

# KNOCKING DOWN THE BIG DOOR

**Breaking Authentication and Segregation of  
Production and Non-Production Environments**

Buenos Aires, 27 de Abril 2018

Nahuel Grisolia  
Cinta Infinita, Founder / CEO  
@cintainfinita  
[nahuel@cintainfinita.com.ar](mailto:nahuel@cintainfinita.com.ar)





# nahuel@cintainfinita\$ whoami

- Cinta Infinita Founder and CEO
- (Web) Application Security specialist & enthusiast
- Many vulnerabilities discovered in Open Source and Commercial software: Vmware, Websense, OSSIM, Cacti, McAfee, Oracle VM, etc.
- Gadgets and Electronics Lover (RFID!)
- <http://ar.linkedin.com/in/nahuelgrisolia>
- <http://cintainfinita.com>
- <http://www.exploit-db.com/author/?a=2008>
- <http://www.proxmark.org/forum/profile.php?id=3000>





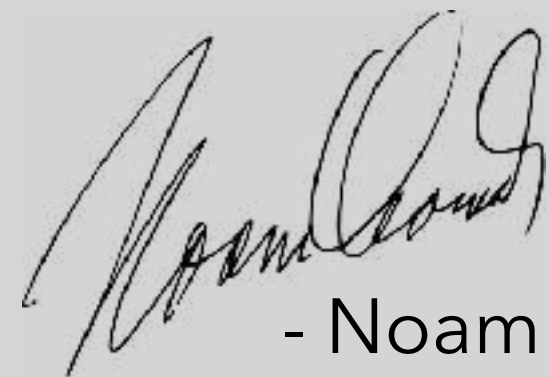
# MOTIVATION

*“The Purpose of Education” - Enlightenment Sense*

“The highest goal in life is to inquire and create”

“Education is really aimed at helping students get to the point where they can learn on their own”


“It’s you the learner who is going to achieve in the course of education and it’s really up to you to determine how you’re going to master and use it.”

A handwritten signature in black ink, appearing to read 'Noam Chomsky', with a stylized, cursive script.

- Noam Chomsky

# MOTIVATION

*“The Purpose of Education” - Enlightenment Sense*

 RadioFreeEurope  
RadioLiberty

**Hackers are free people, just like artists who wake up in the morning in a good mood and start painting.**

# MOTIVATION

*“The Purpose of Education” - Enlightenment Sense*

RadioFreeEurope  
RadioLiberty



Hackers are free people, just like artists who wake up in the morning in a good mood and start painting.





FUCK THE SYSTEM

# Agenda



The Old Toad

He lives under the stone. He has been asleep all winter. The warm sun called to him, and he awoke from his long winter nap.



# Agenda

- ★ Introduction (boring but necessary)



The Old Toad

He lives under the stone. He has been asleep all winter. The warm sun called to him, and he awoke from his long winter nap.



# Agenda

- ★ Introduction (boring but necessary)
- ★ Case 1: Be careful while **impersonating users**. Seriously



The Old Toad

He lives under the stone. He has been asleep all winter. The warm sun called to him, and he awoke from his long winter nap.

# Agenda

- ★ Introduction (boring but necessary)
- ★ Case 1: Be careful while **impersonating users**. Seriously
- ★ Case 2: **Authentication Bypass** vulnerability in the **Auth0** platform



The Old Toad

He lives under the stone. He has been asleep all winter. The warm sun called to him, and he awoke from his long winter nap.



# Agenda

- ★ Introduction (boring but necessary)
- ★ Case 1: Be careful while **impersonating users**. Seriously
- ★ Case 2: **Authentication Bypass** vulnerability in the **Auth0** platform
- ★ Case 3: **Observations** in MS Azure and IIS installations running .NET Web Applications using **SAML** Authentication  
**Machine Keys**? Is that a new rock band?



The Old Toad

He lives under the stone. He has been asleep all winter. The warm sun called to him, and he awoke from his long winter nap.

# Agenda

- ★ Introduction (boring but necessary)
- ★ Case 1: Be careful while **impersonating users**. Seriously
- ★ Case 2: **Authentication Bypass** vulnerability in the **Auth0** platform
- ★ Case 3: **Observations** in MS Azure and IIS installations running .NET Web Applications using **SAML Authentication Machine Keys**? Is that a new rock band?
- ★ Final Conclusions & Recommendations



The Old Toad

He lives under the stone. He has been asleep all winter. The warm sun called to him, and he awoke from his long winter nap.



# Agenda

- ★ Introduction (boring but necessary)
- ★ Case 1: Be careful while **impersonating users**. Seriously
- ★ Case 2: **Authentication Bypass** vulnerability in the **Auth0** platform
- ★ Case 3: **Observations** in MS Azure and IIS installations running .NET Web Applications using **SAML** Authentication  
**Machine Keys**? Is that a real band?
- ★ Final Conclusions & Recommendations



The Old Toad



...es under the stone. He has  
...all winter. The warm  
...m, and he awoke from  
...nap.





# Authentication (AuthN)

Restrictions on Who (or What) can Access a System



## Authentication (AuthN)

Restrictions on Who (or What) can Access a System

YOU SHALL



NOT PASS



## Authorization (AuthZ)

Restrictions on Actions of Authenticated Users

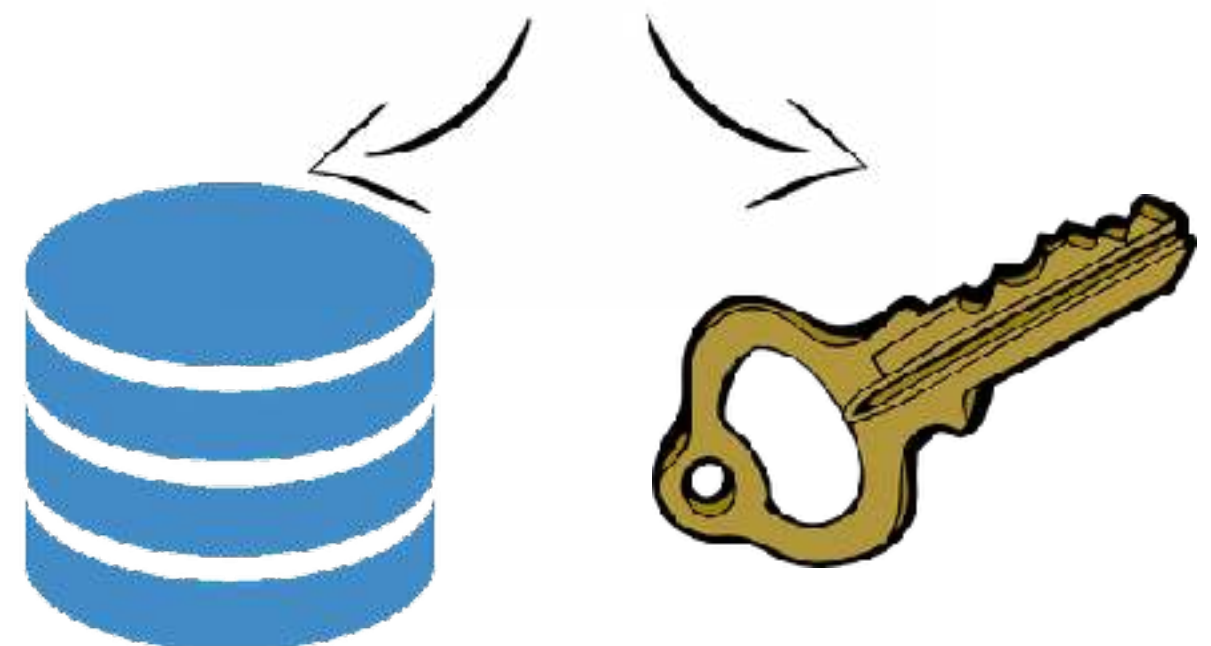




## We usually Pentest in Staging / Development Environments

**Full Isolation / Complete Segregation between Environments?**

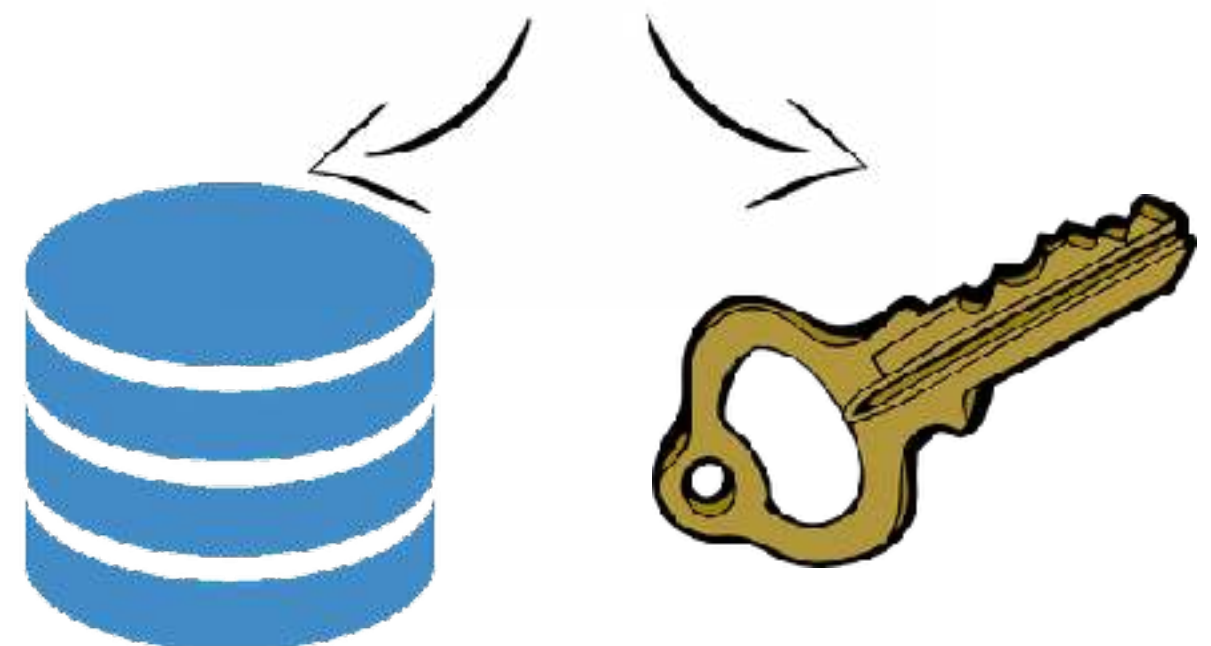
**Shared Secrets? Which secrets exactly?  
Shared Databases?**



## **We usually Pentest in Staging / Development Environments**

**Full Isolation / Complete Segregation  
between Environments?**

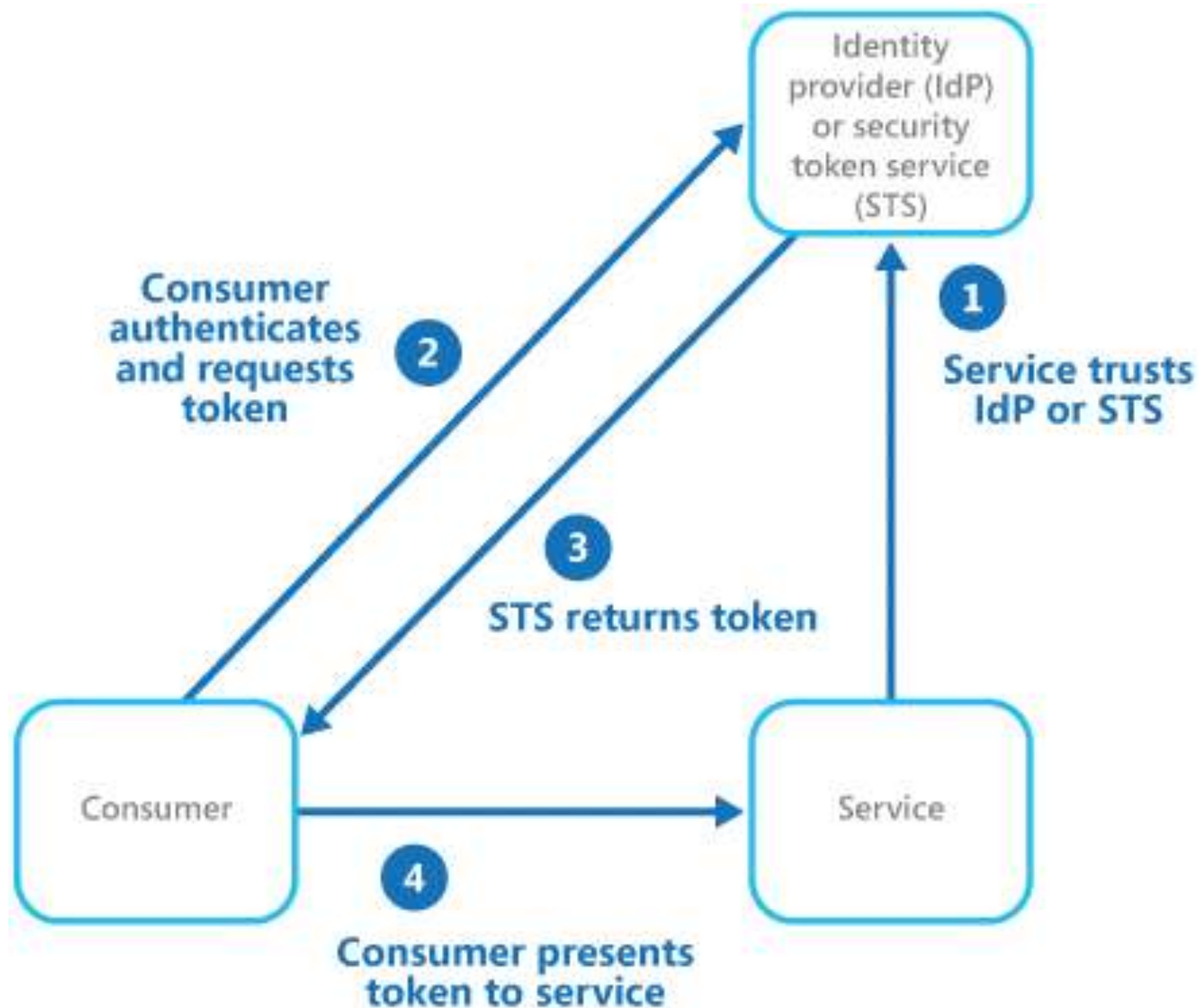
**Shared Secrets? Which secrets exactly?  
Shared Databases?**





# Federated Identity pattern

“Delegate authentication to an external identity provider”



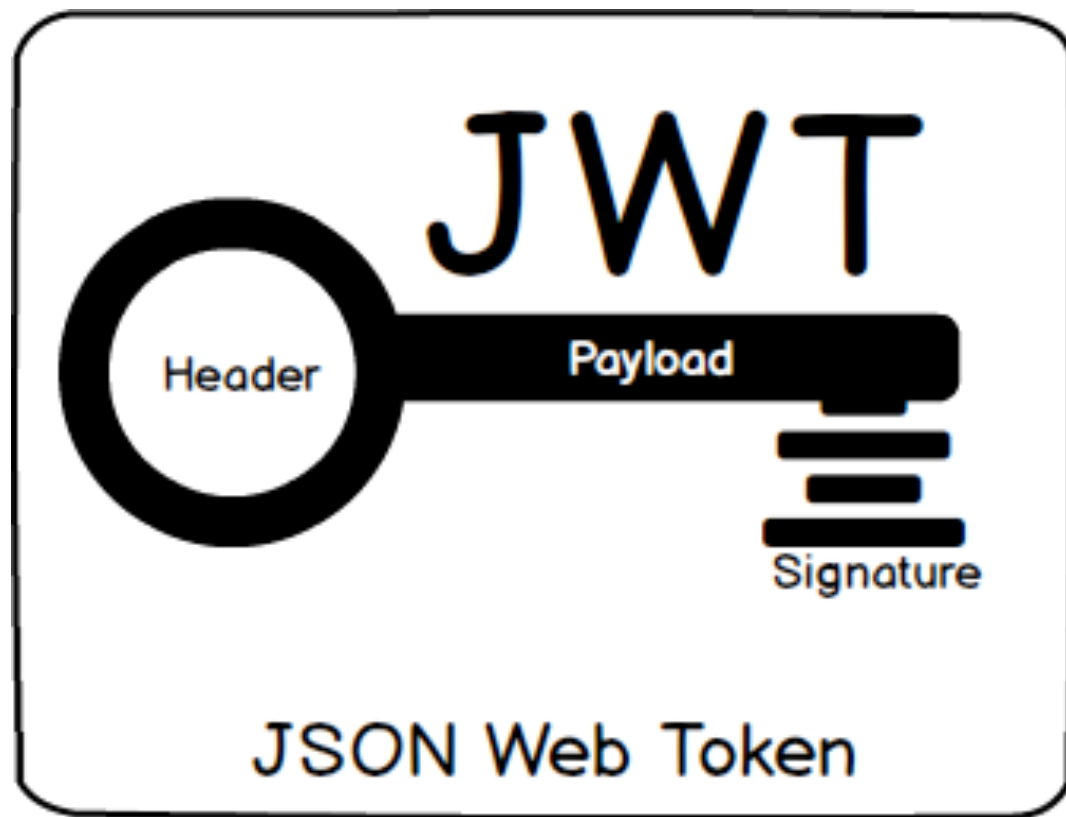
ALGORITHM

HS256

## Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.XbPfbIHM  
I6arZ3Y922BhjWgQzWXcXNrZ8ogtVhfEd2o
```



## Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM &amp; TOKEN TYPE

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

PAYLOAD: DATA

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "iat": 1516239022  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  secret  
) ☐ secret base64 encoded
```

Signature Verified

<https://jwt.io>



# Security Assertion Markup Language (SAML)

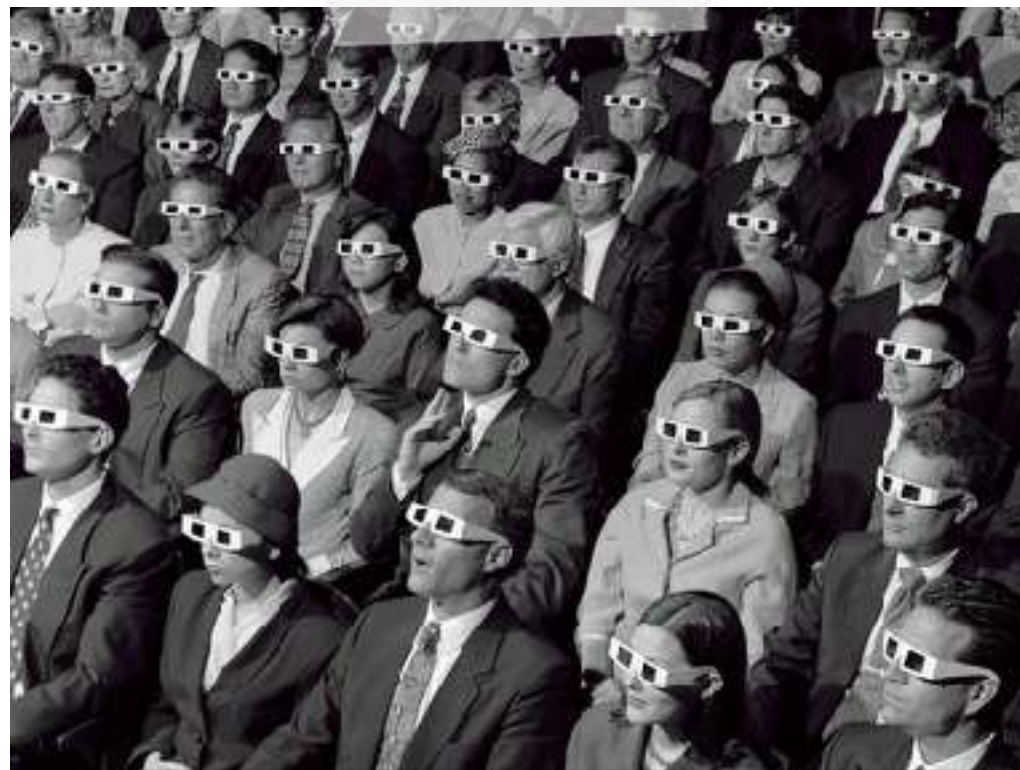
“XML-based framework for communicating user authentication, entitlement, and attribute information”



**Signed**

*Based  
Signature*

**Audience**

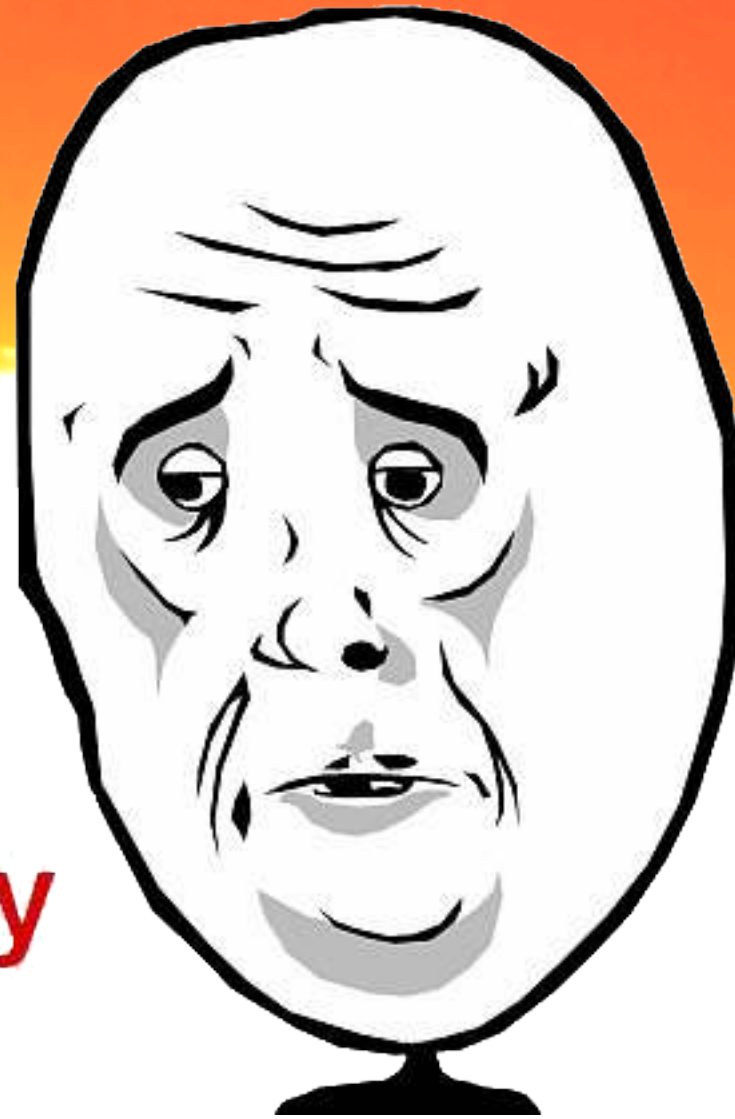


**And more...**



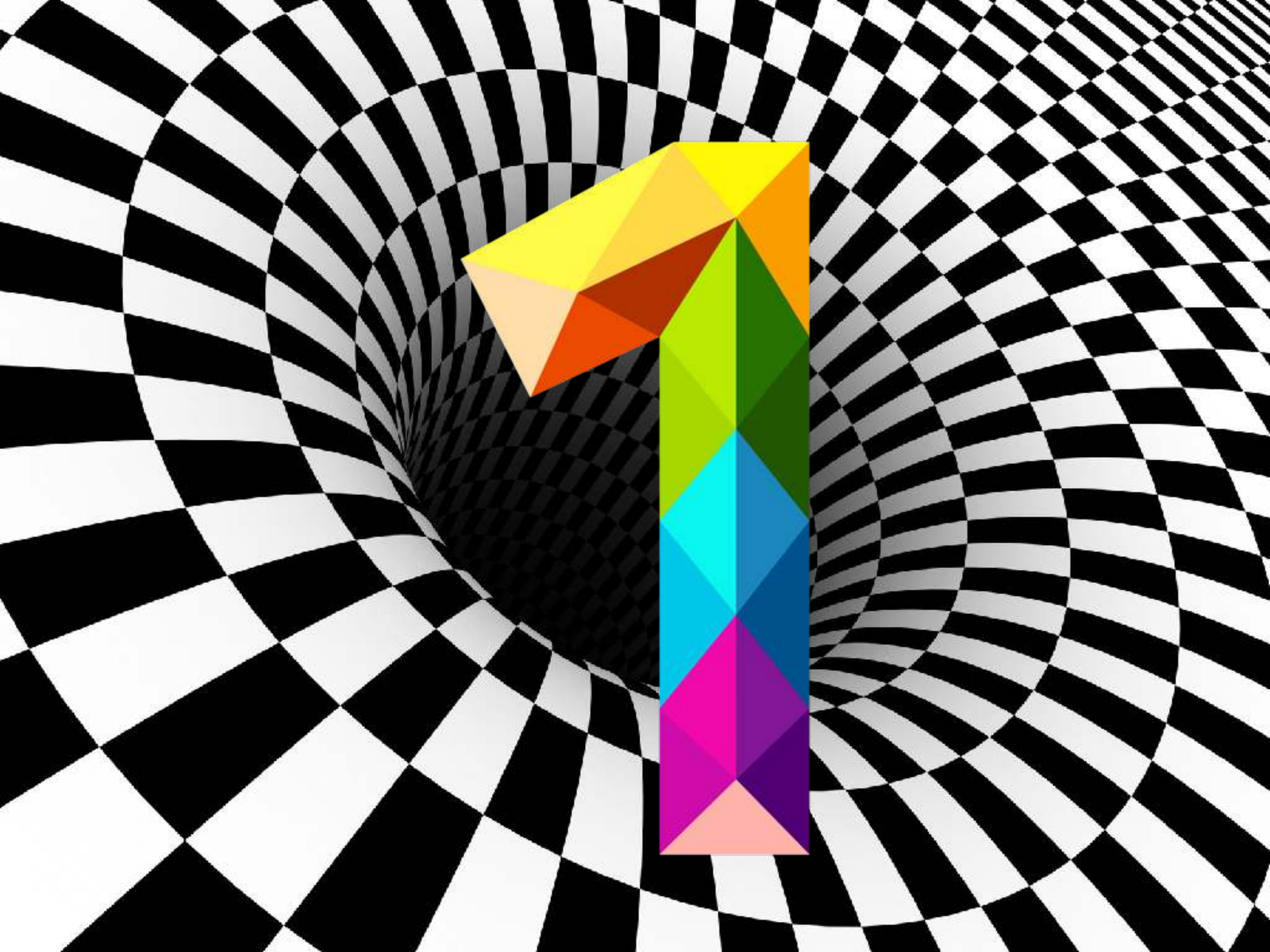
NOTHING NEW HERE





Okay

NOTHING NEW HERE





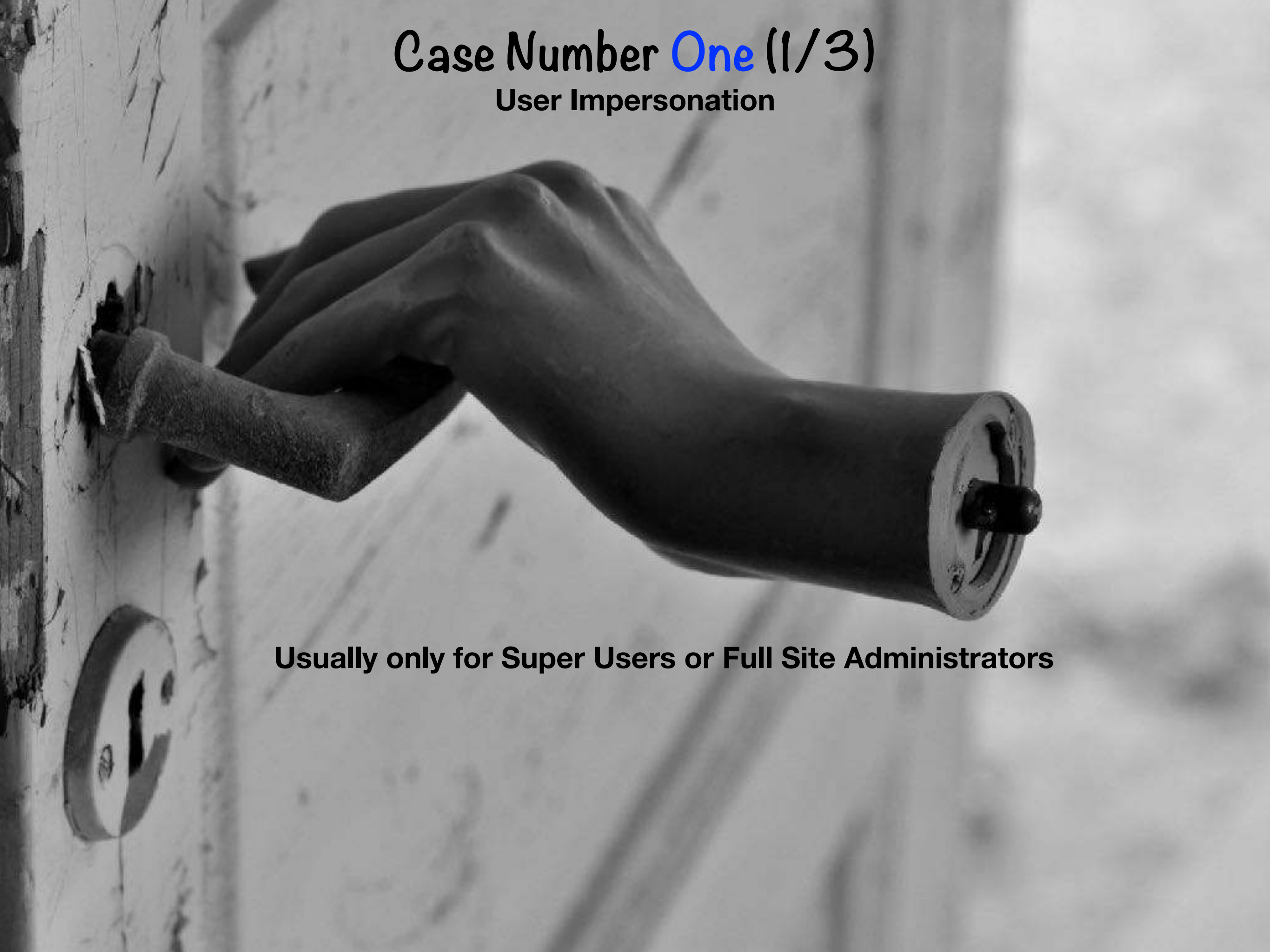
# Case Number **One** (1/3)

## User Impersonation



# Case Number **One** (1/3)

## User Impersonation

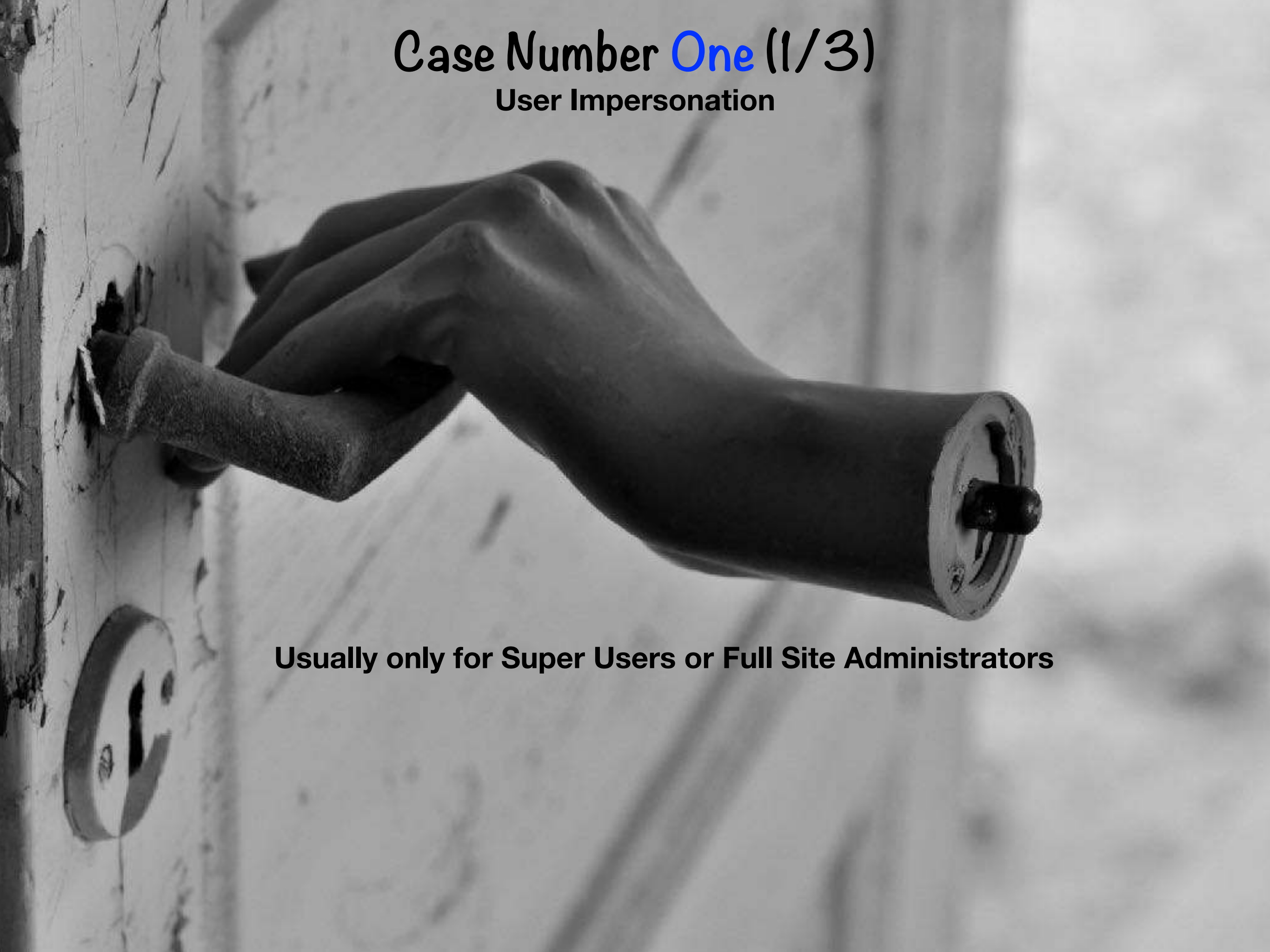


**Usually only for Super Users or Full Site Administrators**



# Case Number **One** (1/3)

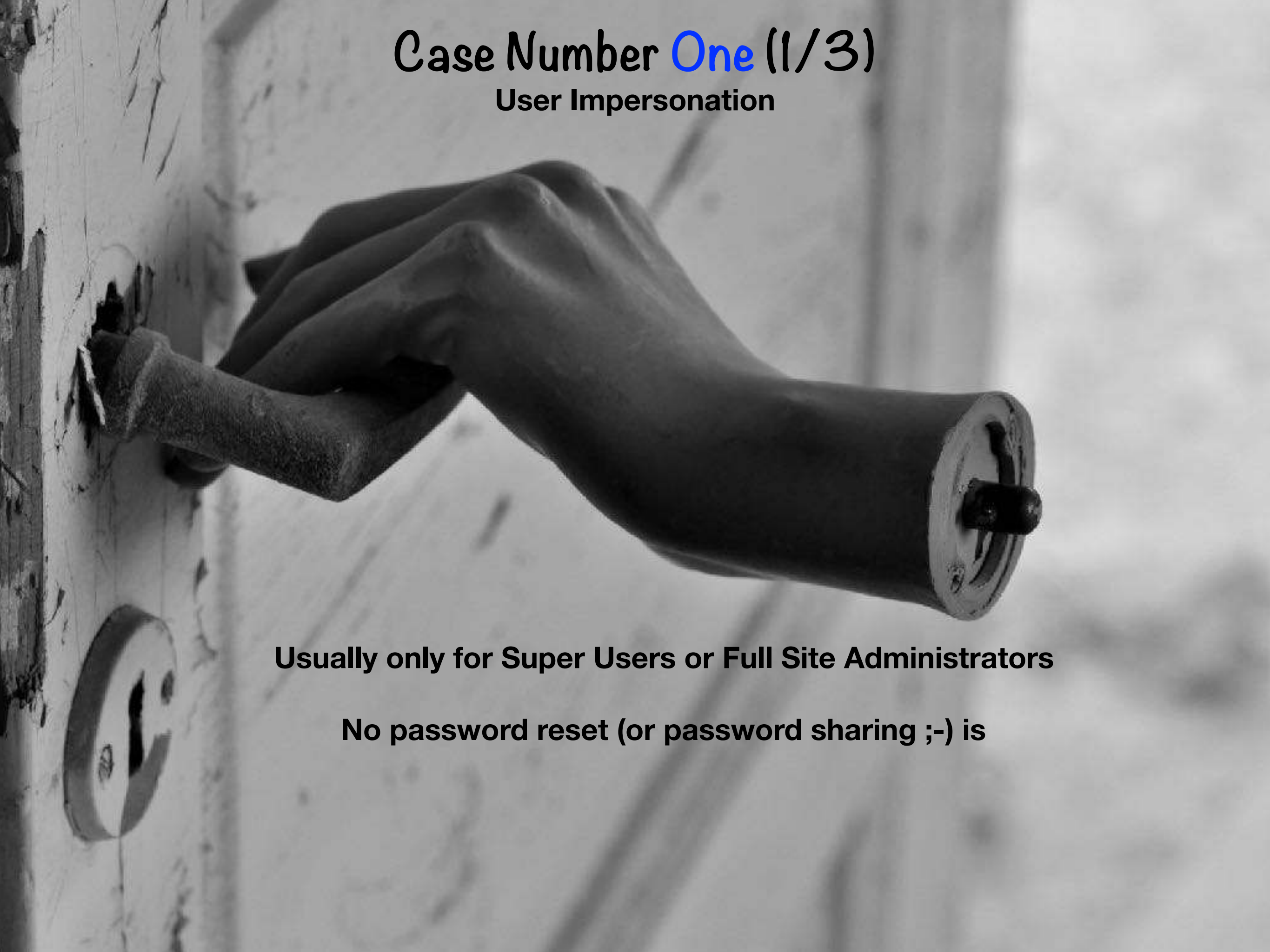
## User Impersonation



**Usually only for Super Users or Full Site Administrators**

# Case Number **One** (1/3)

## User Impersonation

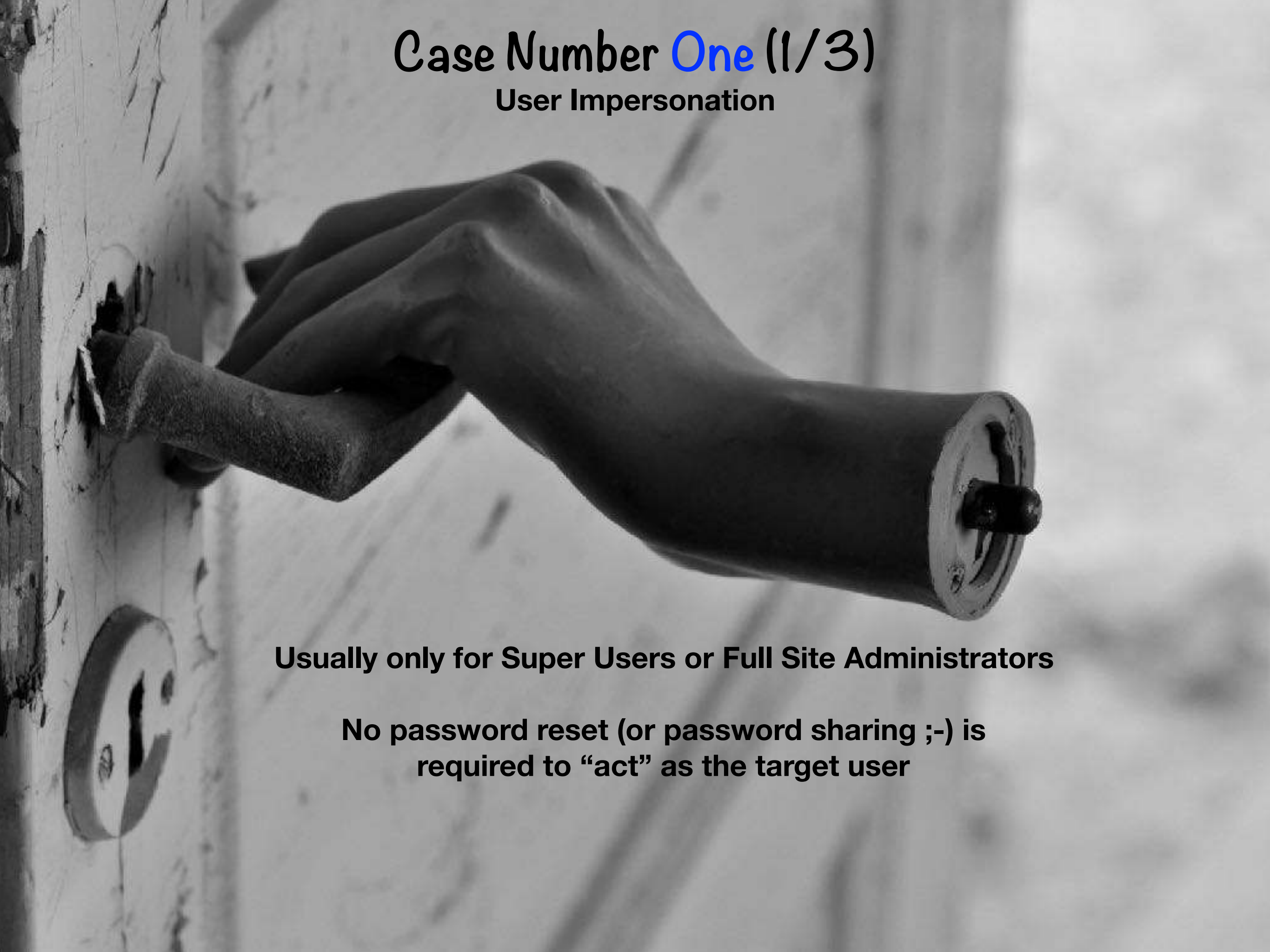


**Usually only for Super Users or Full Site Administrators**

**No password reset (or password sharing ;-)** is

# Case Number **One** (1/3)

## User Impersonation



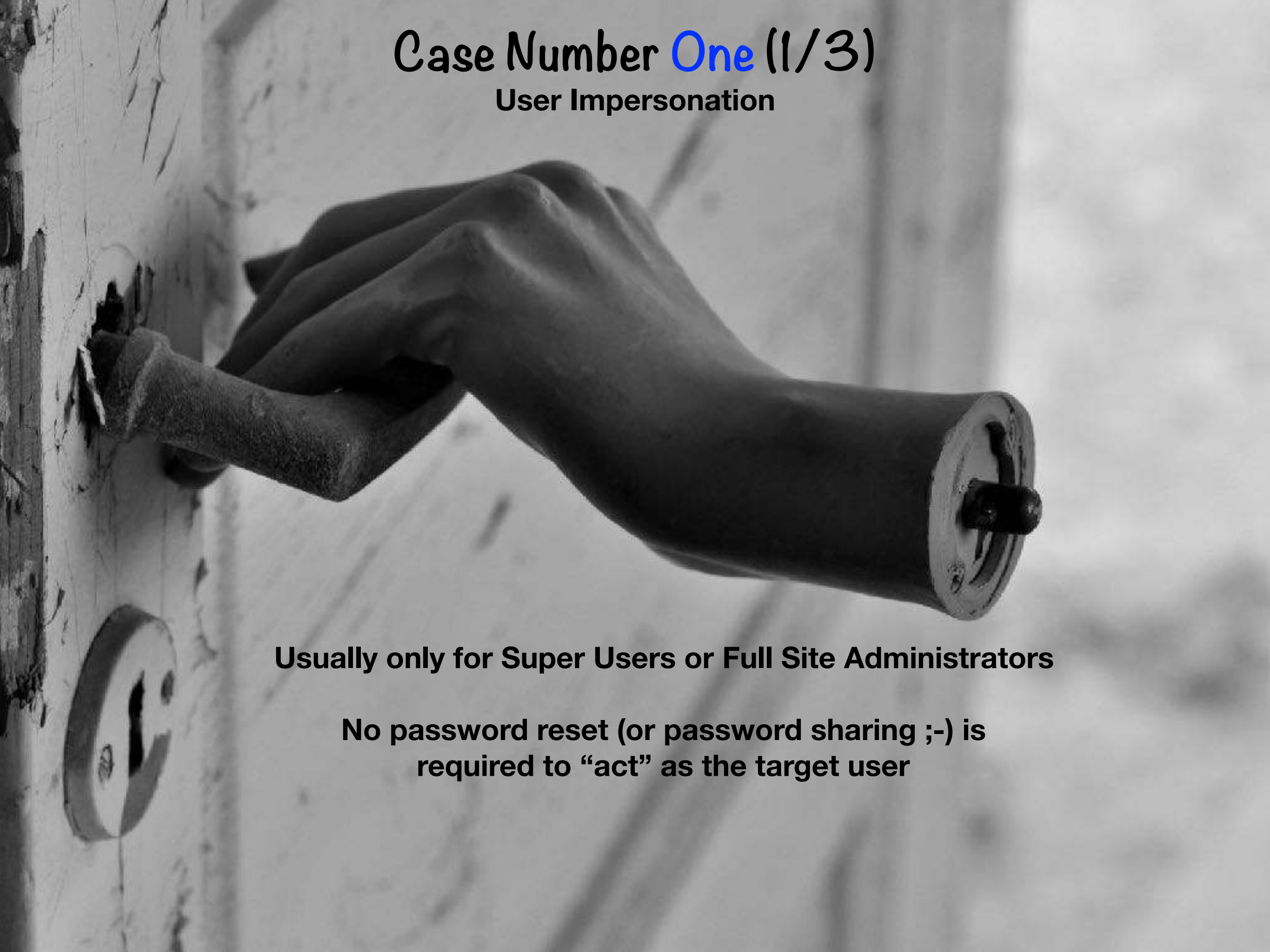
**Usually only for Super Users or Full Site Administrators**

**No password reset (or password sharing ;- ) is  
required to “act” as the target user**



# Case Number **One** (1/3)

## User Impersonation

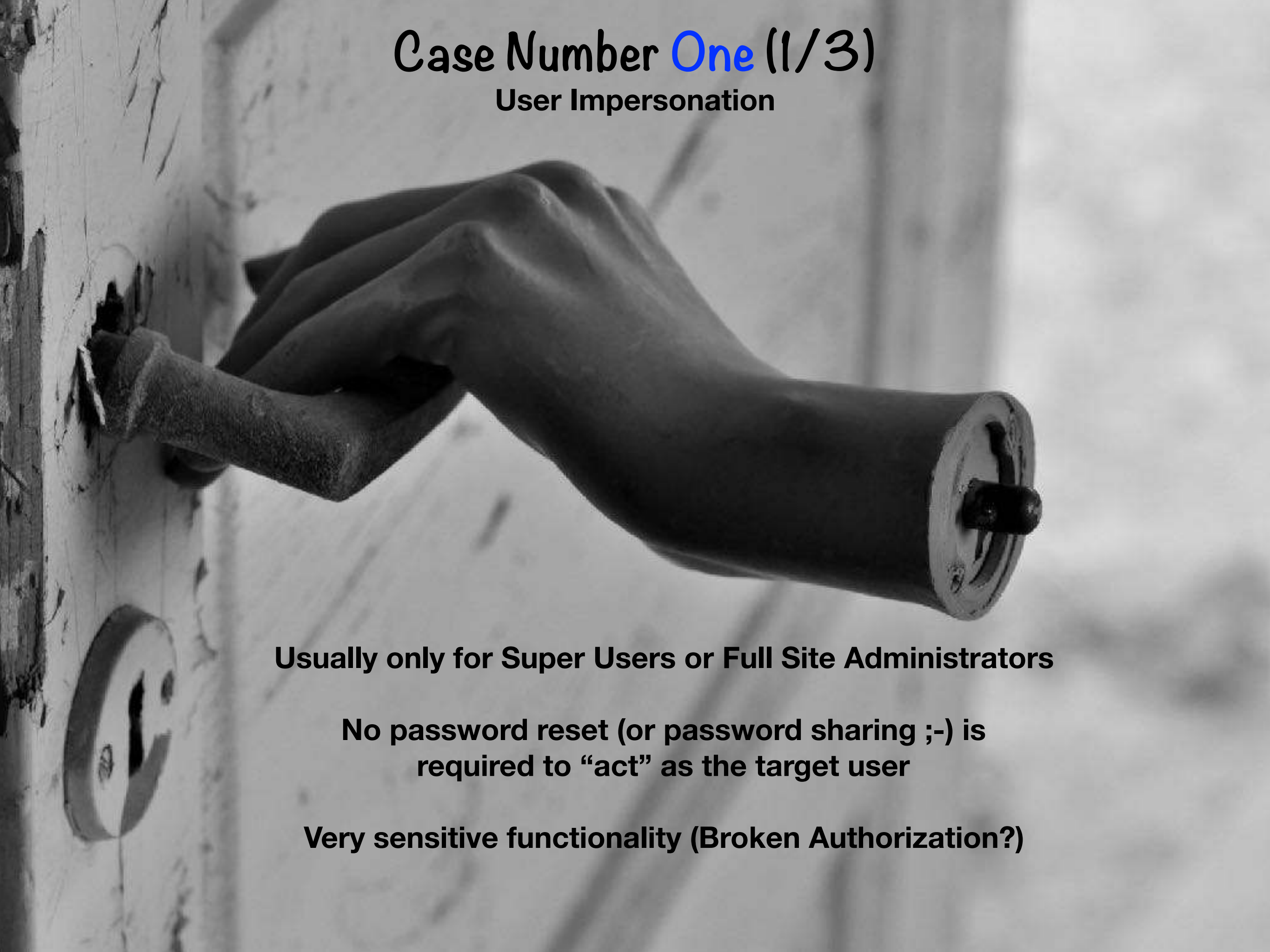


**Usually only for Super Users or Full Site Administrators**

**No password reset (or password sharing ;- ) is  
required to “act” as the target user**

# Case Number **One** (1/3)

## User Impersonation



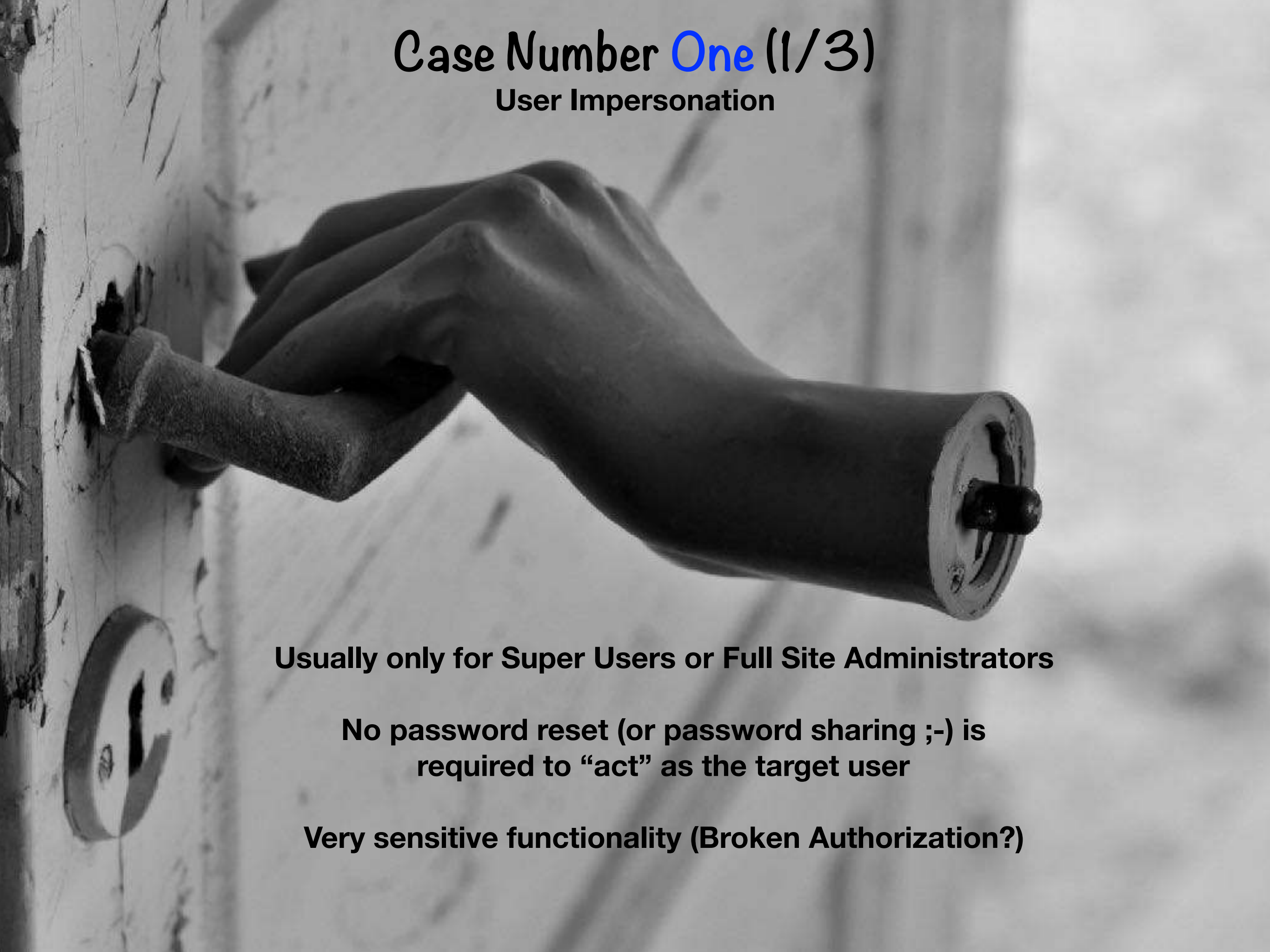
**Usually only for Super Users or Full Site Administrators**

**No password reset (or password sharing ;- ) is required to “act” as the target user**

**Very sensitive functionality (Broken Authorization?)**

# Case Number **One** (1/3)

## User Impersonation



**Usually only for Super Users or Full Site Administrators**

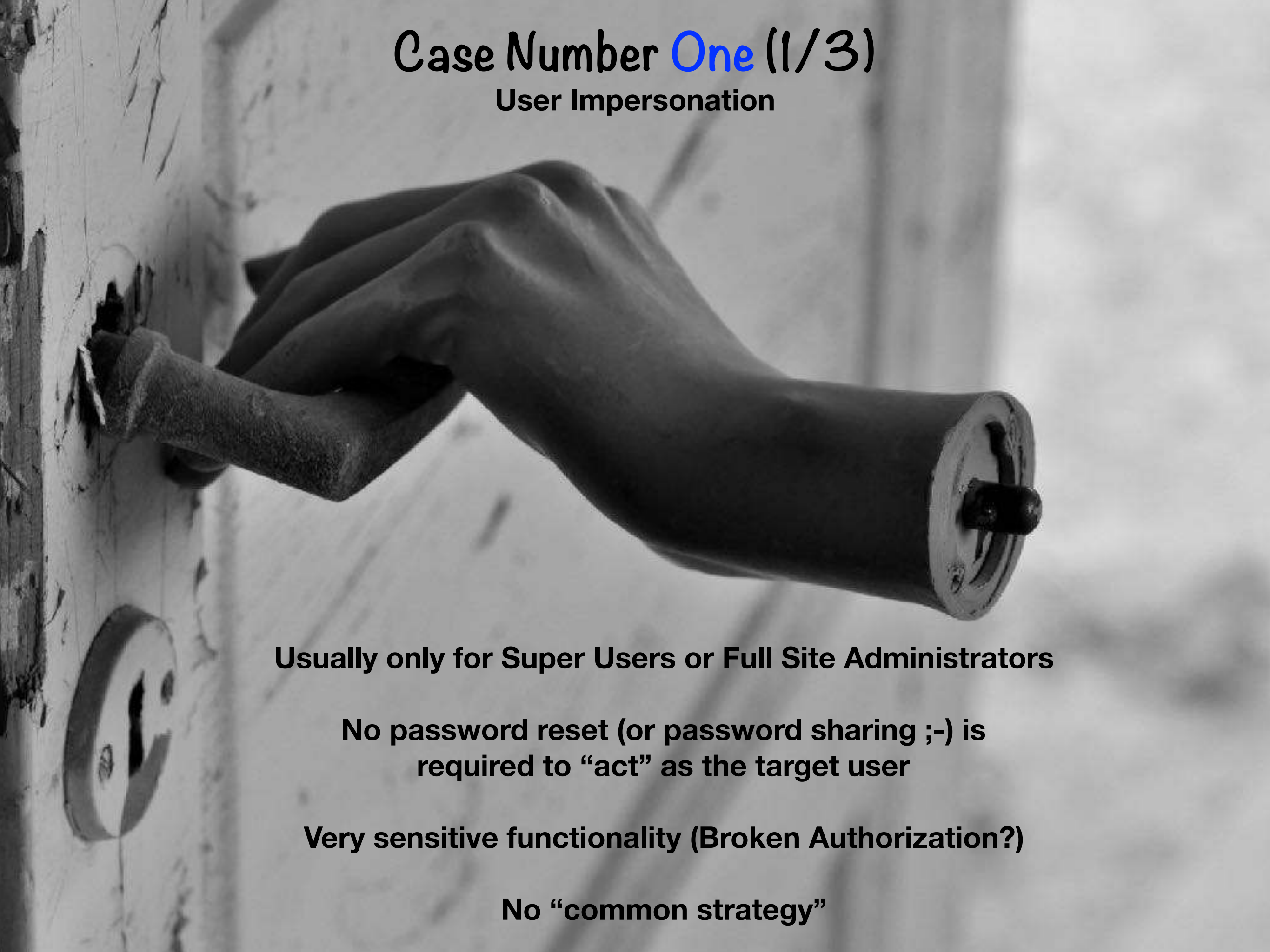
**No password reset (or password sharing ;- ) is required to “act” as the target user**

**Very sensitive functionality (Broken Authorization?)**



# Case Number **One** (1/3)

## User Impersonation



**Usually only for Super Users or Full Site Administrators**

**No password reset (or password sharing ;- ) is required to “act” as the target user**

**Very sensitive functionality (Broken Authorization?)**

**No “common strategy”**

# Case Number **One** (2/3)

## User Impersonation



# Case Number **One** (2/3)

## User Impersonation

### Request:

POST /api/user/1753/impersonate HTTP/1.1  
Host: **test.crazy.net**  
[...]

### Response:

HTTP/1.1 200 OK  
Server: Microsoft-IIS/8.5  
Date: Tue, 16 Jan 2018 15:28:17 GMT  
Connection: close  
Content-Length: 245

```
{ "username": "1753_user", "passkey": "OMRDSPWTM  
2X6KNM3KYHINET6MHL3XHNLYORN3VOK7EFJBFWXHX54H  
FLQRF7XSVEGOJGZ6G4YHTMPNEBTKKIEGLSC4WUCTVDV [  
redacted] " }
```





# Case Number One (2/3)

## User Impersonation

### Request:

POST /api/user/1753/impersonate HTTP/1.1  
Host: **test.crazy.net**  
[...]

### Response:

HTTP/1.1 200 OK  
Server: Microsoft-IIS/8.5  
Date: Tue, 16 Jan 2018 15:28:17 GMT  
Connection: close  
Content-Length: 245  
  
{ "username": "1753\_user", "passkey": "OMRDSPWTM2X6KNM3KYHINET6MHL3XHNLYORN3VOK7EFJBFWXHX54HFLQRF7XSVEGOJGZ6G4YHTMPNEBTKKIEGLSC4WUCTVDV[redacted]" }

### Request II:

POST /api/authentication/token HTTP/1.1  
Host: **test.crazy.net**

[...]grant\_type=password&username=**admin**&password=OMRDSPWTM2X6KNM3KYHINET6MHL3XHNLYORN3VOK7EFJBFWXHX54HFLQRF7XSVEGOJGZ6G4YHTMPNEBTKKIEGLSC4WUCTVDV[redacted]

### Response II:

HTTP/1.1 200 OK  
Server: Microsoft-IIS/8.5  
Date: Tue, 16 Jan 2018 15:31:12 GMT  
Connection: close  
Content-Length: 1169

{ "access\_token": "dxjPlvTBeSg9ztuzMq8Ja\_FKcgNaSV-SVHct49OXxL2FOkALjeD-Aq3dOEH4fnOgADjfiHgmmOsChuAkXY2OQbrlUnZfotfKePcLhcY8BJxcJukPlHuJCwtUo6kj\_7IR81-MQ4cbOARDG9N81FUaP45VHcYxexLGS8JMzEscPJBe[redacted]", "token\_type": "bearer", "expires\_in": 1209599, "userName": "**admin**", ".issued": "Tue, 16 Jan 2018 15:31:12 GMT", ".expires": "Tue, 30 Jan 2018 15:31:12 GMT" }



# Case Number **One** (2/3)

## User Impersonation

### Request:

POST /api/user/1753/impersonate HTTP/1.1  
Host: **test.crazy.net**  
[...]

### Response:

HTTP/1.1 200 OK  
Server: Microsoft-IIS/8.5  
Date: Tue, 16 Jan 2018 15:28:17 GMT  
Connection: close  
Content-Length: 245

```
{ "username": "1753_user", "passkey": "OMRDSPWTM  
2X6KNM3KYHINET6MHL3XHNLYORN3VOK7EFJBFWXHX54H  
FLQRF7XSVEGOJGZ6G4YHTMPNEBTKKIEGLSC4WUCTVDV[  
redacted]" }
```

### Request II:

POST /api/authentication/token HTTP/1.1  
Host: **test.crazy.net**

[...]grant\_type=password&username=**admin**&passw  
ord=OMRDSPWTM2X6KNM3KYHINET6MHL3XHNLYORN3VO  
K7EFJBFWXHX54HFLQRF7XSVEGOJGZ6G4YHTMPNEBTKK  
IEGLSC4WUCTVDV[redacted]

### Response II:

HTTP/1.1 200 OK  
Server: Microsoft-IIS/8.5  
Date: Tue, 16 Jan 2018 15:31:12 GMT  
Connection: close  
Content-Length: 1169

```
{ "access_token": "dxjPlvTBeSg9ztuzMq8Ja_FKcg  
NaSV-SVHct49OXxL2FOkALjeD-  
Aq3dOEh4fnOgADjfiHgmmOsChuAkXY2OQbrlUnZfotf  
KePcLhcY8BJxcJukPlHuJCwtUo6kj_7IR81-  
MQ4cbOARDG9N81FUaP45VHcYxexLGS8JMzEscPJBe[r  
edacted]  
", "token_type": "bearer", "expires_in":  
1209599, "userName": "admin", ".issued": "Tue,  
16 Jan 2018 15:31:12 GMT", ".expires": "Tue,  
30 Jan 2018 15:31:12 GMT" }
```

**OK, this is bad, but...**



# Case Number One (2/3)

## User Impersonation

### Request:

POST /api/user/1753/impersonate HTTP/1.1  
Host: test.crazy.net  
[...]

### Response:

HTTP/1.1 200 OK  
Server: Microsoft-IIS/8.5  
Date: Tue, 16 Jan 2018 15:28:17 GMT  
Connection: close  
Content-Length: 245  
  
{ "username": "1753\_user", "password": "Dm2X6KNM3KYHINET6MHL3XHNLYORN3VOFLQRF7XSVEGOJGZ6G4YHTMPNEBTKKIEGLSC [redacted]" }

### Request II:

POST /api/authentication/token HTTP/1.1  
Host: test.crazy.net  
[...]  
Content-Type: application/json  
{"username": "admin", "password": "Dm2X6KNM3KYHINET6MHL3XHNLYORN3VOFLQRF7XSVEGOJGZ6G4YHTMPNEBTKKIEGLSC [redacted]" }

HTTP/1.1 200 OK  
Server: Microsoft-IIS/8.5  
Date: Tue, 16 Jan 2018 15:31:12 GMT  
Connection: close  
Content-Length: 1169

{ "access\_token": "dxjPlvTBeSg9ztuzMq8Ja\_FKcgNaSV-SVHCT49OXxL2FOkALjeD-Aq3dOEh4fnOgADjfiHgmmOsChuAkXY2OQbrlUnZfotfKePcLhcY8BJxcJukPlHuJCwtUo6kj\_7IR81-MQ4cbOARDG9N81FUaP45VHcYxexLGS8JMzEscPJBe [redacted]", "token\_type": "bearer", "expires\_in": 1209599, "userName": "admin", ".issued": "Tue, 16 Jan 2018 15:31:12 GMT", ".expires": "Tue, 30 Jan 2018 15:31:12 GMT" }





# Case Number **One** (3/3)

User Impersonation



# Case Number **One** (3/3)

## User Impersonation



### Request III:

POST /api/authentication/token HTTP/1.1

Host: **prod.crazy.net**

[...]grant\_type=password&username=**admin**&password=OMRDS PWTM2X6KNM3KYHINET6MHL3XHN  
LYORN3VOK7EFJBFWXHX54HFLQRF7XSVEGOJGZ6G4YHTMPNEBTKKIEGLSC4WUCTVDV [redacted]

### Response III:

HTTP/1.1 200 OK

Server: Microsoft-IIS/8.5

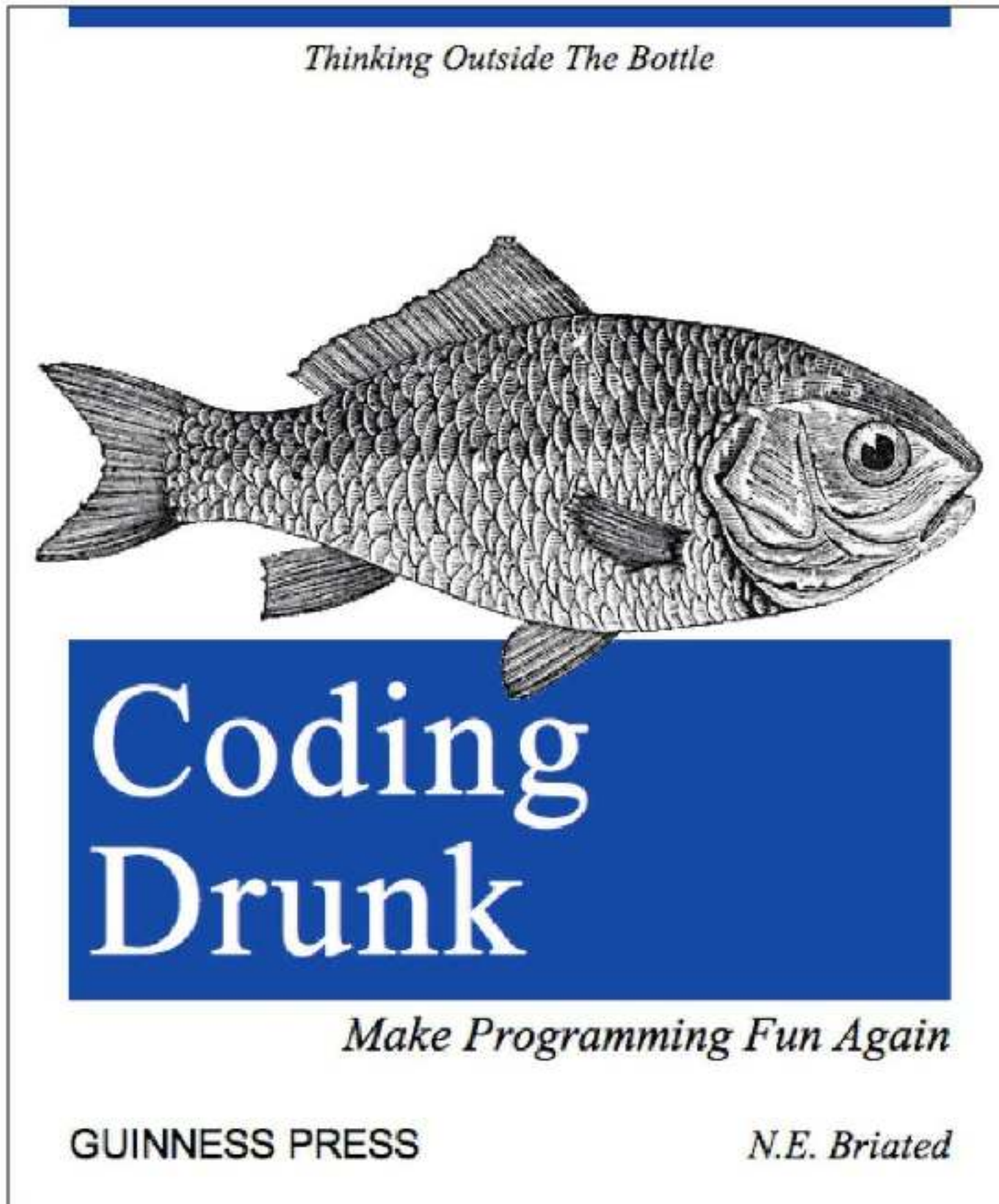
Connection: close

Content-Length: 1169

{"access\_token":"RssDFG44gGfDs6548Ja\_FKcgNaSV-SVHCt49OXxL2FOkALjeD-  
Aq3dOEh4ffdsfdRFCGU5456DDDuJCwtUo6kj\_7IR81-  
MQ4cbOARDDdfGER345VHcYxexLGS8JMzEscPJBe [redacted]  
", "token\_type":"bearer", "expires\_in":1209599, "userName":"**admin**", ".issued": [...]

# Case Number **One** - Conclusion

## User Impersonation



**Passkey WTF?**

**Not Bound to the User for whom  
it was generated**

**Testing and Production are Sharing  
The Decryption Keys**

**Code will grant access if Password  
Or Passkey are correct  
(same parameter name)**





# Case Number **Two** (0/5)

## Bypassing the Auth0 Authentication Process

With more than 2000 enterprise customers and managing 42 million logins every single day, **Auth0** is one of the biggest Identity Platforms ([auth0.com](https://auth0.com))

I found an **Authentication Bypass vulnerability** that affected **any application** using **Auth0** in the context of an independent non-profitable research

The described vulnerability would allow **malicious users to run cross-company attacks, allowing them to access any portal / application protected with Auth0 with minimum knowledge**

I will demonstrate the flaw **attacking the Auth0 Management Console**  
(used as one exploitable example application)

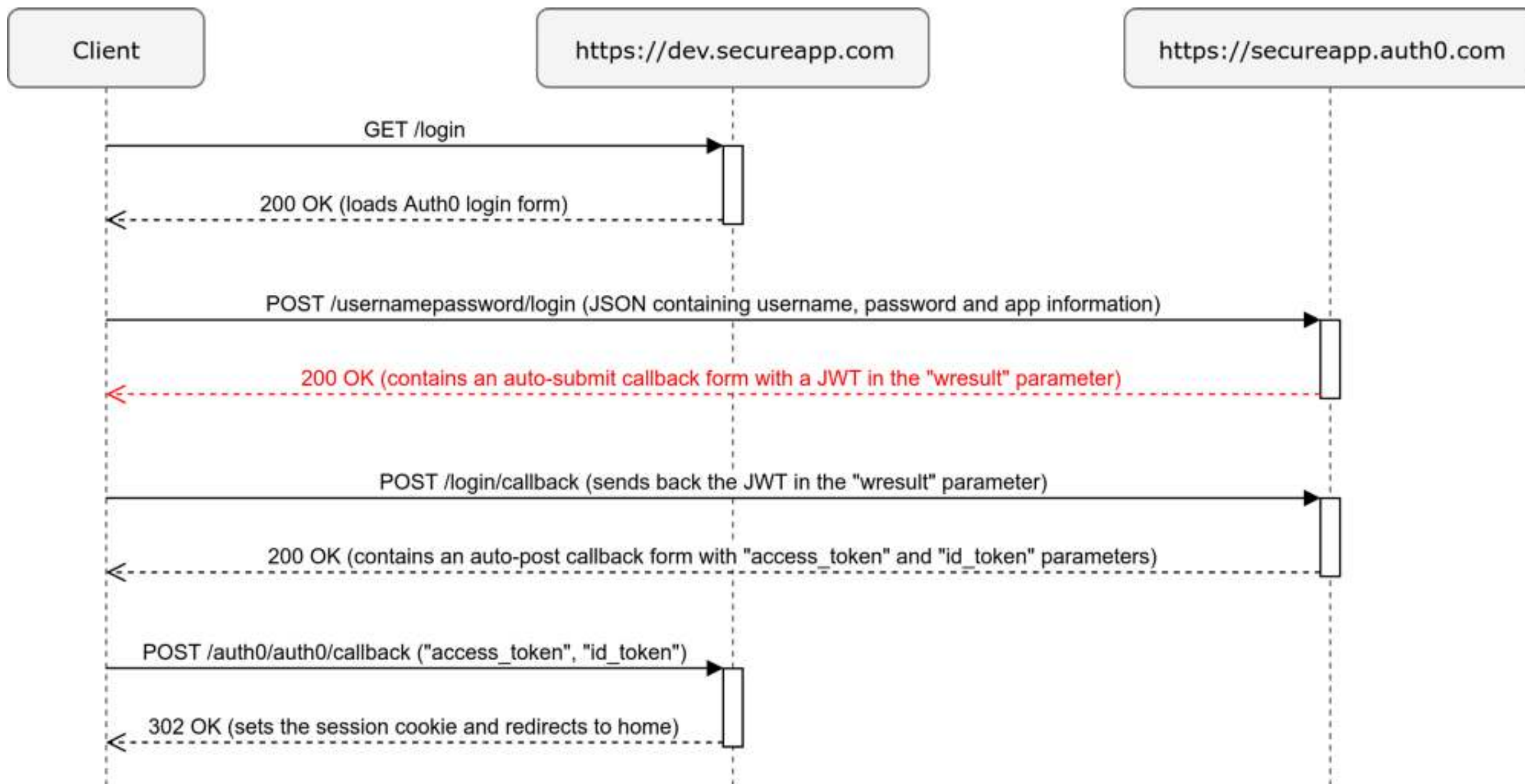


# Case Number **Two** (1/5)

## Bypassing the Auth0 Authentication Process

The story begins in **September 2017**, while I was pentesting an application which we will call “**SecureApp**”. The application was already in production but we were testing in a **DEV** environment, and it used **Auth0** for authentication.

The authentication flow looked like the following:



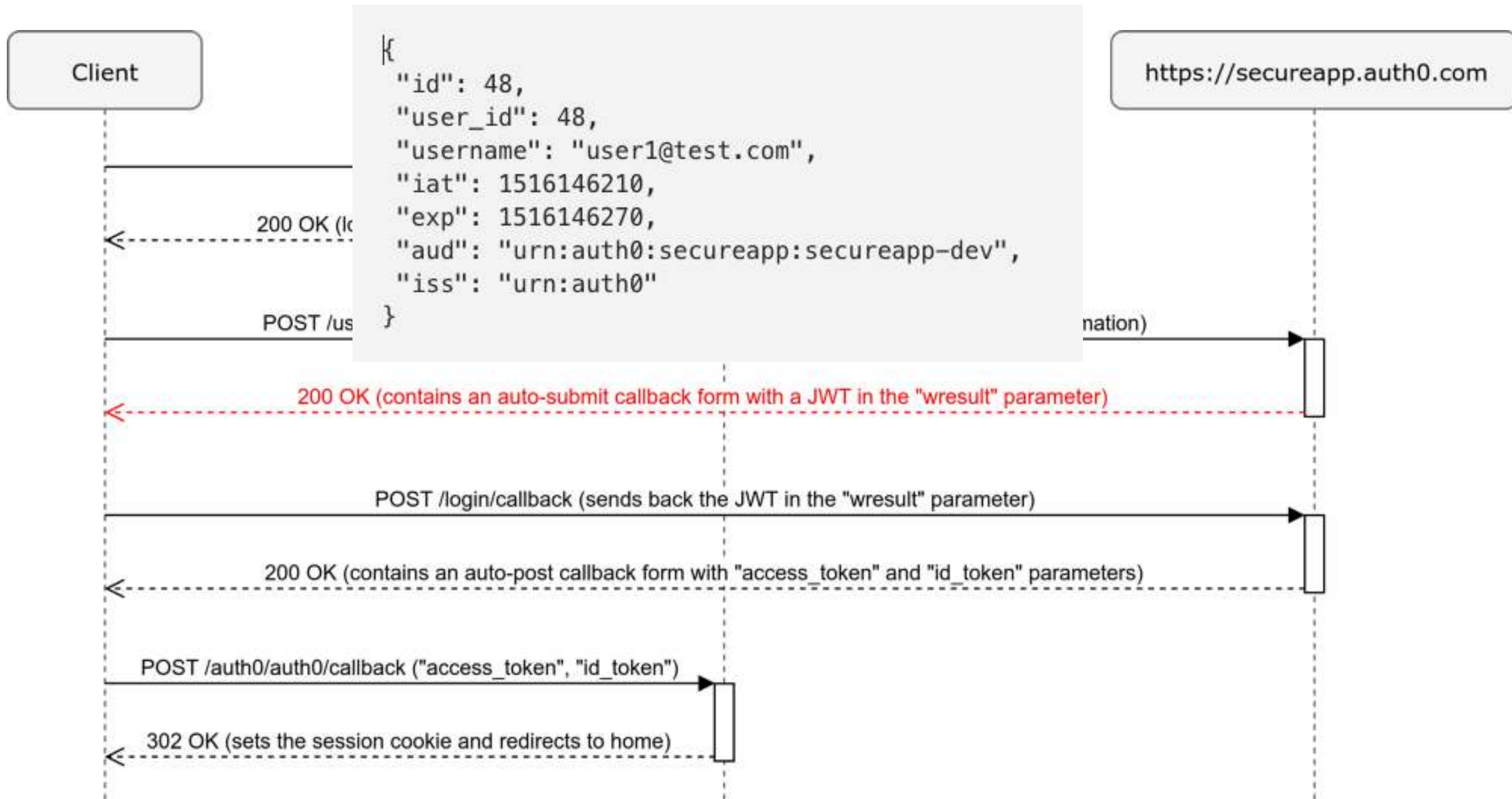


# Case Number **Two** (1/5)

## Bypassing the Auth0 Authentication Process

The story begins in **September 2017**, while I was pentesting an application which we will call “**SecureApp**”. The application was already in production but we were testing in a **DEV** environment, and it used **Auth0** for authentication.

The authentication flow looked like the following:



# Case Number **Two** (2/5)

## Bypassing the Auth0 Authentication Process

# Case Number **Two** (2/5)

## Bypassing the Auth0 Authentication Process

We couldn't modify this payload because it had been signed, but we could try to reuse it.



# Case Number **Two** (2/5)

## Bypassing the Auth0 Authentication Process

We couldn't modify this payload because it had been signed, but we could try to reuse it.

# Case Number **Two** (2/5)

## Bypassing the Auth0 Authentication Process

We couldn't modify this payload because it had been signed, but we could try to reuse it.

So, armed with a proxy, we captured **a valid “wresult” JWT from the DEV environment and injected it into a login flow in PROD**, and **it worked!** We were able to access the account for that user in the production environment.

# Case Number **Two** (2/5)

## Bypassing the Auth0 Authentication Process

We couldn't modify this payload because it had been signed, but we could try to reuse it.

So, armed with a proxy, we captured **a valid “wresult” JWT from the DEV environment and injected it into a login flow in PROD**, and **it worked!** We were able to access the account for that user in the production environment.



# Case Number **Two** (2/5)

## Bypassing the Auth0 Authentication Process

We couldn't modify this payload because it had been signed, but we could try to reuse it.

So, armed with a proxy, we captured **a valid “wresult” JWT from the DEV environment and injected it into a login flow in PROD**, and **it worked!** We were able to access the account for that user in the production environment.

**The question is then, are DEV and PROD environments using the same signing keys / certificates? What else is wrong?**

# Case Number **Two** (2/5)

## Bypassing the Auth0 Authentication Process

We couldn't modify this payload because it had been signed, but we could try to reuse it.

So, armed with a proxy, we captured **a valid “wresult” JWT from the DEV environment and injected it into a login flow in PROD**, and **it worked!** We were able to access the account for that user in the production environment.

**The question is then, are DEV and PROD environments using the same signing keys / certificates? What else is wrong?**

# Case Number **Two** (2/5)

## Bypassing the Auth0 Authentication Process

We couldn't modify this payload because it had been signed, but we could try to reuse it.

So, armed with a proxy, we captured **a valid “wresult” JWT from the DEV environment and injected it into a login flow in PROD**, and **it worked!** We were able to access the account for that user in the production environment.

**The question is then, are DEV and PROD environments using the same signing keys / certificates? What else is wrong?**

**Jump through different apps/envs** within the organization?????!!!!!!



# Case Number **Two** (2/5)

## Bypassing the Auth0 Authentication Process

We couldn't modify this payload because it had been signed, but we could try to reuse it.

So, armed with a proxy, we captured **a valid “wresult” JWT from the DEV environment and injected it into a login flow in PROD**, and **it worked!** We were able to access the account for that user in the production environment.

**The question is then, are DEV and PROD environments using the same signing keys / certificates? What else is wrong?**

**Jump through different apps/envs** within the organization?????!!!!!!

Think of a “user\_id” value that identifies an internal user, and multiple applications that rely on that identifier.

# Case Number **Two** (2/5)

## Bypassing the Auth0 Authentication Process

We couldn't modify this payload because it had been signed, but we could try to reuse it.

So, armed with a proxy, we captured **a valid “wresult” JWT from the DEV environment and injected it into a login flow in PROD**, and **it worked!** We were able to access the account for that user in the production environment.

**The question is then, are DEV and PROD environments using the same signing keys / certificates? What else is wrong?**

**Jump through different apps/envs** within the organization?????!!!!!!

Think of a “user\_id” value that identifies an internal user, and multiple applications that rely on that identifier.

We could now access all of them even when without valid credentials.

**What else can go wrong?**



**WE'RE GONNA NEED A BIGGER  
WALL**



# Case Number **Two** (3/5)

## Bypassing the Auth0 Authentication Process - **Attacking the Auth Management Console**

```
{
  "user_id": "59c5a39c5315152c967cc031",
  "email": "nahuel@cintainfinita.com.ar",
  "email_verified": true,
  "iat": 1506952673,
  "exp": 1506952733,
  "aud": "urn:auth0:auth0:auth0",
  "iss": "urn:auth0"
}
```

**“wresult” parameter**

In order to hijack an account, we would need to forge a valid JWT with that user’s information.

We don’t have access to:

1. the “user\_id” (not trivial like an email address or an incremental integer, but for other applications this could be the case) —> TENANT INVITE, ACCEPT, **DELETE**
2. **the signing key (or private certificate)**

# Case Number **Two** (4/5)

## Bypassing the Auth0 Authentication Process - **Attacking the Auth Management Console**

We found a **functionality** that could be used (or abused) as an oracle to generate valid **JWTs** with arbitrary payloads

The Management Console allows you to create Database Action Scripts that are executed every time a user logs in. We created a simple “Database Action Script” that returned the needed values for the profile, **signed** ;-))



```
{
  "user_id": "59d60fef8025c603ce735e02",
  "email": "nahuel+victim@cintainfinita.com.ar",
  "email_verified": true,
  "iat": 1507203410,
  "exp": 1507203470,
  "aud": "urn:auth0:atacante:Username-Password-Authentication",
  "iss": "urn:auth0"
}
```



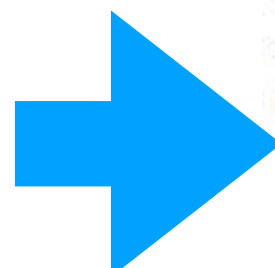
So, now we had the ability to forge a valid signed JWT with the “email” and “user\_id” of the victim.

*What about the AUD?*



# Case Number Two (5/5)

## Bypassing the Auth0 Authentication Process - Attacking the Auth Management Console



The image shows the Auth0 management console interface. The left sidebar contains a navigation menu with items: Dashboard, Clients, APIs, SSO Integrations, **Connections**, Users, Rules, Hooks, Multifactor Auth, Hosted Pages, Emails, Logs, Anomaly Detection, Extensions, and Get Support. The 'Connections' menu is highlighted with a blue arrow. The main content area is titled 'Username-Password-Authentication' and includes a search bar, navigation tabs (Settings, Password Policy, Custom Database, Clients, Try connection), and a toggle switch for 'Use my own database' which is currently turned on. Below this is the 'Database Action Scripts' section, showing a 'Login' script. The script code is as follows:

```
1 function login(email, password, callback) {  
2  
3   var profile =  
4   {  
5     'user_id': '55c60faf8025e603ce735e02',  
6     'email': 'nate1-victims@containinfinite.com.ar',  
7     'email_verified': true,  
8   };  
9  
10  callback(null, profile);  
11  
12 }  
13
```



# Case Number **Two** - Conclusion

Bypassing the Auth0 Authentication Process - **Attacking the Auth Management Console**



The screenshot displays the Auth0 Management Console dashboard. At the top, there's a navigation bar with the Auth0 logo, a search bar, and links for Help & Support, Documentation, Talk to Sales, and a user profile. A message at the top right indicates a 21-day trial for the Free plan. The main section is titled 'Dashboard' and includes a 'NEW CLIENT' button. Below this is a 'Login Activity' chart showing a grid of login events. A summary section shows 'USERS: 1', 'LOGINS: 0', and 'NEW SIGNUPS: 0'. The 'Latest Logins' table lists a user 'nahuel+vicima@cintaininit a.com.ar' who logged in 3 minutes ago using 'Username-Password Authentication'. The 'New Signups' table shows a similar entry for the same user, also 3 minutes ago. A sidebar on the left contains various management options like Clients, APIs, and Users. At the bottom right, there's a link to 'Continue with this tutorial'.



# Case Number **Two** - Conclusion

Bypassing the Auth0 Authentication Process - **Attacking the Auth Management Console**



The screenshot displays the Auth0 Management Console dashboard. At the top, there's a navigation bar with the Auth0 logo, a search bar, and links for Help & Support, Documentation, Talk to Sales, and a user profile. A message at the top right indicates a 21-day trial for the Free plan. The main section is titled 'Dashboard' and includes a 'NEW CLIENT' button. Below this is a 'Login Activity' chart showing a grid of login events. A summary section shows 'USERS: 1', 'LOGINS: 0', and 'NEW SIGNUPS: 0'. The 'Latest Logins' section lists a login for 'nahuel+vicima@cintaininit a.com.ar' 3 minutes ago. The 'New Signups' section lists a new signup for 'nahuel+vicima@cintaininit a.com.ar' 3 minutes ago. A sidebar on the left contains links to various features like Clients, APIs, SSO Integrations, etc. At the bottom right, there's a link to 'Continue with this tutorial'.









# Case Number **Three** (0/6)

**Observations in MS Azure (and Standard IIS) running .NET Apps & SAML AuthN**

# Case Number **Three** (0/6)

## Observations in MS Azure (and Standard IIS) running .NET Apps & SAML AuthN

benjamin perkins

```
List<Solution> solutions = changes.Where(c => c.CeterisParibus != true)
```

### Machine Keys on an Azure App Service, machineKey multiple instances Azure

Rate this article ★★★★★



benjaminperkins May 8, 2017

Share 1

12

in 0

1

When you run an ASP.NET application on multiple instances of an App Service Plan ([ASP](#)) you do not need to worry about machineKeys as the App Service Platform will use the same one across all your instances and therefore will not need to make any changes to your application.

# Case Number **Three** (0/6)

Observations in MS Azure (and Standard IIS) running .NET Apps & SAML AuthN

benjamin perkins

```
List<Solution> solutions = changes.Where(c => c.CeterisParibus != true)
```

## Machine Keys on an Azure App Service, machineKey multiple instances Azure



benjamin

When you run the App Service application.

### Clone your existing App using Azure portal

I'm going to show you how easy it is to Clone in Azure Portal. With this feature you can setup a new instance of your app in seconds and start using it.

Clone in Azure Portal feature allows you to move the App you want to clone to a different region or keep it in the same region.

When you clone, Azure will also clone all the App Settings, Connection strings and Deployment sources and Certificates, etc too, so the new cloned app is more or less up and running.



1

Keys as



# Case Number **Three** (0/6)

## Observations in MS Azure (and Standard IIS) running .NET Apps & SAML AuthN

benjamin perkins

```
List<Solution> solutions = changes.Where(c => c.CeterisParibus != true)
```

Machine Keys on an Azure App Service, machineKey  
multiple instances Azure








benjamin

**Clone your existing App using Azure portal**

1

## Set up staging environments in Azure App Service

📅 12/16/2016 • ⌚ 10 minutes to read • Contributors     

When you deploy your web app, web app on Linux, mobile back end, and API app to [App Service](#), you can deploy to a separate deployment slot instead of the default production slot when running in the **Standard** or **Premium** App Service plan tier. Deployment slots are actually live apps with their own hostnames. App content and configurations elements can be swapped between two deployment slots, including the production slot.

When you clone, Azure will also clone all the App Settings, Connection strings and Deployment sources and Certificates, etc too, so the new cloned app is more or less up and running.

# Case Number **Three** (1/6)

Observations in MS Azure (and Standard IIS) running .NET Apps & SAML AuthN

Machine Keys?

## machineKey Element (ASP.NET Settings Schema)

.NET Framework 3.0 | [Other Versions](#) ▾

Configures keys to use for encryption and decryption of forms authentication cookie data and view-state data, and for verification of out-of-process session state identification.

```
<machineKey
  validationKey="AutoGenerate,IsolateApps" [String]
  decryptionKey="AutoGenerate,IsolateApps" [String]
  validation="SHA1" [SHA1 | MD5 | 3DES | AES]
  decryption="Auto" [Auto | DES | 3DES | AES]
/>
```

# Case Number **Three** (1/6)

Observations in MS Azure (and Standard IIS) running .NET Apps & SAML AuthN

## Machine Keys?

### machineKey Element (ASP.NET Settings Schema)

.NET Framework 3.0 | [Other Versions](#) ▾

Configures keys to use for encryption and decryption of forms authentication cookie data and view-state data, and for verification of out-of-process session state identification.

```
<machineKey
  validationKey="AutoGenerate,IsolateApps" [String]
  decryptionKey="AutoGenerate,IsolateApps" [String]
  validation="SHA1" [SHA1 | MD5 | 3DES | AES]
  decryption="Auto" [Auto | DES | 3DES | AES]
/>
```

Home > App Services > SingleSignOnProd - Deployment slots > Add a slot

#### Add a slot

Deployment slots let you deploy different versions of your web app to different URLs. You can test a certain version and then swap content and configuration between slots.

\* Name ⓘ

staging ✓

Configuration Source

Don't clone configuration from an existi... ▾

Slot swapping?



**Staging**



**Production**



# Case Number **Three** (2/6)

**Observations in MS Azure (and Standard IIS) running .NET Apps & SAML AuthN**

Web Application written in .NET on MS Azure  
(ASP.NET\_SessionId + .ASPXAUTH + FedAuth cookies)

Identity Provider for the above (using SAML)

**Staging** + **Production** SLOTS

(Swapping is easy my friend..., by default they share the same secrets  
-MachineKeys-, and they have to!?)

Common Certificates, easier, faster



**THIS CONCEPT ALSO WORKS IN WEBAPPS NOT RUNNING ON MS AZURE  
(STANDARD MS IIS INSTALLATION)**

# Case Number **Three** (2/6)

Observations in MS Azure (and Standard IIS) running .NET Apps & SAML AuthN



Web Application written in .NET on MS Azure  
(ASP.NET\_SessionId + .ASPXAUTH + FedAuth cookies)

Identity Provider for the above (using SAML)

**Staging** + **Production** SLOTS

(Swapping is easy my friend..., by default they share the same secrets  
-MachineKeys-, and they have to!?)

Common Certificates, easier, faster



THIS CONCEPT ALSO WORKS IN WEBAPPS NOT RUNNING ON MS AZURE  
(STANDARD MS IIS INSTALLATION)

# Case Number **Three** (2/6)

Observations in MS Azure (and Standard IIS) running .NET Apps & SAML AuthN



Web Application written in .NET on MS Azure  
(ASP.NET\_SessionId + .ASPXAUTH + FedAuth cookies)

Identity Provider for the above (using SAML)

**Staging** + **Production** SLOTS

(Swapping is easy my friend..., by default they share the same secrets  
-MachineKeys-, and they have to!?)

Common Certificates, easier, faster



THIS CONCEPT ALSO WORKS IN WEBAPPS NOT RUNNING ON MS AZURE  
(STANDARD MS IIS INSTALLATION)



# Case Number **Three** (2/6)


Observations in MS Azure (and Standard IIS) running .NET Apps & SAML AuthN



Web Application written in .NET on MS Azure  
(ASP.NET\_SessionId + .ASPXAUTH + FedAuth cookies)



Identity Provider for the above (using SAML)



**Staging** + **Production** SLOTS  
(Swapping is easy my friend..., by default they share the same secrets  
-MachineKeys-, and they have to!?)

Common Certificates, easier, faster



THIS CONCEPT ALSO WORKS IN WEBAPPS NOT RUNNING ON MS AZURE  
(STANDARD MS IIS INSTALLATION)

# Case Number **Three** (2/6)


## Observations in MS Azure (and Standard IIS) running .NET Apps & SAML AuthN



Web Application written in .NET on MS Azure  
(ASP.NET\_SessionId + .ASPXAUTH + FedAuth cookies)



Identity Provider for the above (using SAML)



**Staging** + **Production** SLOTS  
(Swapping is easy my friend..., by default they share the same secrets  
-MachineKeys-, and they have to!?)



Common Certificates, easier, faster



THIS CONCEPT ALSO WORKS IN WEBAPPS NOT RUNNING ON MS AZURE  
(STANDARD MS IIS INSTALLATION)

# Case Number **Three** (2/6)


## Observations in MS Azure (and Standard IIS) running .NET Apps & SAML AuthN



Web Application written in .NET on MS Azure  
(ASP.NET\_SessionId + .ASPXAUTH + FedAuth cookies)



Identity Provider for the above (using SAML)



**Staging** + **Production** SLOTS  
(Swapping is easy my friend..., by default they share the same secrets  
-MachineKeys-, and they have to!?)



Common Certificates, easier, faster



THIS CONCEPT ALSO WORKS IN WEBAPPS NOT RUNNING ON MS AZURE  
(STANDARD MS IIS INSTALLATION)



# Case Number **Three** (3/6)

Observations in MS Azure (and Standard IIS) running .NET Apps & SAML AuthN

## MS Azure

All WebApps Deployed in App Services, with No specific configuration (Web.config), within the Same Resource Group (Slots config!)

=

Will Share Machine Keys

## IIS

All WebApps Deployed, with No specific configuration (Web.config), Same or Different Application Pool

=

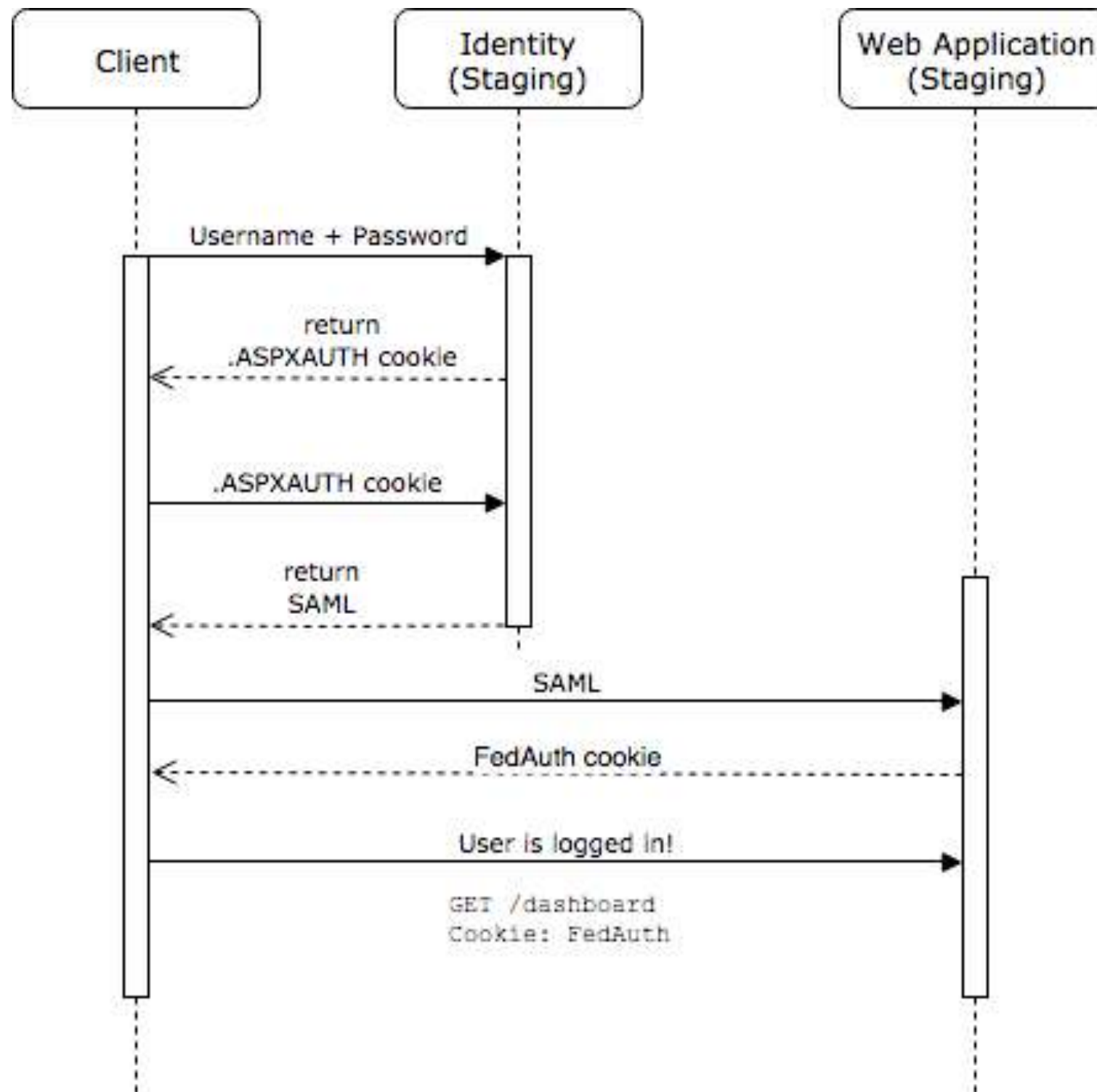
Will Share Machine Keys



# Case Number **Three** (4/6)

Observations in MS Azure (and Standard IIS) running .NET Apps & SAML AuthN

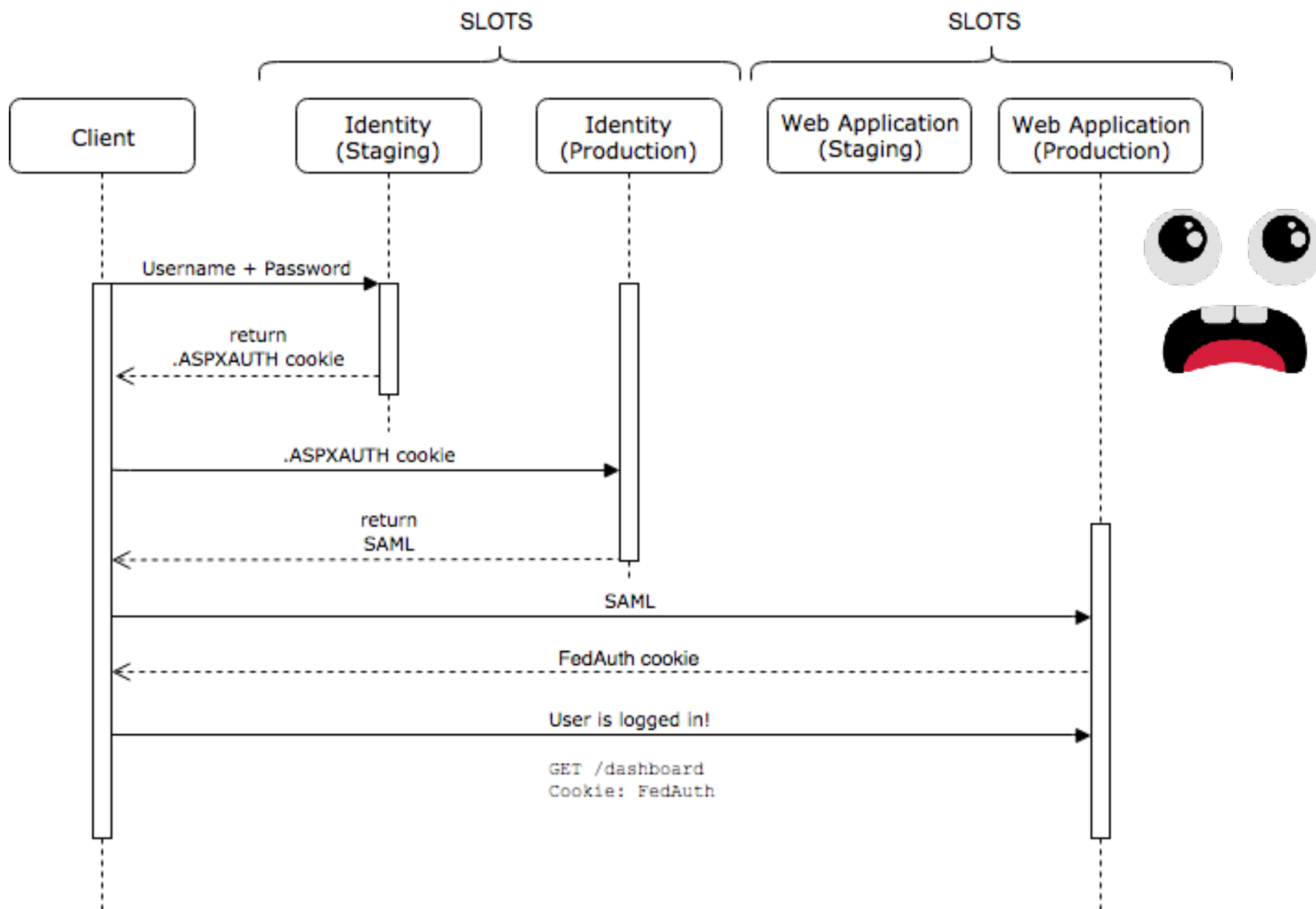
## Standard Authentication Flow



# Case Number **Three** (5/6)

Observations in MS Azure (and Standard IIS) running .NET Apps & SAML AuthN

## Modified Authentication Flow Try 1

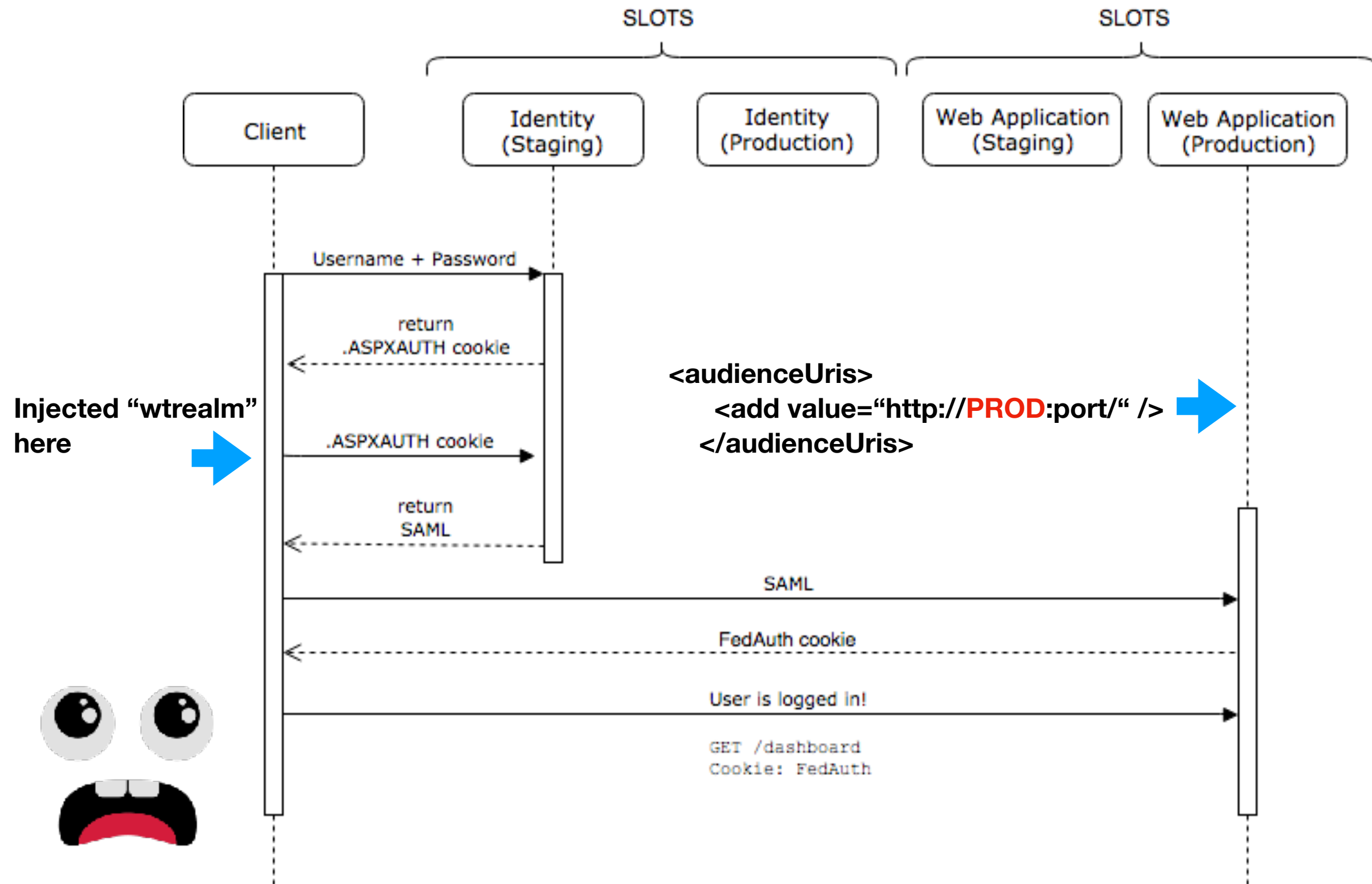




# Case Number **Three** (6/6)

Observations in MS Azure (and Standard IIS) running .NET Apps & SAML AuthN

## Modified Authentication Flow Try 2



# Case Number **Three** - Conclusion

Observations in MS Azure (and Standard IIS) running .NET Apps & SAML AuthN

Resource Groups?

No Slot Swapping?

```
<machineKey  
validationKey="AutoGenerate,IsolateApps"  
decryptionKey="AutoGenerate,IsolateApps"  
validation="SHA1"  
/>
```

[https://msdn.microsoft.com/en-us/library/w8h3skw9\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/w8h3skw9(v=vs.85).aspx)



# Conclusions

---

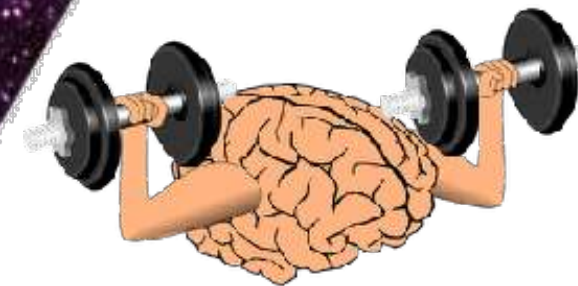
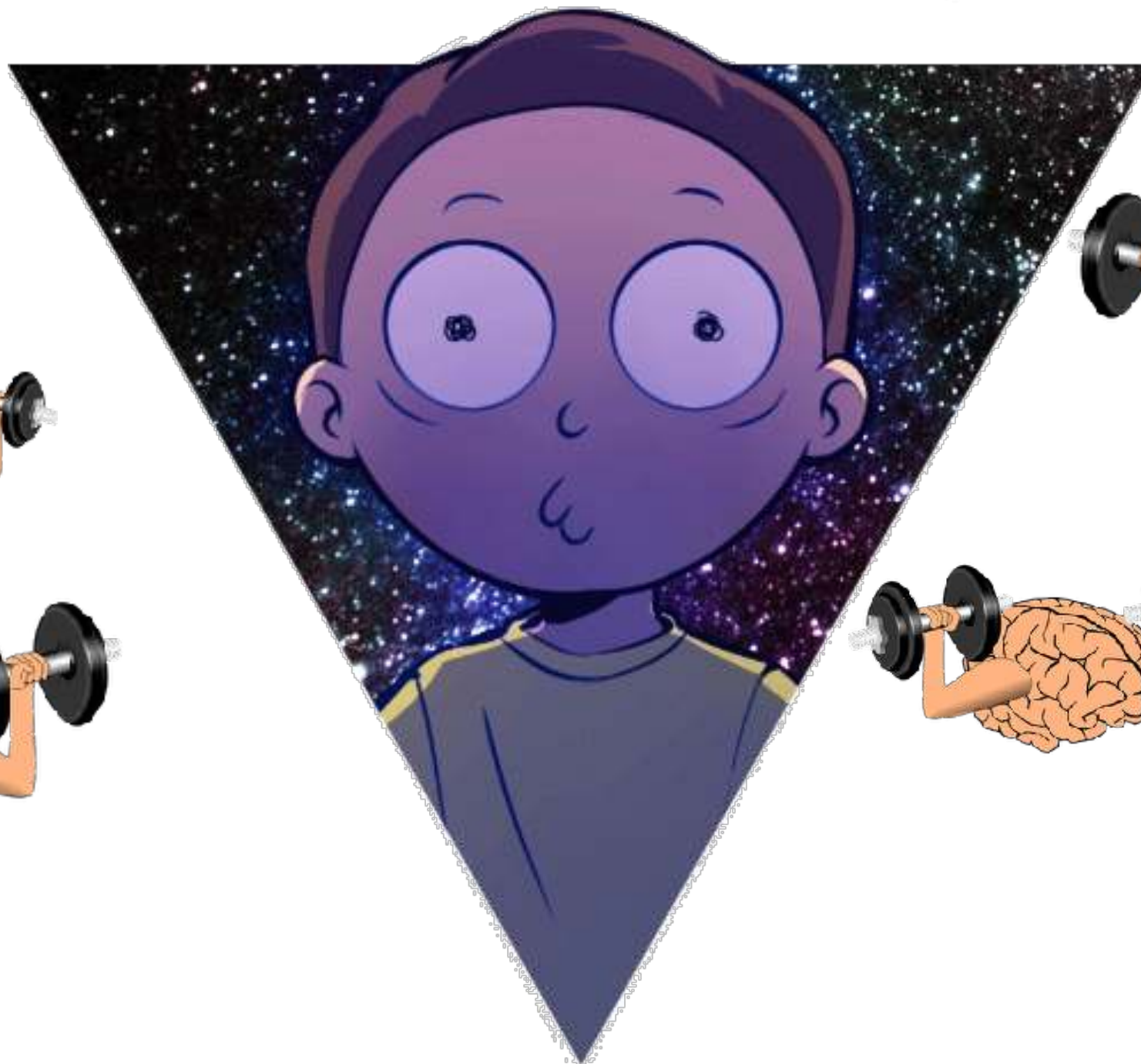
- ★ Isolate and Segregate Environments
  - ★ DO NOT share Secrets
  - ★ Verify the Audience of Claims
  - ★ Educate Developers and SysAdmins about Security (crypto, unicorns, etc.)
  - ★ Understand what you are doing in the “Cloud” (eg. Azure Governance)
- <https://docs.microsoft.com/en-us/azure/security/governance-in-azure>
- ★ Run Penetration Tests





Shoot your Question!

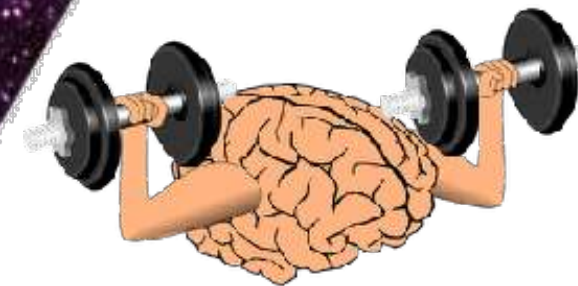
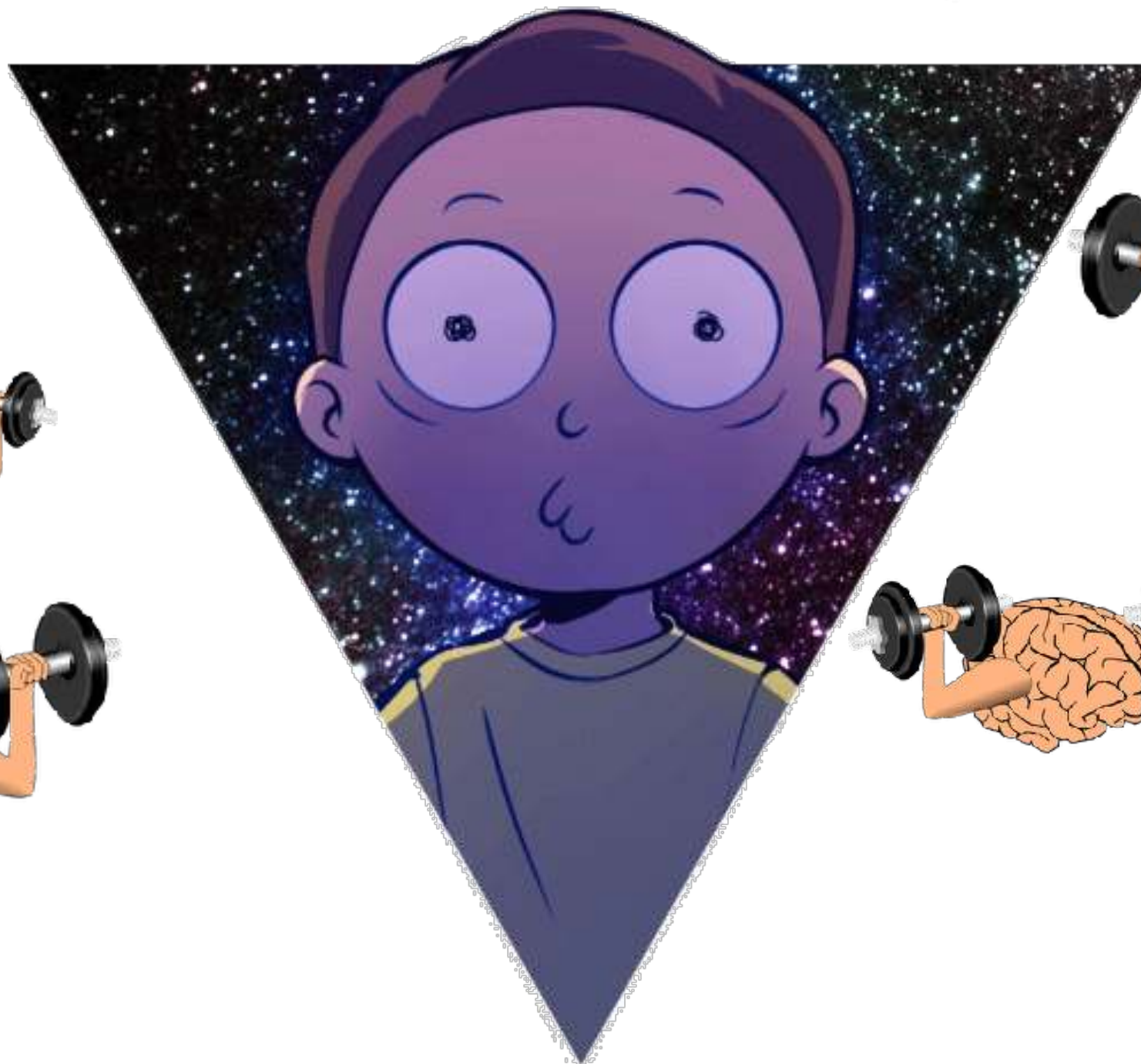
Shoot your Question!



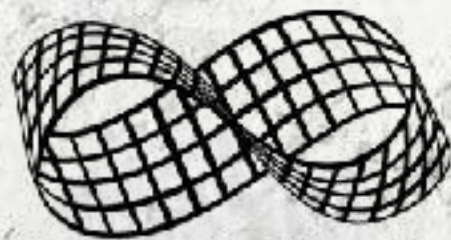


Shoot your Question!

Shoot your Question!







Cinta Infinita  
Information Security

# KNOCKING DOWN THE BIG DOOR

**Breaking Authentication and Segregation of  
Production and Non-Production Environments**

*Thanks*

Nahuel Grisolia  
Cinta Infinita Founder / CEO  
@cintainfinita