TLP: AMBER

Analysis Report on CVE-2022-26134 Vulnerability

- Atlassian Confluence OGNL Injection

V1.0

AhnLab Security Emergency Response Center (ASEC)

Aug. 11, 2022

AhnLab

Classification

Publications or provided content can only be used within the scope allowed for each classification as shown below.

Classification	Distribution Targets	Precautions
TLP: RED	Reports only provided for certain clients and tenants	**Documents that can only be accessed by the recipient or the recipient department** Cannot be copied or distributed except by the recipient
TLP: AMBER	Reports only provided for limited clients and tenants	**Can be copied and distributed within the recipient organization (company) of reports** Must seek permission from AhnLab to use the report outside the organization, such as for educational purposes
TLP: GREEN	Reports that can be used by anyone within the service	**Can be freely used within the industry and utilized as educational materials for internal training, occupational training, and security manager training** Strictly limited from being used as presentation materials for the public
TLP: WHITE	Reports that can be freely used	Cite source Available for commercial and non-commercial uses Can produce derivative works by changing the content

Remarks

The version information of this report is as follows:

Version	Date	Details
1.0	2022-08-11	First release

Contents

Overview 4
Background Knowledge 6
Vulnerability Attack Process 11
  Interceptor 11
  OGNL Injection 11
  Proof of Concept 13
Confluence Attack Cases 16
  CVE-2022-26134 16
  CVE-2021-26084 17
Vulnerability Patch 19
Vulnerability Update 21
  Temporary Mitigation 21
  Version Update 21
AhnLab Response Overview 23
Indicators Of Compromise (IOC) 23
  File Hashes (MD5) 23
References 24

Overview

The CVE-2022-26134 vulnerability occurs when XWork—a web framework of Atlassian's Confluence Server and Confluence Data Center—processes URI values requested by users. During this process, inadequate verification measures cause Object-Graph Navigation Language Injection (OGNL) to occur. This attack allows threat actors to execute arbitrary codes.

The product versions affected by this vulnerability are outlined in Table 1 below.

Product Name	Version
Confluence Data Center	Versions 1.3 - 7.4.17 Versions 7.13.0 - 7.13.6 Versions 7.14.0 - 7.14.2 Versions 7.15.0 - 7.15.1 Versions 7.16.0 - 7.16.3 Versions 7.17.0 - 7.17.3 Version 7.18.0
Confluence Server	Versions 1.3 - 7.4.17 Versions 7.13.0 - 7.13.6 Versions 7.14.0 - 7.14.2 Versions 7.15.0 - 7.15.1 Versions 7.16.0 - 7.16.3 Versions 7.17.0 - 7.17.3 Version 7.18.0

Table 1. Versions affected by this vulnerability

From version 1.3 released in 2005 to the most recent version of 7.18.0, almost all Confluence products can be affected. Also, because the same XWork package is used for Linux and Windows environments, the impact of this vulnerability may be critical. The vulnerability was measured to be 9.8 (Critical) according to the CVSSv3, which evaluates the severity of vulnerabilities. As this vulnerability allows system control such as uploading web shells and executing arbitrary codes through attacks, users must update their product to the version where this vulnerability has been patched.

This report explains the causes of the CVE-2022-26134 vulnerability, attack cases, and countermeasures.

Background Knowledge

XWork is a web framework used by Confluence servers, and it uses "Action" as the unit of tasks.

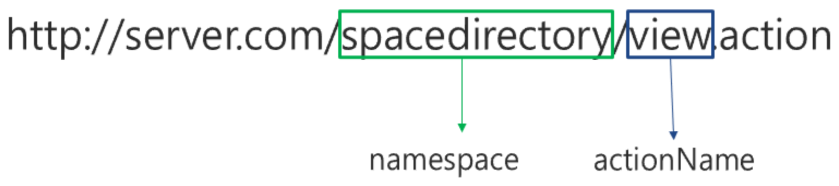
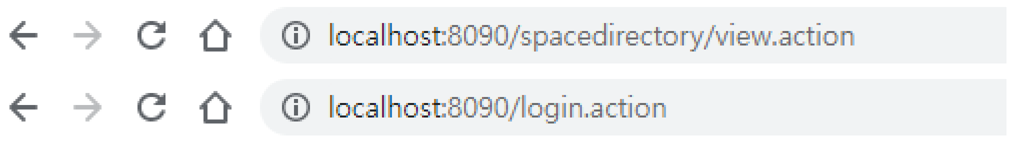


Figure 1. ActionName and NameSpace expressed in the URI

The server internally performs different tasks according to each action and sends the result values as a reply to the client. Figure 2 shows the URI path of the Confluence server's task space page and login page.



```
localhost:8090/spacedirectory/view.action
localhost:8090/login.action
```

Figure 2. Various ActionNames available

The server configures an environment that can handle various actions such as "view" and "login" to process requests from clients. The detailed process of the XWork web framework performing actions is shown in Figure 3.

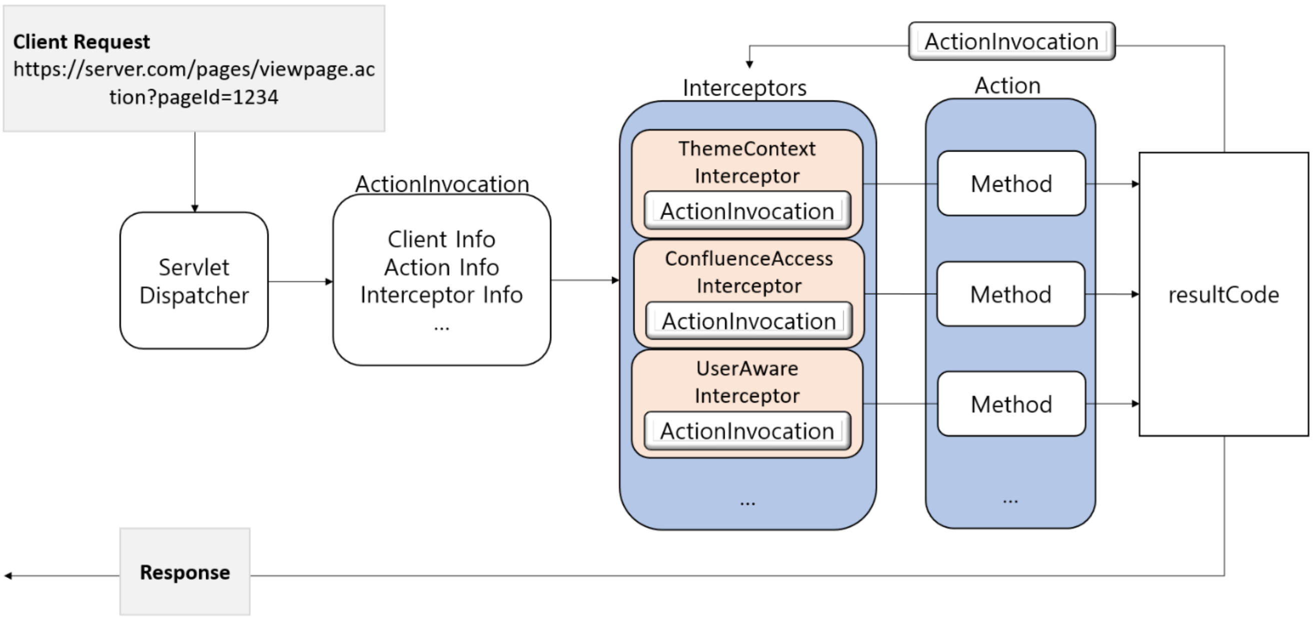
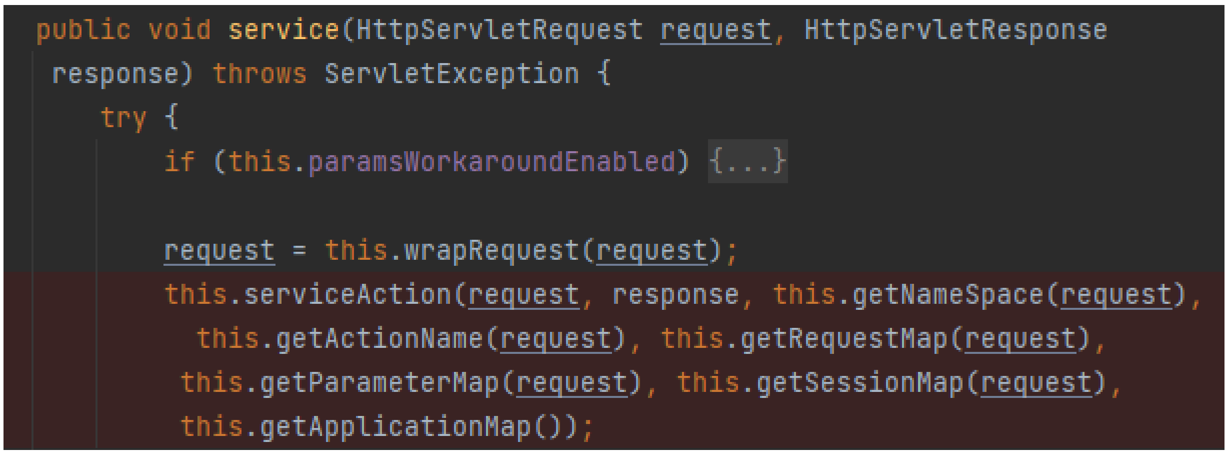


Figure 3. Detailed work process of XWork

When a Request is received, ServletDispatcher first imports the information sent by the client, including namespace and actionName.



```
public void service(HttpServletRequest request, HttpServletResponse
 response) throws ServletException {
    try {
        if (this.paramsWorkaroundEnabled) {...}

        request = this.wrapRequest(request);
        this.serviceAction(request, response, this.getNameSpace(request),
          this.getActionName(request), this.getRequestMap(request),
         this.getParameterMap(request), this.getSessionMap(request),
         this.getApplicationMap());
```

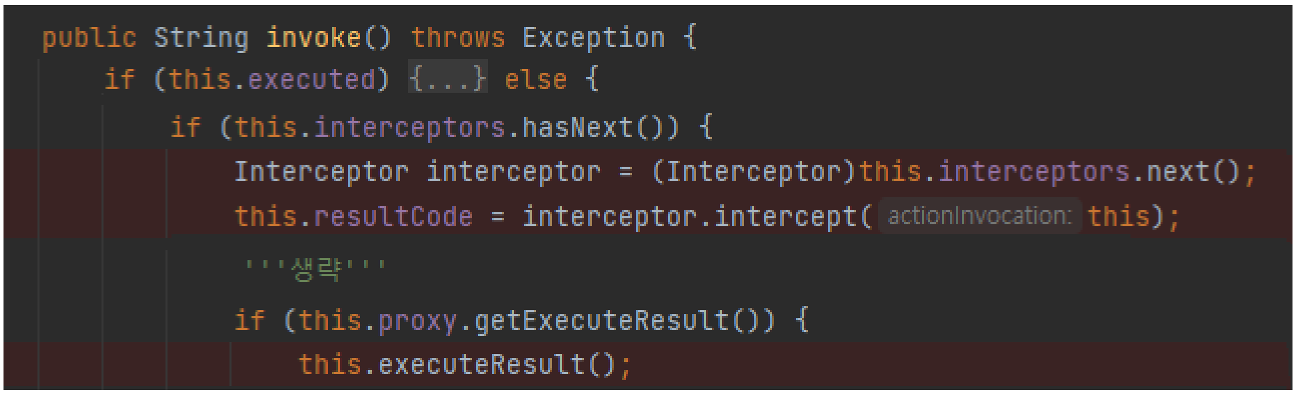
Code 1. Inside ServletDispatcher.class

These pieces of information are formed into an argument to call the serviceAction method. Within the serviceAction method, the values received as an argument are objectified with ActionInvocation. Then, an interceptor is called on this ActionInvocation.

This interceptor is responsible for performing the Action corresponding to the actionName included in the request URI. In this step, the client information received in the

Request is used. ActionInvocation is an object made to contain the information on the Action to be used by the interceptor and client information, alongside the information on the interceptor to be called and the performance status/results of the interceptor.

Code 2 below is a code that calls the Interceptor according to ActionInvocation.



```
public String invoke() throws Exception {
    if (this.executed) {...} else {
        if (this.interceptors.hasNext()) {
            Interceptor interceptor = (Interceptor)this.interceptors.next();
            this.resultCode = interceptor.intercept( actionInvocation: this);
            '''생략'''
            if (this.proxy.getExecuteResult()) {
                this.executeResult();
```

Code 2. Calling the interceptor

Using interceptors.next, the list of interceptors within ActionInvocation is read and called with the “intercept” method. Different methods are called by each interceptor, and this process is the stage where the task requested by the client is actually performed by the server.

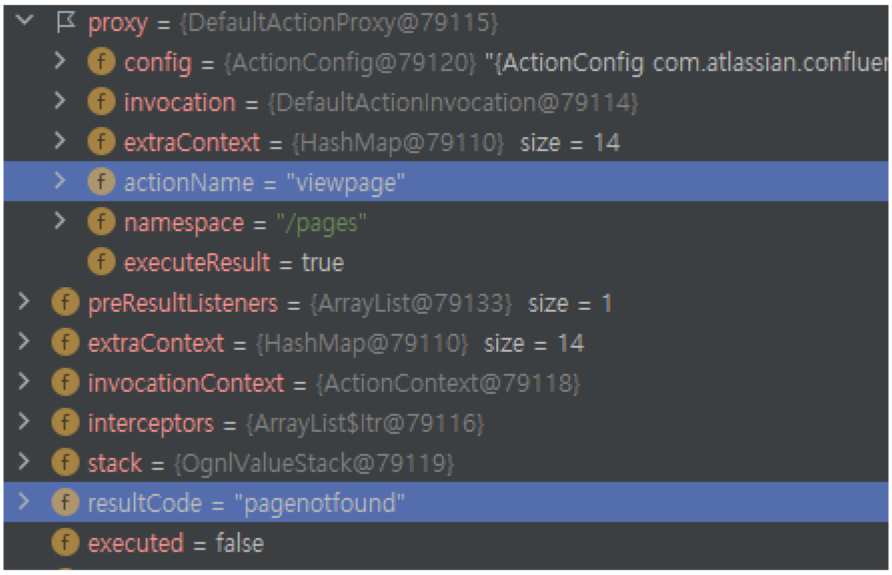
Table 2 shows some of the 28 interceptors called by IndexAction in Confluence 7.13.6.

Interceptor	Description
SetupIncompleteInterceptor	Check if initial settings have been configured for Confluence
ConfluenceAccessInterceptor	Allow or deny actions performed by the current user
PageAwareInterceptor	Insert a page into the current action
PrepareInterceptor	Perform tasks that must precede the current action

Table 2. Interceptor descriptions

When a method is called through the interceptor, a resultCode is returned. If this value is null, the next interceptor is called, and if there exists a value, this return value is used to create a new ActionInvocation object. executeResult in Code 2 fulfills this role. Interceptors are called for the newly created ActionInvocation.

Figure 4 shows the ActionInvocation members according to a request for a nonexistent page in the Confluence server.

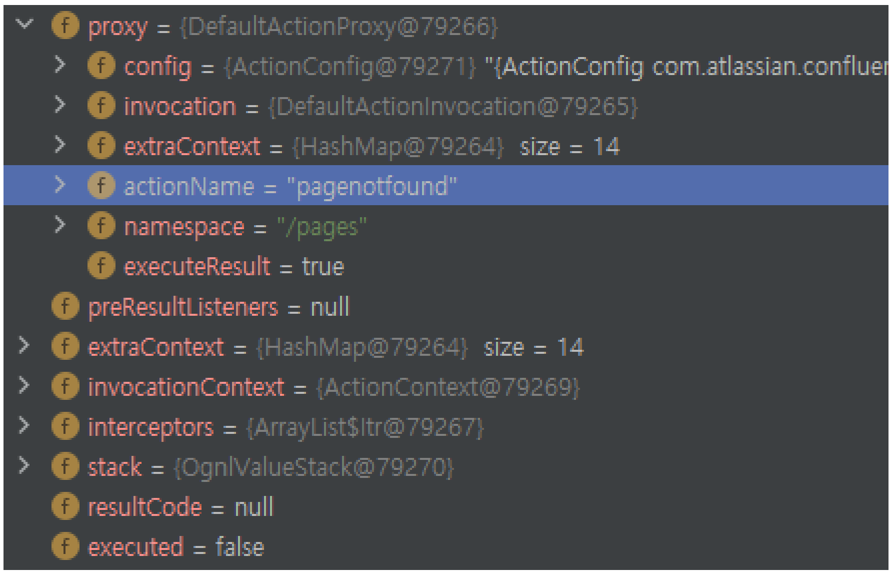


```
v  proxy = {DefaultActionProxy@79115}
   >  config = {ActionConfig@79120} "{ActionConfig com.atlassian.conflue
   >  invocation = {DefaultActionInvocation@79114}
   >  extraContext = {HashMap@79110}  size = 14
   >  actionName = "viewpage"
   >  namespace = "/pages"
      executeResult = true
>  preResultListeners = {ArrayList@79133}  size = 1
>  extraContext = {HashMap@79110}  size = 14
>  invocationContext = {ActionContext@79118}
>  interceptors = {ArrayList$Itr@79116}
>  stack = {OgnlValueStack@79119}
>  resultCode = "pagenotfound"
   executed = false
```

Figure 4. ActionInvocation object members according to the viewpage Action

viewpage is the actionName used when requesting a page. Out of the many interceptors called, PageAwareInterceptor is responsible for identifying the fact that the page requested by the client does not exist, and it returns a resultCode of pagenotfound.

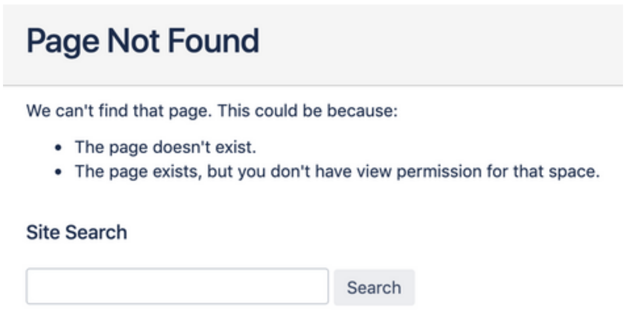
Figure 5 shows the values of ActionInvocation members at the point of executing the very next interceptor.



```
proxy = {DefaultActionProxy@79266}
  config = {ActionConfig@79271} "{ActionConfig com.atlassian.confluer
  invocation = {DefaultActionInvocation@79265}
  extraContext = {HashMap@79264}  size = 14
  actionName = "pagenotfound"
  namespace = "/pages"
  executeResult = true
preResultListeners = null
extraContext = {HashMap@79264}  size = 14
invocationContext = {ActionContext@79269}
interceptors = {ArrayList$Itr@79267}
stack = {OgnlValueStack@79270}
resultCode = null
executed = false
```

Figure 5. New ActionInvocation object members according to the pagenotfound result

We can see that based on resultCode pagenotfound, a new ActionInvocation is created, and an interceptor is called accordingly. The response of the Confluence server to the pagenotfound action is shown in Figure 6.



Page Not Found

We can't find that page. This could be because:

- The page doesn't exist.
- The page exists, but you don't have view permission for that space.

Site Search

Figure 6. pagenotfound response

To recap the operational flow of XWork, the server reads the actionName in the client request and performs the corresponding Action. This process involves creating the information sent by the client into an ActionInvocation object to call various interceptors and return a resultCode. If a resultCode exists, ActionInvocation is made again for this resultCode and a corresponding interceptor is called. Through this process, when the tasks of the interceptor that first handles the request and the task of the interceptor handling the resultCode are all complete, the server sends a response to the client.

Vulnerability Attack Process

Interceptor

The interceptor with the vulnerability is ConfluenceAccessInterceptor. This interceptor checks the privileges of the current user's request to allow or deny access. If the user has access privileges, it returns a null value as the resultCode, and if not, returns notpermitted.

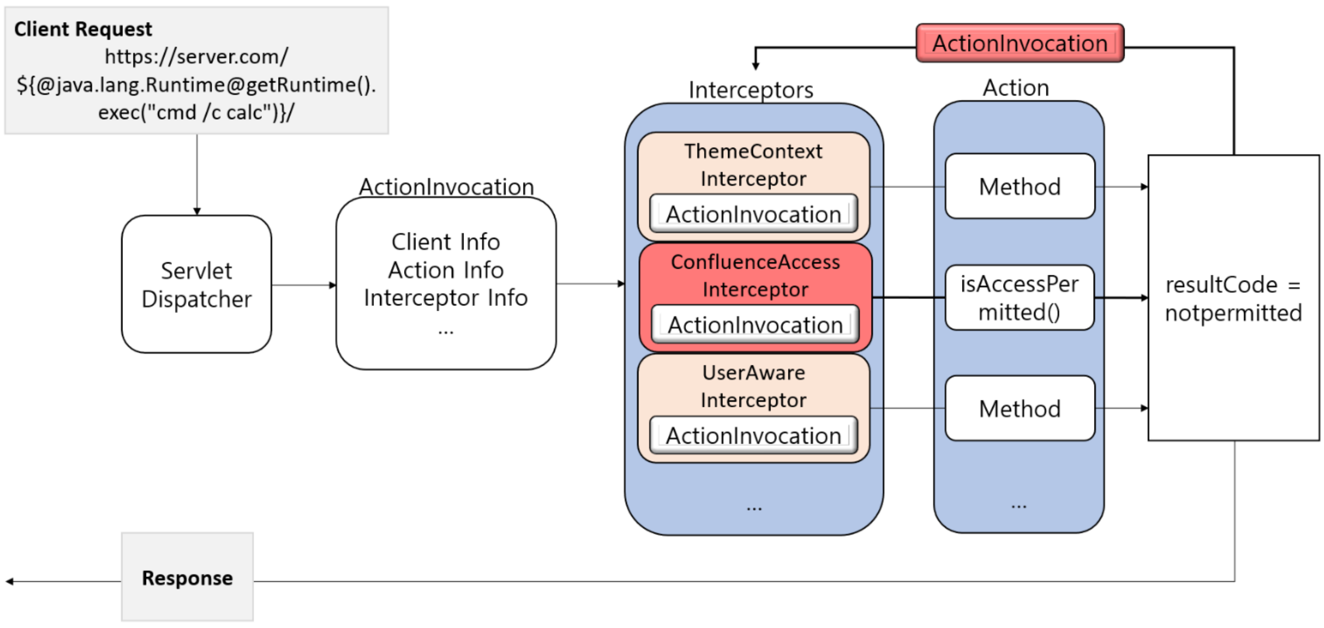


Figure 7. Attack process using ConfluenceAccessInterceptor

As a resultCode of notpermitted is returned to a request made without the appropriate privilege, a new ActionInvocation is formed and objectified based on this result, and interceptors are called again on this object. The reason the OGNL Injection attack occurs is because OGNL expressions are interpreted without any verification process while objectifying the ActionInvocation following the notpermitted result.

OGNL Injection

The CVE-2022-26134 vulnerability interprets OGNL expressions in namespace of URIs without verification.

Figure 8 shows a simple process of OGNL Injection. A code with malicious intent can be written into OGNL expressions to affect the server system.

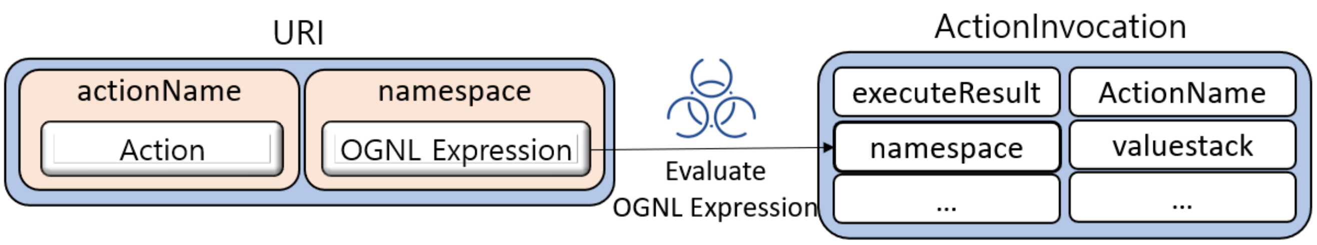


Figure 8. OGNL Injection process

Object-Graph Navigation Language (OGNL) is a simple expression that allows access to Java objects. Thus, the following tasks are possible.

- Accessing object properties
- Accessing variables
- Accessing arrays
- Calling methods
- Operations (arithmetic, logical, comparison, etc.)

These features can be combined, so almost all tasks that can be done through Java are available.

When XWork processes the OGNL expression read from a user request, it first checks whether the object properties contain contents that match the expression. XWork saves the properties of the object in a place called “ValueStack”. If a property that matches the expression exists in this ValueStack, the value of the property is returned, and if not, the expression is executed as-is and the result is returned. Executing the expression as-is means that accessing variables and arrays, or calling methods becomes possible. Figure 9 shows the findValue method process that interprets OGNL expressions.

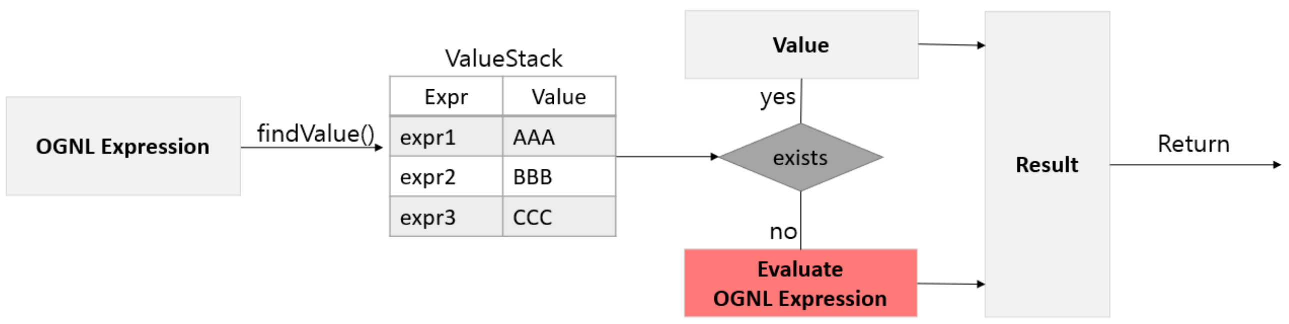
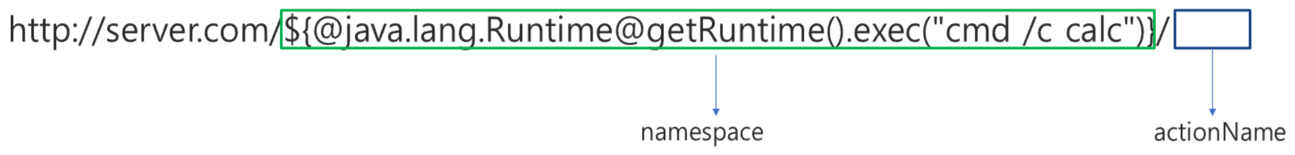


Figure 9. OGNL expression interpretation process

The fundamental cause behind the CVE-2022-26134 vulnerability is that the URI namespace field, which can be manipulated as the client wishes, can contain OGNL expressions; the server can use these expressions as-is without verifying whether they are safe. Thus, an attack code can be executed during the process of interpreting OGNL expressions.

Proof of Concept

The Proof of Concept (PoC) URI is shown in Figure 10.



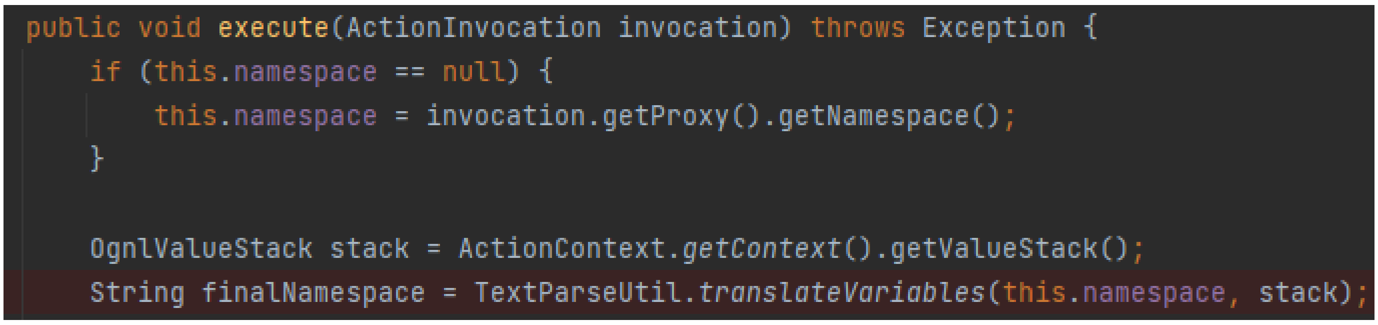
```
http://server.com/${@java.lang.Runtime@getRuntime().exec("cmd /c calc")}/
```

Figure 10. CVE-2022-26134 PoC URI

The namespace contains an OGNL expression, and the actionName is blank. If there is no actionName, Confluence processes this as the basic Index Action.

Index Action is an action that requests for the main page. As threat actors would not be logged into the Confluence server, they do not have access. As a result, “notpermitted” resultCode is returned from ConfluenceAccessInterceptor which checks the privilege when making an ActionInvocation and calling various interceptors. Because a resultCode exists, a new ActionInvocation is made on this result.

Code 3 shows the details of the class that makes an ActionInvocation when the resultCode is notpermitted.

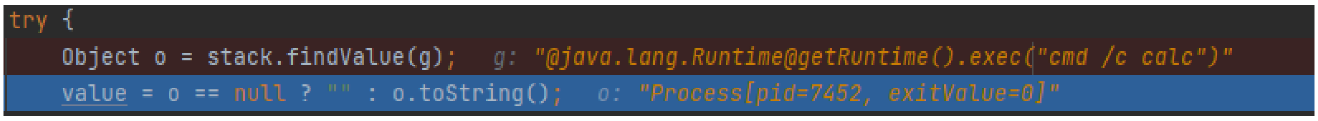


```java
public void execute(ActionInvocation invocation) throws Exception {
    if (this.namespace == null) {
        this.namespace = invocation.getProxy().getNamespace();
    }

    OgnlValueStack stack = ActionContext.getContext().getValueStack();
    String finalNamespace = TextParseUtil.translateVariables(this.namespace, stack);
```

Code 3. ActionInvocation object creation process within the DefaultActionInvocation.class

The translateVariables method is called. The first argument is the namespace in the client request URI, and the second argument is the ValueStack containing the object properties.

Code 4 shows the contents of translateVariables. If we look at the findValue method, it is checking for whether a property that matches the OGNL expression exists in the ValueStack.

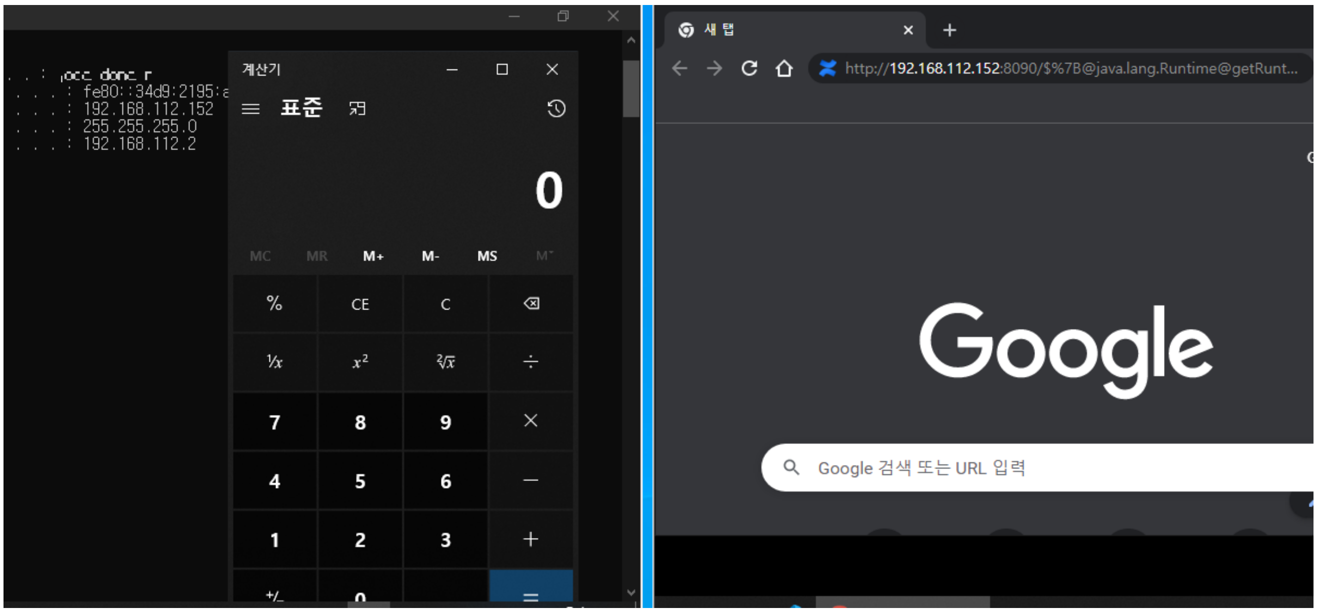


```java
try {
    Object o = stack.findValue(g);   g: "@java.lang.Runtime@getRuntime().exec("cmd /c calc")"
    value = o == null ? "" : o.toString();   o: "Process[pid=7452, exitValue=0]"
```

Code 4. OGNL expression contained within variable “g” and the execution result

The expression contains the exec method which executes processes, and thus this property cannot be found in the ValueStack. Therefore, the expression is executed as-is, and the result is returned. We can see that the process has been executed normally and the Process ID has been returned. If an attack code was written into the expression, this code would have been executed as written.

Figure 11 below shows the transmission of the PoC URI to the server, where the client uses the java.lang.Runtime@getRuntime().exec() method to run Calculator.



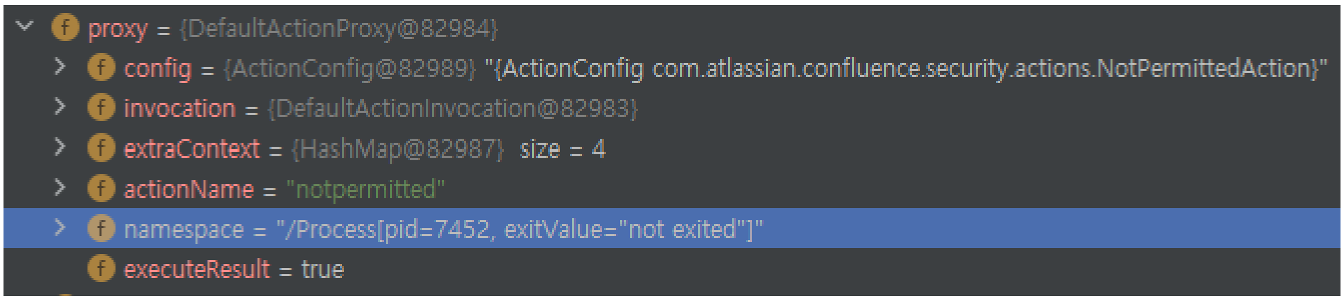
```
fe80::34d9:2195:a
192.168.112.152
255.255.255.0
192.168.112.2
```

```
http://192.168.112.152:8090/$%7B@java.lang.Runtime@getRunt...
```

Figure 11. Possible to execute arbitrary processes on the server

The right is the client and the left is the server environment. We can see the Calculator process has been run on the server.

Figure 12 shows the ActionInvocation object member that was newly made after the OGNL expression interpretation process.



```
proxy = {DefaultActionProxy@82984}
  config = {ActionConfig@82989} "{ActionConfig com.atlassian.confluence.security.actions.NotPermittedAction}"
  invocation = {DefaultActionInvocation@82983}
  extraContext = {HashMap@82987}  size = 4
  actionName = "notpermitted"
  namespace = "/Process[pid=7452, exitValue="not exited"]"
  executeResult = true
```

Figure 12. Execution result of the OGNL expression in the namespace

We can see that the execution result of the OGNL expression being used again as a namespace. The original intent of XWork seems to be the following: the client sends a request with a namespace containing an OGNL expression for which a matching value is sought in the ValueStack, then the resulting value in turn is made into a namespace and an ActionInvocation formed from this.

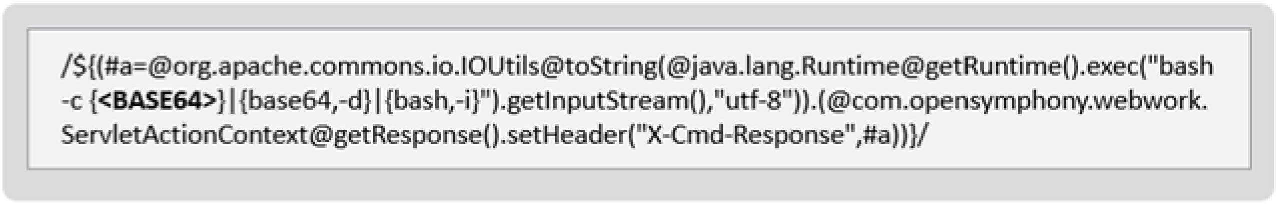
However, threat actors can freely include OGNL expressions in the namespace of the URI, and as a property matching this expression cannot be found in the ValueStack, an arbitrary method can be executed. Therefore, threat actors can execute malicious codes that perform process execution, web shell uploads, modification of system variables, and persistence maintenance without restriction.

Confluence Attack Cases

Recently, vulnerabilities of Atlassian's Confluence Server and Confluence Data Center are continuously being discovered, [1] and attacks using these vulnerabilities are also continuing. Confluence is a server that provides a common space for work and sharing. Thus, corporations and organizations are the main victims of these attacks, and there have been identified attacks against vulnerable Confluence servers of multiple corporations.

CVE-2022-26134

On June 4, 2022, a proof-of-concept code of the CVE-2022-26134 vulnerability was published by a security researcher.[2] A few days following this disclosure, many threat groups utilized this vulnerability. Among them, 8220 Gang abused the vulnerability in their initial access stage of the attack.



```
/${(#a=@org.apache.commons.io.IOUtils@toString(@java.lang.Runtime@getRuntime().exec("bash
-c {<BASE64>}|{base64,-d}|{bash,-i}").getInputStream(),"utf-8")).(@com.opensymphony.webwork.
ServletActionContext@getResponse().setHeader("X-Cmd-Response",#a))}/
```

Figure 13. 8220 Gang's attack Request

Source: https://blog.checkpoint.com/2022/06/09/crypto-miners-leveraging-atlassian-zero-day-vulnerability/

The attack case of 8220 Gang published by CheckPoint[3] shows the transmittal of a malicious request after a vulnerable Confluence server is found as seen in Figure 13. Out of the OGNL expressions, the part where the actual malicious behavior is performed is encoded in BASE64.

[1] https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=confluence

[2] https://github.com/Nwqda/CVE-2022-26134

[3] https://blog.checkpoint.com/2022/06/09/crypto-miners-leveraging-atlassian-zero-day-vulnerability/

When this expression reaches the server, it is decoded and executed.

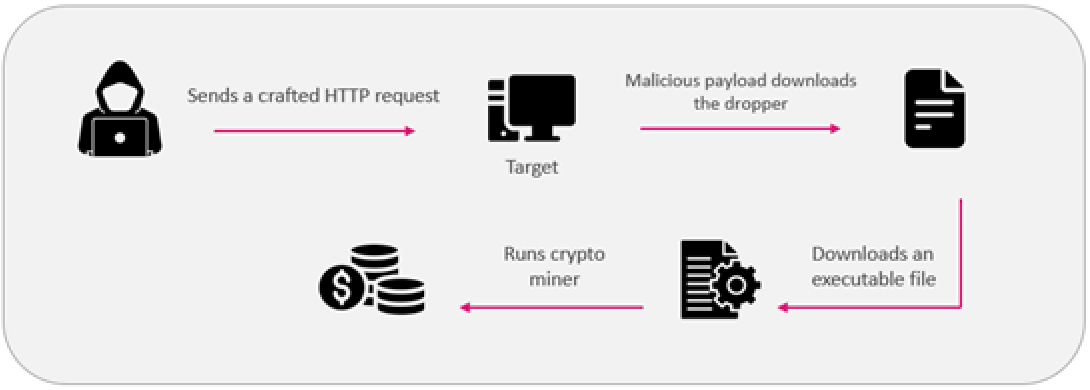

Sends a crafted HTTP request
Target
Malicious payload downloads the dropper
Downloads an executable file
Runs crypto miner

Figure 14. 8220 Gang's attack flow
Source: https://blog.checkpoint.com/2022/06/09/crypto-miners-leveraging-atlassian-zero-day-vulnerability/

Ultimately, Monero CoinMiner (XMRig) is executed. There have been identified cases of attacks exploiting the CVE-2022-26134 vulnerability not just overseas but also in Korea. More attack cases can be found on the ASEC blog.[4]

CVE-2021-26084

The CVE-2021-26084 vulnerability, which also allows OGNL Injection attack on Confluence servers, has been disclosed in the past. Attacks using this vulnerability have a similar method to the ones using CVE-2022-26134.

According to a report published by Microsoft,[5] earlier this year, Lapsus$ group exploited this vulnerability to search for accounts and credentials with access to sensitive information.

As such, the CVE-2022-26134 vulnerability shows a potential to be used in attacks for a long time, just like the CVE-2021-26084 vulnerability. On this vulnerability, AhnLab TIP had published a CVE-2021-26084 vulnerability analysis report. Please refer to the <Analysis Report

[4] https://asec.ahnlab.com/en/36820/

[5] https://www.microsoft.com/security/blog/2022/03/22/dev-0537-criminal-actor-targeting-organizations-for-data-exfiltration-and-destruction/

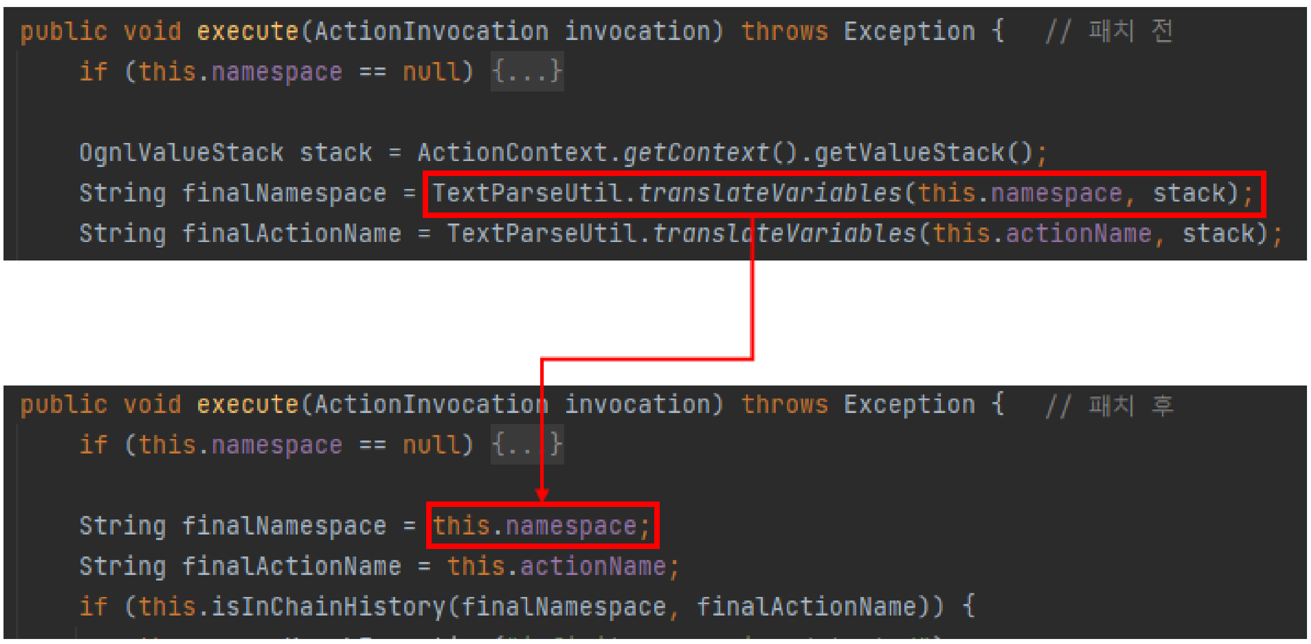
on ATIP CVE-2021-26084 Vulnerability> for more details.[6]

[6] https://atip.ahnlab.com/ti/contents/issue-report/vulnerability?i=4397d090-94e2-407a-b30c-99d517faa66c

Vulnerability Patch

The CVE-2022-26134 vulnerability was reported to Atlassian on May 31, 2022, and a security patch was released on June 2, 2022. The details show that the vulnerability was mitigated largely through two methods.

• Using the string of the namespace value as it is



```
public void execute(ActionInvocation invocation) throws Exception {   // 패치 전
    if (this.namespace == null) {...}

    OgnlValueStack stack = ActionContext.getContext().getValueStack();
    String finalNamespace = TextParseUtil.translateVariables(this.namespace, stack);
    String finalActionName = TextParseUtil.translateVariables(this.actionName, stack);
```

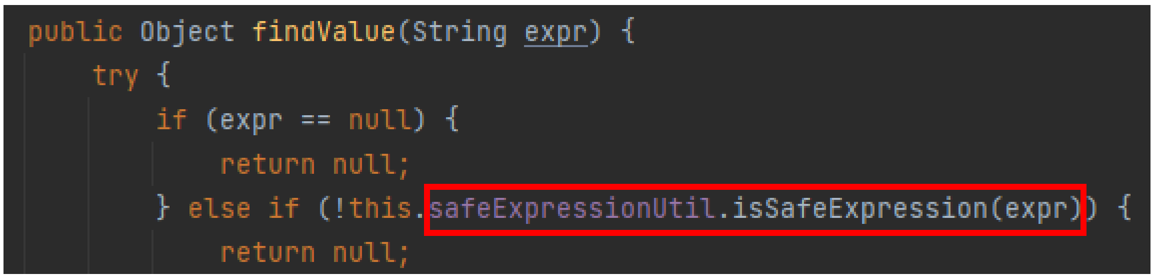
```
public void execute(ActionInvocation invocation) throws Exception {   // 패치 후
    if (this.namespace == null) {...}

    String finalNamespace = this.namespace;
    String finalActionName = this.actionName;
    if (this.isInChainHistory(finalNamespace, finalActionName)) {
```

Code 5. translateVariables not being used

In Code 5, the top image is before the patch and the bottom is after the patch. The patch made a change so that when a new ActionInvocation is made following the notpermitted result, the translateVariables method containing the process of interpreting OGNL expressions is not used but rather the namespace string is used as-is. Thus, even if a threat actor includes an OGNL expression in the URI, it is only regarded as a string and not interpreted and executed.

• Verifying expressions

The findValue method is used not only by ConfluenceAccessInterceptor, but also by many other interceptors to interpret OGNL expressions. A syntax that verifies expressions was added to the top of this method.



```
public Object findValue(String expr) {
    try {
        if (expr == null) {
            return null;
        } else if (!this.safeExpressionUtil.isSafeExpression(expr)) {
            return null;
```

Code 6. Verification of OGNL expressions

With the isSafeExpression method shown in Code 6, a scan is run to see whether an expression contains packages, properties, or methods that can affect the system.

unsafePackageNames	ognl java.io java.net org.apache.velocity java.sql javax.sql org.apache.commons.exec org.apache.commons.io org.apache.velocity org.apache.tomcat …
unsafePropertyNames	java.lang.Runtime java.lang.System java.lang.ClassLoader java.lang.Shutdown java.lang.ProcessBuilder java.lang.Thread java.lang.Package java.lang.Processorg.apache.tomcat …

Table 3. Packages and properties verified by the isSafeExpression method

We can see that it includes org.apache.commons.io, java.lang.Runtime, and java.lang.System which were used in the 'Confluence Attack Cases' in this report.

Vulnerability Update

On June 2, 2022, a package that has resolved the CVE-2022-26134 vulnerability and a version of Confluence with this package applied was released. Two measures of countermeasures are proposed.

Temporary Mitigation

If it is not possible to update Confluence immediately, the file that is the cause of the vulnerability can be exchanged as a temporary solution.

Version	File
7.15.0 - 7.18.0	https://packages.atlassian.com/maven-internal/opensymphony/xwork/1.0.3-atlassian-10/xwork-1.0.3-atlassian-10.jar
6.0.0 - 7.14.2	https://packages.atlassian.com/maven-internal/opensymphony/xwork/1.0.3-atlassian-10/xwork-1.0.3-atlassian-10.jar https://packages.atlassian.com/maven-internal/opensymphony/webwork/2.1.5-atlassian-4/webwork-2.1.5-atlassian-4.jar https://confluence.atlassian.com/doc/files/1130377146/1137639562/3/1654274890463/CachedConfigurationProvider.class

Table 4. Replaced files

The version update includes patches for other security issues aside from the CVE-2022-26134 vulnerability, so it is recommended to run the version update instead of switching out only the files. More details on the temporary mitigation measures are available in the Confluence security advice.[7]

Version Update

Fixed Version
7.4.17 or later
7.13.7 or later
7.14.3 or later

[7] https://confluence.atlassian.com/doc/confluence-security-advisory-2022-06-02-1130377146.html

7.15.2 or later
7.16.4 or later
7.17.4 or later
7.18.1 or later
7.19.0 or later

Table 5. Products with the CVE-2022-26134 vulnerability patched

You can download the latest version from the official Confluence website.[8] The website also provides a list of security checks that must be performed after installing the new update, as well as how to perform said checks.[9] It is recommended to run the check to prevent attacks and minimize damage.

The Confluence Server and Confluence Data Center versions that are currently in use can be seen at the bottom of the page after connecting to the server.

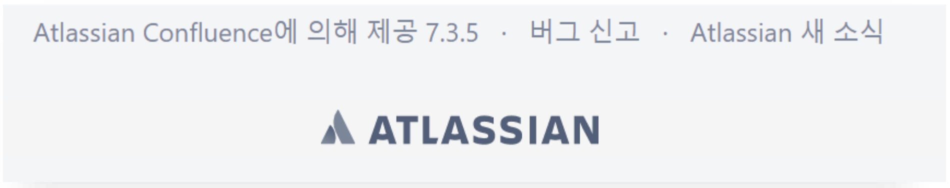

Atlassian Confluence에 의해 제공 7.3.5 · 버그 신고 · Atlassian 새 소식
ATLASSIAN

Figure 15. Checking the Confluence Server version

[8] https://www.atlassian.com/ko/software/confluence

[9] https://confluence.atlassian.com/doc/confluence-7-19-release-notes-1141976784.html#Confluence7.19ReleaseNotes-Securitycheck-up

AhnLab Response Overview

The aliases and the engine version information of AhnLab products are shown below.

File Diagnosis
8220 Gang
- Downloader/PS.Miner.R226842 (2022.07.15.00)
- Downloader/Win.MSIL.R504724 (2022.07.15.00)
- Trojan/Win.Generic.C5154950 (2022.06.02.02)
- Trojan/Win64.XMR-Miner (2019.12.11.01)

Network Detection
AIPS/HIPS/MDS
- Atlassian Confluence Server and Data Center xwork OGNL Injection-1
- Atlassian Confluence Server and Data Center xwork OGNL Injection-2
- Atlassian Confluence Server and Data Center xwork OGNL Injection-3
- Atlassian Confluence Server and Data Center xwork OGNL Injection-4

Indicators Of Compromise (IOC)

File Hashes (MD5)

The MD5 of the related files are as follows. (However, sensitive samples may have been excluded.)

```
51ac2e4df1978c3fadaf3654f0f91462
8e211d1701e0e16cd30a414f5e5a384c
af0b85c176c7c32f0e9585b7eeaa6629
dbda412cf6bf74af449ecb0b3bac7aa8
```

References

[1] OGNL Expression Injection
https://vulncat.fortify.com/ko/detail?id=desc.dataflow.java.ognl_expression_injection

[2] XWork Document
https://svn.apache.org/repos/asf/struts/attic/xwork/trunk/docs/wikidocs/Documentation.html

[3] Confluence Class Description - ConfluenceAccessInterceptor
https://docs.atlassian.com/ConfluenceServer/javadoc/6.7.0/com/atlassian/confluence/security/interceptors/ConfluenceAccessInterceptor.html

[4] ActionInvocation Description
http://devdoc.net/javaweb/struts/Struts_2.3.8-site/xwork-core/apidocs/com/opensymphony/xwork2/ActionInvocation.html

[5] NVD CVE-2022-26134 Detail
https://nvd.nist.gov/vuln/detail/CVE-2022-26134

More security, More freedom

AhnLab, Inc.

220, Pangyoyeok-ro, Bundang-gu, Seongnam-si, Gyeonggi-do, Korea
Tel : +82 31 722 8000 | Fax : +82 31 722 8901
www.ahnlab.com
www.asec.ahnlab.com/en

About ASEC

AhnLab Security Emergency Response Center(ASEC), through our team of highly skilled cyber threat analysts and incident responders, delivers timely and accurate threat intelligence and state-of-the-art response on a global scale. The ASEC provides the most contextual and relevant threat intelligence backed by our groundbreaking research on malware, vulnerabilities, and threat actors to help the global community stay ahead of evolving cyber-attacks.

About AhnLab

AhnLab is a leading cybersecurity company with a reliable reputation for delivering advanced cyber threat intelligence and threat detection and response (TDR) capabilities with cutting-edge technology. We offer a cybersecurity platform comprised of purpose-built products securing endpoint, network, and cloud, which ensures extended threat visibility, actionable insight, and optimal response. Our best-in-class researchers and development professionals are always fully committed to bringing our security offerings to the next level and future-proofing our customers' business innovation against cyber risks.