July 24, 2017

EasyCrypt passes an independent security audit

EasyCrypt, a Swiss-based email encryption and privacy service, announced that it has passed an independent security audit. The audit was sponsored and commissioned by Open Technology Fund and performed by Include Security while EasyCrypt was still in beta. The vulnerabilities discovered in the audit have been fixed in the current version of the service, with the exception of two low priority vulnerabilities that were disputed.

The highly usable EasyCrypt service allows non-technical users to communicate using OpenPGP end-to-end encryption, using their existing email service and address, without installing anything and with no need to manage encryption keys.

EasyCrypt team would like to express its sincere thanks to OTF for sponsoring the audit.

Scope

Include Security team was given access to the entire client and server source code of EasyCrypt as well as to architecture and design documents. While EasyCrypt Hamlet secure webmail client relies on RoundCube for parts of the code, RoundCube itself was not included in the audit. In addition to the code, a live instance of EasyCrypt server, identical to the production version except for some additional developer/debug information, was made available to Include Security in order to enable its team to try and execute exploits of any vulnerabilities discovered while inspecting the code. The assessment was performed using Light Grey Box Assessment Methodology.

Summary of the discovered vulnerabilities

Finding	Include Security assessment	Status
C1: Persistent Cross-Site Scripting (XSS)	Critical	Fixed
C2: Insecure CORS Origin Match	Critical	Fixed
H1: Redis Key Query Injection	High	Fixed
M1: Temporary 502 Denial of Service Condition Identified	Medium	Fixed
M2: Authentication System Does Not Protect Against Brute-Force Attacks	Medium	Fixed
M3: Sensitive Information Stored in LocalStorage	Medium	Fixed
L1: Application Functionality Can Be Used to Confirm Valid Accounts	Low	Fixed
L2: Cryptographic Secrets Stored in Source Code	Low	Fixed
L3: Insecure Mailgun WebHook Implementation	Low	Fixed
L4: Weak User Password Requirements	Low	Disputed
L5: Credentials for External Services Stored in Source Code	Low	Fixed
L6: Content Security Policy Not Implemented	Low	Fixed
L7: HTTP Strict Transport Security Not Implemented	Low	Fixed
L8: Missing X-XSS-Protection Header	Low	Fixed
L9: Missing X-Frame-Options Header	Low	Fixed
L10: AES CFB Mode Encryption Used Without Integrity Checking	Low	Disputed

Details and status of the discovered vulnerabilities

C1: Persistent Cross-Site Scripting (XSS)

Finding: one instance of persistent XSS was discovered. For example, malicious code can be sent to the victim through an email with the following subject: 123'<script>alert(1)</script>.

EasyCrypt team response: Acknowledged

EasyCrypt team action: the client-side sanitization process has been improved. The entire email content is now being scanned after decryption. JavaScript code and HTML tags that are considered harmful are filtered out.

Status: fixed.

C2: Insecure CORS Origin Match

Finding: a misconfiguration was identified in the implementation of the Origin header match, which allows an attacker to bypass any restrictions and gain access to sensitive information such as user emails. This vulnerability can be exploited by an attacker by creating custom JavaScript code that will use CORS (Cross-Origin Resource Sharing) to read emails from the EasyCrypt webmail server and then convince EasyCrypt users into clicking on a link to the attacker's domain. If the victim has an active EasyCrypt session, the attacker will be able to access all his emails. The application code is using a regular expression such as .*.easycrypt.co when it should be using .*\.easycrypt.co.

EasyCrypt team response: Acknowledged

EasyCrypt team action: Nginx configuration has been updated and the CORS regex has been fixed.

Status: fixed.

H1: Redis Key Query Injection

Finding: two instances of Redis key query injection were identified in the EasyCrypt application. This vulnerability occurs when an adversary manages to manipulate variables inside the secured Jason Web Token (JWT) that is sent to the client, and these are used to create a string which is then used to query for Redis keys. The vulnerability can be exploited to perform various actions such as retrieve encrypted PGP private keys on all users.

EasyCrypt team response: Acknowledged; however, it is worth noting that this vulnerability could be only exploited if the attacker has access to the encryption and signing secrets of the JWT that are stored on the server in volatile memory at run-time. Moreover, this vulnerability cannot be exploited to hack into user accounts.

EasyCrypt team action: All JWT fields, although signed and encrypted, are now considered to be unsafe and pass validation before being used. Specifically, attributes that are used in SQL or Redis queries are checked more thoroughly, and escape and wildcard characters are removed.

Status: fixed.

M1: Temporary 502 Denial of Service Condition Identified

Finding: Denial of Service vulnerability was identified. The issue was triggered using Burp Suite's automated vulnerability scanner and resulted in the account.easycrypt.co domain returning HTTP code 502 for all requests. The servers were inaccessible for at least 15 minutes after stopping the automated scan process. Given the low bandwidth required to trigger this denial of service, it seems to be a performance issue in the application code.

EasyCrypt team response: Acknowledged

EasyCrypt team action: We have reviewed the code that could potentially cause the issue and reached the conclusion that the issue is not related to the code. To fix it, we modified the service architecture and reconfigured nginx. Stress testing in production environment indicated that the issue has been resolved.

Status: fixed.

M2: Authentication System Does Not Protect Against Brute-Force Attacks

Finding: the authentication system does not prevent brute-force attacks against accounts. Given a large number of authentication attempts, an attacker may be able to use an automated brute-force attack to successfully guess users' authentication credentials.

EasyCrypt team response: Acknowledged

EasyCrypt team action: A new access control and throttling mechanism has been implemented that monitors login attempts and kicks in when excessive failed attempts are detected. The mechanism throttles subsequent login attempts.

Status: fixed.

M3: Sensitive Information Stored in LocalStorage

Finding: sensitive information, including the user's EasyCrypt password, cached emails and PGP keys are stored in the browser's LocalStorage without encryption. EasyCrypt user password is the most critical piece of information since it can be used to login into the EasyCrypt account and decrypt any existing emails.

EasyCrypt team response: Acknowledged

EasyCrypt team action: EasyCrypt user password is no longer stored in LocalStorage. All other sensitive data stored in LocalStorage, including emails and private keys, is now encrypted.

Status: fixed.

L1: Application Functionality Can Be Used to Confirm Valid Accounts

Finding: the login functionality is implemented in such a way that would allow an anonymous user to confirm system user's email addresses as valid accounts. In the current implementation, the application responds differently depending on whether the input supplied was recognized as associated with a valid user or not. This behavior could be used as part of a two-stage automated attack. During the first stage, an attacker would iterate through a list of account names to determine which correspond with valid accounts. During the second stage, the attacker would use a list of common passwords to attempt to brute force credentials for accounts that were recognized by the system in the first stage.

EasyCrypt team response: This vulnerability indeed existed in the test environment because we provide debug information there, but it does not exist in production where strict API responses are enforced.

EasyCrypt team action: Not required.

Status: fixed.

L2: Cryptographic Secrets Stored in Source Code

Cryptographic secrets used for signing and encrypting JWT in the staging environment were found within the EasyCrypt server source code. As access to the source code may be exposed due to another exploit (e.g., file traversal) or via a shared source code repository, any attacker or malicious insider would have access to these secrets. This may enable them to attack the cryptographic systems used by the application.

EasyCrypt team response: This vulnerability doesn’t exist in production environment. All secrets are instantiated at run-time and stored only in volatile memory of the service instances.

EasyCrypt team action: not required

Status: fixed.

L3: Insecure Mailgun WebHook Implementation

EasyCrypt uses the Mailgun service to handle emails sent to registerpublickey@easycrypt.co. When an email is received by Mailgun for this email account an HTTP POST request is sent to EasyCrypt servers where it is handled by a method in the authentication service. Using source code review, it was possible to identify that the webhook handler does not authenticate the received information in any way before processing it. This allows an attacker to send specially crafted information to the controller, which will process it just as if it were sent by Mailgun.

EasyCrypt team response: Acknowledged

EasyCrypt team action: An access control measure has been implemented that expects a pre-shared key to be provided by incoming requests to the API endpoint. The pre-shared key in hard-coded in the URL that we set in MailGun. Therefore, only authentic requests from MailGun are eligible and are processed.

Status: fixed.

L4: Weak User Password Requirements

Finding: users are required to choose a password with at least 8 characters, containing a minimum of 2 digits and 2 letters. Given the sensitive information protected by this password, the security assessment team believes that the password strength requirements must be increased.

EasyCrypt team response: this is a security/usability trade-off by design. Users can make their passwords as secure as they like as we set no upper password length or character limitations.

EasyCrypt team action: not required.

Status: no change.

L5: Credentials for External Services Stored in Source Code

Finding: credentials for external services were found within the EasyCrypt source code. As access to the source code may be exposed due to another exploit (e.g., file traversal) or via a shared source code repository, any attacker or malicious insider would have access to these secrets. This may enable them to attack the services used by the application.

EasyCrypt team response: Acknowledged

EasyCrypt team action: We are no longer using external services.

Status: fixed.

L6: Content Security Policy Not Implemented

Finding: Content Security Policy is not implemented. Content Security Policy is a W3C specification which offers the possibility for an application to instruct the client browser about which locations and/or which type of resources are allowed to be loaded within the application. CSP is a very effective way to prevent Cross-Site Scripting vulnerabilities and also forces developers to split view and controller code.

EasyCrypt team response: Acknowledged

EasyCrypt team action: Content security policy has been deployed in learning mode.

Status: fixed.

L7: HTTP Strict Transport Security Not Implemented

Finding: Several EasyCrypt subdomains do not implement HSTS. This could be exploited by an attacker to perform protocol downgrade attacks which could be used to intercept and/or modify HTTP traffic between the user and EasyCrypt servers.

EasyCrypt team response: Acknowledged

EasyCrypt team action: HSTS policy was configured to enforce HTTPS connection for MAX time (1yr) over all easycrypt.co subdomains.

Status: fixed.

L8: Missing X-XSS-Protection Header

The EasyCrypt application doesn't set the HTTP X-XSS-Protection header. This header enables the XSS filter within the browser so that the browser will prevent certain types of Cross-Site Scripting attacks.

EasyCrypt team response: Acknowledged

EasyCrypt team action: We configured a header that enables XSS filtering. If an attack is detected, the page load will be blocked.

Status: fixed.

L9: Missing X-Frame-Options Header

EasyCrypt's web application is vulnerable to clickjacking. Clickjacking attacks typically use a combination of stylesheets, iframes, and form elements to convince a targeted user that they are interacting with an innocuous page when instead, they are typing into or clicking on an invisible frame controlled by an attacker. A successful clickjacking attack could circumvent cross-site request forgery (CSRF) protections that attempt to confirm transactions with the user, resulting in an unwanted transaction.

EasyCrypt team response: Acknowledged

EasyCrypt team action: We configured a header that causes the browser to prevent loading of our webmail inside an iframe if the page’s origin is different than the iframe’s origin.

Status: fixed.

L10: AES CFB Mode Encryption Used Without Integrity Checking

The EasyCrypt application was found in several source code sections to use AES' Cipher Feedback (CFB) mode of operation to encrypt and decrypt information. AES' CFB operation mode is considered more secure than ECB and CBC, but it does not verify the integrity of the data received before decrypting it, which could be abused by an attacker to perform complex cryptographic attacks on the application.

EasyCrypt team response: we are doing multi-level integrity checking of the encrypted content. Therefore, Galois Counter Mode is not needed. EasyCrypt signs the AES content, which prevents tampering with the encrypted block.

EasyCrypt team action: not required.

Status: no change.

###