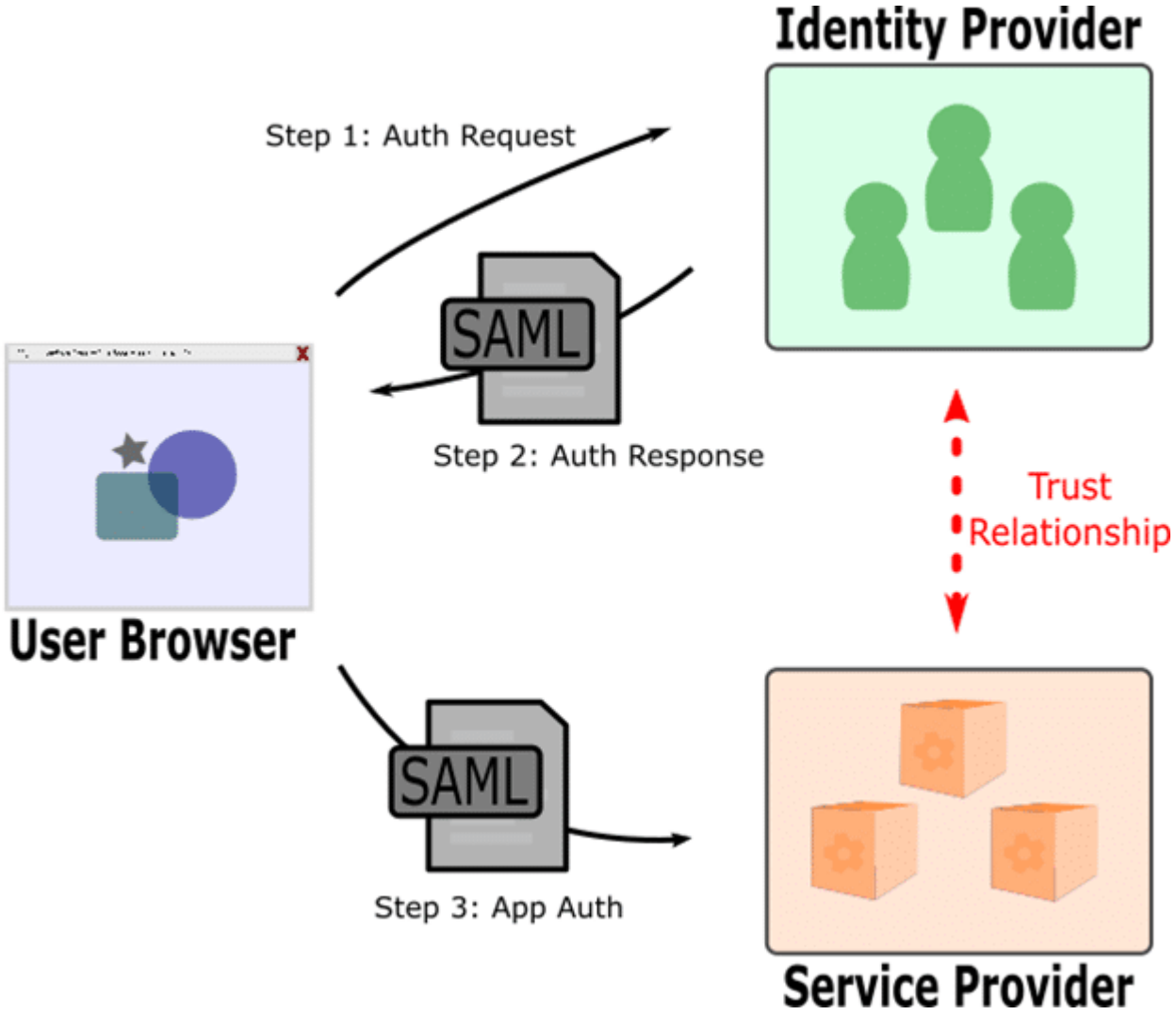
Attacking SSO: Common SAML Vulnerabilities and Ways to Find Them

In this blog I'll share some pointers that can be used when testing Single Sign-On (SSO) solutions that utilize SAML. The centralized nature of SSO provides a range of security benefits, but also makes SSO a high-profile target to attackers. The majority of SSO implementations I have seen in the past year pass SAML messages as part of the authentication process. There is nothing inherently wrong with this approach but a small misconfiguration in an SSO implementation can lead to some large vulnerabilities.

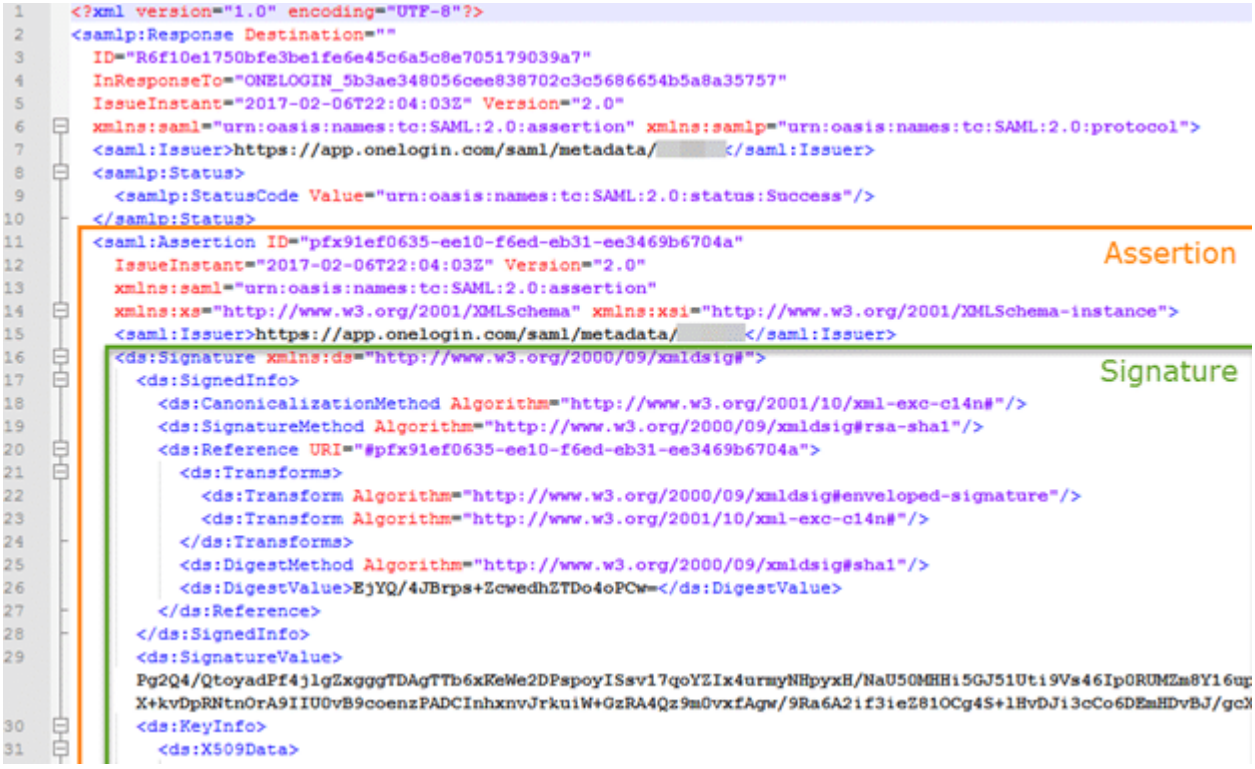
A Simplified Overview of SAML



A traditional application may implement authentication checks before allowing a user to access protected functions of the application. In the SSO model, the authentication functions are moved to an external Identity Provider (IP) application that performs authentication before allowing the user to access the protected functions in the Service Provider (SP) application. In order for the two applications to communicate with other, some messaging must pass through a user's browser, which gives the user

an opportunity to tamper with the message. One common flow is for the Identity Provider to return a SAML message to the browser, which forwards the message to the Service Provider.

SAML is a markup language implemented in XML. SAML messages are base64 encoded but that is easily decoded to view the message contents. In my experience, the two most common areas in SAML messages that are prone to tampering are signatures and assertions. The signature enforces the trust relationship between the IP and SP. The assertion instructs the SP on what trusted operations to perform, usually to allow you to access the application as a certain user.



Common Implementation Mistakes & Testing Tips

- Message Expiration: SAML messages should contain a timestamp of when the request was issued, when it expires or both. If the SAML message never expires or if the expiration is not honored, there is a greater risk of a message falling into the hands of an attacker. Check the message for timestamps, such as an **IssueInstant** or **NotOnOrAfter** assertion. Pause the request until after the expiration has passed and then allow the request through to the SP. Also make sure the expiration window is reasonable, like 1-5 minutes.
- Message Replay: Assertions should contain a unique ID that is only accepted once by the application. Try replaying a SAML message to create multiple sessions.
    - Missing Signature: Messages without signatures can be freely edited to tamper with permissions on the SP application. Make sure a signature exists in the SAML and that the signature is required by the application. If there is one, try to resend the message without a signature.
    - Invalid Signature: Signatures which are not signed by a real CA are prone to cloning. Ensure the signature is signed by a real CA. If the certificate is self-signed, you may be able to clone the certificate or create your own self-signed certificate to replace it.
    - SAML from Different Recipient: An application should only accept a SAML message

intended for the SP application. If the application does not perform this check, it may honor a SAML message generated from authenticating to another application and allow you into the application as the user from the other application. If you have a valid login for another application which uses the same IP, login to the other SP application and record the message. Replay the message intended for the other SP to your target SP.

- Signature Wrapping: Some implementations check for a valid signature and match it to a valid assertion, but do not check for multiple assertions, multiple signatures, or behave differently depending on the order of assertions. The following are eight of the most common XML Signature Wrapping attacks. You can edit the original SAML file manually to perform these attacks but it is much quicker with the use of a tool. The short names (ex: XSW1) map to the names used in the SAML Raider tool, discussed below.
  - XSW1 – Applies to SAML Response messages. Add a cloned unsigned copy of the Response after the existing signature.
  - XSW2 – Applies to SAML Response messages. Add a cloned unsigned copy of the Response before the existing signature.
  - XSW3 – Applies to SAML Assertion messages. Add a cloned unsigned copy of the Assertion before the existing Assertion.
  - XSW4 – Applies to SAML Assertion messages. Add a cloned unsigned copy of the Assertion after the existing Assertion.
  - XSW5 – Applies to SAML Assertion messages. Change a value in the signed copy of the Assertion and adds a copy of the original Assertion with the signature removed at the end of the SAML message.
  - XSW6 – Applies to SAML Assertion messages. Change a value in the signed copy of the Assertion and adds a copy of the original Assertion with the signature removed after the original signature.
  - XSW7 – Applies to SAML Assertion messages. Add an “Extensions” block with a cloned unsigned assertion.
  - XSW8 – Applies to SAML Assertion messages. Add an “Object” block containing a copy of the original assertion with the signature removed.
- XML External Entity (XXE): A SAML message is just a user-provided XML message that is processed by the Service Provider. Be sure to check all standard XML attack vectors. XXE is a very common XML attack and I find it frequently through SAML messages.

Exploiting SAML Vulnerabilities

Some attacks, such as replaying expired messages or replaying messages for another application, will yield their own limited results. Most of the vulnerabilities described above allow an assertion to be tampered with, which requires one last step to fully exploit the discovered vulnerability. If you are able to tamper with a SAML message in such a way as to send your own assertions, try the following:

- Change the expiration date on an expired message to make it valid again
- Change the UserId to a different valid user – Bonus points if that user is an admin
- Change the UserId to a different invalid user – Sometimes an application will grant default permissions or higher privileges to an unmapped user

SAML Raider

One very helpful tool for testing SAML is the SAML Raider extension for Burp Suite. It automatically highlights proxied requests containing SAML messages and adds a proxy tab with the decoded payload. SAML Raider also adds a pane to Repeater which allows you to quickly issue popular signature wrapping (XSW) attacks. Finally, SAML Raider adds a Certs tab which makes cloning certificates easy. You can either clone the certificate outright or create a self-signed version of the certificate.

Summary

SAML security is an often-overlooked area of SSO applications. Successful SAML attacks result in severe exploits such as replaying sessions and gaining unauthorized access to application functions. SAML attacks are varied but tools such as SAML Raider can help in detecting and exploiting common SAML issues. I hope that by using these techniques you can improve your detection and correction of SAML vulnerabilities in your applications.

References

- https://www.owasp.org/index.php/SAML_Security_Cheat_Sheet
- https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final91.pdf