

## **The Red Teamer's Guide To Deception**

Building effective internal honeypots

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



**Security  
Research  
Labs**



---

## 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

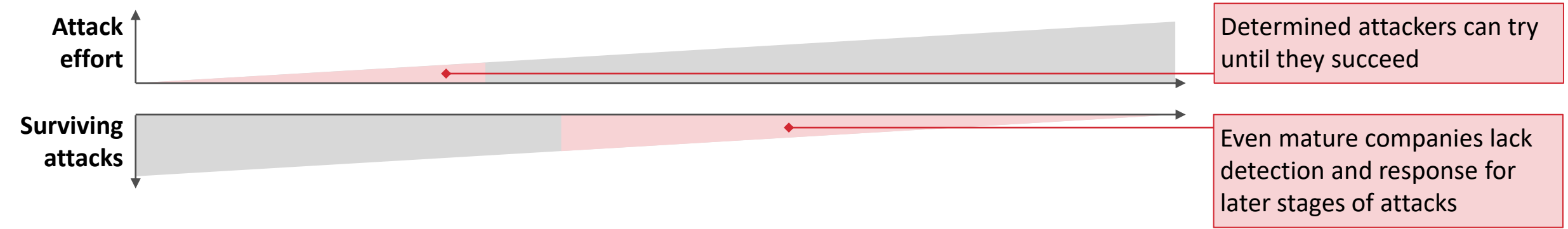
Balthasar's mistake

```
74945 ?      Ss      0:00
74958 ?      Ss      0:00
74973 pts/1   R+      0:00
oot@:/home/centos# ls [5:43 PM] Fabian Becker
ls: cannot access '[5:43]': No such file or directory
ls: cannot access 'PM]': No such file or directory
ls: cannot access 'Fabian': No such file or directory
ls: cannot access 'Becker': No such file or directory
oot@:/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
<div> <b>Telco</b></div> <div>Assumed breach in test environment</div>	SSH keys for various users → Escalated privileges on shared server → Account for production automation	<div><b>Subscriber data and SMS access</b></div> <div> No alert triggered  SOC didn't have a chance</div>
<div> <b>Financial</b></div> <div>Malware and credential phishing </div>	Exploit Java servers, spray external pws → Abused left-over AD permissions → Admin for identity management	<div><b>Control over banking interface</b></div> <div> EDR caught malware  Continued other foothold</div>
<div> <b>Manufac.</b></div> <div>Assumed breach with basic account</div>	Credentials in code repositories → Extensive persistence, local recon 	<div><b>Ransomware via full AD compromise</b></div> <div> EDR &amp; identity monitoring  Categorized as harmless</div>



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

Problem	Details	Consequences
Effort to achieve EDR and log coverage	<ul style="list-style-type: none"><li>Requires much leg-work and communication</li><li><b>Pareto principle: last 20% take 80% of work</b></li></ul>	<ul style="list-style-type: none"><li>Attackers with time or luck can find <b>“that under-monitored system”</b></li></ul>
Complex corporate networks	<ul style="list-style-type: none"><li>Large volume of alerts that is hard to tune</li><li><b>“Weird” things happen regularly</b></li></ul>	<ul style="list-style-type: none"><li>Not every alert can be investigated in-depth</li><li><b>True positive alerts are overseen or not followed-up</b> upon with full response</li></ul>
Application-specific knowledge gap	<ul style="list-style-type: none"><li><b>SOC has limited knowledge about applications</b></li><li>Requires domain-expert support to write rules or evaluate alerts</li></ul>	<ul style="list-style-type: none"><li><b>Incorrect classification of alerts</b></li><li>Example: alert for activity by built-in domain admin but analyst doesn't realize because it was renamed</li></ul>
Analyst Turnover	<ul style="list-style-type: none"><li>Undesirable work style (shift work, factory style)</li><li><b>Trained analysts leave</b> for better positions</li></ul>	<ul style="list-style-type: none"><li>Lower analysis quality in general</li></ul>
Analysis	<ul style="list-style-type: none"><li><b>Attackers with time/skill/luck trigger few alerts</b></li><li>SOCs are designed to handle large volume with okay-ish coverage and investigation result precision</li></ul>	<ul style="list-style-type: none"><li>Attack chains with e.g. few “medium” alerts have a <b>realistic chance get through</b></li></ul> <p>→ <b>Blue team needs a “smoke detector” to catch these cases just before the fire is out of control</b></p>

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

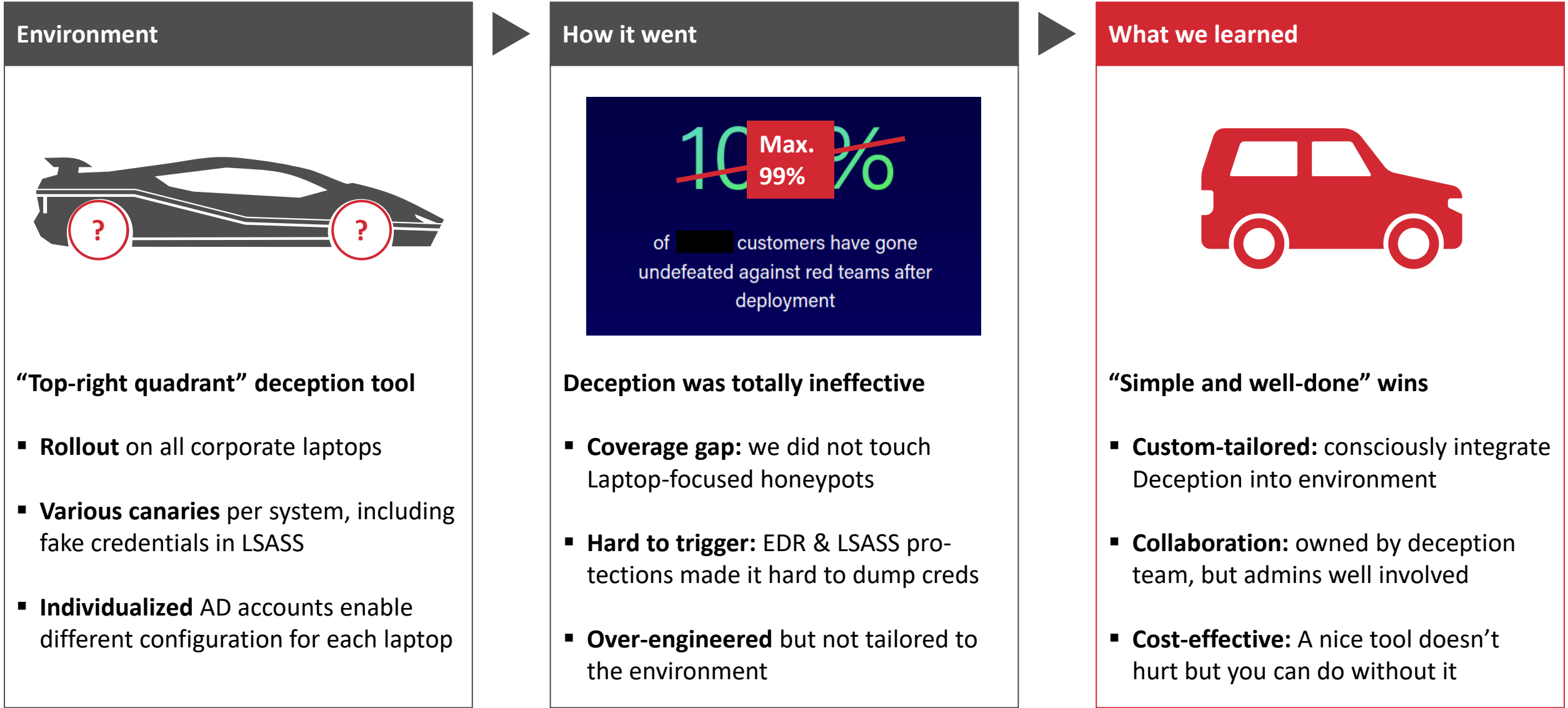
Definition	<b>Internal honeypot</b> (aka. canary, aka. deception tech): A strategically placed system, account, or vulnerability designed to mimic legitimate assets, serving as a trap for attackers	
Example	A pair of invalid credentials placed on a server, triggering an alert when used	
Advantages	1. Low roll-out complexity and maintenance	<ul style="list-style-type: none"><li>▪ Deploy once to a few easily-discovered locations</li><li>▪ Use existing technologies like a SIEM</li><li>▪ Low footprint, limited maintenance</li></ul>
	2. Low-noise detections	<ul style="list-style-type: none"><li>▪ Honeypots are not used by legitimate users</li><li>▪ They can be set up to only trigger on clearly malicious activity</li></ul>
	3. High-relevance alerts	<ul style="list-style-type: none"><li>▪ Are triggered during lateral movement and privilege escalation</li><li>▪ Honeypot exploitation likely indicates a significant threat</li><li>▪ Allows to trigger critical alerts, directly to a senior analyst</li></ul>
Strategic Impact	<ul style="list-style-type: none"><li>▪ <b>Effective alerting that can prevent the worst</b> in cases where initial infection stays undetected</li><li>▪ <b>Great cost-benefit</b> ratio for catching attackers</li><li>▪ <b>Slowing down attackers</b> by forcing them to second-guess their attacks</li></ul>	

- 
- Introduction

- ▶ **Deception strategy**

- Must-have AD honeypots
  - Tool release: ADCS deception
-

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





# Effective honeypots are easily encountered and suggest a worthwhile attack path

## Design Goal

Discoverability

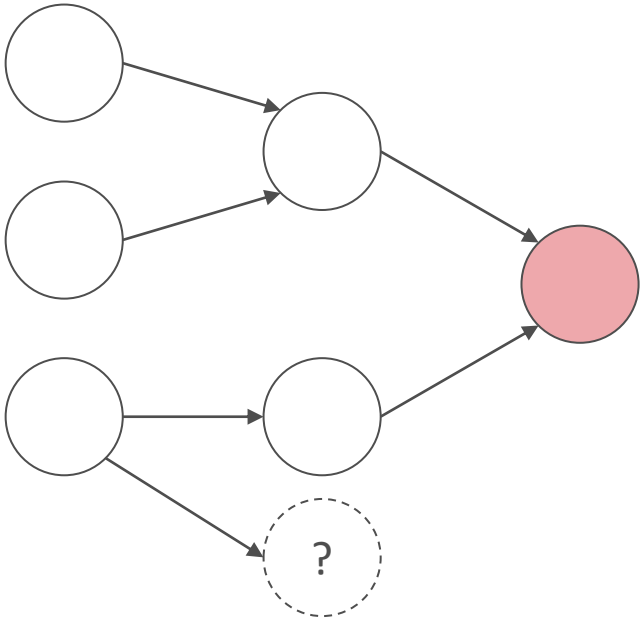
## Description

- Easy for attackers to find
- Ensuring it serves its purpose as a trap

## Example how to mess it up

- Fake credential injected to memory
- Deployed to laptops only



This is your network, where to place the honeypot?

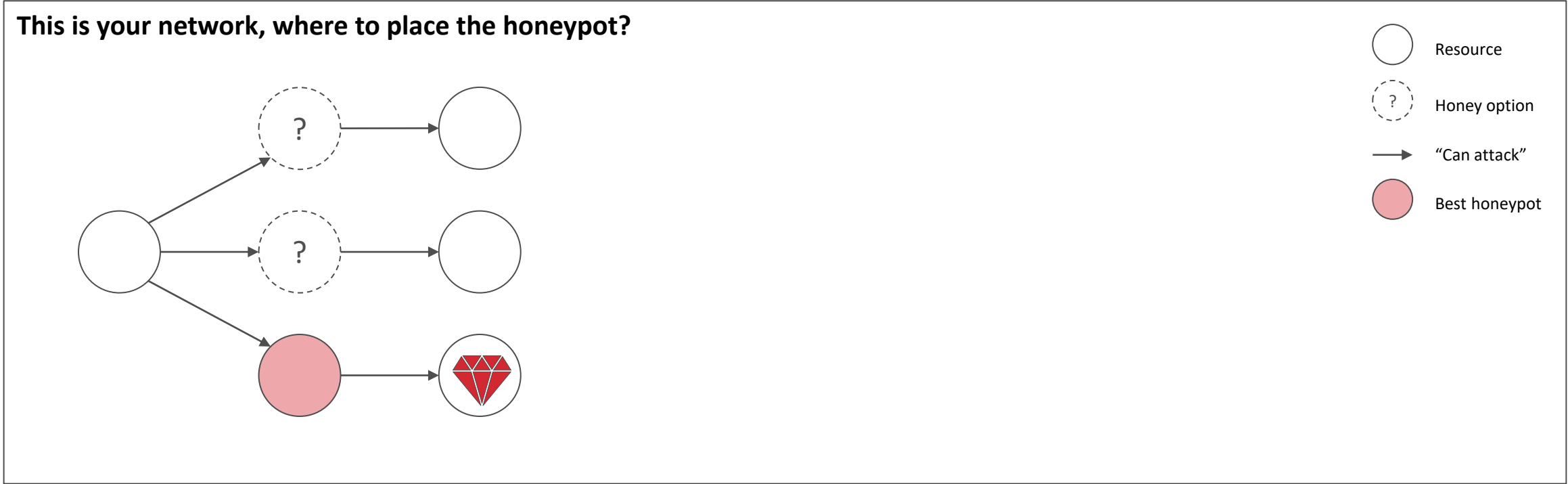


- Resource
- ? Honey option
- "Can attack"
- Best honeypot




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

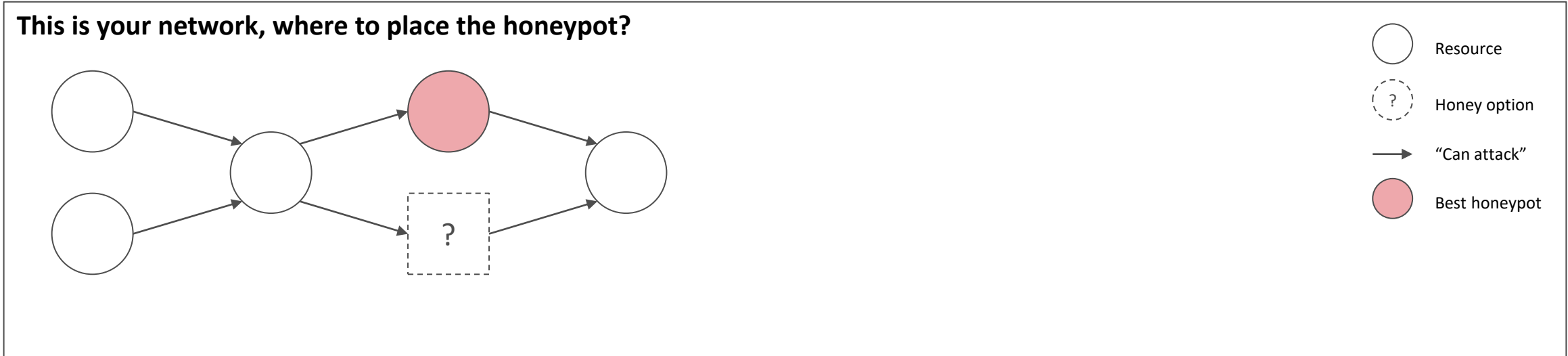
# Effective honeypots are easily encountered and suggest a worthwhile attack path

Design Goal	Description	Example how to mess it up
<b>Discoverability</b> 	<ul style="list-style-type: none"><li>▪ <b>Easy for attackers to find</b></li><li>▪ Ensuring it serves its purpose as a trap</li></ul>	<ul style="list-style-type: none"><li>▪ Fake credential injected to memory</li><li>▪ Deployed to laptops only</li></ul>
<b>Appeal to Attackers</b> 	<ul style="list-style-type: none"><li>▪ <b>Appears valuable</b> to attackers</li><li>▪ Illusion of advancing access or privileges</li></ul>	<ul style="list-style-type: none"><li>▪ Honey accounts seem like basic users</li><li>▪ But basic users can be obtained by external password spraying → not worth the risk</li></ul>








# Effective honeypots are easily encountered and suggest a worthwhile attack path

Design Goal	Description	Example how to mess it up
<b>Discoverability</b> 	<ul style="list-style-type: none"><li>▪ <b>Easy for attackers to find</b></li><li>▪ Ensuring it serves its purpose as a trap</li></ul>	<ul style="list-style-type: none"><li>▪ Fake credential injected to memory</li><li>▪ Deployed to laptops only</li></ul>
<b>Appeal to Attackers</b> 	<ul style="list-style-type: none"><li>▪ <b>Appears valuable</b> to attackers</li><li>▪ Illusion of advancing access or privileges</li></ul>	<ul style="list-style-type: none"><li>▪ Honey accounts seem like basic users</li><li>▪ But basic users can be obtained by external password spraying → not worth the risk</li></ul>
<b>Authenticity</b> 	<ul style="list-style-type: none"><li>▪ <b>Blends into the environment</b> realistically</li><li>▪ Hard to identify as a honeypot</li></ul>	<ul style="list-style-type: none"><li>▪ Last logon long ago for “normal” user</li><li>▪ More cached credentials on machine than <i>CachedLogonsCount</i> would allow</li></ul>

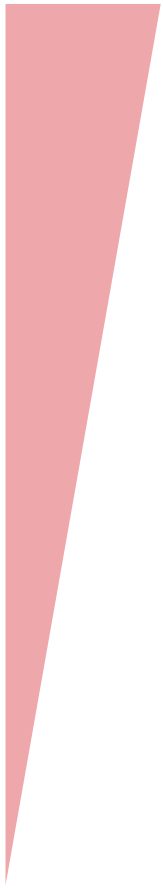


# Effective honeypots are easily encountered and suggest a worthwhile attack path

Design Goal	Description	Example how to mess it up
<b>Discoverability</b> 	<ul style="list-style-type: none"><li>▪ <b>Easy for attackers to find</b></li><li>▪ Ensuring it serves its purpose as a trap</li></ul>	<ul style="list-style-type: none"><li>▪ Fake credential injected to memory</li><li>▪ Deployed to laptops only</li></ul>
<b>Appeal to Attackers</b> 	<ul style="list-style-type: none"><li>▪ <b>Appears valuable</b> to attackers</li><li>▪ Illusion of advancing access or privileges</li></ul>	<ul style="list-style-type: none"><li>▪ Honey accounts seem like basic users</li><li>▪ But basic users can be obtained by external password spraying → not worth the risk</li></ul>
<b>Authenticity</b> 	<ul style="list-style-type: none"><li>▪ <b>Blends into the environment</b> realistically</li><li>▪ Hard to identify as a honeypot</li></ul>	<ul style="list-style-type: none"><li>▪ Last logon long ago for “normal” user</li><li>▪ More cached credentials on machine than <i>CachedLogonsCount</i> would allow</li></ul>
<b>Safety</b> 	<ul style="list-style-type: none"><li>▪ Honeypot is <b>not exploitable</b></li><li>▪ Limit risk of things going wrong</li></ul>	<ul style="list-style-type: none"><li>▪ High privilege account with password in description but logon hours deny</li><li>▪ Admin changes logon hours for testing</li></ul>
<b>Alert precision</b> 	<ul style="list-style-type: none"><li>▪ Strongly <b>limit false positive alerts</b></li><li>▪ Logs should enable investigation</li></ul>	<ul style="list-style-type: none"><li>▪ Normal users can find honey files</li><li>▪ Source IP who accessed honey account is hidden by gateway</li></ul>

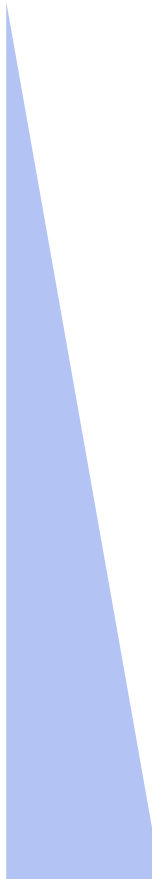
Start small and test, then add more over time!  
But where to start?

# Different types of deception vary in effectiveness

Type	Description	Alert Mechanism	Examples	Pros / Cons	Usage
Honey network services	<ul style="list-style-type: none"><li>▪ <b>Imitate network service</b></li><li>▪ Containers, VMs or separate hardware</li></ul>	<ul style="list-style-type: none"><li>▪ <b>Alert on access</b></li><li>▪ Or based on attack patterns (high-interaction)</li></ul>	<ul style="list-style-type: none"><li>▪ <b>Web or SSH login</b> that accepts all credentials</li><li>▪ SMB file share</li><li>▪ Many options on GitHub</li></ul>	<ul style="list-style-type: none"><li>+ <b>Insights</b> on attacker behavior</li><li>- <b>Discoverability</b> (effort for good coverage)</li></ul>	
Honeytokens Files	<ul style="list-style-type: none"><li>▪ <b>Files that trigger alerts when opened</b></li></ul>	<ul style="list-style-type: none"><li>▪ <b>DNS request</b></li><li>▪ File open event in log</li></ul>	<ul style="list-style-type: none"><li>▪ <b>PDF or office documents</b></li><li>▪ World-readable ssh keys</li></ul>	<ul style="list-style-type: none"><li>+ <b>Flexible</b> location (O365, file system...)</li><li>- <b>FPs and traceability</b></li></ul>	
Auth secrets	<ul style="list-style-type: none"><li>▪ Credentials or API tokens</li></ul>	<ul style="list-style-type: none"><li>▪ Alert upon attempted authentication</li></ul>	<ul style="list-style-type: none"><li>▪ AWS token in Github repo</li><li>▪ Hardcoded pw in mobile app</li></ul>	<ul style="list-style-type: none"><li>+ Flexible, <b>less FPs</b></li><li>- <b>Traceability</b> for cloud</li></ul>	
Active Directory honeypots	<ul style="list-style-type: none"><li>▪ AD object suggesting easy attack path</li></ul>	<ul style="list-style-type: none"><li>▪ Sysmon (or EDR)</li><li>▪ Monitor specific Event IDs in SIEM</li></ul>	<ul style="list-style-type: none"><li>▪ AD user credentials<sup>[1]</sup></li><li>▪ Kerberoastable user</li><li>▪ Group with fake RDP privileges</li></ul>	<ul style="list-style-type: none"><li>+ <b>Fit most attackers'</b> toolset</li><li>+ Easy and effective</li><li>- <b>Require AD admin</b></li></ul>	

[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	<ul style="list-style-type: none"><li>▪ Useful as internet-connected honeypots for threat-intelligence</li><li>▪ <b>Hard to discover for attackers</b> in large networks, high roll-out effort for good coverage</li><li>▪ Often don't look very attractive</li></ul> <p>→ <b>Do this last or don't do it</b></p>	
3 Honeytokens Files	<ul style="list-style-type: none"><li>▪ <b>Can flexibly cover many environments:</b> cloud, file shares, code repositories, local filesystems, ...</li><li>▪ Need to ensure a detection can be traced back to attacker</li><li>▪ How much sense it makes depends a bit on your environment</li></ul> <p>→ <b>Effective to set up with reasonable effort and cost using a SAAS product</b></p>	
2 Auth secrets		
1 Active Directory honeypots	<ul style="list-style-type: none"><li>▪ <b>Most attack chains touch Active Directory</b> at some point</li><li>▪ Attacker tooling – especially of ransomware gangs – is optimized for it</li><li>▪ Requires Sysmon+SIEM, EDR or a solution like MDI to alert on AD events</li></ul> <p>→ <b>Perfect location for deception – let's see what we can do here!</b></p>	
Pro-tip	Red team reports can provide inspiration for what honeypots to build	

- 
- 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 : WDAGUtilityAccount
description : A user account managed by Windows Defender
```

```
check_ldap_conn.ps1 - Notepad
File Edit Format View Help
$password = ConvertTo-SecureString "Super#S3cure"
$creds = New-Object System.Management.Automation.PSCredential($password)
$conn = "LDAP://DC01.mycorp.int"

try {
    # Create the DirectoryEntry object
    $DirectoryEntry = New-Object System.DirectoryServices.DirectoryEntry($conn)
    $password = ConvertTo-SecureString $password
    # Create a DirectorySearcher object to perform the search
    $Searcher = New-Object System.DirectoryServices.DirectorySearcher($DirectoryEntry)
    $Searcher.Filter = "(objectClass=*)"
    $Searcher.SizeLimit = 1

    # Perform the search operation
```

## Design Goal

### Discoverability



### Appeal to attackers



### Authenticity



### Safety



### Alert precision



## Guidance

#### ▪ Get creative where to hide fake credentials

- Description field in AD object, PowerShell script on SYSVOL, code repos, file of rolled out to endpoints

- Should be a privileged account (or at least seem like it)
- Could be from group membership, permissions visible in LDAP, or naming scheme

#### a. Active account with very rare failed logons

#### b. Dedicated honey account by recycling old account for RID, lastlogon, BadPasswordTime, ...

#### ▪ Password hint should be wrong

- We advise against real creds with logon hours deny

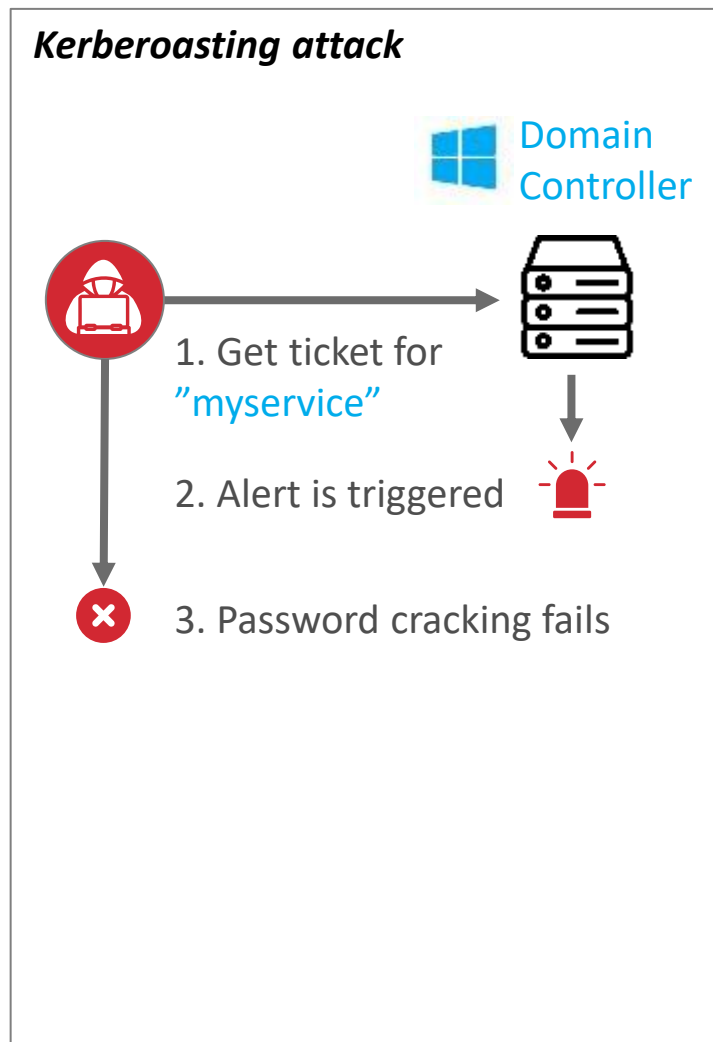
- Windows event ID 4625 (failed logon)

- Windows event ID 4768 (TGT request)

- SIEM can find suitable accounts with few failed logins



## 2 Kerberoasting honeypots appeal to a common attack vector



### Design Goal

#### Discoverability



#### Appeal to attackers



#### Authenticity



#### Safety



#### Alert precision



### Guidance

- Attackers query LDAP for users with SPN

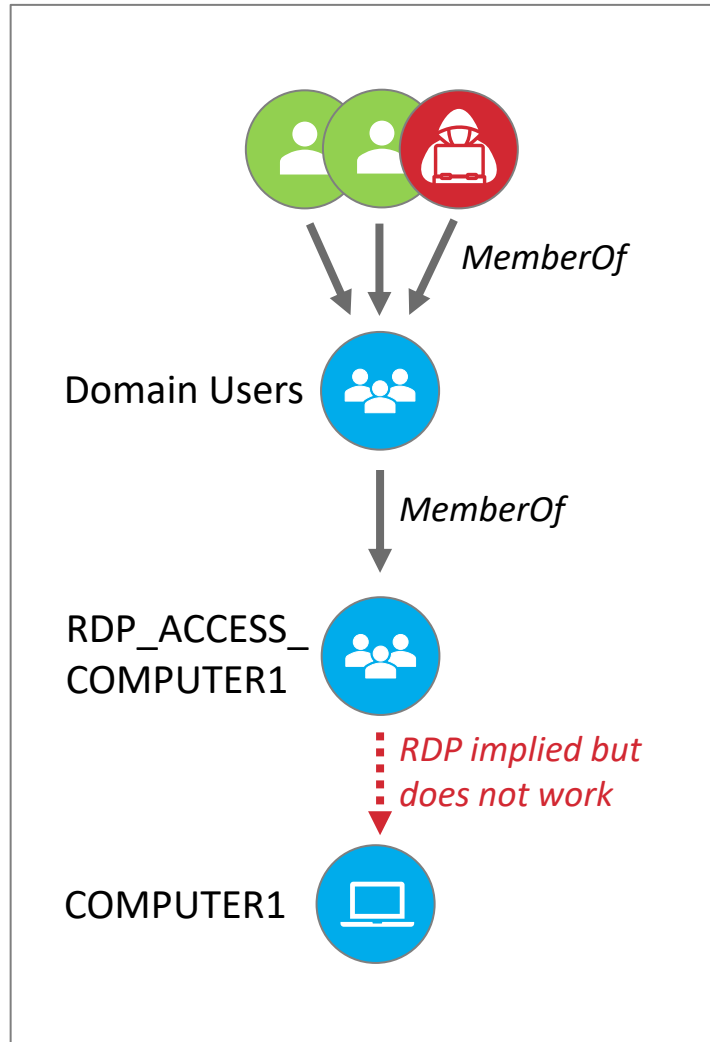
- Accounts with **older passwords** are more attractive
- **Human accounts** are attractive in general

- Set SPN on normal user** as if it was a forgotten test
  - Dedicated honey account** by recycling
- Consider how common RC4 is in your environment

- **Ensure account has strong, auto-generated password**
- Account owner needs to be aware

- Event ID 4769 (service ticket request)

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



#### Design Goal

##### Discoverability



##### Appeal to attackers



##### Authenticity



##### Safety



##### Alert precision



#### Guidance

- Attackers usually review group membership
- **All users are member of “Domain Users” in LDAP**
- **Group name suggests RDP access privileges**
- Could also do the same with local admin
- Ideally, machine seems important
- **Machine OS >= Windows Server 2016** (no easy RDP privilege enumeration anymore)
- Pick description, OUs, etc. to make it fit in
- **No actual RDP access**
- **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...



- 
- Introduction
  - Deception strategy
  - Must-have AD honeypots

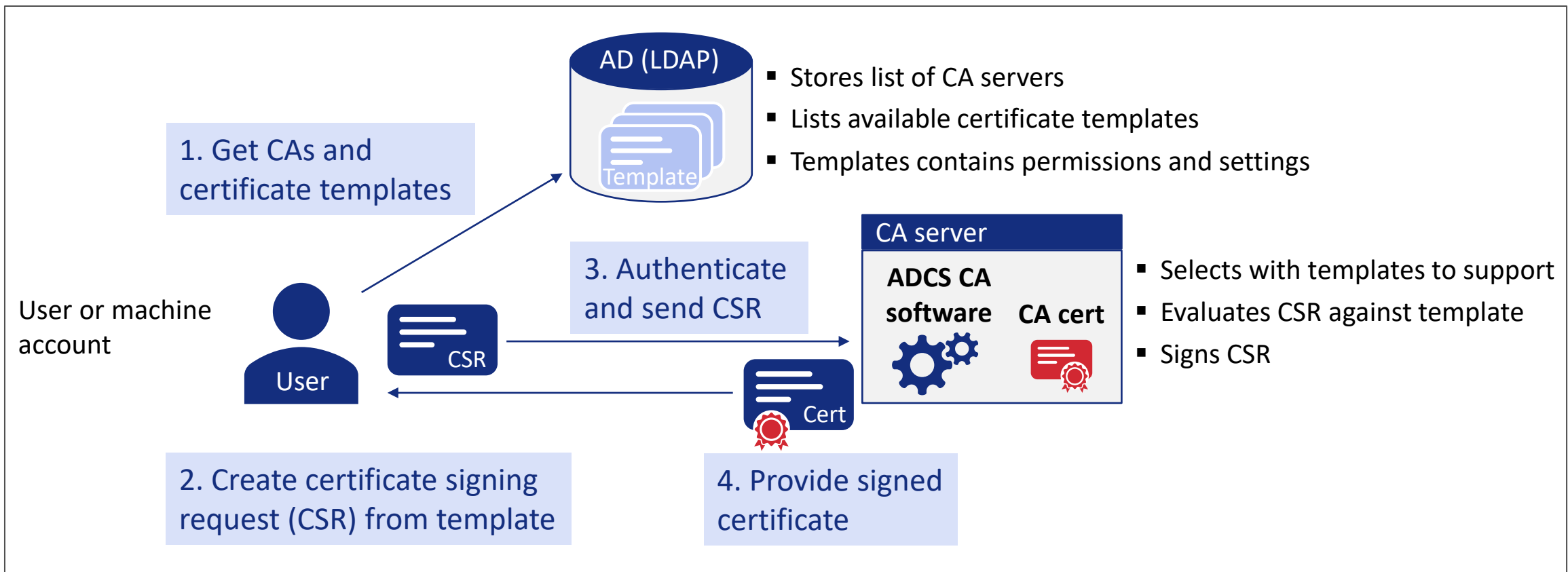
 **Tool release: ADCS deception**

---

# 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



# ADCS is complex to configure, and mistakes have high impact

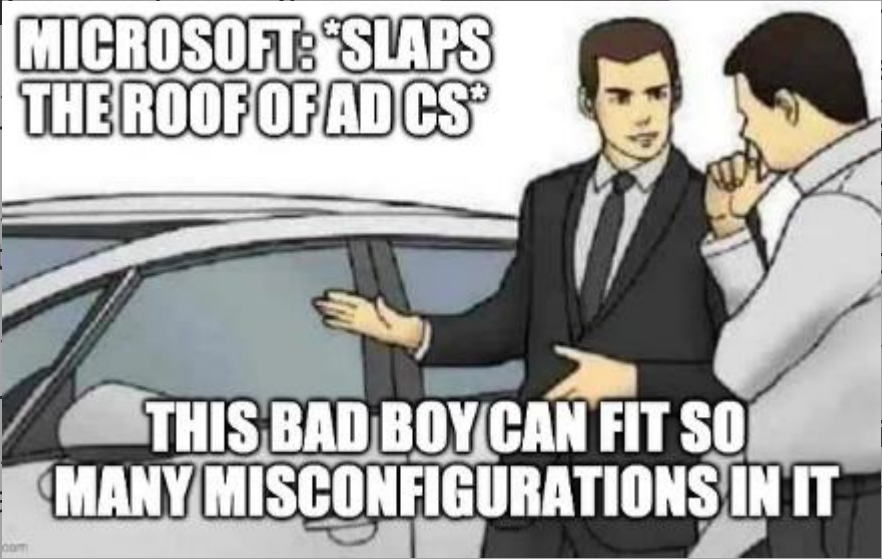
Common misconfigurations in ADCS	
ESC-1	Certificate template allows enrolling user to specify who the certificate is valid for → “Domain admin”
ESC-2	User certificate can be used to enroll new certificates → Create one for Domain Admin
ESC-3	
ESC-4	User has write permission to certificate template → 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	No protection against relay attacks → Compromise account when coercing authentication
ESC-11	
ESC-9	Obtain certificate as any Domain user by modifying the UPN of a controlled user
ESC-10	
ESC-12	Chain of conditions and quite complicated, you probably did not read this far → ignored on this slide
ESC-13	

 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 enrollment for any user →	Manager approval on certificate templates → it
ESC-2	User certificate can be used to enroll →	on against relay attacks → Compromise
ESC-3	→ Create one for Domain Admin	when coercing authentication
ESC-4	User has write permission to certificate → introduce ESC1	certificate as any Domain user by modifying a controlled user
ESC-5	Compromise one of the ADCS objects (computer object, container, ...)	conditions and quite complicated, you did not read this far → ignored on this slide
ESC-6	CA-level setting that basically enables	Previously exploited in client engagements



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

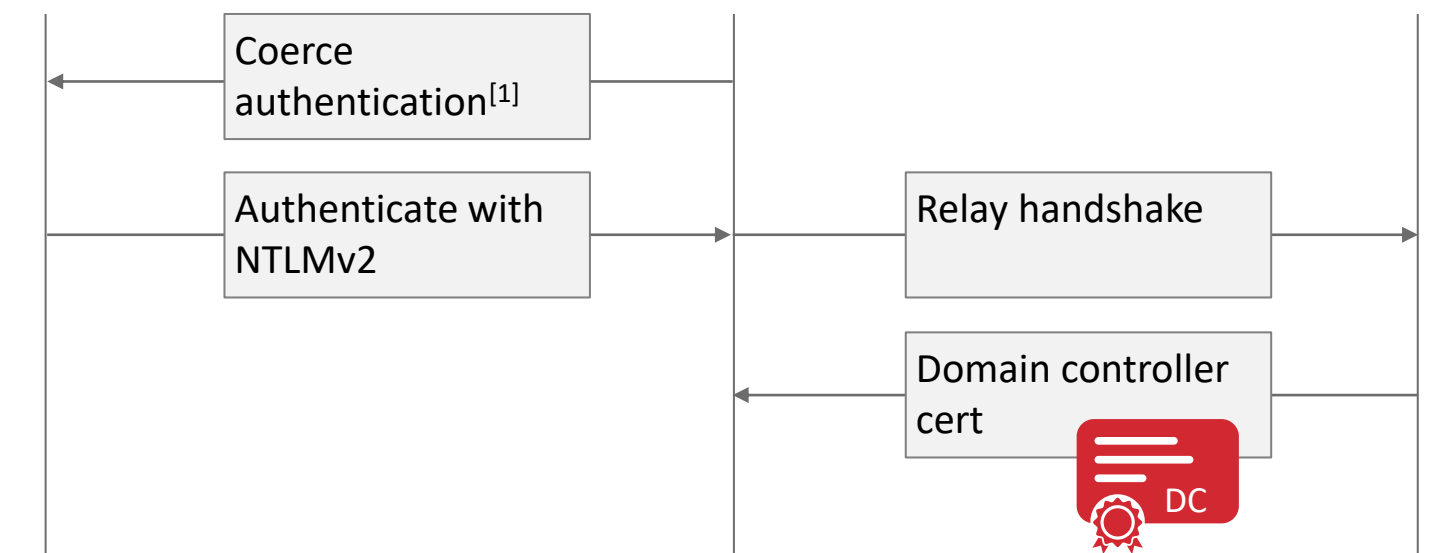
# An ESC8 honeypot is feasible but was not effective enough for us

## 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



## 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**



# 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	<ul style="list-style-type: none"><li>▪ Feasible and safe option</li><li>▪ Exploitation needs auth coercion (tricky)</li></ul> → Harder for hackers to step into trap
ESC3 with enrolment restrictions	<ul style="list-style-type: none"><li>▪ Place restrictions on second required cert</li><li>▪ Attacker still obtains enrolment certificate</li></ul> → Too risky
Auto-revocation	<ul style="list-style-type: none"><li>▪ Dangerous time window with valid cert</li><li>▪ An OCSP setup could work</li></ul> → We don't understand revocation enough

## The TameMyCerts policy module saved the day

ADCS policy modules	<ul style="list-style-type: none"><li>▪ Receives and evaluate certificate requests</li><li>▪ Can issue or deny</li><li>▪ Implemented as a DLL on the CA</li></ul>
---------------------	---

TameMyCerts <sup>[1]</sup>	<ul style="list-style-type: none"><li>▪ Policy module developed and maintained by Uwe Gradenegger<sup>[2]</sup></li><li>▪ Developed for fine grained and automated certificate issuance checks</li><li>▪ Rules for evaluation are specified as XML</li></ul>
----------------------------	--

README Apache-2.0 license

### 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

```
16      <SubjectAlternativeName>
17          <SubjectRule>
18              <Field>sAMAccountName</Field>
19              <Mandatory>>false</Mandatory>
20              <Patterns>
21                  <Pattern>
22                      <Expression>^.*$</Expression>
23                      <Action>Deny</Action>
24                  </Pattern>
25              </Patterns>
26          </SubjectRule>
27      </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 <sup>[1]</sup>	4886 – Certificate enrollment requested	<b>Medium</b> 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	<b>Critical</b> Attempted exploitation
	Future plan – adapt events to honeypot use	

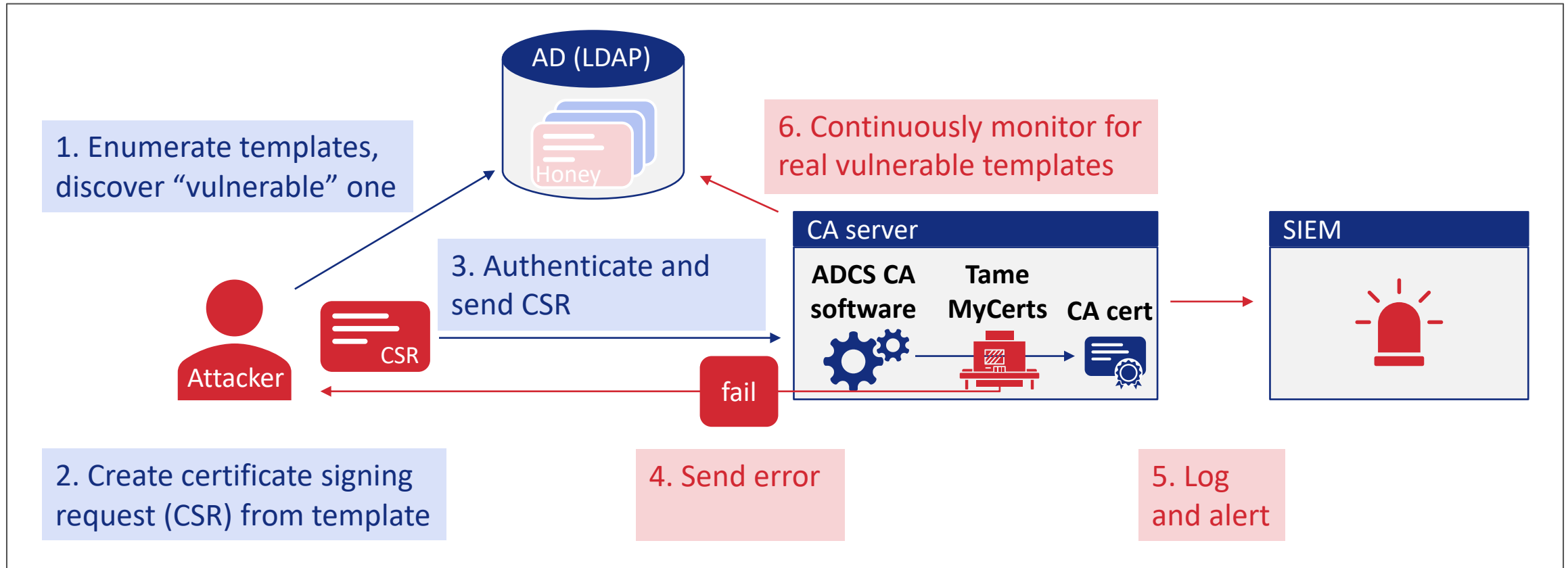
- SIGMA rules to be SIEM-agnostic
- Improvements planned or the future when supporting various honey templates

[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



# 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

```
7 # parameters to customize your honeypot
8 host_name: honeypotCA
9 host_ip: 192.168.56.238
10 ca_name: honeypot-CA4
11 path: DC=mydomain,DC=local
12 computer_name: honeypotCA
13 computer_fqdn: honeypotCA.mydomain.local
14 computer_path: OU=Computers,DC=mydomain,DC=local
15 template_name: ESC1Template
16 template_display_name: ESC1Template4
17 vuln_detector_account_name: ServiceAccount
```

3

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

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

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

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





Stealing  
credentials  
from LSASS



Asking a CA  
for a certificate



Stepping into an  
ADCS honeypot



# Offensive security tooling recognizes Certiception as a vulnerable ESC1 template

Discovery

Certify

```
[!] 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 : TESTLAB\Domain Users S-1-  
  Enrollment Rights : NT AUTHORITY\SYSTEM S-1-  
  All Extended Rights : TESTLAB\Domain Admins S-1-  
  TESTLAB\Domain Admins S-1-
```

[!] Vulnerable Certificates Templates

Certipy

```
Permissions  
Enrollment Permissions : TESTLAB.CORP\Domain Users  
Enrollment Rights : TESTLAB.CORP\Domain Users  
Object Control Permissions : TESTLAB.CORP\Local System  
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

BloodHound

```
graph TD  
  A((COMPROMISED@TESTLAB.CORP)) -- MemberOf --> B((DOMAIN USERS@TESTLAB.CORP))  
  B -- ADCSESC1 --> C((TESTLAB.CORP))
```

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

```
$certipy req -u compromised@testlab.corp -dc-ip 192.168.56.10 -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: code: 0x800b0114 - CERT_E_INVALID_NAME - The certificate has an invalid name. The name is not included in the permitted 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 key? (y/N)  
[-] 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](mailto:infosec.exchange)/@balthasar & @BalthasarMartin>

Niklas van Dornick <[@n1v4d0](mailto:@n1v4d0)>