TROOPERS

The Red Teamer’s Guide To Deception

Building effective internal honeypots

Balthasar Martin <infosec.exchange/@balthasar & @BalthasarMartin>
Niklas van Dornick <@n1v4d0>

Security Research Labs

Agenda

- **Introduction**
- Deception strategy
- Must-have AD honeypots
- Tool release: ADCS deception

Despite ample opportunities, our attacks are barely detected and responded to effectively

Balthasar Martin

- Red team lead @SRLabs
- Built a dedicated team for red, purple and TIBER
- Cool hacks between PowerPoint, Excel & Word

Niklas van Dornick

- Working student @SRLabs
- Builds and breaks protocols and authentication
- Watched too much Winnie-the-Pooh

Thanks, team!

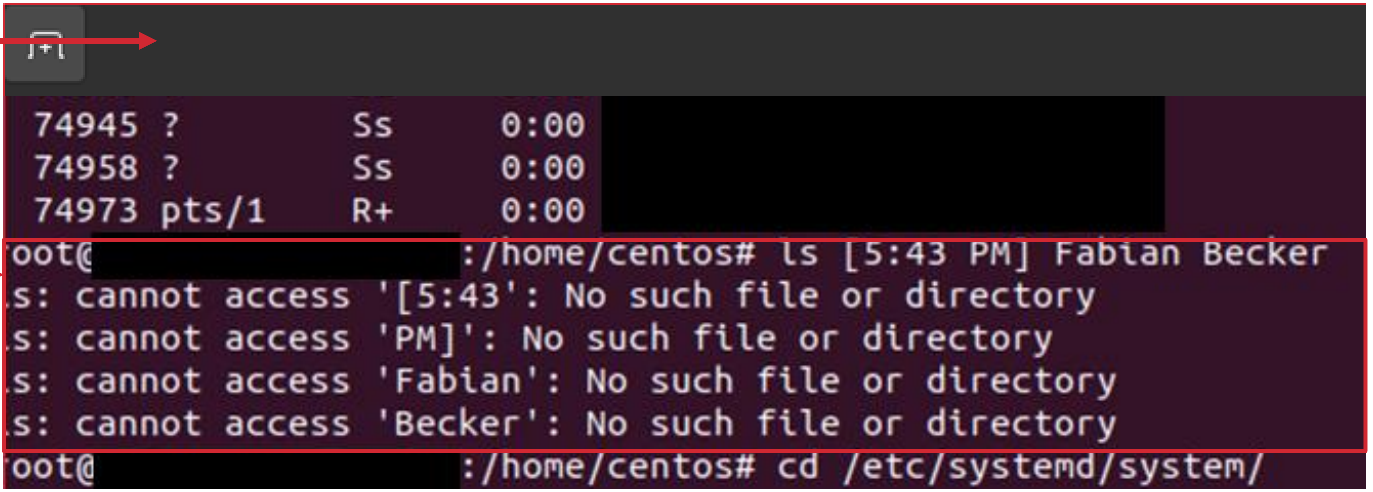
Ali

Fabian

Jorge

Root shell on targeted server

Balthasar’s mistake

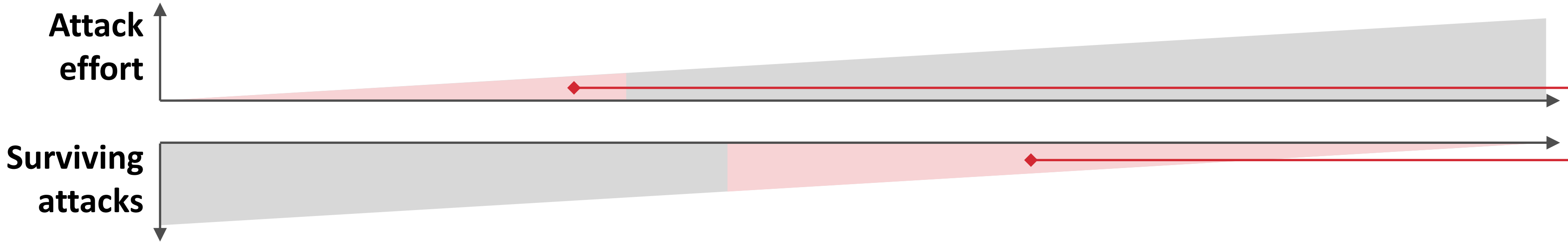


```
74945 ?         Ss     0:00 [redacted]
74958 ?         Ss     0:00
74973 pts/1     R+     0:00
oot@[redacted]:/home/centos# ls [5:43 PM] Fabian Becker
s: cannot access '[5:43': No such file or directory
s: cannot access 'PM]': No such file or directory
s: cannot access 'Fabian': No such file or directory
s: cannot access 'Becker': No such file or directory
oot@[redacted]:/home/centos# cd /etc/systemd/system/
```

- As attackers, we are only human and make mistakes
- There is ample opportunity to detect us
- Nevertheless, we compromise most target environments

We need better detection and response for the few threats that make it past initial defenses

Situation		Attack Path			Objective	
Telco	RCE in web API	SSH keys for various users	Escalated privileges on shared server	Account for production automation	**Subscriber data and SMS access**	👁 No alert triggered SOC didn't have a chance
Financial	Malware and credential phishing 👁	Exploit Java servers, spray external pws	Abused left-over AD permissions	Admin for identity management	**Control over banking interface**	👁 EDR caught malware Continued other foothold
Manufac.	Assumed breach with basic account	Credentials in code repositories	Extensive persistence, local recon 👁	Various Active directory attacks 👁	**Ransomware via full AD compromise**	👁 EDR & identity monitoring Categorized as harmless



Determined attackers can try until they succeed

Even mature companies lack detection and response for later stages of attacks

Why is that?

SOC is hard and corporations struggle to build effective monitoring and detections

Problem	Details	Consequences
Effort to achieve EDR and log coverage	▪ Requires much leg-work and communication ▪ **Pareto principle: last 20% take 80% of work**	▪ Attackers with time or luck can find **“that under-monitored system”**
Complex corporate networks	▪ Large volume of alerts that is hard to tune ▪ **“Weird” things happen regularly**	▪ Not every alert can be investigated in-depth ▪ **True positive alerts are overseen or not followed-up** upon with full response
Application-specific knowledge gap	▪ **SOC has limited knowledge about applications** ▪ Requires domain-expert support to write rules or evaluate alerts	▪ **Incorrect classification of alerts** ▪ Example: alert for activity by built-in domain admin but analyst doesn’t realize because it was renamed
Analyst Turnover	▪ Undesirable work style (shift work, factory style) ▪ **Trained analysts leave** for better positions	▪ Lower analysis quality in general
Analysis	▪ **Attackers with time/skill/luck trigger few alerts** ▪ SOCs are designed to handle large volume with okay-ish coverage and investigation result precision	▪ Attack chains with e.g. few “medium” alerts have a **realistic chance get through** **→ Blue team needs a “smoke detector” to catch these cases just before the fire is out of control**

Well-placed honeypots provide a high-quality detection signal for low costs

Definition

Internal honeypot (aka. canary, aka. deception tech):
A strategically placed system, account, or vulnerability designed to mimic legitimate assets, serving as a trap for attackers

Example

A pair of invalid credentials places on a server, triggering an alert when used

Advantages

1. Low roll-out complexity and maintenance

- Deploy once to a few easily-discovered locations
- Use existing technologies like a SIEM
- Low footprint, limited maintenance

2. Low-noise detections

- Honeypots are not used by legitimate users
- They can be set up to only trigger on clearly malicious activity

3. High-relevance alerts

- Are triggered during lateral movement and privilege escalation
- Honeypot exploitation likely indicates a significant threat
- Allows to trigger critical alerts, directly to a senior analyst

Strategic Impact

- **Effective alerting that can prevent the worst** in cases where initial infection stays undetected
- **Great cost-benefit** ratio for catching attackers
- **Slowing down attackers** by forcing them to second-guess their attacks

Agenda

- Introduction
- **Deception strategy**
- Must-have AD honeypots
- Tool release: ADCS deception

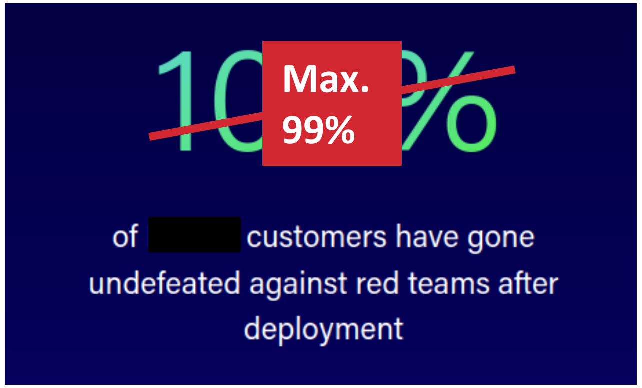
Case study: deception is not solved with a shiny product roll-out

Environment

“Top-right quadrant” deception tool

- **Rollout** on all corporate laptops
- **Various canaries** per system, including fake credentials in LSASS
- **Individualized** AD accounts enable different configuration for each laptop

How it went



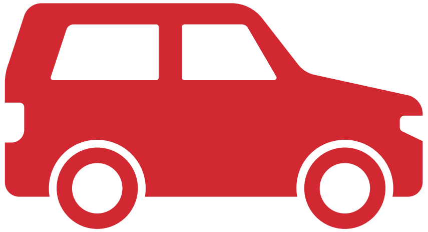
~~100%~~ Max. 99%

of [redacted] customers have gone undefeated against red teams after deployment

Deception was totally ineffective

- **Coverage gap:** we did not touch Laptop-focused honeypots
- **Hard to trigger:** EDR & LSASS protections made it hard to dump creds
- **Over-engineered** but not tailored to the environment

What we learned



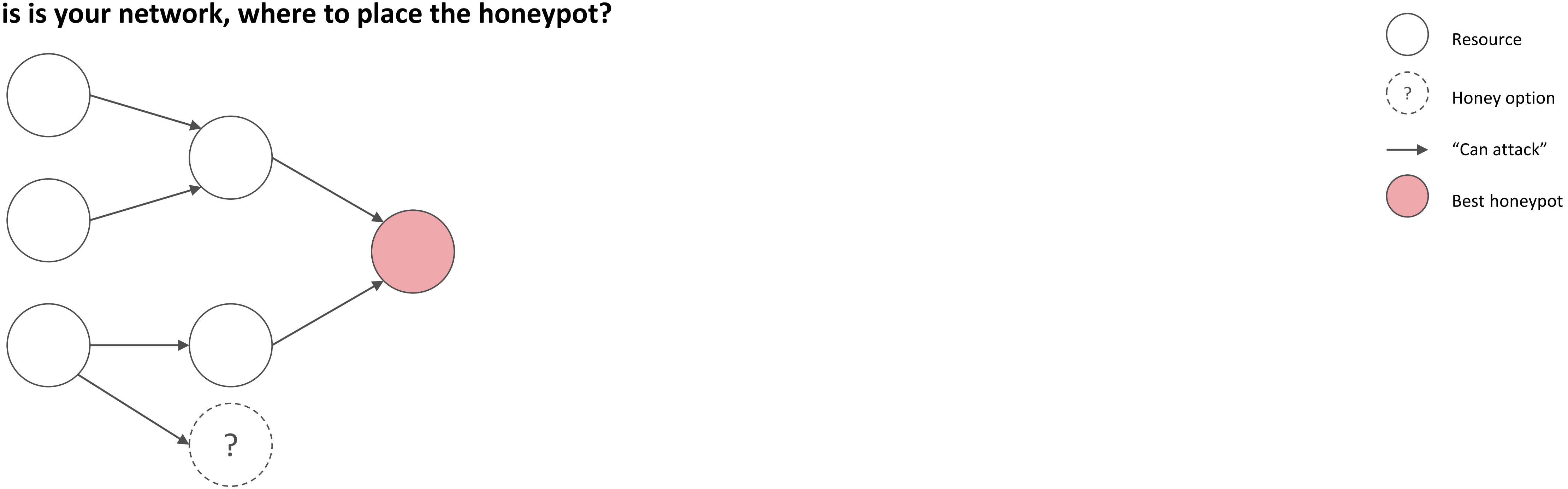
“Simple and well-done” wins

- **Custom-tailored:** consciously integrate Deception into environment
- **Collaboration:** owned by deception team, but admins well involved
- **Cost-effective:** A nice tool doesn’t hurt but you can do without it

Effective honeypots are easily encountered and suggest a worthwhile attack path

Design Goal	Description	Example how to mess it up
Discoverability	▪ **Easy for attackers to find** ▪ Ensuring it serves its purpose as a trap	▪ Fake credential injected to memory ▪ Deployed to laptops only

This is your network, where to place the honeypot?

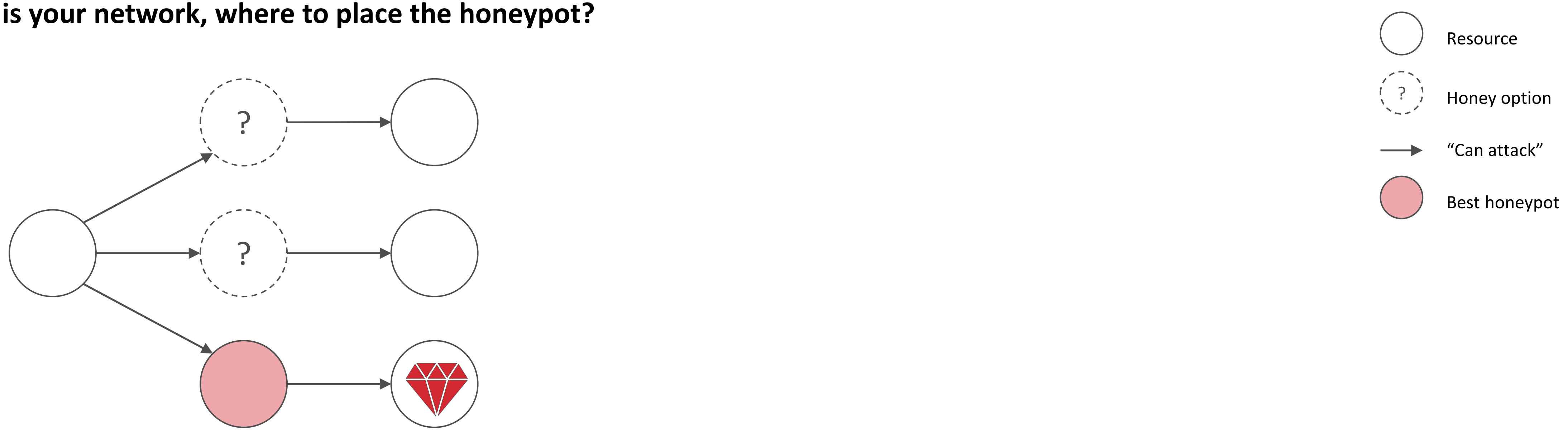


Defenders think in lists. Attackers think in graphs.
As long as this is true, attackers win.
– John Lambert

Effective honeypots are easily encountered and suggest a worthwhile attack path

Design Goal	Description	Example how to mess it up
Discoverability	▪ **Easy for attackers to find** ▪ Ensuring it serves its purpose as a trap	▪ Fake credential injected to memory ▪ Deployed to laptops only
Appeal to Attackers	▪ **Appears valuable** to attackers ▪ Illusion of advancing access or privileges	▪ Honey accounts seem like basic users ▪ But basic users can be obtained by external password spraying → not worth the risk

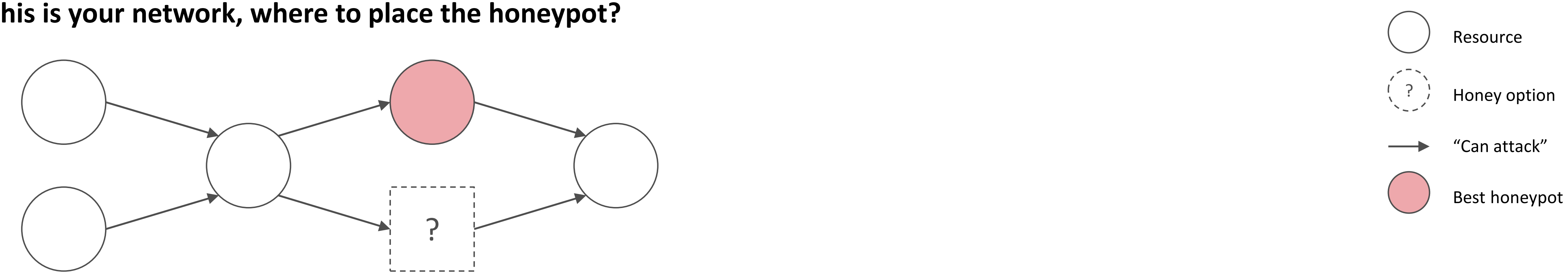
This is your network, where to place the honeypot?



Effective honeypots are easily encountered and suggest a worthwhile attack path

Design Goal	Description	Example how to mess it up
Discoverability	▪ **Easy for attackers to find** ▪ Ensuring it serves its purpose as a trap	▪ Fake credential injected to memory ▪ Deployed to laptops only
Appeal to Attackers	▪ **Appears valuable** to attackers ▪ Illusion of advancing access or privileges	▪ Honey accounts seem like basic users ▪ But basic users can be obtained by external password spraying → not worth the risk
Authenticity	▪ **Blends into the environment** realistically ▪ Hard to identify as a honeypot	▪ Last logon long ago for “normal” user ▪ More cached credentials on machine than *CachedLogonsCount* would allow

This is your network, where to place the honeypot?



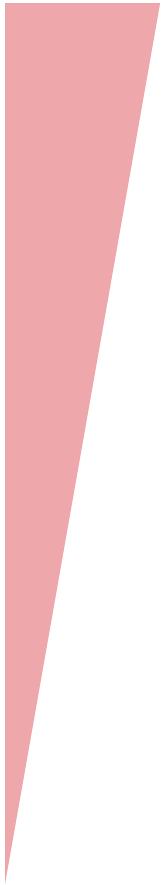
Active Directory authenticity factors: https://www.hub.trimarcsecurity.com/post/the-art-of-the-honeypot-account-making-the-unusual-look-normal

Effective honeypots are easily encountered and suggest a worthwhile attack path

Design Goal	Description	Example how to mess it up
Discoverability	▪ **Easy for attackers to find** ▪ Ensuring it serves its purpose as a trap	▪ Fake credential injected to memory ▪ Deployed to laptops only
Appeal to Attackers	▪ **Appears valuable** to attackers ▪ Illusion of advancing access or privileges	▪ Honey accounts seem like basic users ▪ But basic users can be obtained by external password spraying → not worth the risk
Authenticity	▪ **Blends into the environment** realistically ▪ Hard to identify as a honeypot	▪ Last logon long ago for "normal" user ▪ More cached credentials on machine than *CachedLogonsCount* would allow
Safety	▪ Honeypot is **not exploitable** ▪ Limit risk of things going wrong	▪ High privilege account with password in description but logon hours deny ▪ Admin changes logon hours for testing
Alert precision	▪ Strongly **limit false positive alerts** ▪ Logs should enable investigation	▪ Normal users can find honey files ▪ Source IP who accessed honey account is hidden by gateway

Start small and test, then add more over time!
But where to start?

Different types of deception vary in effectiveness

Type	Description	Alert Mechanism	Examples	Pros / Cons	Usage
Honey network services	▪ **Imitate network service** ▪ Containers, VMs or separate hardware	▪ **Alert on access** ▪ Or based on attack patterns (high-interaction)	▪ **Web or SSH login** that accepts all credentials ▪ SMB file share ▪ Many options on GitHub	+ **Insights** on attacker behavior - **Discoverability** (effort for good coverage)	
Honeytokens Files	▪ **Files that trigger alerts when opened**	▪ **DNS request** ▪ File open event in log	▪ **PDF or office documents** ▪ World-readable ssh keys	+ **Flexible** location (O365, file system…) - **FPs and traceability**	
Auth secrets	▪ Credentials or API tokens	▪ Alert upon attempted authentication	▪ AWS token in Github repo ▪ Hardcoded pw in mobile app	+ Flexible, **less FPs** - **Traceability** for cloud	
Active Directory honeypots	▪ AD object suggesting easy attack path	▪ Sysmon (or EDR) ▪ Monitor specific Event IDs in SIEM	▪ AD user credentials[1] ▪ Kerberoastable user ▪ Group with fake RDP privileges	+ **Fit most attackers' toolset** + Easy and effective - **Require AD admin**	

[1] Technically also a credential, but implementation more like an AD honeypot

Prioritize your roll-out by deception effectiveness and implementation cost

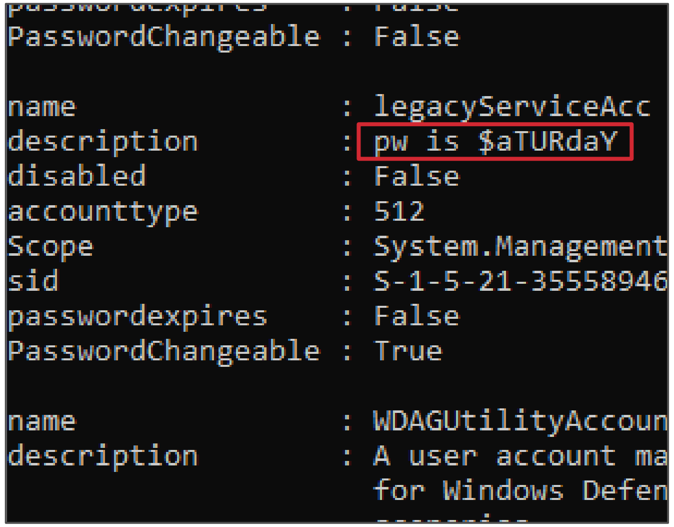
Type	Analysis	Effect
4 **Honey network services**	▪ Useful as internet-connected honeypots for threat-intelligence ▪ **Hard to discover for attackers** in large networks, high roll-out effort for good coverage ▪ Often don’t look very attractive → **Do this last or don’t do it**	
Honeytokens 3 Files 2 Auth secrets	▪ **Can flexibly cover many environments:** cloud, file shares, code repositories, local filesystems, … ▪ Need to ensure a detection can be traced back to attacker ▪ How much sense it makes depends a bit on your environment → **Effective to set up with reasonable effort and cost using a SAAS product**	
1 **Active Directory honeypots**	▪ **Most attack chains touch Active Directory** at some point ▪ Attacker tooling – especially of ransomware gangs – is optimized for it ▪ Requires Sysmon+SIEM, EDR or a solution like MDI to alert on AD events → **Perfect location for deception – let’s see what we can do here!**	
Pro-tip	Red team reports can provide inspiration for what honeypots to build	

N Suggested deployment order for “standard” environments

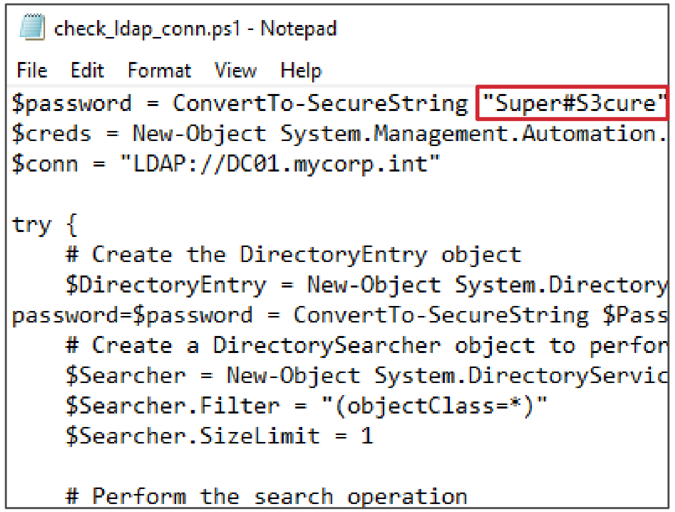
Agenda

- Introduction
- Deception strategy
- **Must-have AD honeypots**
- Tool release: ADCS deception

1 Hiding credentials for attractive AD accounts is simple yet effective



```
passwordexpires      : False
PasswordChangeable : False

name                 : legacyServiceAcc
description          : pw is $aTURdaY
disabled             : False
accounttype          : 512
Scope                : System.Management
sid                  : S-1-5-21-35558946
passwordexpires      : False
PasswordChangeable : True

name                 : WDAGUtilityAccoun
description          : A user account ma
                       for Windows Defen
```

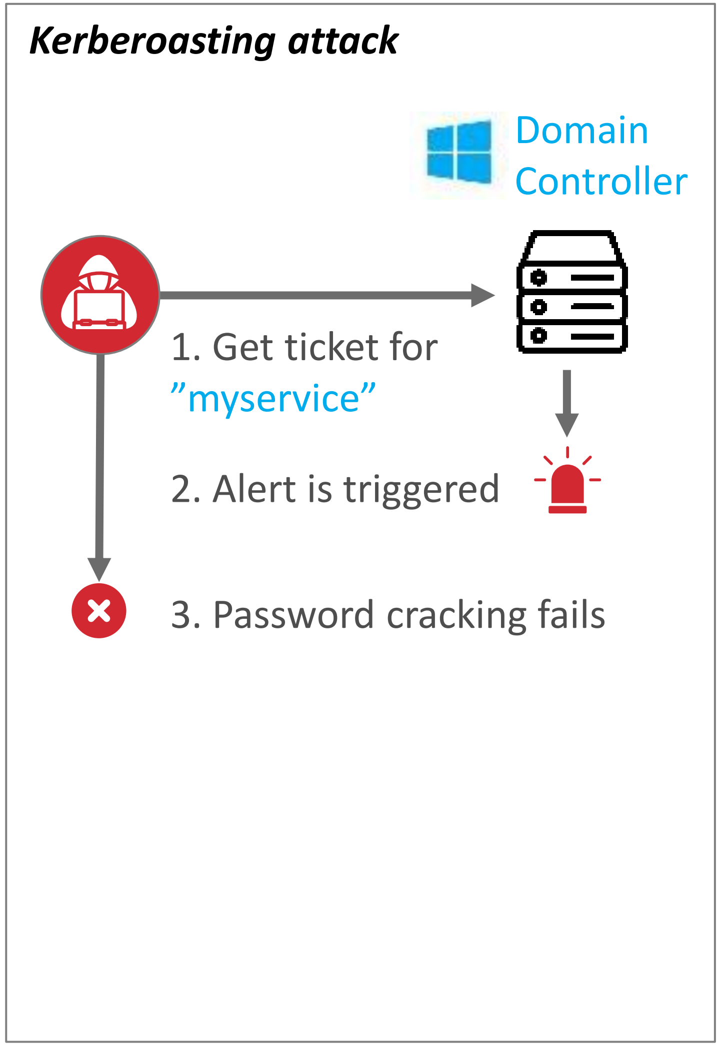


```
$password = ConvertTo-SecureString "Super#S3cure"
$creds = New-Object System.Management.Automation.
$conn = "LDAP://DC01.mycorp.int"

try {
    # Create the DirectoryEntry object
    $DirectoryEntry = New-Object System.Directory
password=$password = ConvertTo-SecureString $Pass
    # Create a DirectorySearcher object to perfor
    $Searcher = New-Object System.DirectoryServic
    $Searcher.Filter = "(objectClass=*)"
    $Searcher.SizeLimit = 1

    # Perform the search operation
```

Design Goal	Guidance
Discoverability	▪ **Get creative where to hide fake credentials** ▪ Description field in AD object, PowerShell script on SYSVOL, code repos, file of rolled out to endpoints
Appeal to attackers	▪ Should be a privileged account (or at least seem like it) ▪ Could be from group membership, permissions visible in LDAP, or naming scheme
Authenticity	a. **Active account** with very rare failed logons b. **Dedicated honey account** by recycling old account for RID, lastlogon, BadPasswordTime, …
Safety	▪ **Password hint should be wrong** ▪ We advise against real creds with logon hours deny
Alert precision	▪ Windows event ID 4625 (failed logon) ▪ Windows event ID 4768 (TGT request) ▪ SIEM can find suitable accounts with few failed logins

More details and an additional GPP honeypot: https://www.hub.trimarcsecurity.com/post/the-art-of-the-honeypot-account-making-the-unusual-look-normal

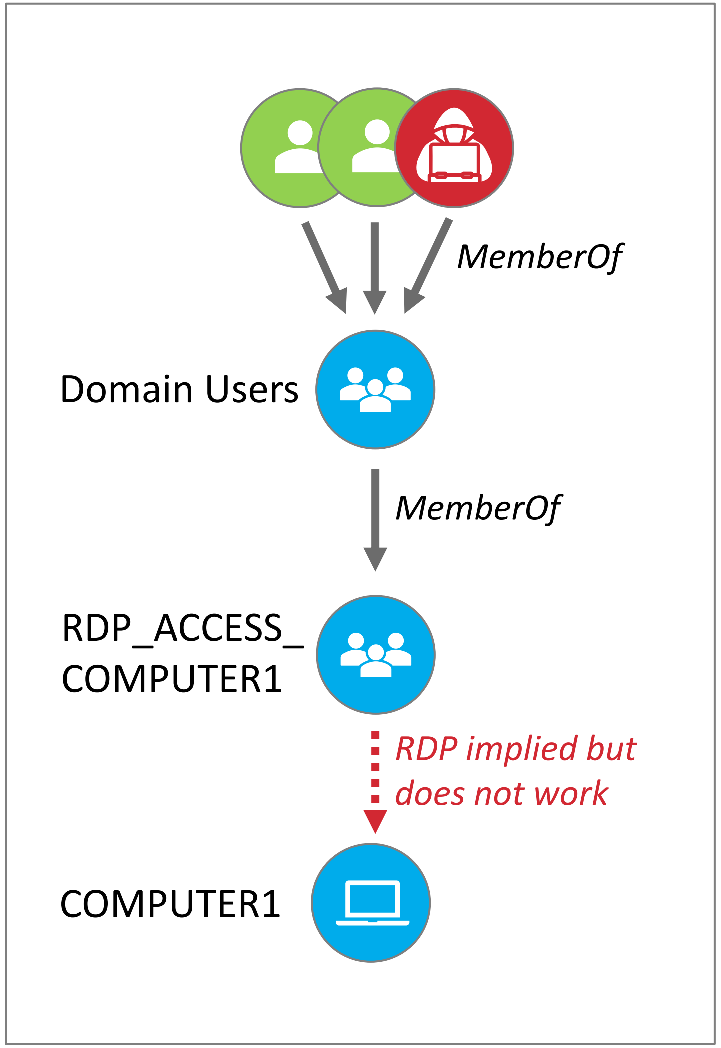
❷ Kerberoasting honeypots appeal to a common attack vector



Kerberoasting attack

Design Goal	**Guidance**
Discoverability	▪ Attackers query LDAP for users with SPN
Appeal to attackers	▪ Accounts with **older passwords are more attractive** ▪ **Human accounts are attractive** in general
Authenticity	a. **Set SPN on normal user** as if it was a forgotten test b. **Dedicated honey account** by recycling ▪ Consider how common RC4 is in your environment
Safety	▪ **Ensure account has strong, auto-generated password** ▪ Account owner needs to be aware
Alert precision	▪ Event ID 4769 (service ticket request)

❸ A group claiming to grant RDP privileges for all users is easy to find for attackers



Design Goal	Guidance
Discoverability	▪ Attackers usually review group membership ▪ **All users are member of “Domain Users”** in LDAP
Appeal to attackers	▪ **Group name suggests RDP access** privileges ▪ Could also do the same with local admin ▪ Ideally, machine seems important
Authenticity	▪ **Machine OS >= Windows Server 2016** (no easy RDP privilege enumeration anymore) ▪ Pick description, OUs, etc. to make it fit in
Safety	▪ **No actual RDP access**
Alert precision	▪ **Event ID 4624/4625** (failed logon) ▪ Might focus on type 10 but if you can, include others ▪ Existing machine with few failed logons or new one

You can do this with all types of failed login you can alert on with low noise (e.g. fake “VCENTER-ADMIN” group)

One more thing...

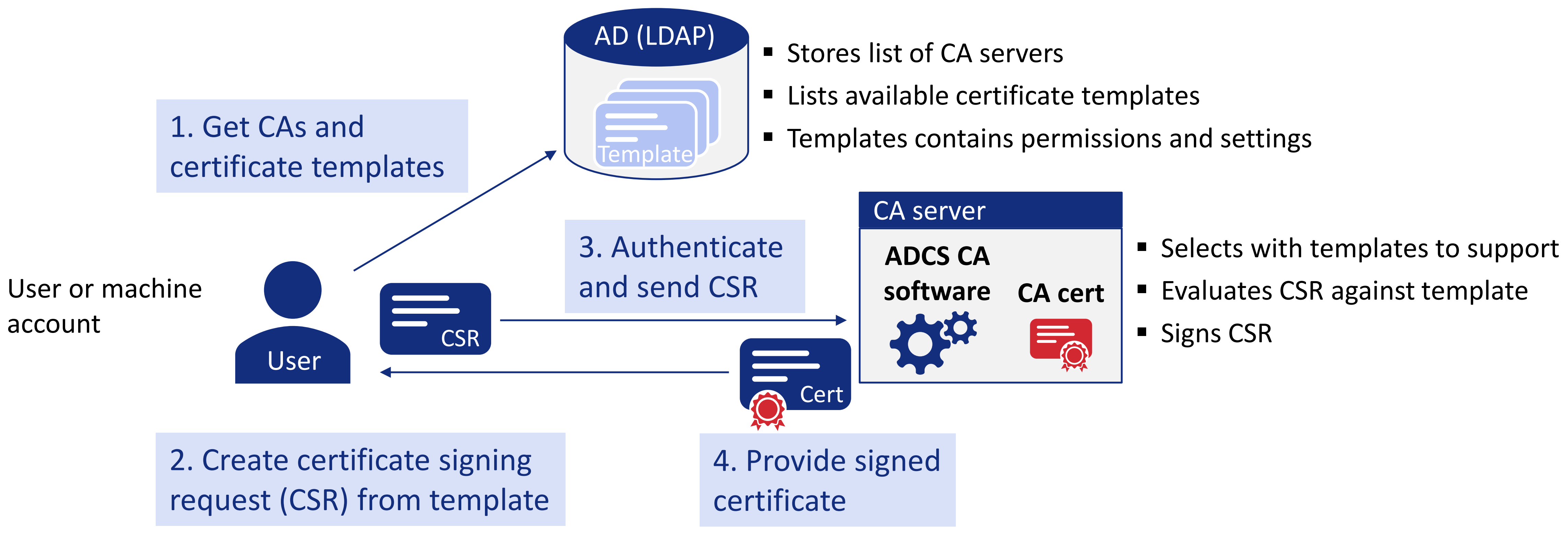
Agenda

- Introduction
- Deception strategy
- Must-have AD honeypots
- **Tool release: ADCS deception**

Active Directory Certificate Services manages critical authentication

What is ADCS?

- Microsoft's solution for public key infrastructure (PKI)
- Creates certificates for authentication, code signing, email, server authentication, …
- Used for device authentication, TLS certificates, smartcard authentication, …
- Can create authentication certificates for everyone → Tier 0



- Stores list of CA servers
- Lists available certificate templates
- Templates contains permissions and settings

User or machine account

1. Get CAs and certificate templates

2. Create certificate signing request (CSR) from template

3. Authenticate and send CSR

4. Provide signed certificate

- Selects with templates to support
- Evaluates CSR against template
- Signs CSR

ADCS is complex to configure, and mistakes have high impact

Common misconfigurations in ADCS

ESC-1	Certificate template allows enrolling user to specify who the certificate is valid for → “Domain admin”
ESC-2 ESC-3	User certificate can be used to enroll new certificates → Create one for Domain Admin
ESC-4	User has write permission to certificate template → introduce ESC1
ESC-5	Compromise one of the ADCS objects in AD (computer object, container, …)
ESC-6	CA-level setting that basically enables ESC1

ESC-7	Bypass manager approval on certificate templates that require it
ESC-8 ESC-11	No protection against relay attacks → Compromise account when coercing authentication
ESC-9 ESC-10	Obtain certificate as any Domain user by modifying the UPN of a controlled user
ESC-12 ESC-13	Chain of conditions and quite complicated, you probably did not read this far → ignored on this slide

▨ Misconfigurations we see the most

ESC 1-8: https://posts.specterops.io/certified-pre-owned-d95910965cd2
ESC 9-10: https://research.ifcr.dk/certipy-4-0-esc9-esc10-bloodhound-gui-new-authentication-and-request-methods-and-more-7237d88061f7
ESC 11: https://blog.compass-security.com/2022/11/relaying-to-ad-certificate-services-over-rpc/
ESC 12: https://pkiblog.knobloch.info/esc12-shell-access-to-adcs-ca-with-yubihsm
ESC 13: https://posts.specterops.io/adcs-esc13-abuse-technique-fda4272fbd53

ADCS is a great location for a honeypot

Common misconfigurations in ADCS

ESC-1	Certificate template allows enroll… who the certificate is valid for →
ESC-2 ESC-3	User certificate can be used to er… → Create one for Domain Admin
ESC-4	User has write permission to cert… → introduce ESC1
ESC-5	Compromise one of the ADCS ob… (computer object, container, …)
ESC-6	CA-level setting that basically ena…

…nager approval on certificate templates …e it

…ion against relay attacks → Compromise …hen coercing authentication

…tificate as any Domain user by modifying … a controlled user

…nditions and quite complicated, you …id not read this far → ignored on this slide

Previously exploited in client engagements

MICROSOFT: *SLAPS THE ROOF OF AD CS*
THIS BAD BOY CAN FIT SO MANY MISCONFIGURATIONS IN IT

Why hackers target ADCS	Why it would be a great honeypot
1. **Easy access** (can be used by all domain users)	**Discoverability** (easily found from different points)
2. **Complex configuration** (hard to configure securely)	**Authenticity** (occurs often in real environments)
3. **Tooling available** (run certipy to find vulns)	**Discoverability** (in the playbook of most TIs)
4. **Significant impact** (full environment compromise)	**Appeal to attackers** (juicy to exploit)
5. **Under-monitored** (likely stay undetected)	**Appeal to attackers** (attacker feels safe to exploit)

Image from https://posts.specterops.io/certified-pre-owned-d95910965cd2

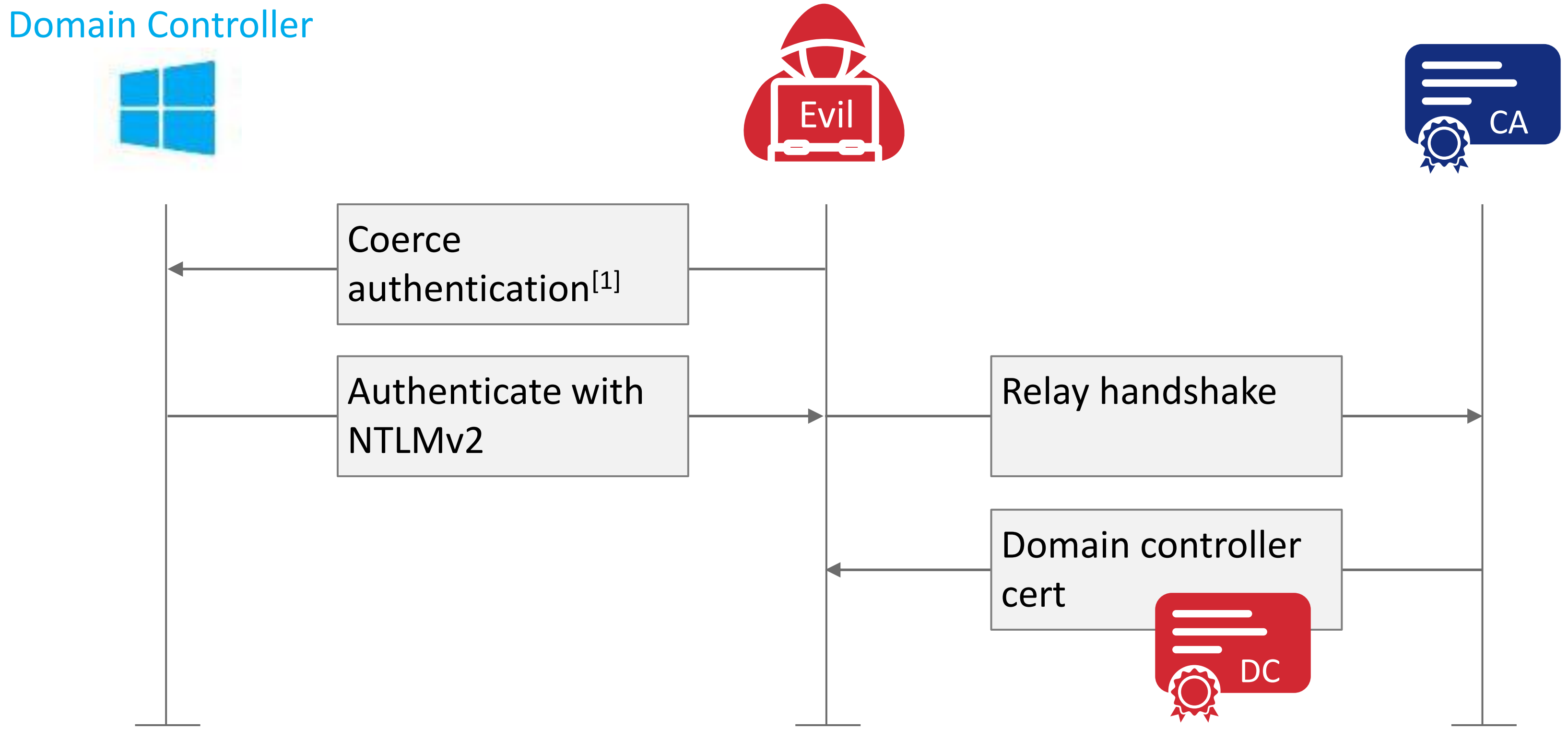
An ESC8 honeypot is feasible but was not effective enough for us

ESC8 issue

- CA server has web enrollment enabled and supports HTTP (or lacks EPA on HTTPS)

→ Attacker that receives NTLMv2 authentication handshake can relay it to receive an authentication certificate

Example attack flow against Domain Controller



Domain Controller

Evil

CA

Coerce authentication[1]

Authenticate with NTLMv2

Relay handshake

Domain controller cert

DC

Analysis

- Attacker tooling checks ESC8 by connecting to the CA on HTTP
- **Honeypot feasible** in a safe way by mocking parts of the CA web server
- **Problem:** relays and coercion can be tricky for attackers → **not super easy to step into the trap**
- **Let's see if we can find a better option**

[1] Via printspooler, petitpotam, dfscoerce, or whatever is found next

ADCS policy modules can evaluate and block CSRs on the CA

We followed many paths for an ADCS honeypot

Mock web enrolment to fake ESC8

- Feasible and safe option
- Exploitation needs auth coercion (tricky)

→ Harder for hackers to step into trap

ESC3 with enrolment restrictions

- Place restrictions on second required cert
- Attacker still obtains enrolment certificate

→ Too risky

Auto-revocation

- Dangerous time window with valid cert
- An OCSP setup could work

→ We don’t understand revocation enough

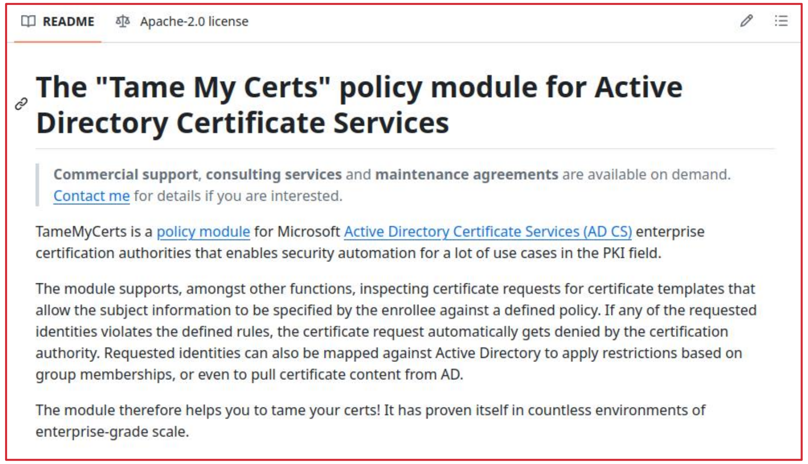
The TameMyCerts policy module saved the day

ADCS policy modules

- Receives and evaluate certificate requests
- Can issue or deny
- Implemented as a DLL on the CA

TameMyCerts[1]

- Policy module developed and maintained by Uwe Gradenegger[2]
- Developed for fine grained and automated certificate issuance checks
- Rules for evaluation are specified as XML



The "Tame My Certs" policy module for Active Directory Certificate Services

> **Commercial support**, **consulting services** and **maintenance agreements** are available on demand. Contact me for details if you are interested.

TameMyCerts is a policy module for Microsoft Active Directory Certificate Services (AD CS) enterprise certification authorities that enables security automation for a lot of use cases in the PKI field.

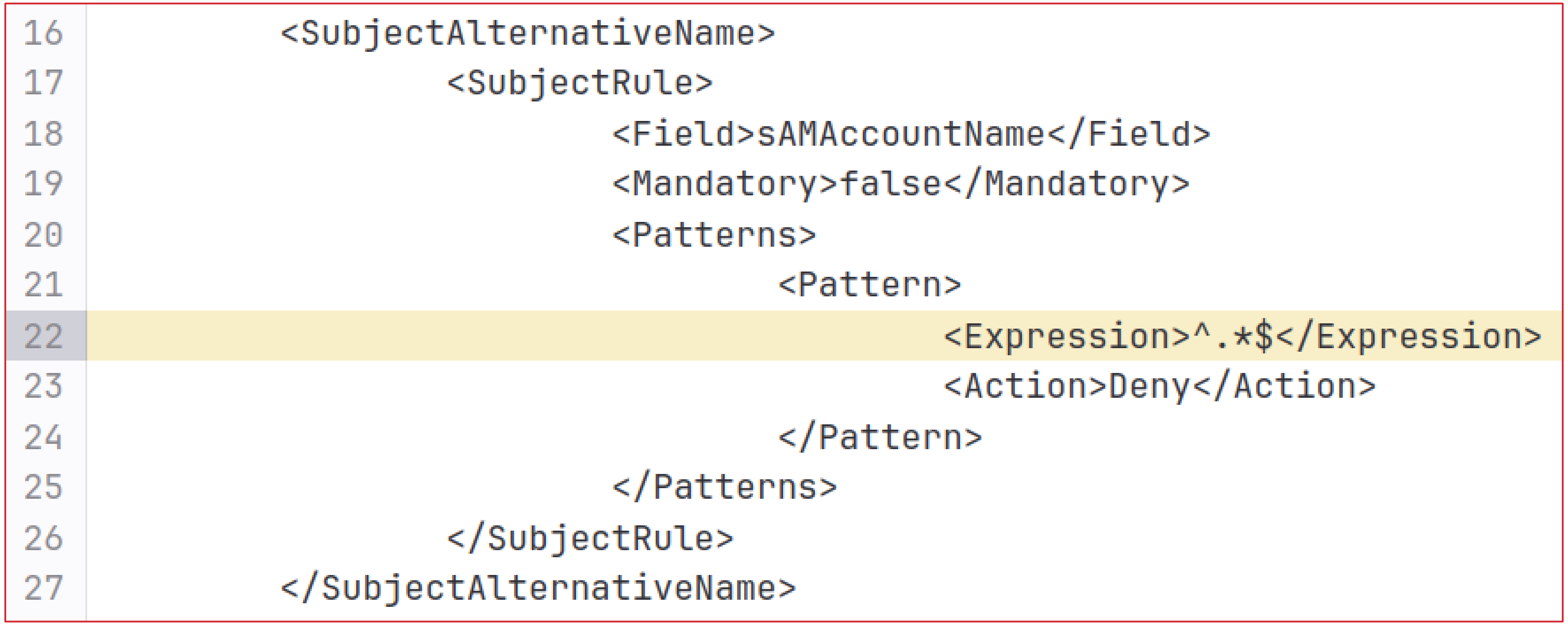
The module supports, amongst other functions, inspecting certificate requests for certificate templates that allow the subject information to be specified by the enrollee against a defined policy. If any of the requested identities violates the defined rules, the certificate request automatically gets denied by the certification authority. Requested identities can also be mapped against Active Directory to apply restrictions based on group memberships, or even to pull certificate content from AD.

The module therefore helps you to tame your certs! It has proven itself in countless environments of enterprise-grade scale.

[1] https://github.com/Sleepw4lker/TameMyCerts

[2] https://www.gradenegger.eu/de/

TameMyCerts enables us to build a simple yet effective ESC1 honeypot



```
        <SubjectAlternativeName>
                <SubjectRule>
                        <Field>sAMAccountName</Field>
                        <Mandatory>false</Mandatory>
                        <Patterns>
                                <Pattern>
                                        <Expression>^.*$</Expression>
                                        <Action>Deny</Action>
                                </Pattern>
                        </Patterns>
                </SubjectRule>
        </SubjectAlternativeName>
```

- In ESC1, the certificate template has the CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT flag set
- It allows the user to specify a subject alternative name (SAN) in the certificate request
- The TameMyCerts policy file above blocks the CSR if it includes a SAN
- This prevents malicious use while still allowing users to create certificates for themselves

We can differentiate between suspicious and clearly malicious use of the honeypot

Event source	Event ID	Alerts
CA built-in[1]	**4886 –** Certificate enrollment requested	**Medium** Honey template used
	4887 – Certificate issued	▪ Possible, but 4886 has more coverage
	4888 – Certificate request denied	▪ Possible, but less precise than TameMyCerts 6
TameMyCerts logging	**6 –** CSR denied due to policy violation	**Critical** Attempted exploitation
	Future plan – adapt events to honeypot use	

- SIGMA rules to be SIEM-agnostic
- Improvements planned or the future when supporting various honey templates

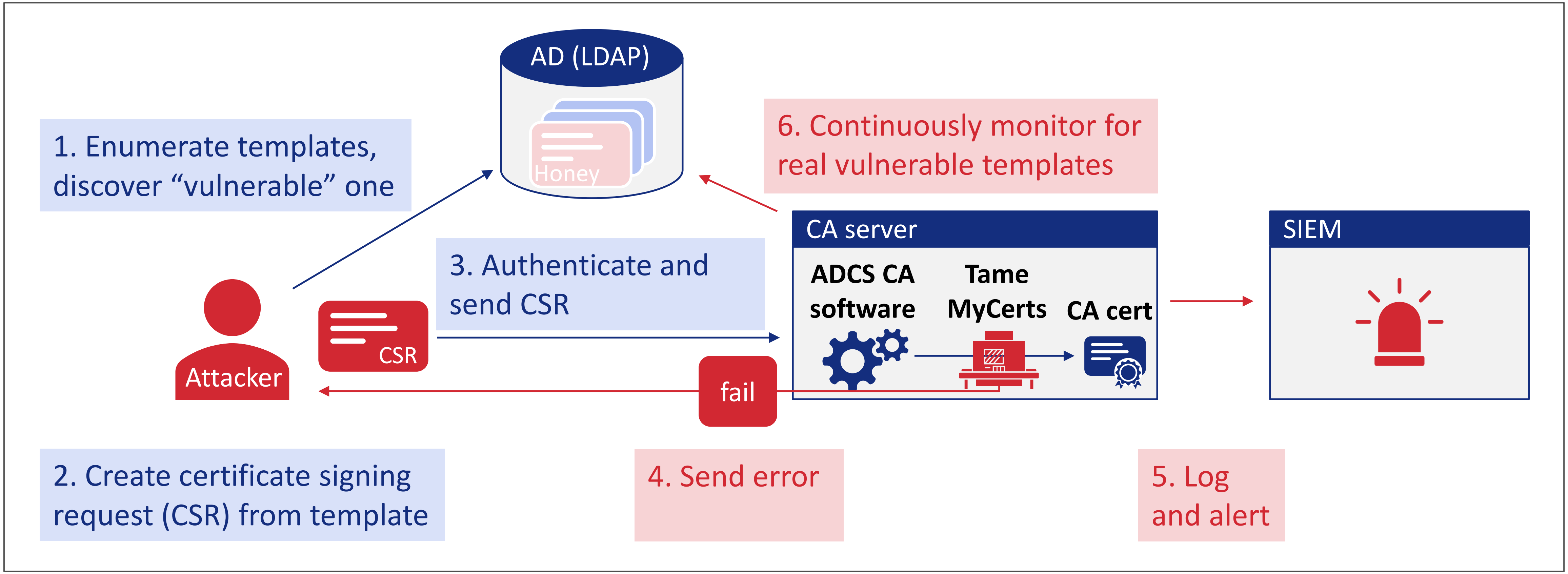
Security Research Labs

[1] Requires extended audit log to be enabled to exist with sufficient information

We release Certiception, our tooling to setup ADCS honeypots

Certiception automates your ADCS honeypot setup

- Set up a new CA, add a “vulnerable” ESC1 template and enable it only on the new CA
- Install and configure TameMyCerts to prevent issuance if CSR contains SAN
- Enable the extended audit log to get template names in CA event logs
- Print a SIGMA rule to set up alerting in your SIEM
- Set up continuous checks to catch any other CA enabling the vulnerable template


AD (LDAP)
Honey
1. Enumerate templates, discover “vulnerable” one
6. Continuously monitor for real vulnerable templates
CA server
SIEM
3. Authenticate and send CSR
ADCS CA software
Tame MyCerts
CA cert
CSR
Attacker
fail
2. Create certificate signing request (CSR) from template
4. Send error
5. Log and alert

https://github.com/srlabs/certiception

We release Certiception, our tooling to setup ADCS honeypots

Prerequisites

- Domain-joined Windows server for CA
- Machine with Ansible and WinRM connectivity to server
- Local admin the CA server
- Enterprise Admin account to create and register CA
- Basic Domain account without any privileges for Certify

Certiception setup flow

How to set up an ADCS honeypot

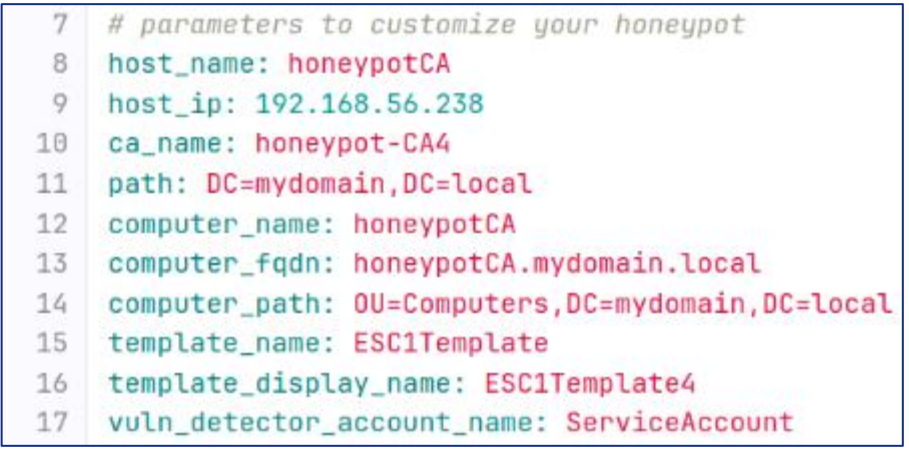
1. Choose unique parameters for your Honeypot
2. (optional) Create EDR exception for future Certify location
3. **Execute Certiception** via Ansible
4. Connect event logs to your SIEM and configure alerts
5. Verify and manually test your setup

Security and safety

Disclaimer

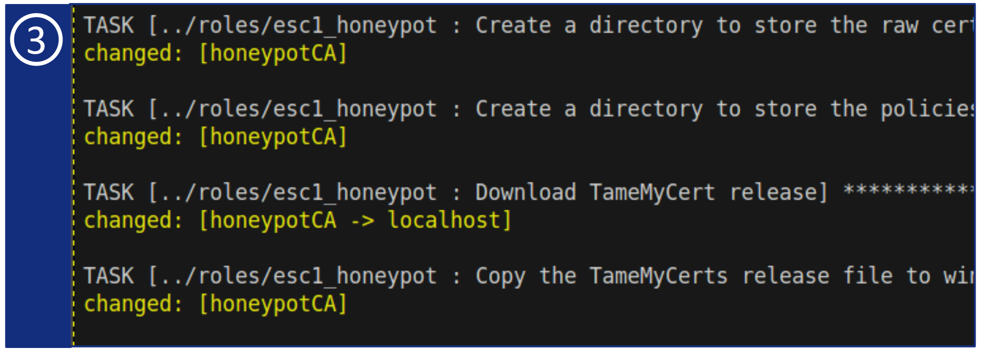
- Use at your own risk – you are responsible for what you set up with Certiception
- Read the code and understand what it does
- We expect potential for improvements after this release
- More on this topic: https://github.com/srlabs/Certiception

1



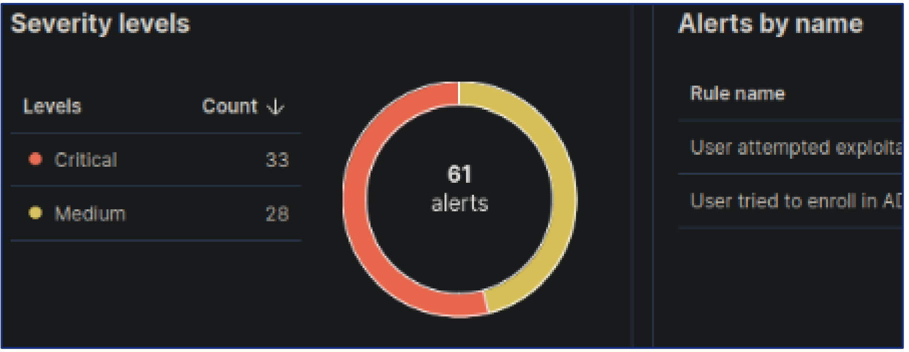
```
# parameters to customize your honeypot
host_name: honeypotCA
host_ip: 192.168.56.238
ca_name: honeypot-CA4
path: DC=mydomain,DC=local
computer_name: honeypotCA
computer_fqdn: honeypotCA.mydomain.local
computer_path: OU=Computers,DC=mydomain,DC=local
template_name: ESC1Template
template_display_name: ESC1Template4
vuln_detector_account_name: ServiceAccount
```

3



```
TASK [../roles/esc1_honeypot : Create a directory to store the raw cert
changed: [honeypotCA]

TASK [../roles/esc1_honeypot : Create a directory to store the policie
changed: [honeypotCA]

TASK [../roles/esc1_honeypot : Download TameMyCert release] **********
changed: [honeypotCA -> localhost]

TASK [../roles/esc1_honeypot : Copy the TameMyCerts release file to wi
changed: [honeypotCA]
```

4



Severity levels

Levels	Count ↓
● Critical	33
● Medium	28

Alerts by name

Rule name
User attempted exploita
User tried to enroll in AD

Demo Time!

Stealing credentials from LSASS
Asking a CA for a certificate
imgflip.com

Stepping into an ADCS honeypot

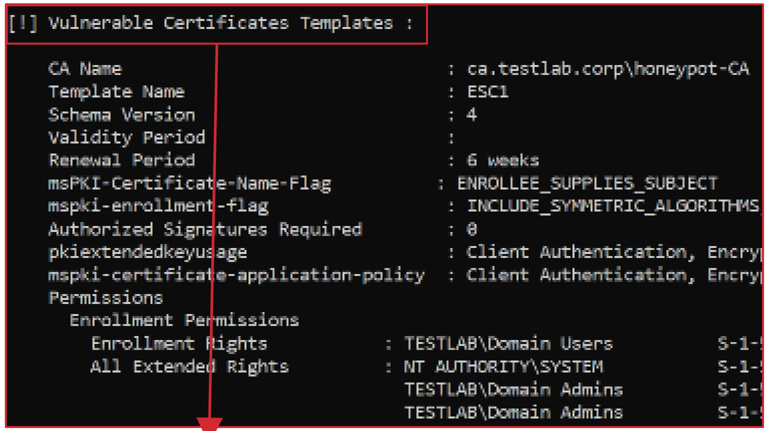
Offensive security tooling recognizes Certiception as a vulnerable ESC1 template

Certify

Certipy

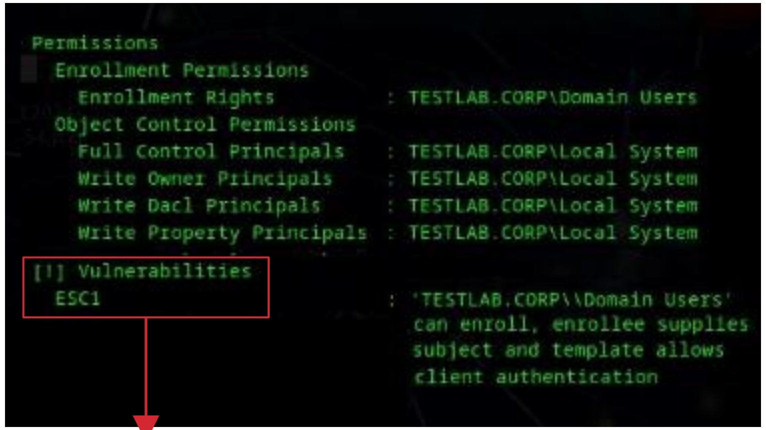
BloodHound

Discovery



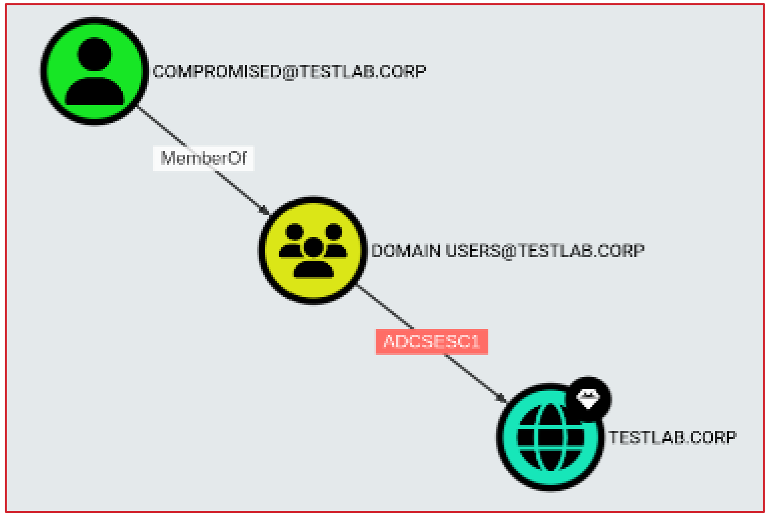
```
[!] Vulnerable Certificates Templates :

    CA Name                               : ca.testlab.corp\honeypot-CA
    Template Name                         : ESC1
    Schema Version                        : 4
    Validity Period                       :
    Renewal Period                        : 6 weeks
    msPKI-Certificate-Name-Flag          : ENROLLEE_SUPPLIES_SUBJECT
    mspki-enrollment-flag                 : INCLUDE_SYMMETRIC_ALGORITHMS
    Authorized Signatures Required        : 0
    pkiextendedkeyusage                   : Client Authentication, Encry
    mspki-certificate-application-policy  : Client Authentication, Encry
    Permissions
      Enrollment Permissions
        Enrollment Rights           : TESTLAB\Domain Users          S-1-
        All Extended Rights         : NT AUTHORITY\SYSTEM           S-1-
                                      TESTLAB\Domain Admins         S-1-
                                      TESTLAB\Domain Admins         S-1-
```

```
[!] Vulnerable Certificates Templates
```

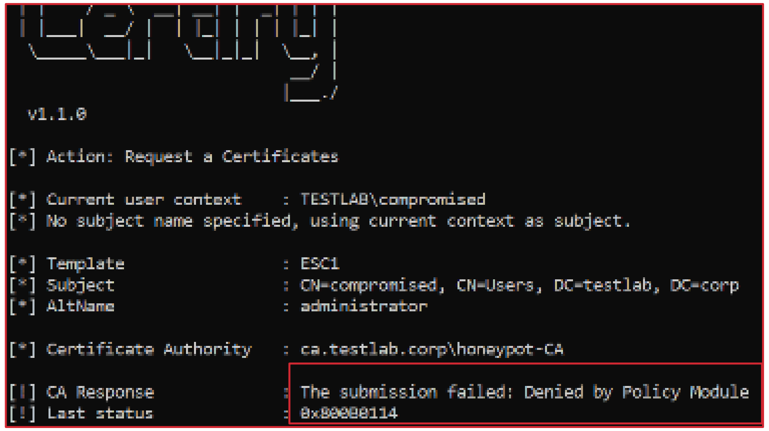


```
Permissions
  Enrollment Permissions
    Enrollment Rights           : TESTLAB.CORP\Domain Users
  Object Control Permissions
    Full Control Principals     : TESTLAB.CORP\Local System
    Write Owner Principals      : TESTLAB.CORP\Local System
    Write Dacl Principals       : TESTLAB.CORP\Local System
    Write Property Principals   : TESTLAB.CORP\Local System
[!] Vulnerabilities
  ESC1                          : 'TESTLAB.CORP\\Domain Users'
                                  can enroll, enrollee supplies
                                  subject and template allows
                                  client authentication
```

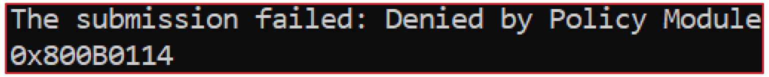
```
[!] Vulnerabilities
```



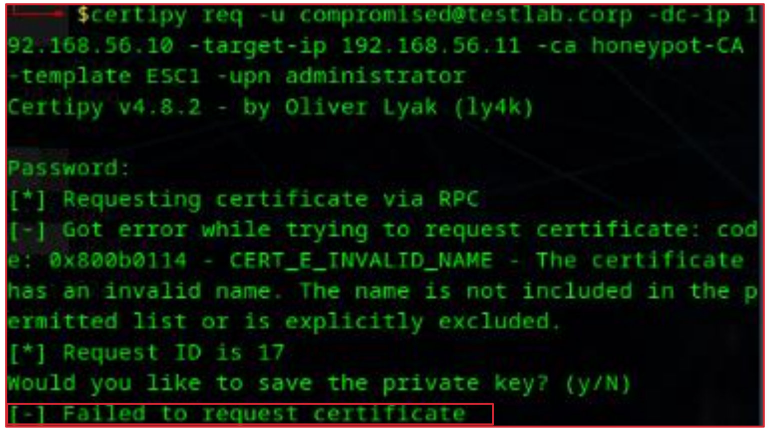
Exploitation attempt



```
  v1.1.0

[*] Action: Request a Certificates

[*] Current user context    : TESTLAB\compromised
[*] No subject name specified, using current context as subject.

[*] Template                : ESC1
[*] Subject                 : CN=compromised, CN=Users, DC=testlab, DC=corp
[*] AltName                 : administrator

[*] Certificate Authority   : ca.testlab.corp\honeypot-CA

[!] CA Response             : The submission failed: Denied by Policy Module
[!] Last status             : 0x80080114
```



```
The submission failed: Denied by Policy Module
0x800B0114
```



```
$certipy req -u compromised@testlab.corp -dc-ip 1
92.168.56.10 -target-ip 192.168.56.11 -ca honeypot-CA
-template ESC1 -upn administrator
Certipy v4.8.2 - by Oliver Lyak (ly4k)

Password:
[*] Requesting certificate via RPC
[-] Got error while trying to request certificate: cod
e: 0x800b0114 - CERT_E_INVALID_NAME - The certificate
has an invalid name. The name is not included in the p
ermitted list or is explicitly excluded.
[*] Request ID is 17
Would you like to save the private key? (y/N)
[-] Failed to request certificate
```

```
would you like to save the private
[-] Failed to request certificate
```

Not applicable

Future work

Us

- Support placing honey templates on existing CAs
- Implement other ESC misconfigurations
- Investigate additional hardening options
- Add less suspicious error message on denied CSR
- Setup with lower priv. accounts instead of enterprise admin

We need you

- Let community scrutinize safety of the honeypot
- Investigate and mitigate ways of fingerprinting

Takeaways

1 **Honeypots provide meaningful high-relevance alerts**
for threats that make it past initial defenses

2 **Custom-tailoring is necessary**
to make deception appealing to attackers

3 **SRLabs' Certiception is the ADCS honeypot you always wanted**

Questions?

Certicept your threats

https://github.com/srlabs/Certiception

Balthasar Martin <infosec.exchange/@balthasar & @BalthasarMartin>
Niklas van Dornick <@n1v4d0>