkError.equals("")) {
143    getOrmHelper().getWritableDatabase(getOrmHelper().getPassword()).rawExecSQL("PRAGMA
         rekey = '" + pw + "';");
```

Listing 3: Vulnerable code from `Global.java`

Short Term Solution: Switch to using a parameterized SQL statement. More information for preventing SQLi vulnerabilities can be found in the OWASP SQLi Prevention Cheatsheet.[7]

Long Term Solution: Update all database operation functions to use parameterized SQL statements (with constant query and variable parameters). This is an industry best practice in defending against SQLi vulnerabilities.

[6]https://www.owasp.org/index.php/SQL_Injection
[7]https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

5. SSLv3 enabled and vulnerable to POODLE attack

Class: Configuration **Severity:** Informational

Finding ID: iSEC-UMBRELLA-2

Targets: The API server at `https://api.secfirst.org`

Description: The Umbrella application is vulnerable to the Padding Oracle On Downgraded Legacy Encryption (POODLE) attack. First disclosed in September 2014, this attack is based on a Padding Oracle flaw in some block cipher SSL/TLS protocols.[8] The POODLE vulnerability requires an attacker to be able to run arbitrary code via a vulnerability such as Cross-Site Scripting. Once this is achieved, they are able to perform a Padding Oracle attack on the SSLv3 protocol as implemented on the Security First Umbrella platform. This allows an attacker to decrypt the cipher text and steal sensitive information. Additional information can be found at the following links:

- `https://www.openssl.org/~bodo/ssl-poodle.pdf`
- `https://www.us-cert.gov/ncas/alerts/TA14-290A`

This vulnerability is classified as Informational - POODLE allows an attacker to extract private data from a SSL/TLS session, most commonly a cookie. In its current design Umbrella does not use cookies or other user-sensitive data in its connections, thus there is nothing significant to steal. An attacker could conceivably use POODLE to learn the type of data the user is accessing (which news source), but this is more easily done using the Cross-Site Scripting vulnerability necessary to exploit POODLE, or metadata analysis on the encrypted stream.

Short Term Solution: Disable SSLv3 to mitigate this vulnerability.

Long Term Solution: Perform routine scanning of the application to ensure that SSL/TLS implementations are not vulnerable to POODLE. This scanning can be performed with SSLyze,[9] originally developed by iSEC Partners, or `https://www.ssllabs.com/`.

[8] Although POODLE primarily affects SSL, some TLS implementations have been shown to be vulnerable as well.
[9] `https://github.com/nabla-c0d3/sslyze`

6. Hard-coded encryption key in source code

Class: Data Exposure **Severity:** Informational

Finding ID: iSEC-UMBRELLA-6

Targets: The `DATABASE_PASSWORD` public variable of the `OrmHelper` class (app/src/main/java/org/secfirst/umbrella/util/OrmHelper.java; line 22).

Description: iSEC discovered that the default encryption key for the database is hard-coded in source code. Depending on the sensitivity of the data that is being protected, the application should not rely on a default encryption key, but either suggest or require that an encryption key be setup for the user during the product tour. If such a per-user encryption key is not used, and the key hardcoded in the application, there is effectively no encryption.

```
public static final String DATABASE_PASSWORD = "_OMITTED_";
```

Listing 4: Default database password in source code of `OrmHelper.java`

In many cases, best practice is to separate secrets from source code through configuration files that are deployed separately; however, as this is an open-source project and the secrets will still be visible, this is less significant.

Short Term Solution: Consider suggesting or requiring that the user create a encryption key during the tour to improve the security of sensitive data in transit. Requiring such a key would remove the need for a default encryption key.

Long Term Solution: Implement guidelines against persisting operationally sensitive or security relevant secrets in source code. Best practice for all applications, is to break out such operationally sensitive secrets into configuration files. Factor this configuration into your existing system configuration management strategy. However, since this is an open source application, storing default passwords anywhere presents a risk.

7. Verbose debug error messages logged

Class: Error Reporting **Severity:** Informational

Finding ID: iSEC-UMBRELLA-7

Targets: Error messages generated throughout the application

Description: The Security First Umbrella application outputs verbose error messages with full stack traces when an exception is caught. These error messages contain detailed information, such as code paths and line numbers, that can aid an attacker in exploiting other vulnerabilities or otherwise expose information to other applications on the phone. The following code snippet from `MainActivity` demonstrates an example of this:

```
  } catch (SQLException e) {
    e.printStackTrace();
```

Listing 5: Verbose error message example: `MainActivity.java`

As a best practice, the `printStackTrace()` function should not be called directly. The application should use the `android.util.Log` class functionality instead. Additionally, debug logging should be stripped from the production release of the application.

Short Term Solution: Switch to using the `android.util.Log` class functionality for error logging. Additionally, verbose error messages should be disabled in the production release of the application. This can be done with ProGuard[10, 11] to automatically remove any type of logging message desired in production builds of the application.

Long Term Solution: Develop a policy to strip verbose logging from all production application releases.

[10] http://proguard.sourceforge.net/
[11] http://stackoverflow.com/questions/12390466/

8. Reflected Cross-Site Scripting in search functionality

Class: Data Validation **Severity:** Informational

Finding ID: iSEC-UMBRELLA-8

Targets: Cross-Site Scripting points exist in the following locations:

- app/src/main/java/org/secfirst/umbrella/adapters/SearchAdapter.java; line 66
- app/src/main/java/org/secfirst/umbrella/util/Global.java; line 102

Description: The Umbrella application is vulnerable to Cross-Site Scripting (XSS) attacks. XSS is a class of vulnerability related to application input validation and output encoding. In reflected XSS, the application accepts input from an end user and immediately displays it in the response without properly encoding metacharacters. This allows an attacker to inject JavaScript code into the resulting page. This can cause the application to perform unintended actions such as the exfiltration of private data, (session information, application data, or login credentials). For additional information, please see the OWASP XSS guide.[12]

iSEC has marked the severity of this vulnerability as Informational, as the attack can only be performed against the attacker themselves and cannot be exploited to harm to another user (This is often known as 'Self-XSS'). Additionally, this vulnerability requires that the XSS payload already exist in the database.

See the vulnerable code snippets below:

```
65    public void onBindViewHolder(ViewHolder holder, int position) {
66      holder.mSearchText.setText(Html.fromHtml("result while searching for: <b>" +
            mQueries.get(0)+"</b>"));
```

Listing 6: Vulnerable code from `SearchAdapter.java`

```
94    private String searchBody(String body, String query) {
95      String lower = body.toLowerCase(Locale.UK);
96      int start = lower.indexOf(query.toLowerCase(Locale.UK));
97      int offset = 80;
98      int from = (start-offset>0) ? start-offset : 0;
99      int to = (start+query.length()+offset>body.length()) ? body.length() : start+
            query.length()+offset;
100     String returnBody = "..."+body.substring(from, to)+"...";
101
102     returnBody = returnBody.replaceAll("(?i)"+query, "<b>"+query+"</b>");
```

Listing 7: Vulnerable code from `SearchAdapter.java`

Short Term Solution: Since the search term is being included in HTML, the search term should be HTML-entity encoded when included in the output. All user-supplied input should be encoded upon output in a context sensitive nature. Perform input validation on all user input supplied to the application. Input validation should reject input that does not match a whitelist of characters that are acceptable for the particular input, and it might also restrict inputs to a particular size or format. See OWASP XSS Prevention Cheatsheet[13] for more information.

Long Term Solution: Similar to SQL Injection, consider investigating parameterized templates or other more robust solutions to building HTML content before displaying it. These techniques can eliminate Cross-Site Scripting vulnerabilities before they have an opportunity to be written - as opposed to having to remember to escape every single output location in a context-sensitive manner.

[12] https://www.owasp.org/index.php/Cross-site_Scripting_%28XSS%29
[13] https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

9. Certificate pinning not implemented in mobile application

Class: Cryptography **Severity:** Informational

Finding ID: iSEC-UMBRELLA-9

Targets: The Umbrella mobile application.

Description: The Umbrella application does not verify that the certificate presented by the server matches the known, valid certificate for `https://api.secfirst.org`. While it is understood that the application gathers data from sites that Security First does not control, it could pin the certificate of the API server that it does control. Not doing so allows for interception and modification of traffic via custom certificates installed onto the mobile device, or through the use of certificates generated by a compromised Certificate Authority.

Exploit Scenario: An attacker that has compromised a Certificate Authority or that has temporary physical access to a mobile device surreptitiously installs a malicious certificate on the device in order to monitor application traffic going to/from the server. They use this to inject advertising or inappropriate security recommendations, or alter the news feeds from third parties.

Short Term Solution: Implement and utilize certificate pinning for all data transmitted between the mobile device and the server.

Long Term Solution: Update secure development guidelines to recommend the use of certificate pinning when developing mobile applications.

10. Application does not lock when focus is lost

Class: Access Controls **Severity:** Informational

Finding ID: iSEC-UMBRELLA-10

Targets: The Umbrella mobile application.

Description: The Umbrella application does not log the user out when the application focus is lost. This allows an attacker an additional opportunity to access the application should the device become stolen.

Exploit Scenario: An attacker with temporary physical access to a mobile device opens the Umbrella application that the user has been previously using. They are then able to access the data without providing authentication details.

Long Term Solution: While it is understood that this is the current design decision, requiring that the user reauthenticate to the application when focus is lost could be reconsidered as functionality is added.

As other hardening recommendations related to this issue, consider blacking out the screen when focus is lost to prevent a screenshot on the multitasking screen. Additionally, the application should check as to whether a lock is presently set on the phone itself, and if not, recommend the user do so to improve security.