CYCURA™
A WELL Health Company

Safe Software Inc.
FME Server Vulnerability Disclosure

Colin Murdoch
Cyber Security Researcher
Cycura

CYCURA™
A WELL Health Company

While performing a recent penetration test for a client, one of the in-scope modules contained workflows within an instance of the FME Server application. While an assessment of the FME Server instance itself was not originally in-scope, the client signed off on this addition during the final days of the engagement to allow for more excellent coverage of their attack surface.

Safe Software Inc. FME Server is an enterprise tool for creating automated workflows, which run various data integration tasks on a scheduled basis or as an event-driven model. Using a "visual design," workflows enable users to automate data processing tasks without the tedious manual labour of writing code or integrating with various systems. FME Server is a data processing engine that can use data streams to build business insights, transform customer data, and improve operational efficiencies with real-time data processing.

The following is a breakdown of the methodologies and impacts of multiple vulnerabilities we discovered, some of which were critical. Because the discovered vulnerabilities affect software users beyond just our client, Safe Software was contacted for responsible disclosure and to enable them to proceed with remediation of said vulnerabilities.

CYCURA™
A WELL Health Company

ACKNOWLEDGEMENTS

We want to thank Safe Software for being open and transparent during the vulnerability disclosure process. It was a pleasure to work with their team. In addition, Safe Software was quick to jump on implementing fixes and was very receptive to receiving the findings.

Safe Software embraces security and strives to uphold quality standards that ensure their client's sensitive information is secure. This experience was a textbook example of how a vulnerability disclosure should go.

Furthermore, our reviewed overall code employed many excellent and secure coding practices. Of the issues found, many had extra security measures such as reasonably privileged accounts or specialized roles within the tool. These additional security measures significantly reduced the attack surface from the perspective of an external threat actor.

General Inquiries: 1-647-479-8425
Incident Response: 1-844-227-0452

CYCURA.COM

CYCURA™
A WELL Health Company

TESTING METHODOLOGY

Escalating From Black Box

When the initial testing of the FME Server instance began, the scope was a black box with minimal access. It was possible to log into the tool, but the account had limited permissions. While the scope within the client's environment may have been a black box, we effectively turned it into a grey or white box engagement by setting up a custom lab environment and installing a trial version of FME Server matching the client.

In a lab environment, it is possible to gain administrator access and start reviewing the application's functionality; this allowed us to test most components manually. We looked for a variety of vulnerabilities, including cross-site scripting (XSS), XML external entity injection (XXE), SQL injections (SQLi), insecure direct object references (IDOR), input validation, and path traversal issues.

Additionally, with reverse engineering techniques and having control over the executing binaries, it became possible to understand the server-side code and how it interacts with the host system.

Finding XXE

One of the main functionalities exposed on the client instance of FME Server was related to executing Repository workflows. It was impossible to upload or modify the workflows on the client's instance, but it was possible to interact with them. In our lab environment, we observed that a portion of the Repository workflow contained structures similar to an XML document.

We attempted An XXE attack using the modified format, and it became apparent that external entities were parsed, which can lead to server-side request forgery (SSRF) and data exfiltration capabilities. Luckily, it required the `Read` and `Publish` permissions to be on a Repository object set, which not every user would have.

Finding XSS

Reviewing existing vulnerabilities of the application proved sparse, but **`CVE-2020-22789`** and **`CVE-2020-22790`** looked interesting. These vulnerabilities highlighted that stored XSS vectors existed, one of which could be completed as an unauthenticated user.

This vulnerability could pose a risk to our client if it were still possible. Therefore, we quickly tested against the current version. The original fix handled standard XSS injections very well. Still, due to the complexities of the HTML parsing stack and multiple angles of introducing XSS payloads, it can often be difficult to ensure complete coverage.

To be safe, we developed some specialized payloads to test the affected parameters with various non-standard payloads thoroughly, eventually leading to a bypass. Ultimately, we achieved the unauthenticated stored XSS in the lab environment.

We discovered another low-impact vulnerability; server-side validation did not correctly perform when creating new user accounts. As a result, part of the user account details was included in the HTML content when logged in, which could lead to self-XSS. However, it ultimately had minimal impact as the XSS only appeared to affect the user account itself.

As a general takeaway, this low-impact vulnerability highlights the importance of continuously validating or sanitizing data on the server-side first and using those rules to implement client-side controls for the user experience where appropriate.

Client-side controls are highly effective against a typical end-user of the application, but at the end of the day, the end-user is still in control of the validation rules. As a result, they can easily disable the rules and bypass all the intended business logic controls if they have malicious intent.

Fully White Box To Find Path Traversal

Testing against the lab environment was adequate; however, we needed a code review to gain clarity on the application. While the source code didn't exist for FME Server, reverse engineering techniques would be the next best thing, and this would effectively provide near-white box levels of insight into the application.

Diving into the recovered code proved worthwhile, as we discovered a particularly critical path traversal and file upload issue. Luckily, it required administrator access and other unique factors to trigger, which kept it out of reach of the typical user.

CYCURA™
A WELL Health Company

CLOSING THOUGHTS

The above findings highlight from a penetration testing standpoint that getting creative with increasing visibility, given a limited scope, can often pay dividends in the overall hunt for vulnerabilities. When dealing with a black box test, leveraging all external information about the target architecture can significantly improve the chances of finding something interesting.

Because all of these vulnerabilities had some access restrictions and were not available in our client's standard environment and scope, it's unlikely that the discovery of these issues would have happened without setting up an internal lab.

And while it can be tedious, manual review is a very effective method of finding a needle in the haystack; automated tools have a tough time understanding general business logic. But on the other hand, the human component can understand the reason, compensating controls, and intent and can use that information to verify that the application behaves as expected.

One major takeaway from an application developer perspective is validation and data sanitization. Never trust data which an end-user has some form of control over. Building a standard library of validation and sanitization techniques for the various output contexts can be very effective.

When performed close to the data consumption or egress point, sanitization becomes easier to reason about its current context to ensure any dangerous inputs will be escaped or handled. For example, if an internal function assumes that all incoming data is safe and sanitized, it only takes one branch in the application to lead to a potential vulnerability.

Remember that a well-funded threat actor may spend significant time reviewing an application and searching for that single branch. With hardware's processing power and capabilities today, the tiny performance hit for some extra sanitization at the source and removing those branches is invaluable.

Fortunately, most of the vulnerabilities we discovered required special access within the application, significantly reducing the attack surface. However, even if this is the case, a rogue user or compromised account could still impact substantially.

Furthermore, in the worst-case scenario, with the proper access levels, an attacker may be able to achieve remote code execution. For this reason, we will not be releasing technical write-ups or proof of concepts.

The Safe Software team has taken the appropriate measures to correct these issues in FME Server 2021.2.6 and 2022.0.1.1.

CYCURA™
A WELL Health Company

Would you like to find out what a professional Penetration Test could do for your security?

CONTACT A REAL HACKER TODAY!

General Inquiries: 1-647-479-8425
Incident Response: 1-844-227-0452

The below link to all the advisories published by Safe Software related to the discovered vulnerabilities and describe the steps necessary for successful remediation:

* https://community.safe.com/s/article/FME-Server-Stored-Cross-Site-Scripting-XSS-Vulnerabilities
* https://community.safe.com/s/article/Known-Issue-FME-Server-vulnerability-with-arbitrary-path-traversal-and-file-upload
* https://community.safe.com/s/article/Known-Issue-FME-Server-XXE-vulnerability-via-adding-a-repository-item
* https://community.safe.com/s/article/Known-Issue-Arbitrary-file-upload-with-any-authenticated-FME-Server-account
* https://community.safe.com/s/article/Known-Issue-Lack-of-server-side-validation-when-creating-a-new-user-in-FME-Server
* https://community.safe.com/s/article/Known-Issue-FME-Server-missing-validation-which-may-result-in-an-unwanted-redirect-upon-login