Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Pentest-Report Pomerium 03.2021

Cure53, Dr.-Ing. M. Heiderich, N. Hippert, BSc. B. Walny, BSc. T.-C. “Filedescriptor” Hong

Index

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Introduction

“Use identity, device-state, and request context to implement zero-trust, achieve compliance, and secure access to your applications, clusters, and servers without a VPN.”

From https://www.pomerium.com/

This report documents the results of a security assessment targeting the Pomerium complex. Carried out by Cure53 in 2021, this project entailed a thorough penetration test and a dedicated source code audit featuring the Pomerium codebase and a deployed setup running the latest version of Pomerium in a real-life and live environment.

To give some context, the work was requested by Pomerium, Inc. in January 2021 and then promptly scheduled. Cure53 has punctually executed the project in March 2021, namely in CW10 and CW11. As for the resources, a total of twenty person-days were invested to reach the coverage expected for this project, which was prepared, conducted, documented and finalized by a team of four testers, all selected on the basis of their skills and experience matching the Pomerium’s technical traits and objectives.

In terms of optimizing structure, the work was split into two separate work packages (WPs). In WP1, Cure53 performed a thorough source code audit on the latest version of the Pomerium’s codebase, whereas WP2 focused on penetration tests against the Pomerium’s integration and broader setup. Methodology-wise, a white-box approach was chosen and deployed. In particular, Cure53 was given access to the deployed environment as well as all relevant sources and documentation. The environment was rolled-out on GCP and Cure53 was furnished with appropriate access instructions and setups.

The project moved forward efficiently. All preparations were done in late February and early March 2021, namely in CW08 and CW09, indicating that Cure53 could have a smooth start. Communications during the test were done in a dedicated and shared Slack channel which connected the workspaces of Cure53 and Pomerium. All relevant personnel from both teams could partake in the discussion which were very productive. Since the scope was well-prepared and clear, no noteworthy roadblocks were encountered during the test. Nevertheless, Cure53 relayed frequent status updates about the test and the emerging findings.

Very good coverage over the WP1-2 scope items was arguably achieved. Five security-relevant issues were spotted and documented: two representing security vulnerabilities and three classified as general weaknesses with lower exploitation potential. Live-reporting was utilized and the Pomerium team managed to fix the tracked issues while

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

the test was still ongoing. Moreover, the Cure53 team was given all necessary info to also look at the fixes and either verified them or added remarks about possible bypasses and similar problems.

Commenting on the findings, the overall number of items is rather small, which is a great sign for Pomerium and its codebase. However, the severity levels are quite elevated. Although no issues were marked as *Critical,* two discoveries were seen as *High* risks. One of them enveloped a JWT leak and the other pertained to a missing signature verification within middleware.

In the following sections, the report will first shed light on the scope, material available for testing, key examination parameters, as well as the structure and content of the WPs. After that, the document offers a distinct section on testing coverage and methodology, so as to facilitate tracking of the areas covered during the assessment, even if the tests yielded no findings in a given realm.

Next, all findings will be discussed in grouped vulnerability and miscellaneous categories, then following a chronological order in each array. Alongside technical descriptions, PoC and mitigation advice, as well as fix notes, are supplied when applicable. Finally, the report will close with broader conclusions about this March 2021 project. Cure53 elaborates on the general impressions and reiterates the verdict based on the testing team’s observations and collected evidence. Tailored hardening recommendations for the Pomerium complex are also incorporated into the final section.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Scope

- **Penetration-Tests and Source Code Audits against Pomerium**
  - Access to rolled out application
    - https://verify.audit.pomerium.io/
  - GCP Access
    - https://console.cloud.google.com/invitation?project=pomerium-audit-2021q1
  - Public Source Code Repository
    - https://github.com/pomerium/pomerium
  - **Test-supporting material was shared with Cure53**
  - **All relevant sources were shared with Cure53**

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Test Methodology

This section documents the testing methodology applied during this engagement and sheds light on various areas of the Pomerium application complex. Tests were performed both in the form of the server-side API, as well as frontend and infrastructure components. The section further clarifies which areas were examined by Cure53 but did not yield any findings.

WP1: Thorough Source Code Audits against latest version of Pomerium

The enumeration below describes the tests and coverage areas checked through source code audits and focused on the network-based components of Pomerium. The listed areas were investigated through the approaches described per item. Note that the specific commit hash for the version of the Pomerium software examined by Cure53 was: 50dc15de283fc0e864e7e96acce3d4acc346e77c

- Starting with the Pomerium *config*, the parsing logic was reviewed for potential injections or general unexpected behaviors. The policy from and to routes are being passed through the whole software complex, so this needed investigation. These and other parsing and path handling routines which, for instance, match the host of requests to the proxy, have been reviewed for parsing differentials. If found, those would lead to a potential proxy bypass.
- While making requests which go through the proxy, other inputs were reviewed. For example, Cure53 looked at inputs sent through HTTP headers, e.g. *authorization* or *cookie* headers.
- Lua scripts responsible for handling requests, as well as responses from an upstream server through the proxy have been reviewed. In scope were mostly weaknesses that could leak *authorization* tokens to upstream services, or problems due to responses of upstream servers being handled in an improper manner.
- Exposed Pomerium APIs and sites of the proxy, authentication and authorization services have been reviewed for typical web security vulnerabilities like XSS and CSRF.
- The general HTTP security header configuration of the exposed services has been reviewed.
- The employed JWT has been checked for common mistakes, such as improper signature verification due to the “*None”* algorithm, key confusion or signature exclusion. Further, handling of JWT parameters, like the expiry time, has been checked.
- The JWKS implementation was audited for cryptographic issues and proper implementation of the primitives in use.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

- The integration of the IdPs has been reviewed. The configurations for identity providers are compliant with the OAuth/OIDC specifications. Sensitive key information is well-secured.
- Past vulnerability reports for Pomerium have been checked out to spot interesting areas that suffered in the past and these have been verified again.

WP2: Penetration Tests against Pomerium Integration & Setup

The enumeration below describes the tests and coverage areas checked within Pomerium's deployment, both locally and on the provided GKE installation.

- The startup and initialization phase of the Pomerium all-in-one setup has been reviewed and checked for potential misuse by local attackers. The filesystem access routes have been traced and evaluated against potential attack scenarios.
- The configuration stack provided by the Pomerium team, to be used with the GKE installation, has been reviewed.
- Redis configuration and backend used as Databroker were examined for potential NoSQL injection attacks from the application's side, as well as potential SSRF targets from the internal network's side.
- The network topology and connected parts of the overall architecture were examined. This also included consideration of relevant runtime- and environment-specifications that are necessary to run Pomerium
- In terms of security, Pomerium provides the means to configure TLS for the connections between the individual nodes. Due to the requirement of having valid certificates, it is hard to offer this feature by default. However, the documentation on how TLS needs to be configured presented on the website is fairly simple and straightforward. As such, it should be considered by everyone using Pomerium across untrusted networks.
- The authentication flows were checked in detail. The programmatic access feature was found to accept any URL, which led to an Open Redirect and JWT leak (see POM-01-001).

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Identified Vulnerabilities

The following sections list both vulnerabilities and implementation issues spotted during the testing period. Note that findings are listed in chronological order rather than by their degree of severity and impact. The aforementioned severity rank is simply given in brackets following the title heading for each vulnerability. Each vulnerability is additionally given a unique identifier (e.g. *POM-01-001*) for the purpose of facilitating any future follow-up correspondence.

POM-01-001 WP1: JWT leak via Open Redirect in programmatic access *(High)*

Using programmatic access on *verify.audit.pomerium.io*, one can get a signed login URL with *pomerium_redirect_uri* set to an arbitrary URL. Then, if the user has already logged into Pomerium, they will be redirected to the specified *pomerium_redirect_uri* with a JWT attached. This allows an outside attacker to get a signed login URL that, upon visiting, will redirect a victim to the attacker's site. This creates an issue of Open Redirect and an even more serious problem in the form of JWT leakage.

Sample leaked JWT (decoded):

```
{
  "aud": [
    "authenticate.audit.pomerium.io",
    "evil.com",
    "verify.audit.pomerium.io"
  ],
  "exp": 1615442162,
  "iat": 1615392513,
  "iss": "authenticate.audit.pomerium.io",
  "jti": "ID.OqUbtelP7ce6do2Q2DJJj_9htFFpSl8XQEFa-ku95Ks",
  "nbf": 1615392513,
  "programmatic": true,
  "sub": "00u2f6thmxCSSqF104x7",
  "ver": "15750195866874303488"
}
```

With the leaked JWT, the attacker will be able to unveil the victim's identity (e.g., an email address) by supplying the JWT to *authenticate.audit.pomerium.io* or *verify.audit.pomerium.io*. In addition, if an application integrating Pomerium only verifies the *iss* claim but not the *aud* claim, the attacker will be able to access it as the victim.

Steps to reproduce:

1. Navigate to https://verify.audit.pomerium.io/.pomerium/api/v1/login?pomerium_redirect_uri=http://evil.com in order to get a signed login URL
2. Navigate to the login URL

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

3. Sign in via Okta if this has not already been done
4. Observe being redirected to *evil.com* with a *pomerium_jwt* in the URL
5. Replace the *_pomerium* cookie in https://verify.audit.pomerium.io with the one from *Step 4*
6. Refresh https://verify.audit.pomerium.io and confirm the JWT is valid as information is returned

Currently, the programmatic access feature does not restrict what the *pomerium_redirect_uri* can be. It is recommended to implement an allow-list mechanism to prevent its usage in scenarios linked to Open Redirect and leaking JWTs.

Fix Notes: *This issue was addressed by Pomerium and the deployed fix was verified by Cure53.*

POM-01-002 WP1: No verification of *pomerium_signature* in middleware *(High)*

It was found that some API endpoints under */.pomerium/* do not verify parameters with *pomerium_signature*. This could allow modifying parameters intended to be trusted by Pomerium. During communication with the Pomerium team, it was determined that it was caused by a regression that accidentally removed a line of code that enforced the signature check in middleware.

Affected file:
https://github.com/pomerium/pomerium/blob/d653b0156f4a01ea81901bfbdb3bd7c101a16e4f/authenticate/handlers.go#L69

Affected code:
```
v.Use(middleware.ValidateSignature(a.sharedKey))
```

The above code was not found in the master branch. The issue mainly affects routes responsible for sign in/out, but does not introduce an authentication bypass. For example, the issue mentioned in POM-01-001 would still work even if that issue was fixed with the help of this problem.

It is recommended to reintroduce the removed line of code and add a relevant test-case to prevent accidentally deleting critical lines of code.

Fix Notes: *This issue was addressed by Pomerium and the deployed fix was verified by Cure53.*

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Miscellaneous Issues

This section covers those noteworthy findings that did not lead to an exploit but might aid an attacker in achieving their malicious goals in the future. Most of these results are vulnerable code snippets that did not provide an easy way to be called. Conclusively, while a vulnerability is present, an exploit might not always be possible.

POM-01-003 WP2: Hardening recommendations for file-handling *(Medium)*

While auditing the initialization process of an all-in-one installation, it was observed how the *envoy* binary is being placed inside a temporary directory with a fixed filename on the filesystem. Here no checks are being made to ensure this directory is owned by the same user the Pomerium application is running with. A series of other checks is performed but they can all be bypassed due to TOCTOU races. Ultimately, the binary is executed.

Since the attacker model does not include a local attacker, the severity is kept rather low. A sample exploit strategy has been shared with the developers to further increase the awareness of such issues. It is recommended to use random temporary directory names. Paired with the correct access permissions, random temporary directory names would prevent other local users from creating the directory and filling it with malicious contents.

POM-01-004 WP2: Cross-Origin HTTP security headers missing *(Info)*

It was found that the platform is missing certain newer[1] HTTP security headers in HTTP responses. This does not directly lead to a security issue, yet it might aid attackers in their efforts to exploit other problems. The following list enumerates the headers that need to be reviewed to prevent flaws linked to headers.

- **Cross-Origin Resource Policy (CORP)** and *Fetch Metadata Request* headers allow developers to control which sites can embed their resources, such as images or scripts. They prevent data from being delivered to an attacker-controlled browser-renderer process, as seen in *resourcepolicy.fyi* and *web.dev/fetch-metadata*.
- **Cross-Origin Opener Policy (COOP)** lets developers ensure that their application window will not receive unexpected interactions from other websites, allowing the browser to isolate it in its own process. This adds an important process-level protection, particularly in browsers which do not enable full Site Isolation; see *web.dev/coop-coep.*

[1] https://security.googleblog.com/2020/07/towards-native-security-defenses-for.html

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

- **Cross-Origin Embedder Policy (COEP)** ensures that any authenticated resources requested by the application have explicitly opted-in to being loaded. Today, to guarantee process-level isolation for highly sensitive applications in Chrome or Firefox, applications must enable both COEP and COOP; see *web.dev/coop-coep.*
- While the application does use the *X-Frame-Options* header, it should be noted that the CSP framework offers a similar protection to *X-Frame-Options* in ways that overcome some of the shortcomings of the aforementioned header. To optimally protect users of older browsers and modern browsers at the same time, it is recommended to consider deploying the *Content-Security-Policy: frame-ancestors 'none';* header as well.

Overall, missing security headers is a bad practice that should be avoided. It is recommended to add the following headers to every server response, including error responses like *4xx* items. More broadly, it is recommended to reiterate the importance of having all HTTP headers set at a specific, shared and central place rather than setting them randomly. This should either be handled by a load balancing server or a similar infrastructure. If the latter is not possible, mitigation can be achieved by using the web server configuration and a matching module.

In the specific case of Pomerium, it is recommended to advise the users about the existence of those headers and document how they can be used to mitigate side-channels the Pomerium setup might inadvertently expose in case those headers are not set correctly. Resources explaining those headers are available online, explaining both the proper header setup[2] as well as the possible consequences of not setting them after all[3].

[2] https://scotthelme.co.uk/coop-and-coep/
[3] https://web.dev/coop-coep/

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

POM-01-005 WP2: Missing *SameSite* flag for cookie security *(Info)*

During the analysis of the security flags set for cookies, it was noticed that the *SameSite* flag is only used for the CSRF cookie, but it is not used for the cookie containing the JWT token. This behavior could not be turned into an exploitable scenario, however, given the limited demo installation, larger installations might benefit from the additional security provided by properly using the *SameSite* attribute.

Affected cookie:

- _Pomerium
  - Missing *SameSite* flag

It is recommended to offer an option to add the *SameSite* flag with strict settings to all cookies that contain critical data, for instance JWT tokens or other similarly sensitive data. It must be noted however that setting this cookie flag without checking expected functionality afterwards might lead to faulty application behavior, i.e in situations where a redirect is happening as part of the authentication procedure[4].

[4] https://www.ubisecure.com/technical-announcements/samesite-cookies-changes/

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Conclusions

During this March 2021 project, Cure53 examined the Pomerium codebase and a deployed setup running the latest version of Pomerium in a real-life and live environment. As a result, mostly positive conclusions have been reached when it comes to security of the Pomerium application stack. After spending twenty days on the scope in March 2021, the Cure53 team members can conclude that the various examined components and aspects of the applications mostly withstood their scrutiny in regard to security premise. At the same time, two rather severe vulnerabilities were discovered on the scope and should not be disregarded.

As for the general setup and procedures of the test, communication was done using a shared Slack channel and greatly aided the coverage levels and discovery potential of this examination. Questions regarding certain findings and functionality were promptly answered and the engineering team made sure to pass along information to make Cure53's life easier when stumbling onto difficult to understand application flows or certain technical problems. In doing so, the in-house team positively contributed to the overall success of this assignment.

From a meta-level perspective, the overall attack surface related to the usage of the Pomerium software complex is quite small. However, when one looks into details, a potential local privilege escalation was found (see POM-01-003). Fixing this issue further hardens the attack surface against attackers having an advantageous local position.

The security recommendations for the setup and configuration for other third-party software was found to be solid and secure, removing potential threats, such as Redis as an SSRF target. The handling of requests and responses to the proxy was found to be secure and no wiggle room for attacks on the HTTP layer has been found. The Identity Provider Integration appears sound as well. The configurations for identity providers comply with OAuth/OIDC specifications, while sensitive key information is well-secured. The state parameter is correctly verified in the corresponding callback.

Regarding the authentication flow, the programmatic access feature was found to accept any URL, which leads to Open Redirect and JWT leakage described in POM-01-001. It was determined to be a serious (*High)* issue, given the fact that the JWT unveiled the victim's identity and potentially allowed bypassing access control. During the verification of the fix deployed by the Pomerium team, an additional security problem was spotted in the fix implementation. The fault was in trying to maintain backward compatibility with previous local deployments. However, the issue was spotted and rectified quickly with a second patch.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Moreover, it was found that the middleware did not verify the HMAC signature for certain endpoints (POM-01-002) due to regressions in previous versions, offering more potential for exploiting POM-01-001. Simultaneously, the verification of the JWT and parsing of the URL were found solid thanks to the use of good standard libraries, characterized by a proven track record of security consciousness. General client-side security was examined and no issues like XSS, CSRF or insecure HTTP headers were spotted. However, there is some room for improvement, as stated in POM-01-004.

All in all, only two exploitable issues were spotted and none of them would have *Critical* implications. While the Pomerium project should look into improving their automated regression testing, which could have prevented POM-01-002 from being released, the security dedication of the in-house team is praiseworthy, as also evidenced from swift and mostly correct fixes of the problems spotted by Cure53 during this March 2021 inspection. As a consequence, Cure53 sees the application as being on the right track to delivering a proper foundation from a security perspective.

Cure53 would like to thank Bobby DeSimone, Travis Groth, Denis Mishin, Caleb Doxsey and Nathan Hayfield from the Pomerium team for their excellent project coordination, support and assistance, both before and during this assignment.