Open Technology Fund

Mailvelope Firefox Extension

Application Penetration Test

iSECpartners part of nccgroup

Prepared for:

OPEN TECHNOLOGY FUND

Prepared by:

Mark Manning – Senior Security Engineer

Cassie Park – Security Engineer

Table of Contents

1 Executive Summary

OPEN TECHNOLOGY FUND

Application Summary	
Application Name	Mailvelope
Application Version	0.8.1
Application Type	Browser Extension
Platform	Firefox

Engagement Summary	
Dates	April 28 – May 9, 2014
Consultants Engaged	Two
Total Engagement Time	Four person weeks
Engagement Type	Application Penetration Test
Testing Methodology	White Box

Vulnerability Summary	
Total High severity issues	2
Total Medium severity issues	6
Total Low severity issues	2
Total vulnerabilities identified	10

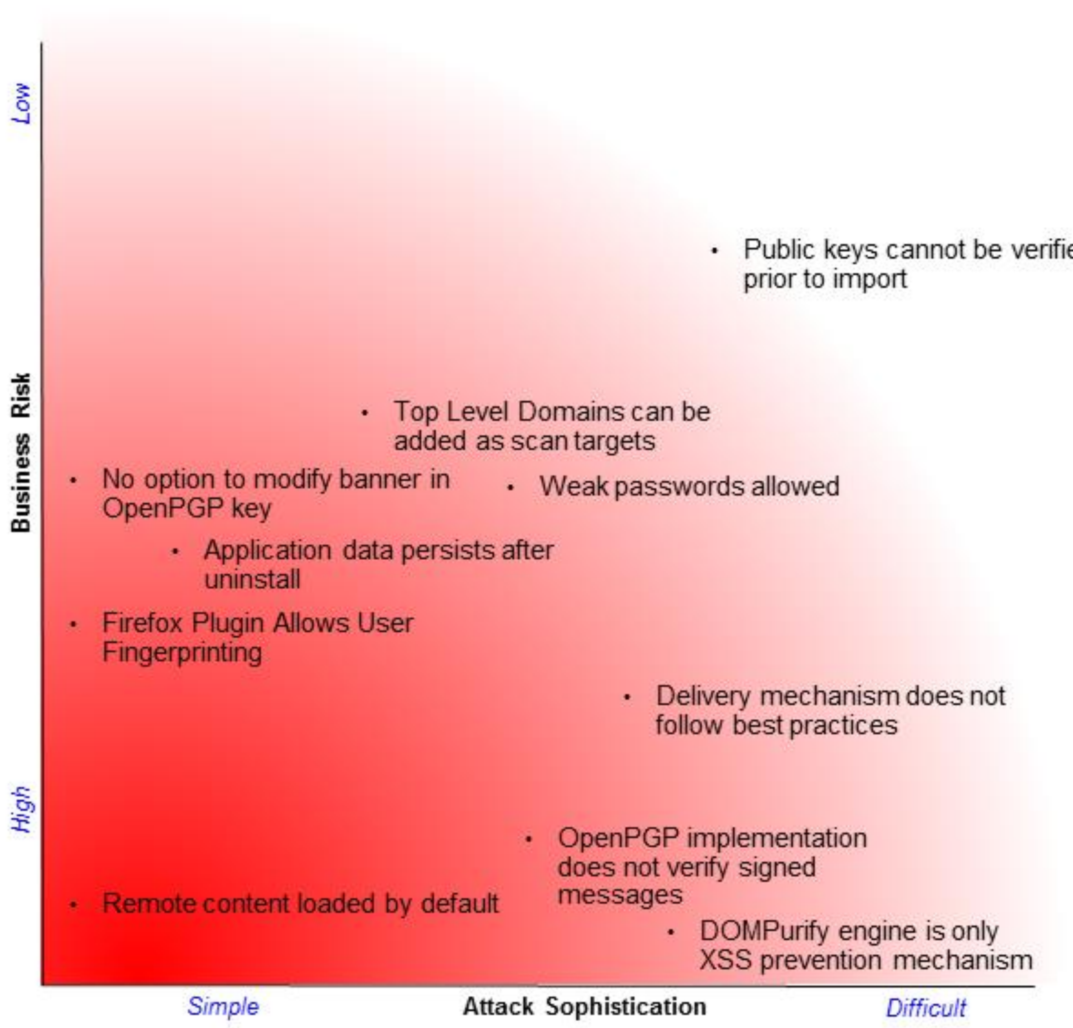
See Section 3.1 for descriptions of these classifications.

Category Breakdown:

Authentication	1 ■
Access Controls	0
Auditing and Logging	1 ■
Configuration	3 ■■■
Cryptography	0
Data Exposure	3 ■■■
Data Validation	3 ■■■
Denial of Service	0
Error Reporting	0
Patching	0
Session Management	0
Timing	0

1.1 iSEC Risk Summary

The iSEC Partners Threat Matrix chart evaluates discovered vulnerabilities according to business 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 business risk.



1.2 Project Summary

The Open Technology Fund[1] (OTF) engaged iSEC Partners to perform a source code assisted review of the Mailvelope Firefox Add-on starting April 28th. The purpose of the engagement was to review the latest release of the extension for common vulnerabilities and to offer suggestions for improvements. The assessment lasted over a two week period during which consultants reviewed version 0.8.1 of the release. Most of the engagement was focused on this version, but during the testing period the 0.8.2 version was released and small portions of its functionality were reviewed, including the decryption function and the use of DOMPurify. The third party library, OpenPGP.js, was not reviewed during this assessment.

iSEC identified the plug-in's threat model, key features, and areas that should be emphasized. Among those target areas were the encryption/decryption functions, public key import functions, and watch list controls. The HTML5 sandboxing initiatives were reviewed as well as the communication functions that transferred data from the standard browser environment to the internal resources of the Mailvelope add-on.

1.3 Findings Summary

The vulnerabilities identified in this report reflect that the "DOMPurify" and "wysiHTML5" filtering libraries for data validation sufficiently minimize the attack surface. However, a significant amount of trust is placed in them, which leads iSEC to be nervous about the possibility of future cross-site scripting attacks. So while these mechanisms minimize the attack surface, iSEC has made recommendations that provide defense-in-depth in the case that DOMPurify or the HTML5 sandbox could eventually be evaded.

As discussed below, further validating the source of postMessage events, standardizing the parseHTML functions used throughout the extension, and considering whitelisting data received even from trusted sources makes the extension less reliant on third party libraries, the cross-origin policy and HTML5 sandbox.

Besides these recommendations, iSEC also identified a few vulnerabilities relating to the loading of remote resources and parsing top level domain suffixes.

1.4 Recommendations Summary

While remediating individual vulnerabilities is expected, it is also important to take a broad perspective to identify root causes of these issues. Based on the vulnerabilities identified in this report, iSEC believes that the following recommendations are in order:

Short Term

Short term recommendations are meant to be relatively easily executed actions, 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.

[1] https://www.opentechfund.org

Perform additional input validation on data coming from the browser environment. While the extension uses postMessage() communications to send data to and from the HTML5 sandbox and code execution is controlled by a cross-origin policy, these features should not be overly relied upon compared to standard input validation functions. iSEC recommends performing further validation of the source of the messages as well as the format of the data. This would help defend against some types of attacks that could, in the future, evade the HTML5 sandbox. It may be possible to standardize the libraries used for input and output filtering. For more information on input validation, see Appendix B.

Implement missing OpenPGP features. Some features such as signature validation are not included in the latest release of Mailvelope. This is an important feature that is planned to be implemented. Without this feature there is no way of validating the authenticity of a signed message, a core aspect of security provided by PGP.

Block remote content by default. Currently, Mailvelope allows content in the form of images, video, and audio to be loaded from arbitrary remote domains. This enables the use of web beacons that indicate when a user has decrypted a message, and can lead to private key confirmation attacks.

Improve UI to help users maintain security. User interface improvements such as allowing users to validate a public key prior to it being imported would help give them control over their OpenPGP environment. Another example is to allow a user to control the header used for each encrypted message. The UI could be further improved to give users the tools necessary to maintain their OpenPGP keyring in a secure way. Possible improvements include the ability to change the key passphrase, implement and modify the private key expiry date, push public keys to a keyserver, and revoke keys.

Enforce common security restrictions on users. The Mailvelope security model is one in which the extension facilitates a secure environment to send and receive encrypted OpenPGP messages. Beyond that, enforcing security best practices helps to ensure that users do not make mistakes when generating keys, choosing passwords, and sending unencrypted content. Further restrictions could be added that force users to follow standard password complexity requirements (as documented in finding 7 on page 20).

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.

Implement development guidelines. Although Mailvelope currently have a single developer managing it, implementing common practices for an SDLC would ensure that proper security standards are followed in the future. Some suggestions include:

- Pre-release checklist
- Input validation library standards
- Unit testing

Better defense against fingerprinting. Future versions of Mailvelope should better defend against fingerprint attacks. Adversaries should not be able to identify that the Mailvelope add-on is installed at all, or disclose what version the user is running. Users should have the ability to customize what information is being disclosed in the OpenPGP header so not to include which versions of the software are used.

2 Engagement Structure

2.1 Internal and External Teams

The iSEC team has the following primary members:

- Mark Manning – iSEC Technical Lead
  mark.manning@intrepidusgroup.com
- Cassie Park – iSEC Security Consultant
  cassie@isecpartners.com
- Tom Ritter – iSEC Account Manager
  tritter@isecpartners.com
- Dana Bost – iSEC Project Manager
  dbost@isecpartners.com

The Mailvelope team has the following primary members:

- Thomas Oberndörfer – Mailvelope
  info@mailvelope.com

2.2 Project Goals and Scope

The goal of this engagement was to review the Firefox extension and corresponding source code for possible vulnerabilities such as code injection, unauthorized configuration changes, UI redressing attacks, and control mechanism evasion. This included attempts to:

- Inject malicious JavaScript and other objects into the decryption process
- Evade the HTML filtering library
- Use the automatic public key import feature to inject malicious content
- Compare the DOM of the environment with Mailvelope extension enabled, with that of one with Mailvelope disabled
- Review the content saved in the extension's storage folder
- Inject information into the watch list, including inappropriate characters
- Build malicious OpenPGP keys and test how they are used for encryption and import

This assessment focused primarily on version 0.8.1 of the release which was the latest at the beginning of testing. A new version 0.8.2 was also released during the testing period, however there was not enough time to assess the newer version fully.

3 Detailed Findings

3.1 Classifications

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

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 protecting the privacy or integrity of 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 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, moderate financial impact, possible legal implications for client
High	Large numbers of users, very bad for client's reputation, or serious legal or financial 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 Vulnerability Overview

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

Vulnerability	Class	Severity
1. Remote content loaded via img, audio, video and other tags	Data Exposure	High
2. DOMPurify engine is the only mechanism to stop Cross Site Scripting	Data Validation	High
3. Application data persists after uninstall	Auditing and Logging	Medium
4. OpenPGP implementation does not verify signed messages	Configuration	Medium
5. Firefox Plugin Allows User Fingerprinting	Data Exposure	Medium
6. Delivery mechanism does not follow best practices	Data Validation	Medium
7. Weak passwords allowed	Authentication	Medium
8. Mailvelope can add Top-Level Domains as Scan Targets	Data Validation	Medium
9. No option to modify banner in OpenPGP key	Data Exposure	Low
10. Public keys cannot be verified prior to import	Configuration	Low
11. Lack of source validation of postMessage communications	Configuration	Informational

3.3 Detailed Vulnerability List

1. Remote content loaded via img, audio, video and other tags

Class: Data Exposure **Severity:** High **Difficulty:** Low

FINDING ID: iSEC-Mailvelope-10

TARGETS: The HTML Sanitization settings

DESCRIPTION: The DOMPurify library[2] employed by version 0.8.2 of Mailvelope, as well as the wysiHTML5 library in 0.8.1, contains a whitelist of allowed HTML tags and elements, and permits the img, audio, video, style, and potentially other tags that allow loading external resources by default. An encrypted email that contains tracking HTML will be displayed and beacon to the sender that the user has decrypted and read the email. Besides general tracking purposes, in the context of encrypted email, this could be used to confirm that a particular user of an email address holds a particular private key. A sample beacon could look like:

```
<img src=”https://isecpartners.com/tracker.gif”>
```

EXPLOIT SCENARIO: An attacker wishes to ascertain if a particular user has control over the private key used by a dissident. They send an email encrypted to the dissident to the user's email account. The user recognizes it is encrypted to their dissident's key because of Mailvelope's dialog, but cannot resist decrypting the message to see what it says. The message contains a hidden beacon that phones home to the attacker, revealing both the user's IP address and confirms ownership of the private key.

SHORT TERM SOLUTION: Remove the img, audio, video, style, and input type=image from the allowed DOMPurify settings, as well as the style attribute. Use the open source Email Privacy Test[3] to attempt to enumerate other mechanisms for remote content loading.

Carefully review all allowed elements and attributes, and remove those that do not need to be included. In particular, there doesn't seem to be a strong case for allowing SVG or MathML elements, which could also contain beacons.

LONG TERM SOLUTION: Add a feature, similar to Gmail's old feature, that would prompt the user to load remote resources if they are detected in the message. The feature can also automatically load remote resources from an approved sender, or be enabled for all senders.

[2] https://github.com/cure53/DOMPurify
[3] https://emailprivacytester.com/

2. DOMPurify engine is the only mechanism to stop Cross Site Scripting

Class: Data Validation **Severity:** High **Difficulty:** High

FINDING ID: iSEC-Mailvelope-11

TARGETS: HTML parsing libraries

DESCRIPTION: To defend against Cross Site Scipting, Mailvelope v0.8.1 uses the wysiHTML5 and v0.8.2 uses DOMPurify.[4] Each of these respective libraries let Mailvelope construct a whitelist of allowed HTML tags and elements which it relies on to sanitize JavaScript emails before displaying them to the user. While both the wysiHTML5 and DOMPurify libraries are considered well-written and have no obvious flaws or architectural mistakes, an extreme amount of trust is placed in each to prevent Cross Site Scripting. This overreliance on a single library to defend against such attacks creates a single point of failure that would potentially result in a complete compromise of the Mailvelope security model. It's unclear if the amount of encrypted HTML email is sufficient to justify the risk of interpreting HTML by default.

EXPLOIT SCENARIO: A vulnerability is discovered in DOMPurify and patched. The library authors alert Mailvelope to the vulnerability, but the public commit made by the project broadcasts the vulnerability to an interested attacker. Because of the Issue 9 No option to modify banner in OpenPGP key, the attacker knows exactly which users can be targeted with this vulnerability, and proceeds to exploit them while Mailvelope is stuck waiting for Mozilla to update the extension in the App Store.

SHORT TERM SOLUTION: Do not parse HTML by default. If Mailvelope detects that a decrypted email contains HTML, strip all HTML tags and present the email in plain text. For simple HTML-formatted messages, this should be sufficient to read the email, and does not encourage users to parse HTML message by default. In case the email is unreadable after being stripped, present an option to the user to parse the encrypted HTML email (using DOMPurify).

LONG TERM SOLUTION: In addition to DOMPurify, use all available mechanisms for sandboxing HTML after it is parsed. DocShell attributes such as allowImages[5] may be useful, as well as using Content Security Policy to prevent JavaScript execution or stylesheet parsing.

[4] https://github.com/cure53/DOMPurify
[5] https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XPCOM/Reference/Interface/nsIDocShell

3. Application data persists after uninstall

Class: Auditing and Logging **Severity:** Medium **Difficulty:** Low

FINDING ID: iSEC-Mailvelope-01

TARGETS: Sensitive information stored in a user's Firefox profile.

DESCRIPTION: During the normal course of operation Mailvelope stores sensitive information within the user's Firefox profile, such as secret keys. When the user uninstalls the extension, this sensitive information is not removed. It persists unless manually deleted by the user.

EXPLOIT SCENARIO: A user wishes to stop using Mailvelope, so they export their Mailvelope-created PGP key pair. They then remove the extension from Firefox and continue to use this key. The user assumes that all secure data has been erased. An attacker gains access to their file system through a malware infection and searches the user's Firefox profile directory for sensitive information. The attacker finds the secret key from Mailvelope and uses it to decrypt sensitive messages.

SHORT TERM SOLUTION: Implement an automated option to delete all stored data prior to extension removal. Document the need to perform this action prior to removing the extension.

LONG TERM SOLUTION: This is an inherent problem with the Mozilla SDK[6]. At minimum, make users aware of the persistence of sensitive information on the file-system. Encourage the use of simple storage editors, such as Simple Storage Editor for Add-on SDK [7]in order to manage and remove application data.

[6] https://bugzilla.mozilla.org/show_bug.cgi?id=627432
[7] https://addons.mozilla.org/en-US/firefox/addon/simple-storage-editor-for-a

4. OpenPGP implementation does not verify signed messages

Class: Configuration **Severity:** Medium **Difficulty:** Low

Finding ID: iSEC-Mailvelope-02

Targets: Message signature verification.

Description: Mailvelope does not currently include the ability to verify OpenPGP signed messages. This is a core feature of OpenPGP: it is the only way to verify the authenticity of OpenPGP messages and without it, users are unable to ensure that messages were sent by their trusted contact, undermining the trust of the system.

Exploit Scenario: An adversary spoofs an email from Alice, intended for Bob. The message is encrypted using Bob's public key. Bob believes the message is from Alice because it appears to be sent from her email address. In reality the attacker has spoofed the message and Bob lacks the ability to verify her identity by cryptographic signature.

Short Term Solution: Implement the OpenPGP.js signing features to allow users to verify signed content.

Long Term Solution: Integrate the signing feature into the UI to verify messages in-line and modal pop-out.

5. Firefox Plugin Allows User Fingerprinting

Class: Data Exposure **Severity:** Medium **Difficulty:** Low

FINDING ID: iSEC-Mailvelope-03

TARGETS: Firefox Plugin accessible via resource:// URLs

DESCRIPTION: The Firefox plugin has its administrative interface and other files accessible at resource:// URLs. The domain name used is done in bootstrap.js and defined as:

```
// Generate the domain name by using jetpack ID, which is the extension ID
// by stripping common characters that doesn't work as a domain name:
```

Defining a file at a resource:// URL makes the file available to any website[8], so a third party can learn if its visitors are using the Mailvelope Firefox Plugin. This information may be used for fingerprinting or engaging in other active attacks. A website operator can use a script similar to the following to detect if its users have the plugin installed:

```
<html>
<body>
<script>
function hasPlugin() {
  document.getElementById('notify').innerHTML = 'Yes.';
}
function noPlugin() {
  document.getElementById('notify').innerHTML = 'No.';
}
</script>
<img src="resource://jid1-aqqsmbyb0a8adg-at-
jetpack/mailvelope/data/common/img/sign-22.png" onload="hasPlugin()"
onerror="noPlugin()">
<div id="notify"></div>
</body>
</html>
```

EXPLOIT SCENARIO: An attacker breaches a high profile website, such as NBC.com. They install a detection script for Mailvelope users, and if the plugin is detected, they attempt to perform a phishing attack by emulating the Mailvelope security UI.

SHORT TERM SOLUTION: Generate a random domain name instead of a deterministic one. Do not use any absolute references to resource URLs, instead use relative references[9]. Alternately, investigate the chrome:// URL scheme as a potential target for migration, as they are not web-accessible.

LONG TERM SOLUTION: Carefully review the data available at Mailvelope resource:// URLs and ensure that it does not allow an attacker to modify settings or data, cause other side effects.

[8] https://developer.mozilla.org/en-US/docs/Chrome_Registration#resource
[9] https://blog.mozilla.org/addons/2012/01/11/sdk-1-4-known-issue-with-hard-coding-resource-uris/

6. Delivery mechanism does not follow best practices

Class: Data Validation **Severity:** Medium **Difficulty:** Low

FINDING ID: iSEC-Mailvelope-04

TARGETS: Distribution of the Mailvelope extension to users.

DESCRIPTION: Firefox has no method to verify the integrity of an add-on after it has been downloaded and installed, as it lacks built-in signatures and verification functions. To ensure authenticity, extensions must be downloaded through the Mozilla add-ons portal.

A user who wishes to install Mailvelope must currently download it from a Github repo. Github offers an unauthenticated hash to verify the file download, but there is no way to verify the authenticity of the product.

NOTE: Version 0.8.2 plans to use AMO for future deployment which helps mitigate this risk.

EXPLOIT SCENARIO: An attacker creates a spoofed version of Mailvelope by forking the Github repo and incrementing the version number. The modified version contains a malicious payload and is distributed to end users for installation.

SHORT TERM SOLUTION: Pursue hosting Mailvelope on the Mozilla add-ons portal (AMO) and completing the review process. This puts the extension through a (minimal) review process to determine the authenticity and integrity of the extension. Recommend that users download this version directly from Mozilla as opposed to the versions hosted on Github.

LONG TERM SOLUTION: Integrate AMO into the development life-cycle so that future releases can compensate for the time that AMO requires to review the extension.

7. Weak passwords allowed

Class: Authentication **Severity:** Medium **Difficulty:** Medium

Finding ID: iSEC-Mailvelope-05

Targets: Private Secret key generation.

Description: The OpenPGP standard requires that users enter a password in order to use a private key. Even if an adversary gains access to the private key, they must also enter a password to use it. For this reason, key-pair passwords are an important element that should follow best practices for password security.

The Mailvelope extension does not restrict users from creating insecure passwords. Single character passwords, common dictionary words, and strings that don't meet standard complexity requirements are all permitted. The following code reflects that the only requirement for passwords are that they are at least one character long.

```
function onKeyPwdChange() {
  var mask = (repwd.val().length > 0) << 1 | (pwd.val().length > 0);
```

Exploit Scenario: An attacker gains access to a user's encrypted private key by compromising their computer. The attacker finds the user's OpenPGP secret key in their Firefox profile directory. Although the key is password-protected, the attacker is able to quickly brute force the password because it is only a single character. The attacker goes on to use the key to decrypt sensitive messages sent to the user.

Short Term Solution: Implement password complexity requirements that include a minimum password length and a variety of characters. For example: passwords should include a combination of upper-case letters, lower-case letters, numbers, and/or special characters. If users cannot be required to make strong passwords, they should at least be warned when an insecure password is created.

Long Term Solution: Ensure that password complexity requirements are added to the future development standards of the Mailvelope extension.

8. Mailvelope can add Top-Level Domains as Scan Targets

Class: Data Validation **Severity:** Medium **Difficulty:** Medium

FINDING ID: iSEC-Mailvelope-09

TARGETS: The reduceHosts() function in controller.js

DESCRIPTION: The reduceHosts() function uses the number of dot-separated labels in a URL to determine where to place a wildcard. The function will disallow wildcarded single-label top-level domains, such as *.com, but does not take into account two-label top-level domains, such as *.co.uk. These domains will then be used as targets for PageMod injection.

EXPLOIT SCENARIO: An attacker registers a domain on a double-label top-level domain, such as attacker.co.uk. They target a user of Mailvelope who has added a UK-based email provider to their Watch List, and then trick the user into visiting the attacker-controlled site.

SHORT TERM SOLUTION: Use the public suffix list[10] to determine which labels are top level domains. Do not allow setting a wildcard on entire top level domains.

LONG TERM SOLUTION: Implement a mechanism to be notified automatically of changes to the public suffix list, and integrate changes promptly into Mailvelope releases.

[10] https://publicsuffix.org/

9. No option to modify banner in OpenPGP key

Class: Data Exposure **Severity:** Low **Difficulty:** High

FINDING ID: iSEC-Mailvelope-06

TARGETS: Identifying extension information

DESCRIPTION: The default OpenPGP banner for keys created within Mailvelope include Mailvelope's name along with the version number. Although it is an industry standard to include the software used to generate an OpenPGP encrypted message, and many users do not mind that this feature exists, it allows an adversary to fingerprint the software and version. Knowing this information may offer a way of either specifically targeting a user, or attributing a message to its owner in the case of a user attempting to maintain their anonymity.

```
-----BEGIN PGP MESSAGE-----
Version: Mailvelope v0.8.1
Comment: Email security by Mailvelope - http://www.mailvelope.com
```

EXPLOIT SCENARIO: An adversary fingerprints Mailvelope users by the software version in the message banner. The attacker targets known vulnerabilities in previous versions for exploitation.

An adversary bent on de-anonymizing a user is more likely able to attribute ownership of an encrypted message to a user that has that particular version of the software installed.

SHORT TERM SOLUTION: Remove the version number from the OpenPGP header.

LONG TERM SOLUTION: Create an option for users to modify the default banner for privacy and security reasons. Allow users to not include any header information or add their own header. For example, users can change the banner to make messages appear they were sent using GPG.

10. Public keys cannot be verified prior to import

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

FINDING ID: iSEC-Mailvelope-07

TARGETS: Public key import function

DESCRIPTION: The import function of the Mailvelope Add-on does not allow a user to verify the authenticity of the key imported.

Users can import keys two ways: through the settings console or via inline processing from text area. In the case of an in-line import, Mailvelope identifies that a public key exists, and prompts the user to import it. When the user clicks the icon, the key is imported without further prompting the user: the software does not display the key's email address, its fingerprint, or any other identifying information. Without this information, there is no way to verify that the key being imported is the one that belongs to the expected user, until after it has been installed.

EXPLOIT SCENARIO: An attacker spoofs an email that includes the public key for an email address. The user clicks on the key and imports it without verifying the key, importing it into the keychain. When the user attempts to encrypt a message to the intended recipient, they will instead be encrypting to the attacker's key.

SHORT TERM SOLUTION: Display the fingerprint and email address of the public key prior to completing the inline import process. Prompt the user to verify the fingerprint to ensure that it matches the expected fingerprint ID.

LONG TERM SOLUTION: Consider general improvements to the UI that make it easier for the user to verify the integrity of a message (signing),validate the authenticity of keys during import, and warn users when they are about to perform an insecure action.

11. Lack of source validation of postMessage communications

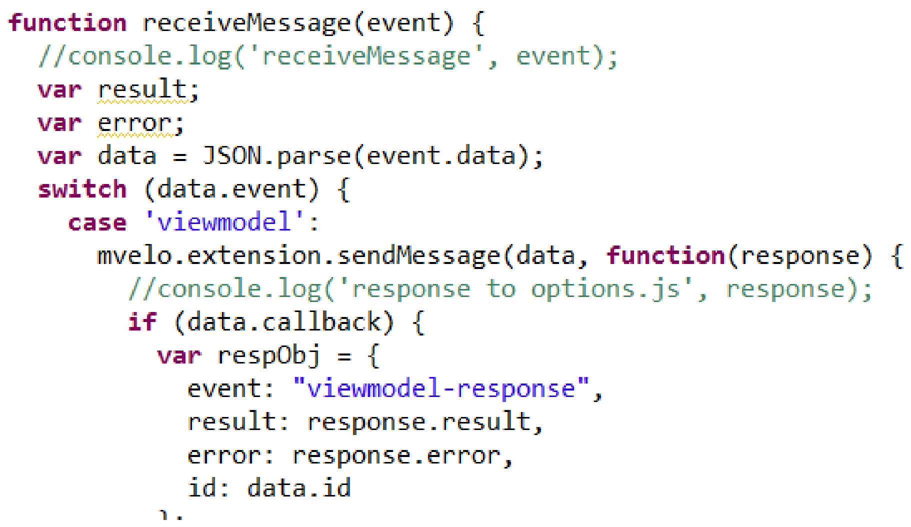
Class: Configuration **Severity:** Informational **Difficulty:** Medium

FINDING ID: iSEC-Mailvelope-08

TARGETS: Cross-domain communication

DESCRIPTION: Mailvelope postMessage event listeners do not validate the source of the messages. A JavaScript postMessage event is sent from the browser to the sandboxed iframe and vice versa. This allows for secure, controlled communication between the standard browser domain, and the trusted extension's domain. Validating the source of postMessages is important for maintaining secure communications.[11] Although a cross-origin policy is in place that restricts execution of unknown scripts as well as other defense measures to never evaluate unknown code, a more defense in-depth approach would be to first validate the source of the messages prior to loading the content. This would further prevent some types of attacks where a malicious webmail would send postMessages on your behalf if the cross-origin policy were to be evaded.

The code block below shows how controller.js creates an function to handle postMessages but does not validate the data.origin value.



```
function receiveMessage(event) {
  //console.log('receiveMessage', event);
  var result;
  var error;
  var data = JSON.parse(event.data);
  switch (data.event) {
    case 'viewmodel':
      mvelo.extension.sendMessage(data, function(response) {
        //console.log('response to options.js', response);
        if (data.callback) {
          var respObj = {
            event: "viewmodel-response",
            result: response.result,
            error: response.error,
            id: data.id
```

EXPLOIT SCENARIO: N/A

SHORT TERM SOLUTION: As shown in the screenshot above, consider adding a control to validate the origin and compare it to an expected sender list.

```
function receiveMessage(event)
{
  // Do we trust the sender of this message?
  if (event.origin !== "http://isecpartners.com:8080")
    return;
```

LONG TERM SOLUTION: Include source validation as part of the standard guidelines when developing postMessage event interfaces.

[11] https://developer.mozilla.org/en-US/docs/Web/API/Window.postMessage

4 Appendices

A Mozilla Firefox Security Considerations

A.1 Summary

Some of the challenges to designing add-ons for Firefox are due to inherent weaknesses in Firefox's design. Unlike Chrome, Firefox has no concept of access levels and allows access to all permissions to all extensions[12]. This appendix is provided for informational purposes to highlight the potential threats that all Firefox extensions are required to deal with.

A.2 Security Model

Add-ons hosted on AMO are reviewed in varying stages; fully reviewed applications are scanned for malware[13]. Non-reviewed applications can be hosted on AMO while queued for testing.

Mozilla documentation states that data stored in simple-storage is private to the add-on[14]; this is misleading, as private simply means the data is stored in a separate file directory and not that there are restrictions in place preventing access by other Firefox add-ons. In fact, this data is not encrypted or access restricted. Other software with access to Firefox's profile directories can access the add-on's private storage.

A.3 Malicious Extensions

If a malicious extension were installed into a browser it would have the ability to read and modify data from other applications, and other areas of the file system. In the case of Mailvelope, this means that other extensions can make modifications, access storage, and even perform keylogging to collect the passwords used by Mailvelope[15]. By utilizing interfaces within the XPCOM API, it is even possible to execute OS commands via Firefox extensions[3].

The threat of a malicious Firefox extension is not mitigated inside the browser. When a user installs an extension, it is given unilateral trust as discussed above. It can be considered an even higher risk due to the fact that most browsers allow extensions to be installed from any location on the Internet. Like Mailvelope's current version that is downloaded from a Github repository, there is no built-in way to verify the authenticity or integrity of extensions hosted on untrusted websites.

[12] Frichot, C., Orru, M., & Alcorn, W. (2014). The Browser Hacker's Handbook. Wiley.
[13] https://blog.mozilla.org/addons/2010/02/04/please-read-security-issue-on-amo
[14] https://developer.mozilla.org/en-US/Add-ons/SDK/High-Level_APIs/simple-storage
[15] http://resources.infosecinstitute.com/keylogger

A.4 Recommendations

Maintain awareness of differences in the threat models amongst various web browsers and implement additional controls when possible to protect sensitive data. Use AMO to host Firefox extensions and submit add-ons for a full AMO review. Encourage users to only install extensions from the AMO store. While these recommendations cannot mitigate all potential threat vectors, they are the best defense to limit the risk of accidentally installing a malicious extension in the Firefox browser.

B Input and output validation improvements

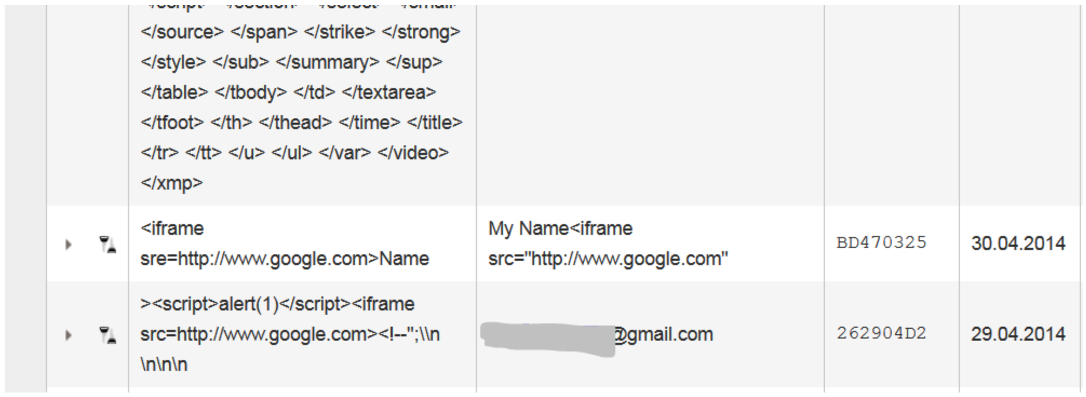
B.1 Summary

This appendix is to provide an overview of, and suggest improvements to, the defensive measures the Mailvelope application currently provides via its HTML5 sandboxed iframe, postMessage cross domain communication, JSON element parsing, and Cross-Origin Request Policy that are used in conjunction to defend against malicious script injection.

B.2 Security Measures

The Mailvelope Add-On implements an HTML5 sandboxed iframe that is designed to compartmentalize data that exists in the normal browser from data that is loaded inside of the extension. Data is sent via JavaScript postMessage between the two domains which is a common security practice. This information is then collected, parsed, and used by the various Mailvelope functions. In the case of importing keys and adding a page to the watch list, the content from the normal browser environment, is stored inside of the extension's trusted storage location. In these cases, JavaScript or other objects are not stripped, but are never evaluated by the JavaScript engine which ensures that injection attacks are mitigated. While this properly defends against most attacks, a more defense in-depth approach would be to also validate the input of the content passed from the browser environment.

In the screenshot below, we see examples of public keys that contained inappropriate content for the display name and email address. While Mailvelope validated that the OpenPGP public key was a valid format, it did not further validate that the extracted information from the key was also in an expected format.



	</source> </span> </strike> </strong> </style> </sub> </summary> </sup> </table> </tbody> </td> </textarea> </tfoot> </th> </thead> </time> </title> </tr> </tt> </u> </ul> </var> </video> </xmp>			
	<iframe sre=http://www.google.com>Name	My Name<iframe src="http://www.google.com"	BD470325	30.04.2014
	><script>alert(1)</script><iframe src=http://www.google.com><!--";\\n \n\n\n	[redacted]@gmail.com	262904D2	29.04.2014

The reason this did not result in a vulnerability was due to the fact that the elements were dynamically converted into JSON objects, and loaded into the properties of the page. This meant that the code was never actually evaluated by Firefox's JavaScript engine nor its HTML processor.

B.3 Improvements

A more defense in-depth approach would be to further validate the format of the information collected, before it is displayed in the settings. While no exploits currently exist for this configuration, it's recommended that even more validation would help defend against the scenario where the HTLM5 sandbox or the cross-origin policy could be circumvented. In this case, an attacker would have access to inject information into the Mailvelope application without restriction.