Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS
San Diego

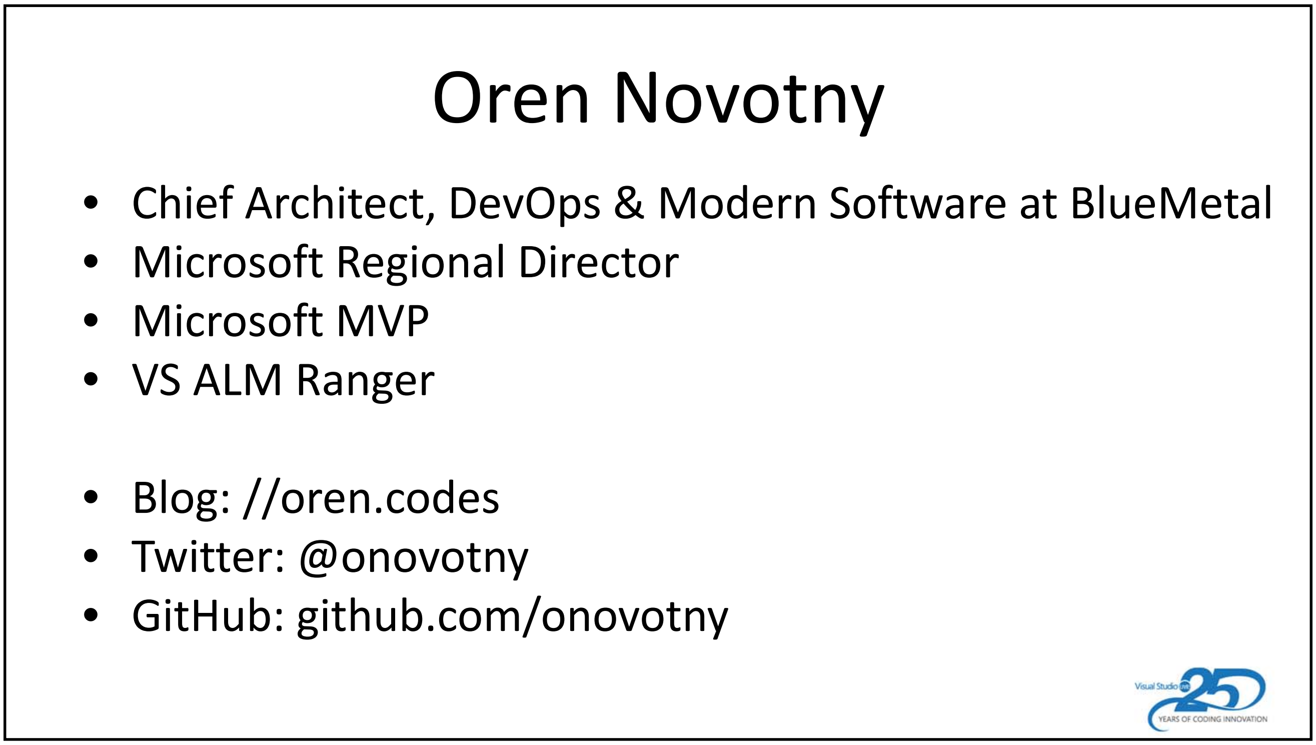
Signing your code the easy way

Oren Novotny

Chief Architect, DevOps & Modern Software
BlueMetal

Level: Intermediate.

Code Again for the First Time!
Visual Studio LIVE! 25 YEARS OF CODING INNOVATION

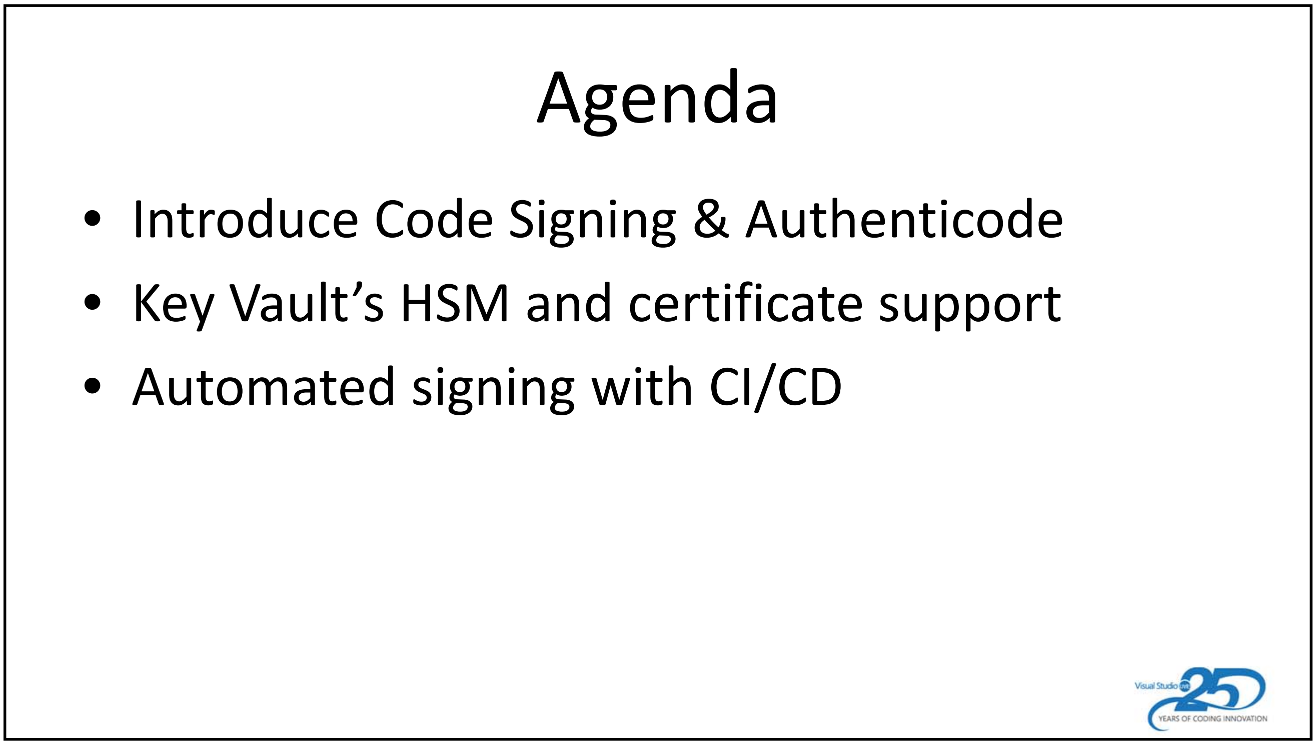


Oren Novotny

- Chief Architect, DevOps & Modern Software at BlueMetal
- Microsoft Regional Director
- Microsoft MVP
- VS ALM Ranger

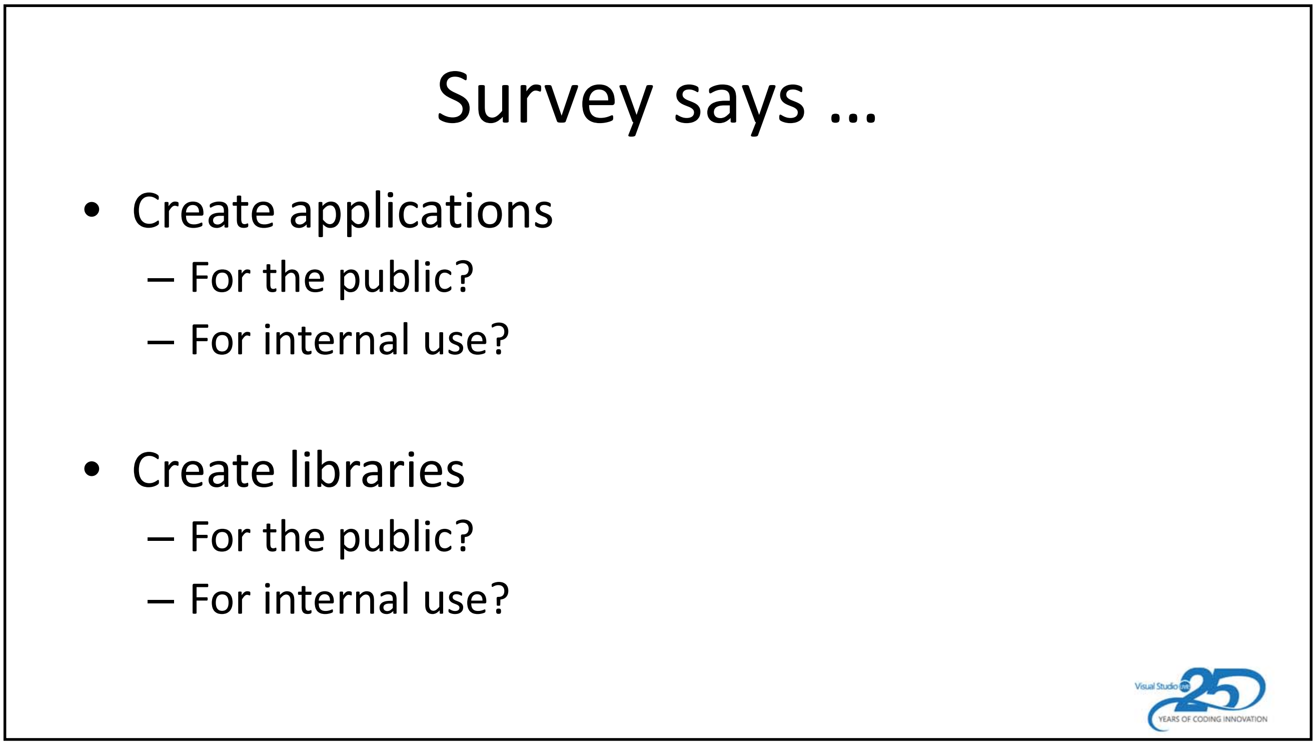
- Blog: //oren.codes
- Twitter: @onovotny
- GitHub: github.com/onovotny

Visual Studio LIVE! 25 YEARS OF CODING INNOVATION



Agenda

- Introduce Code Signing & Authenticode
- Key Vault’s HSM and certificate support
- Automated signing with CI/CD

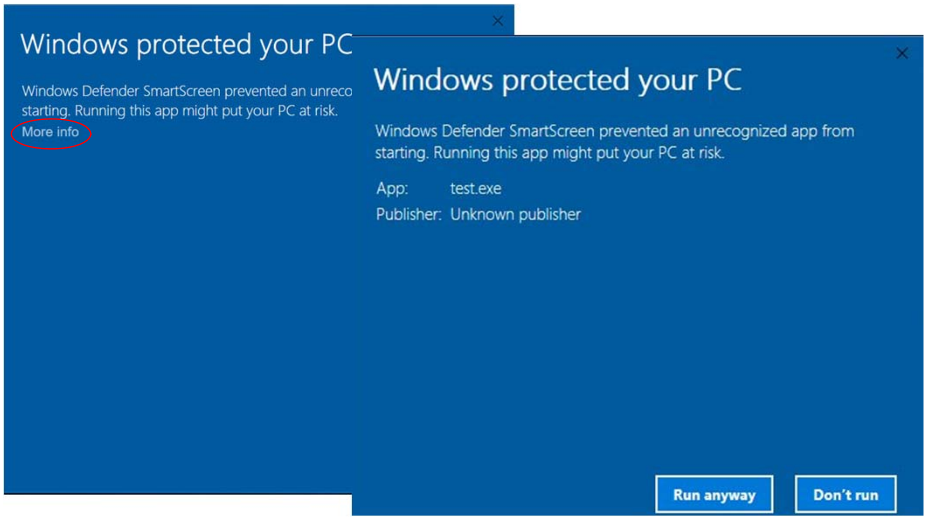


Survey says …

- Create applications
  - For the public?
  - For internal use?

- Create libraries
  - For the public?
  - For internal use?

Why sign your code?



Windows protected your PC

Windows Defender SmartScreen prevented an unreco
starting. Running this app might put your PC at risk.

More info

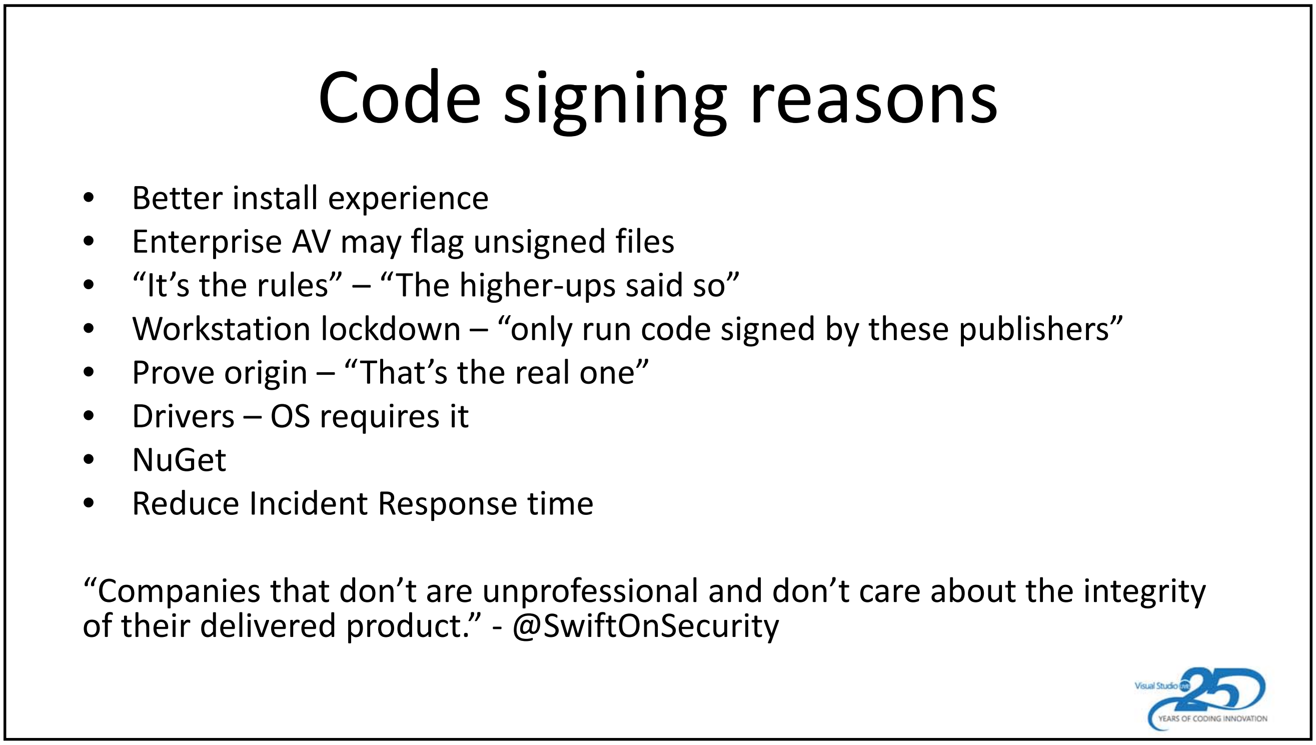
Windows protected your PC

Windows Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.

App: test.exe
Publisher: Unknown publisher

Why sign your code?

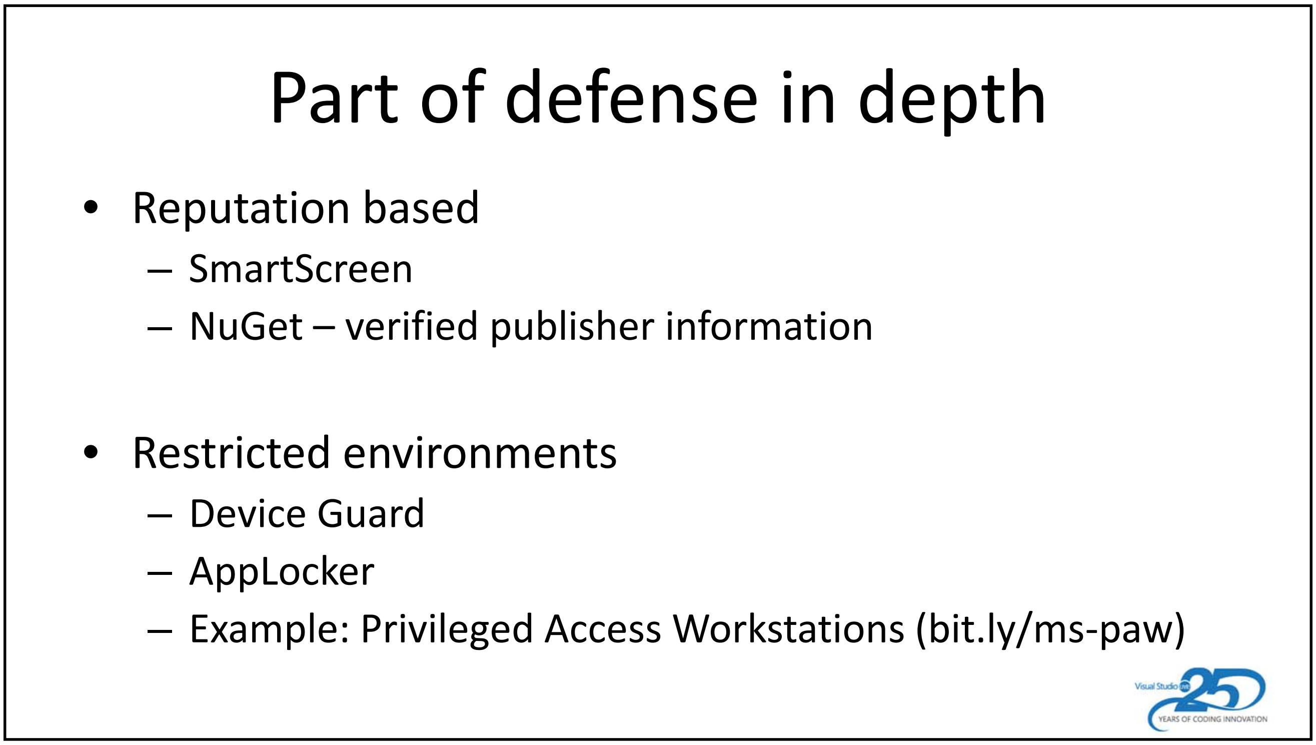
- Prove integrity
- Develop reputation
- Reputation is earned, malware can be signed
- Non-repudiation



Code signing reasons

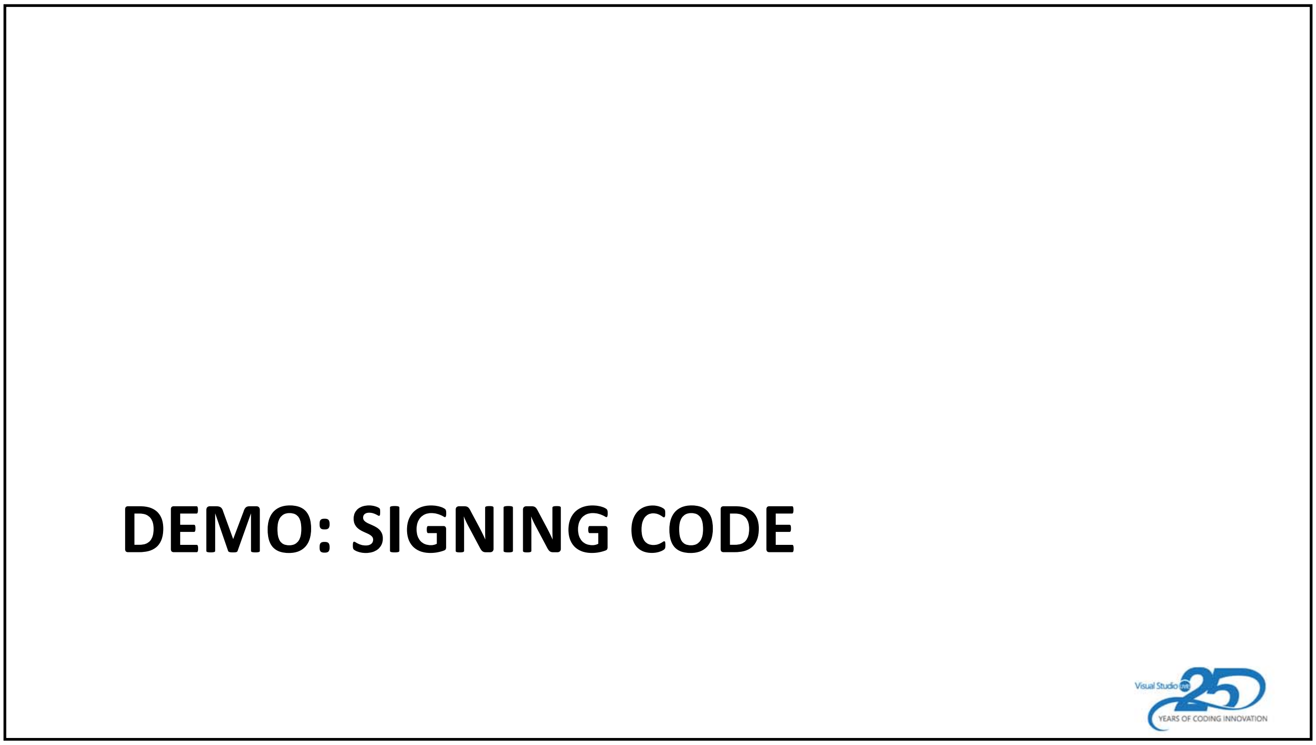
- Better install experience
- Enterprise AV may flag unsigned files
- “It’s the rules” – “The higher-ups said so”
- Workstation lockdown – “only run code signed by these publishers”
- Prove origin – “That’s the real one”
- Drivers – OS requires it
- NuGet
- Reduce Incident Response time

“Companies that don’t are unprofessional and don’t care about the integrity of their delivered product.” - @SwiftOnSecurity

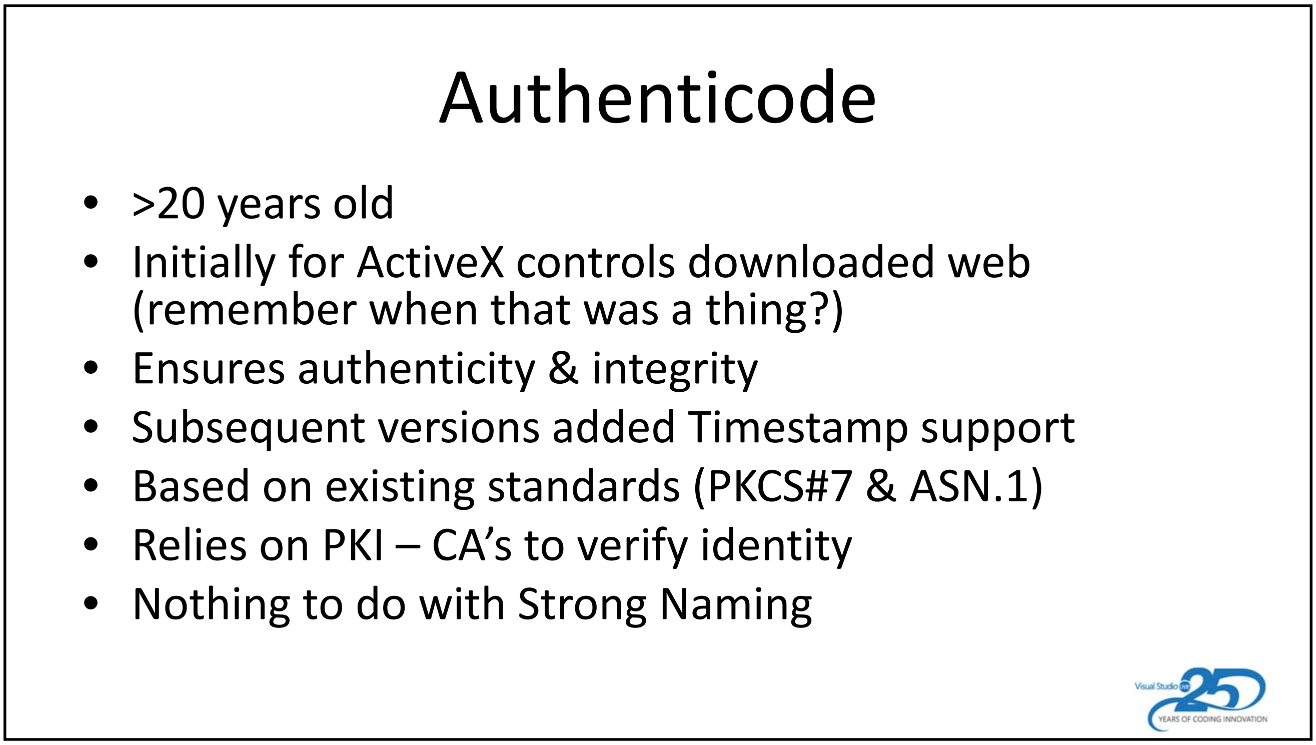


Part of defense in depth

- Reputation based
  - SmartScreen
  - NuGet – verified publisher information
- Restricted environments
  - Device Guard
  - AppLocker
  - Example: Privileged Access Workstations (bit.ly/ms-paw)



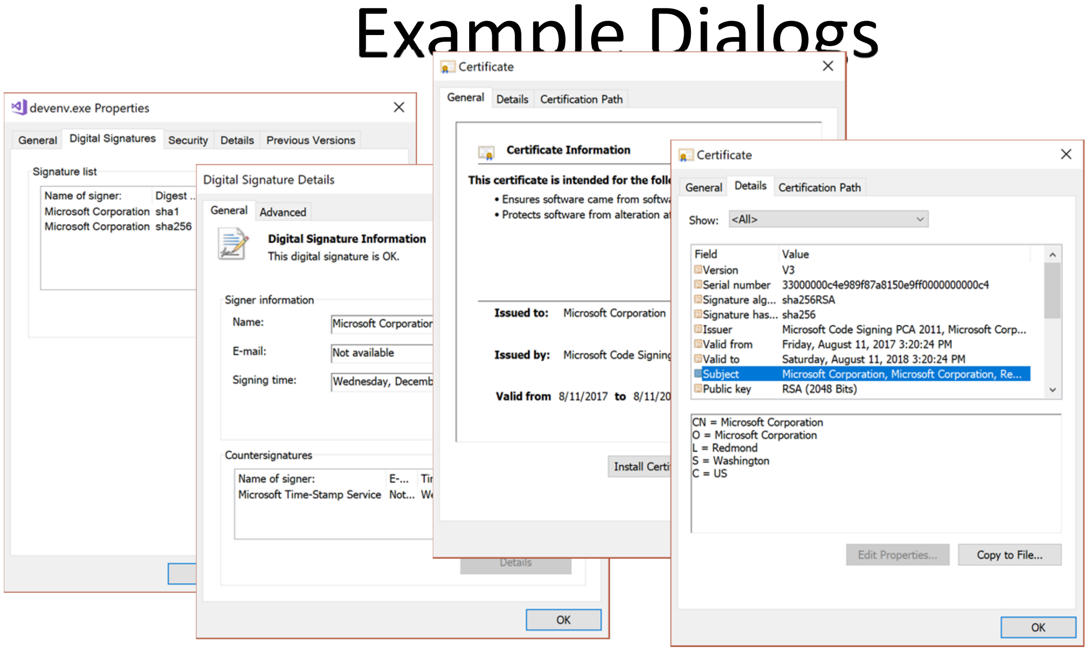
DEMO: SIGNING CODE



Authenticode

- >20 years old
- Initially for ActiveX controls downloaded web (remember when that was a thing?)
- Ensures authenticity & integrity
- Subsequent versions added Timestamp support
- Based on existing standards (PKCS#7 & ASN.1)
- Relies on PKI – CA’s to verify identity
- Nothing to do with Strong Naming

Example Dialogs



Signature list

Name of signer:	Digest ...
Microsoft Corporation	sha1
Microsoft Corporation	sha256

Digital Signature Information
This digital signature is OK.

Signer information
Name: Microsoft Corporation
E-mail: Not available
Signing time: Wednesday, Decemb

Countersignatures

Name of signer:	E-...	Ti
Microsoft Time-Stamp Service	Not...	We

Certificate Information

This certificate is intended for the foll
- Ensures software came from softwa
- Protects software from alteration at

Issued to: Microsoft Corporation

Issued by: Microsoft Code Signing

Valid from 8/11/2017 **to** 8/11/20

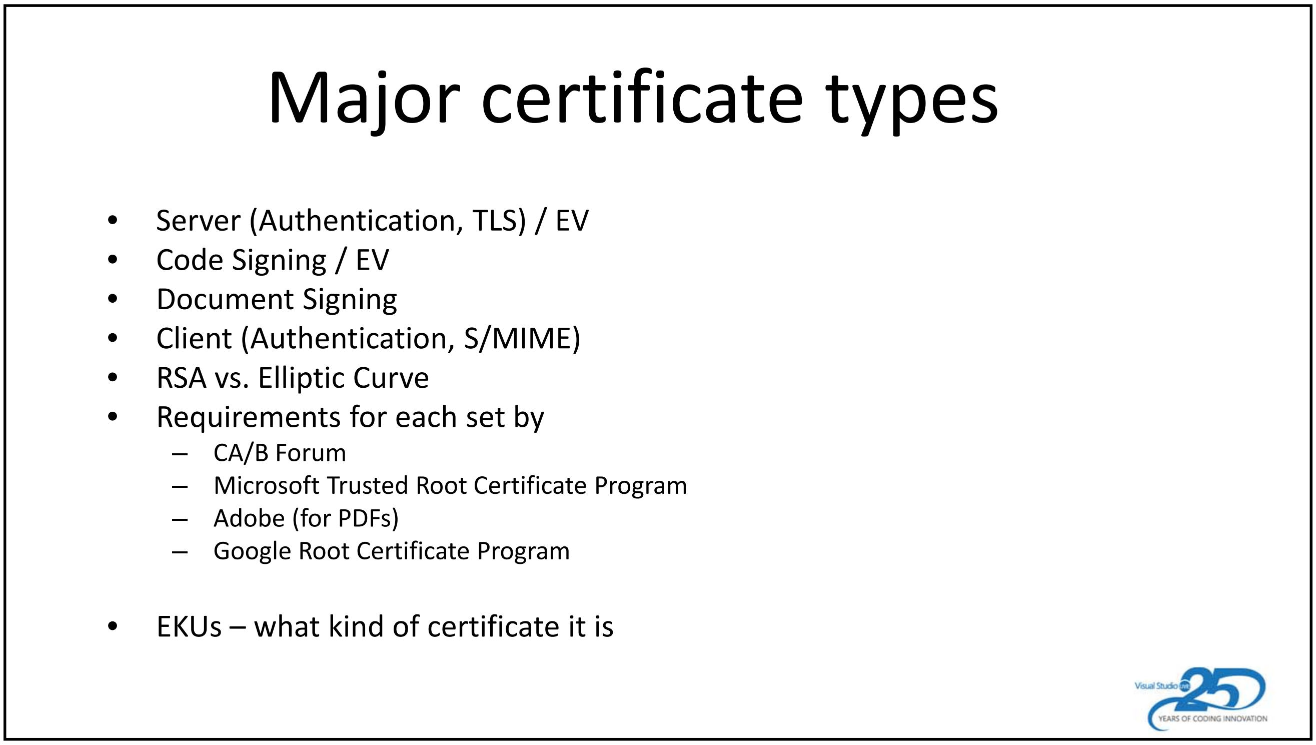
Show: <All>

Field	Value
Version	V3
Serial number	33000000c4e989f87a8150e9ff0000000000c4
Signature alg...	sha256RSA
Signature has...	sha256
Issuer	Microsoft Code Signing PCA 2011, Microsoft Corp...
Valid from	Friday, August 11, 2017 3:20:24 PM
Valid to	Saturday, August 11, 2018 3:20:24 PM
Subject	Microsoft Corporation, Microsoft Corporation, Re...
Public key	RSA (2048 Bits)

CN = Microsoft Corporation
O = Microsoft Corporation
L = Redmond
S = Washington
C = US

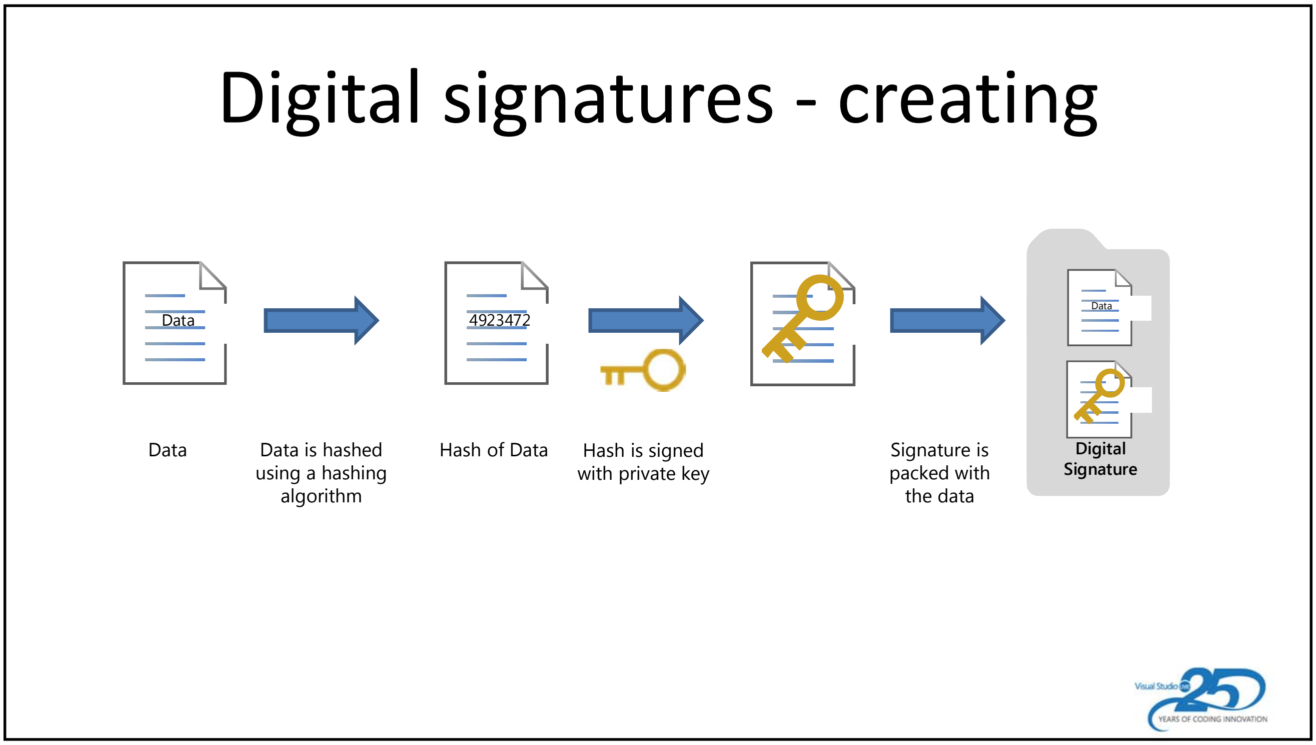
Digital Certificates

- X509 certificate and private key
- Includes identity information in public portion, signed by issuer
  - Chain of trust
- Obtained from public CA
  - DigiCert
  - GlobalSign
  - GoDaddy
- Corporations often have internal CA’s
- Self-signed (for testing only)



Major certificate types

- Server (Authentication, TLS) / EV
- Code Signing / EV
- Document Signing
- Client (Authentication, S/MIME)
- RSA vs. Elliptic Curve
- Requirements for each set by
  - CA/B Forum
  - Microsoft Trusted Root Certificate Program
  - Adobe (for PDFs)
  - Google Root Certificate Program
- EKUs – what kind of certificate it is



Digital signatures - creating

Data

Data is hashed using a hashing algorithm

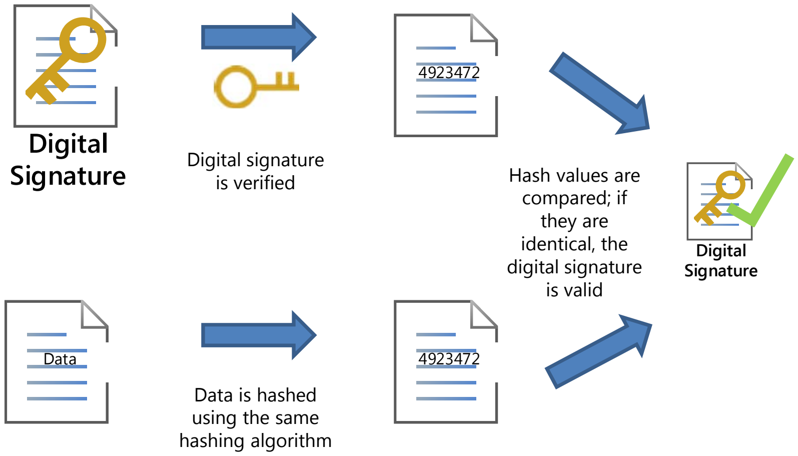
Hash of Data

Hash is signed with private key

Signature is packed with the data

Digital Signature

Digital signatures - verifying



Digital Signature

Digital signature is verified

4923472

Hash values are compared; if they are identical, the digital signature is valid

Digital Signature

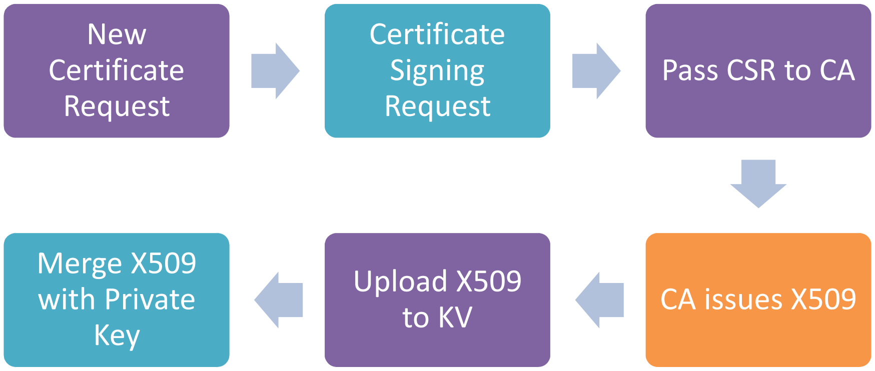
Data

Data is hashed using the same hashing algorithm

4923472

HOW IT WORKS

Getting a certificate



New Certificate Request

Certificate Signing Request

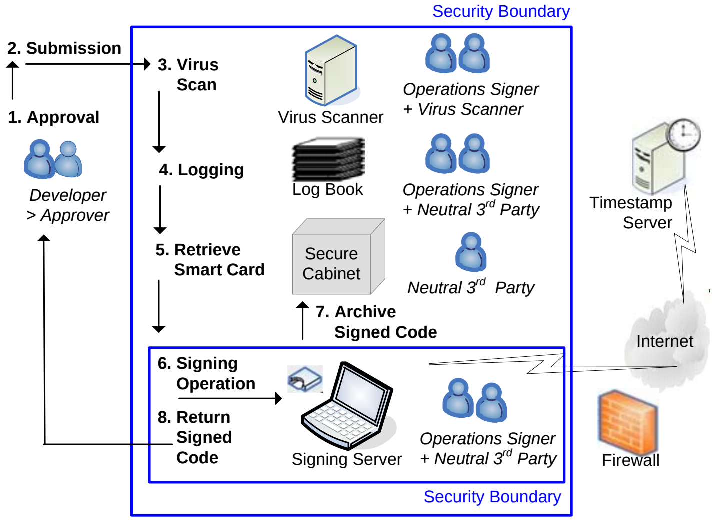
Pass CSR to CA

CA issues X509

Upload X509 to KV

Merge X509 with Private Key

Offline code signing



Security Boundary

1. Approval

Developer > Approver

2. Submission

3. Virus Scan

Virus Scanner

Operations Signer + Virus Scanner

4. Logging

Log Book

Operations Signer + Neutral 3rd Party

5. Retrieve Smart Card

Secure Cabinet

Neutral 3rd Party

7. Archive Signed Code

6. Signing Operation

8. Return Signed Code

Signing Server

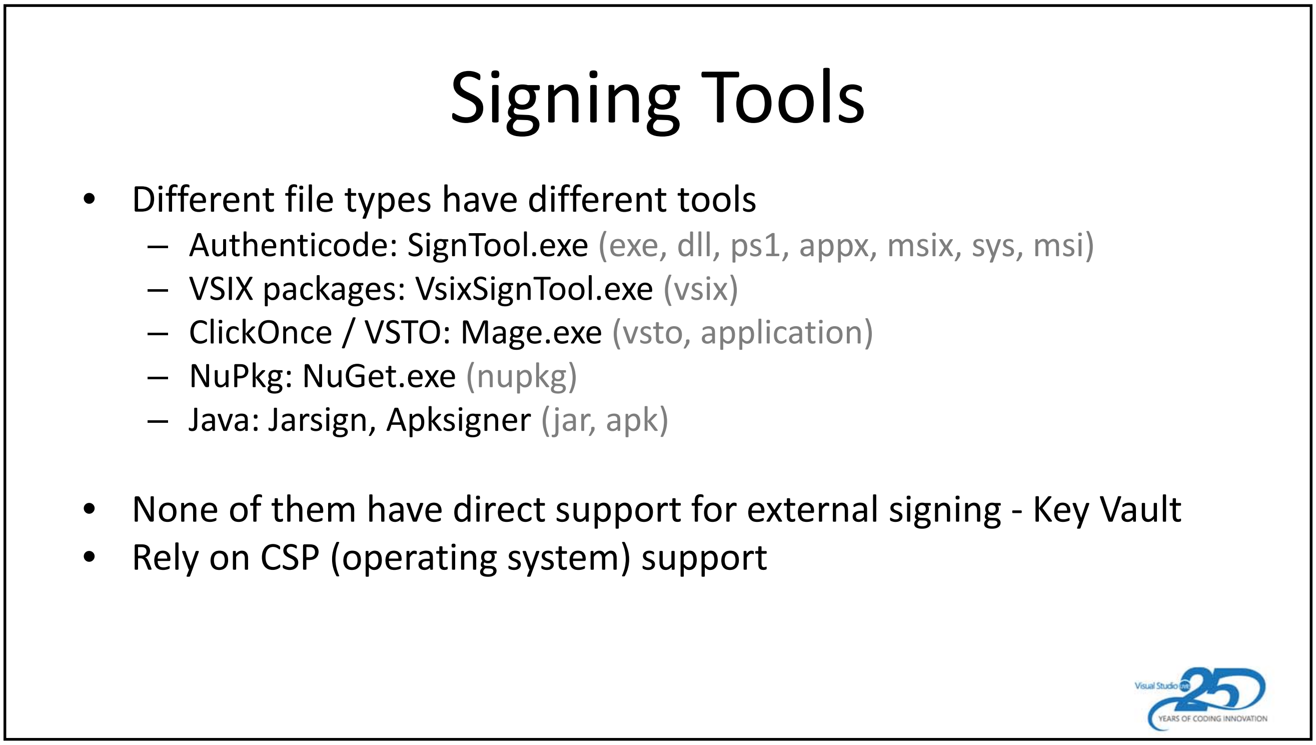
Operations Signer + Neutral 3rd Party

Timestamp Server

Internet

Firewall

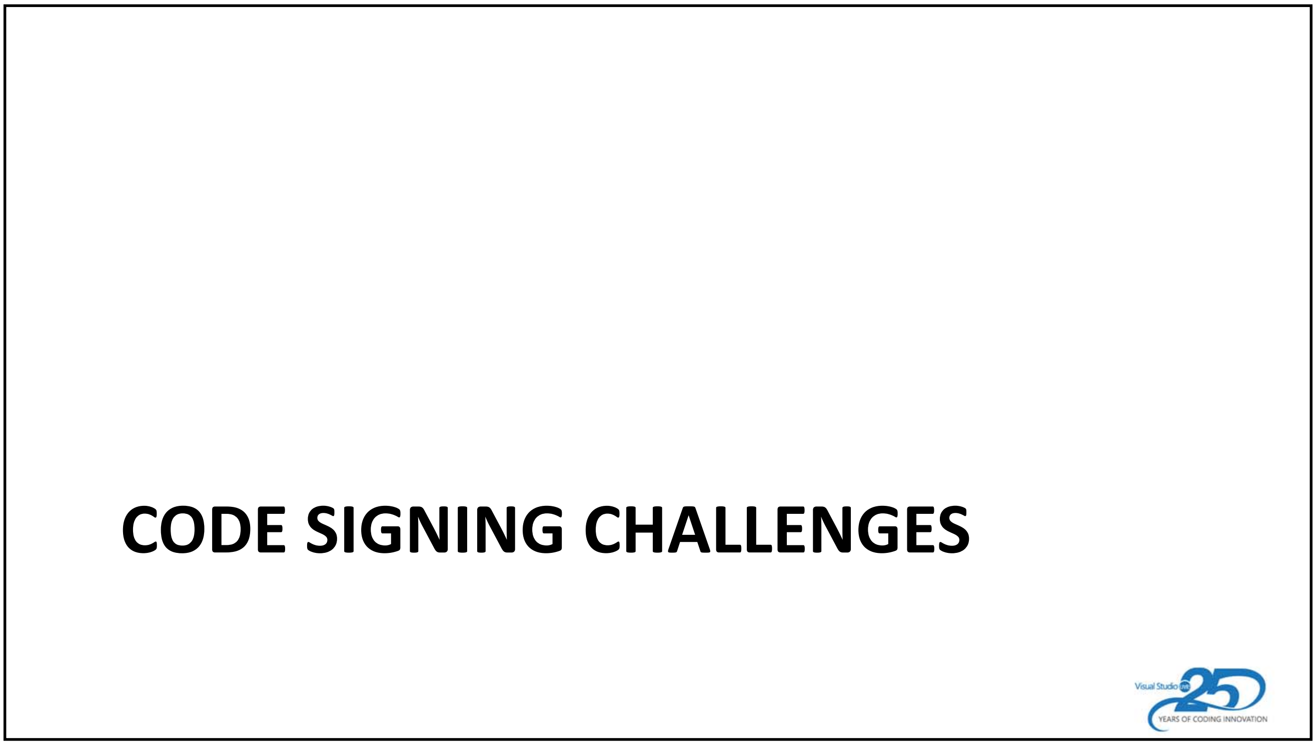
Security Boundary



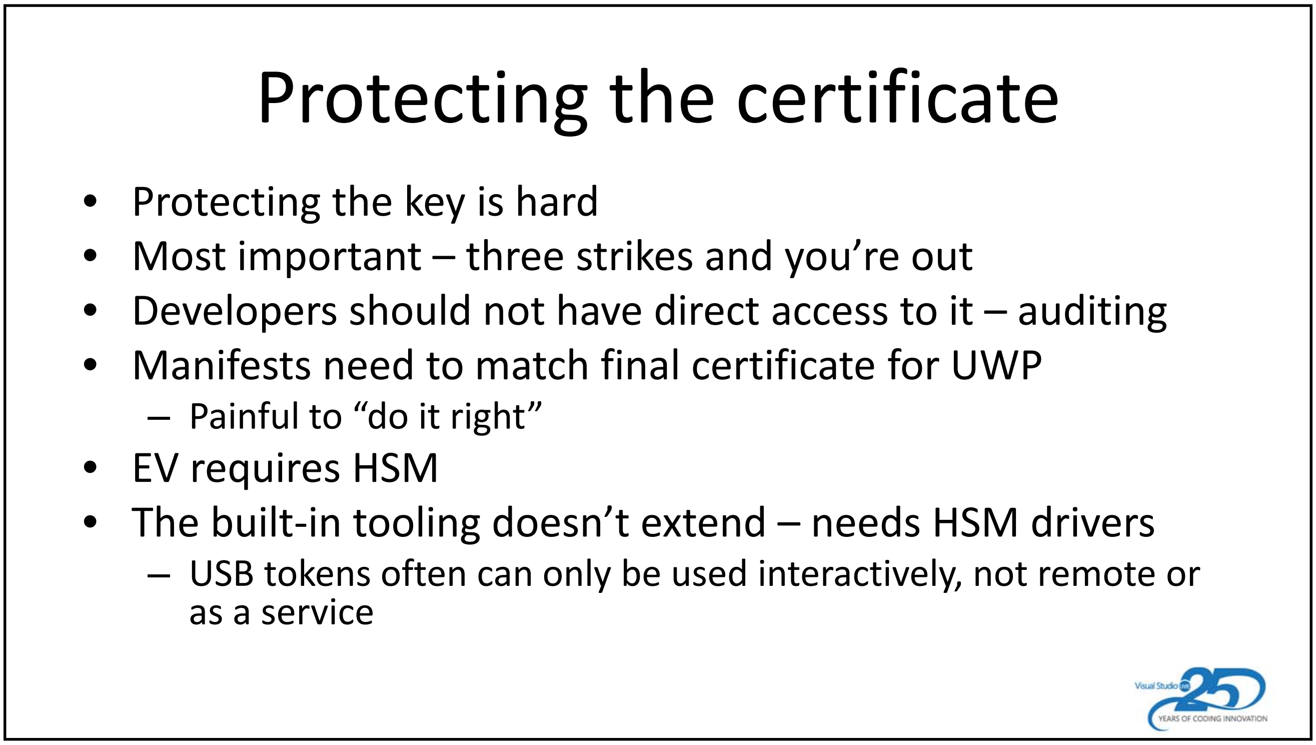
Signing Tools

- Different file types have different tools
  - Authenticode: SignTool.exe (exe, dll, ps1, appx, msix, sys, msi)
  - VSIX packages: VsixSignTool.exe (vsix)
  - ClickOnce / VSTO: Mage.exe (vsto, application)
  - NuPkg: NuGet.exe (nupkg)
  - Java: Jarsign, Apksigner (jar, apk)

- None of them have direct support for external signing - Key Vault
- Rely on CSP (operating system) support

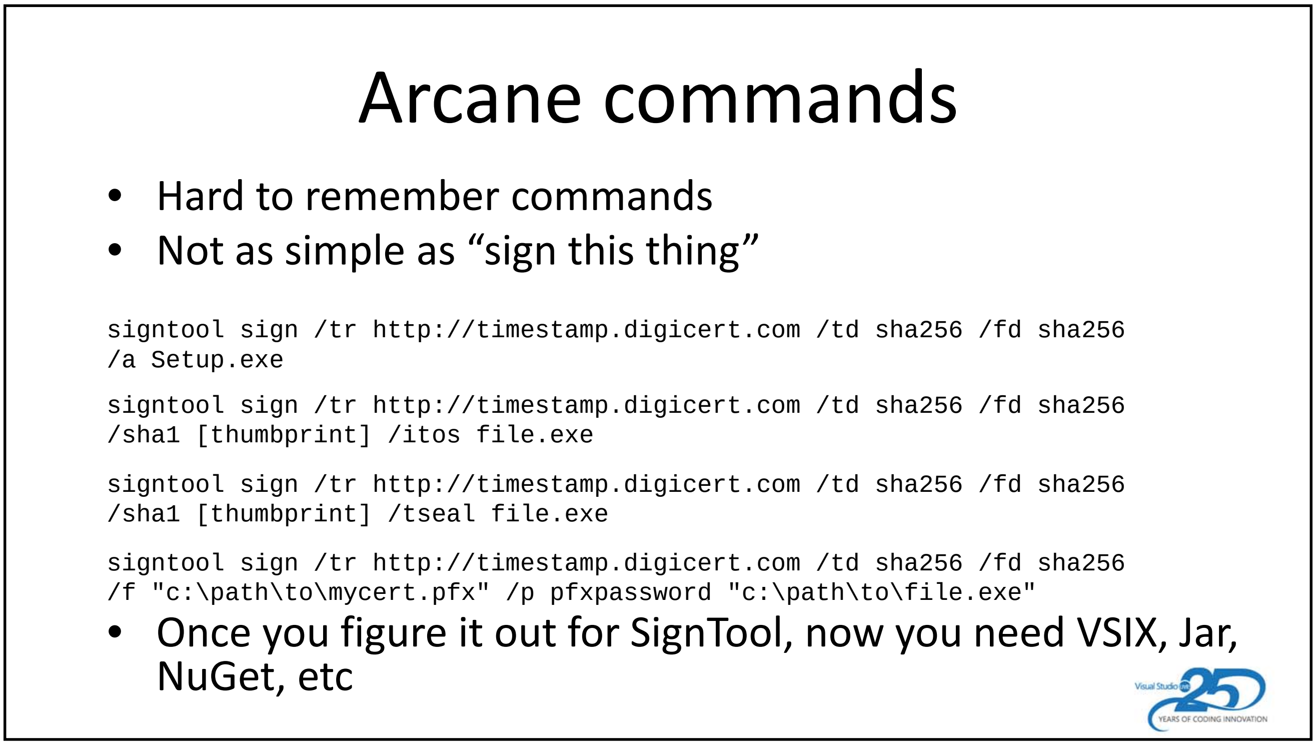


CODE SIGNING CHALLENGES



Protecting the certificate

- Protecting the key is hard
- Most important – three strikes and you're out
- Developers should not have direct access to it – auditing
- Manifests need to match final certificate for UWP
  - Painful to "do it right"
- EV requires HSM
- The built-in tooling doesn't extend – needs HSM drivers
  - USB tokens often can only be used interactively, not remote or as a service



Arcane commands

- Hard to remember commands
- Not as simple as "sign this thing"

```
signtool sign /tr http://timestamp.digicert.com /td sha256 /fd sha256
/a Setup.exe

signtool sign /tr http://timestamp.digicert.com /td sha256 /fd sha256
/sha1 [thumbprint] /itos file.exe

signtool sign /tr http://timestamp.digicert.com /td sha256 /fd sha256
/sha1 [thumbprint] /tseal file.exe

signtool sign /tr http://timestamp.digicert.com /td sha256 /fd sha256
/f "c:\path\to\mycert.pfx" /p pfxpassword "c:\path\to\file.exe"
```

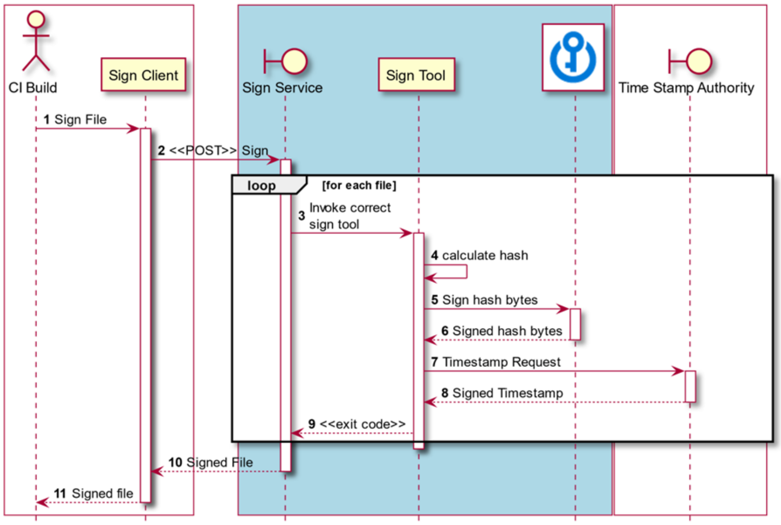
- Once you figure it out for SignTool, now you need VSIX, Jar, NuGet, etc

Orchestrating multiple tools

- VSIX contains DLL’s and a NuPkg, what do you do?
- VSTO contains DLL’s?
- AppX manifest publisher name must match cert.

- Need to sign the “inner” files first, update manifests, sign outer files
- Gets tedious and error-prone

Solution: CI code signing



CI Build

Sign Client

Sign Service

Sign Tool

Time Stamp Authority

1 Sign File

2 <<POST>> Sign

loop [for each file]

3 Invoke correct sign tool

4 calculate hash

5 Sign hash bytes

6 Signed hash bytes

7 Timestamp Request

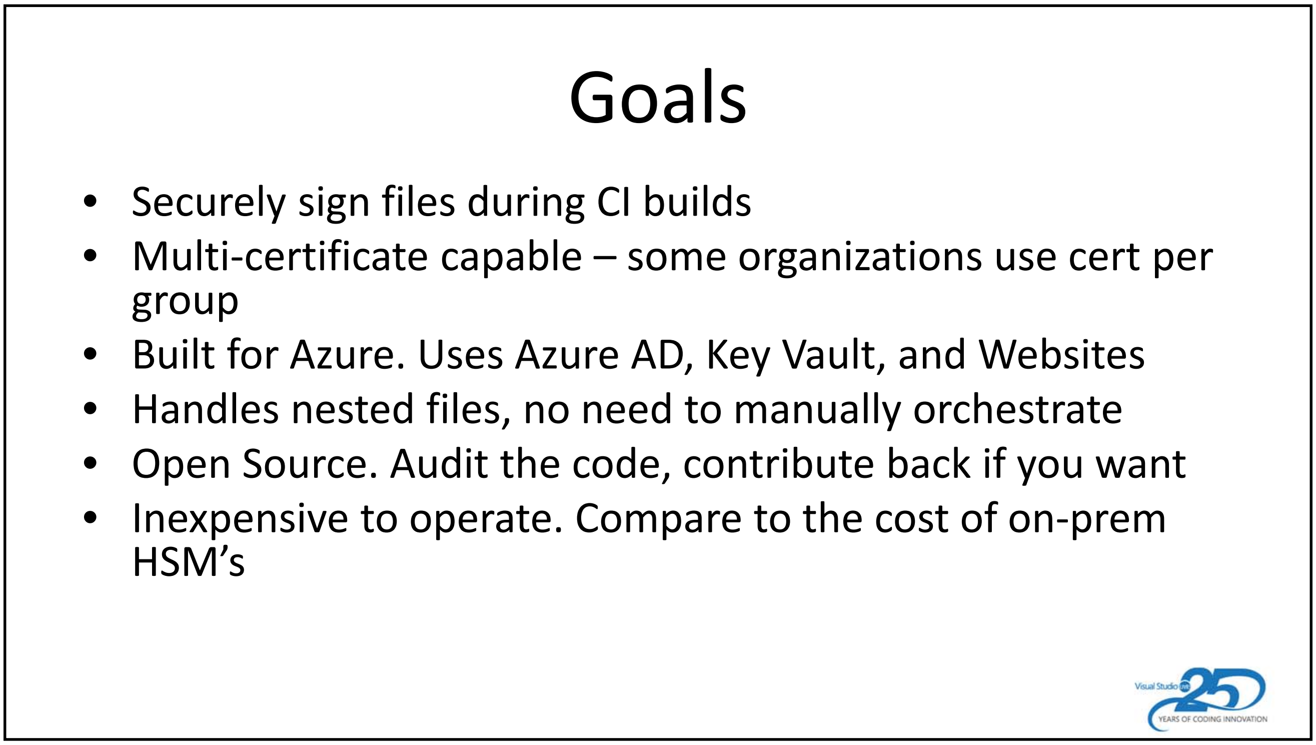
8 Signed Timestamp

9 <<exit code>>

10 Signed File

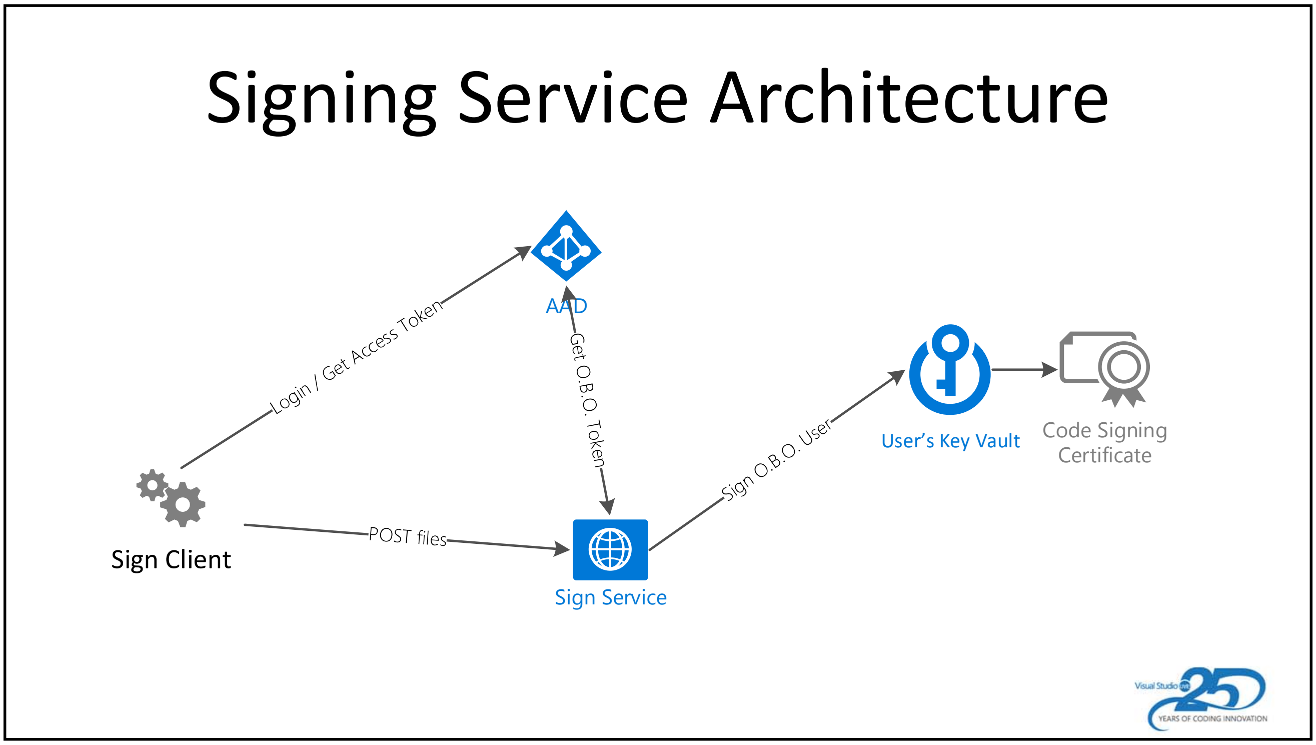
11 Signed file

SIGNING SERVICE



Goals

- Securely sign files during CI builds
- Multi-certificate capable – some organizations use cert per group
- Built for Azure. Uses Azure AD, Key Vault, and Websites
- Handles nested files, no need to manually orchestrate
- Open Source. Audit the code, contribute back if you want
- Inexpensive to operate. Compare to the cost of on-prem HSM’s



Signing Service Architecture

AAD

Login / Get Access Token

Get O.B.O. Token

Sign Client

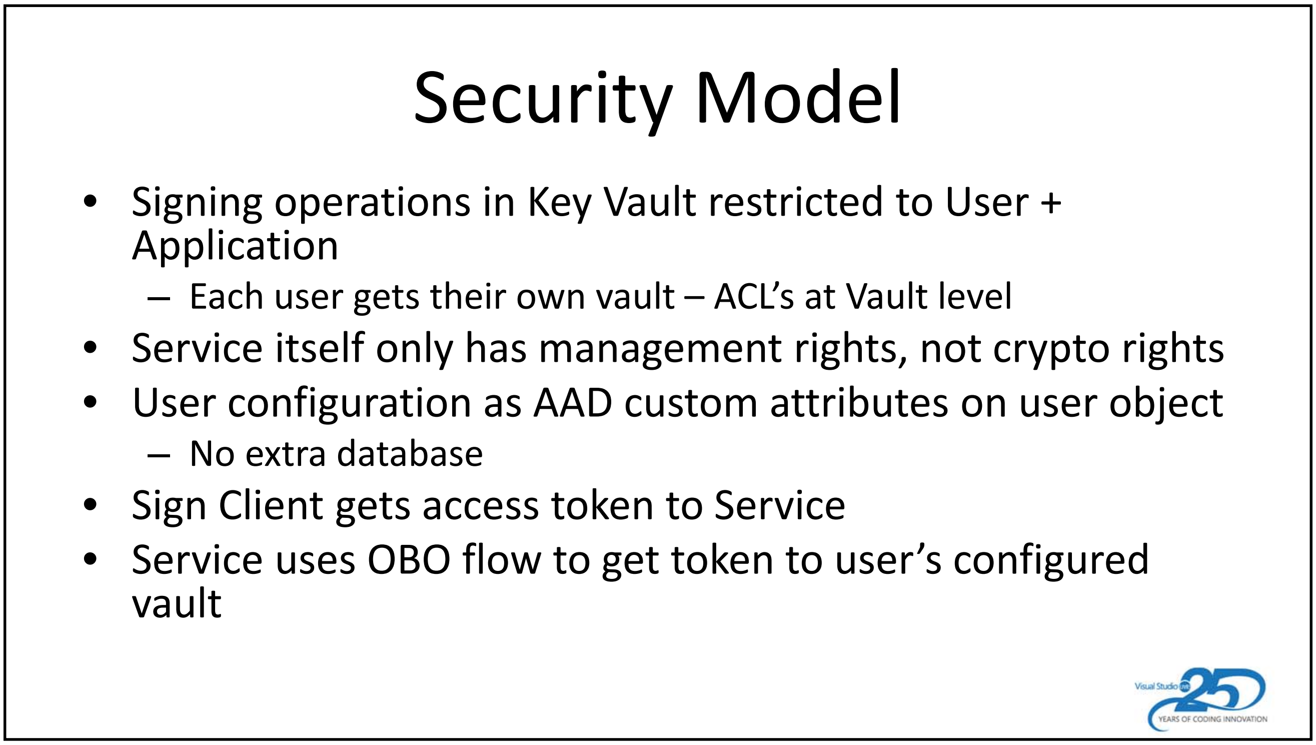
POST files

Sign Service

Sign O.B.O. User

User's Key Vault

Code Signing Certificate



Security Model

- Signing operations in Key Vault restricted to User + Application
  - Each user gets their own vault – ACL's at Vault level
- Service itself only has management rights, not crypto rights
- User configuration as AAD custom attributes on user object
  - No extra database
- Sign Client gets access token to Service
- Service uses OBO flow to get token to user's configured vault

Supporting Key Vault

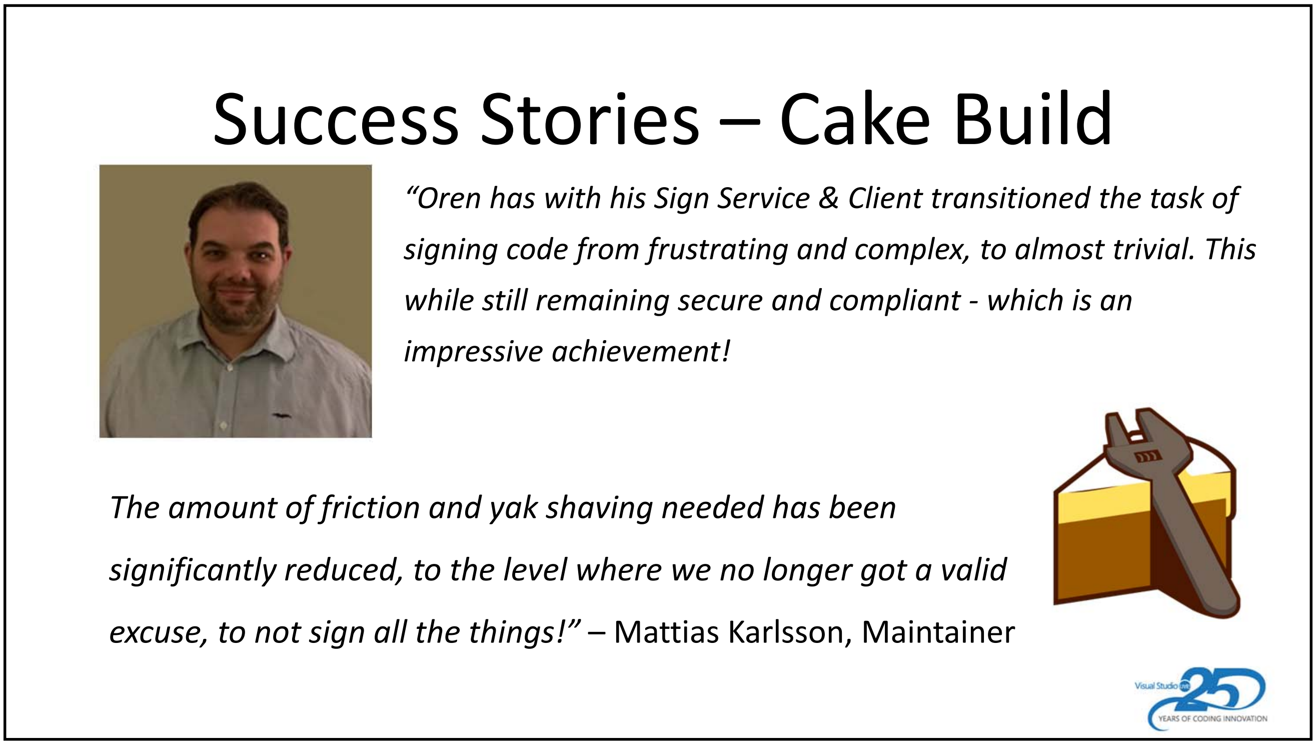
- Kevin Jones created OpenVsixSignTool and AzureSignTool
  - Open source implementations of closed source tools
  - AzureSignTool relies on SignerSignEx3, on Win10/Server 2016
- Mage – adapted signed xml code
- NuGet – custom signing implementation using API
- Jar Sign – coming soon

Success Stories - .NET Foundation

"Code signing can be complicated, and it's not something you want to guess at. Oren has distilled a deep understanding of how to sign code correctly into a service that makes it easy to do it right.

Previously, the .NET Foundation had a cumbersome process relying on virtual machines with signing keys installed, batch files, and luck. Oren's signing service allows us to cleanly integrate code signing directly into our automated builds with keys stored in Key Vault. Very highly recommended!"

- Jon Galloway, Executive Director

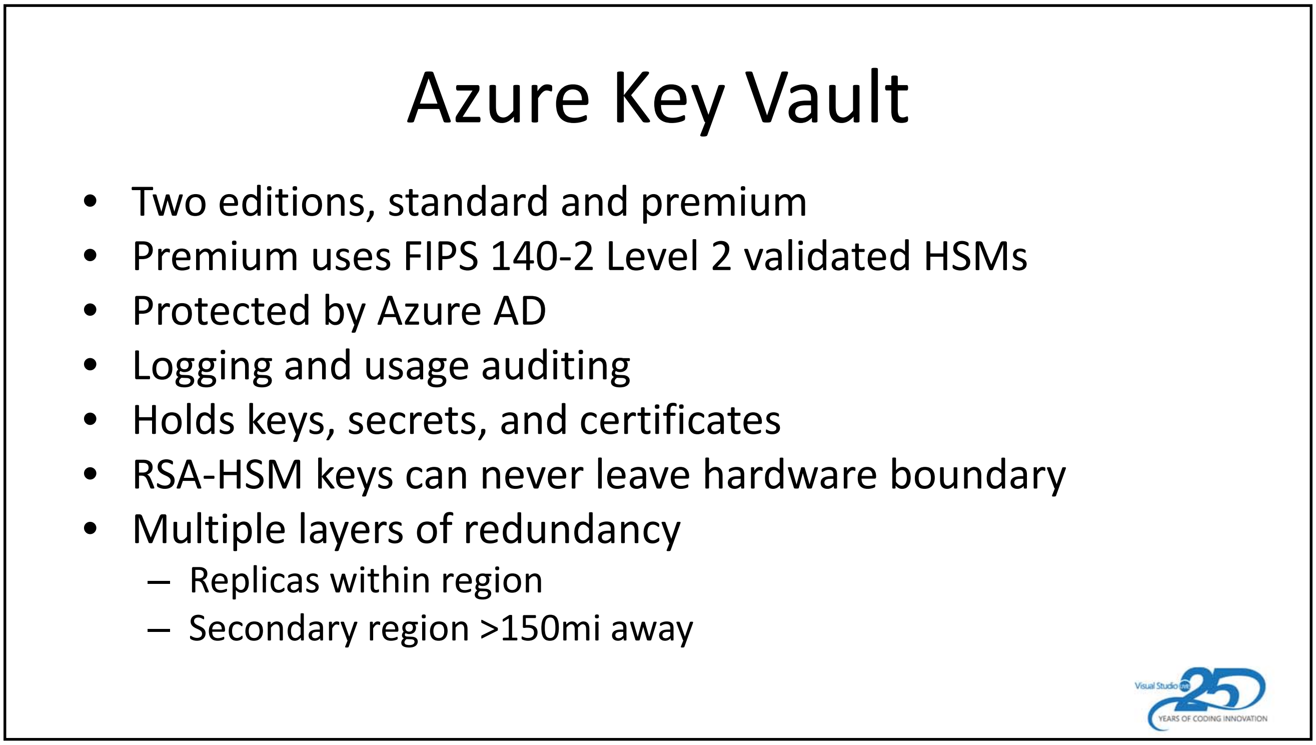


Success Stories – Cake Build

“Oren has with his Sign Service & Client transitioned the task of signing code from frustrating and complex, to almost trivial. This while still remaining secure and compliant - which is an impressive achievement!

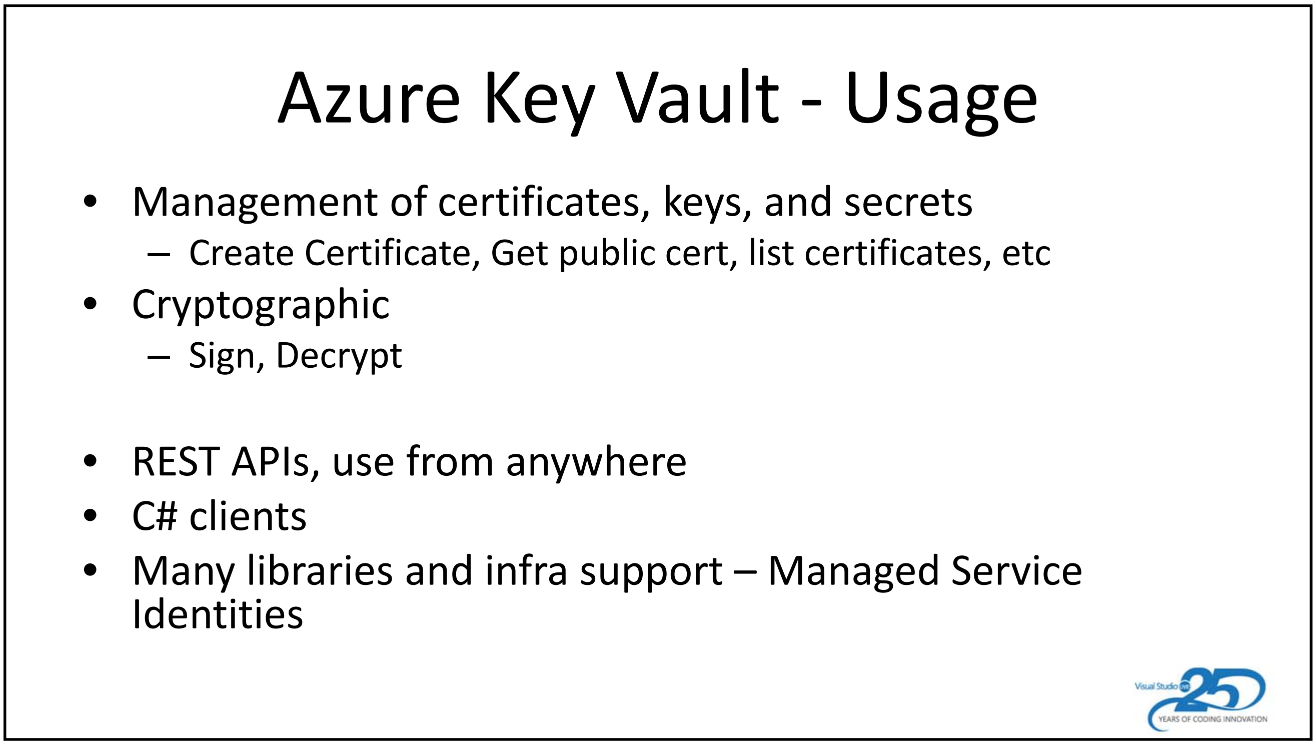
The amount of friction and yak shaving needed has been significantly reduced, to the level where we no longer got a valid excuse, to not sign all the things!” – Mattias Karlsson, Maintainer

CERTIFICATES WITH AZURE KEY VAULT



Azure Key Vault

- Two editions, standard and premium
- Premium uses FIPS 140-2 Level 2 validated HSMs
- Protected by Azure AD
- Logging and usage auditing
- Holds keys, secrets, and certificates
- RSA-HSM keys can never leave hardware boundary
- Multiple layers of redundancy
  - Replicas within region
  - Secondary region >150mi away

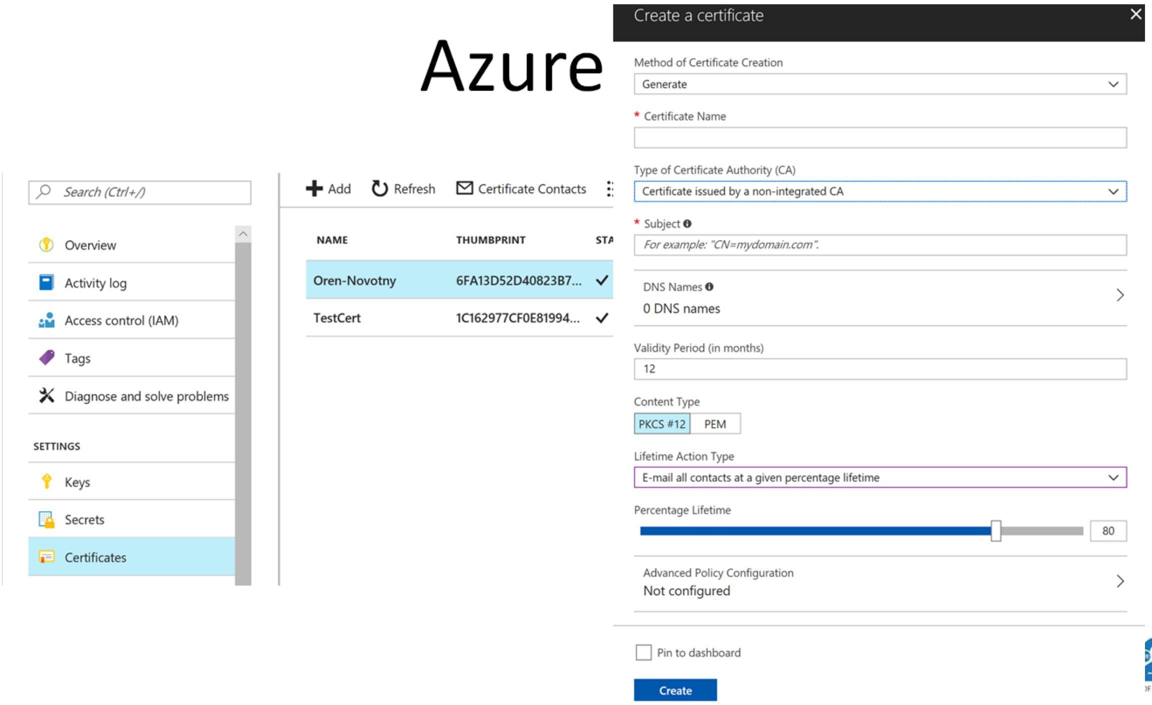


Azure Key Vault - Usage

- Management of certificates, keys, and secrets
  - Create Certificate, Get public cert, list certificates, etc
- Cryptographic
  - Sign, Decrypt

- REST APIs, use from anywhere
- C# clients
- Many libraries and infra support – Managed Service Identities

Azure



NAME	THUMBPRINT	STA
Oren-Novotny	6FA13D52D40823B7...	✓
TestCert	1C162977CF0E81994...	✓

Method of Certificate Creation: Generate

* Certificate Name:

Type of Certificate Authority (CA): Certificate issued by a non-integrated CA

* Subject: *For example: "CN=mydomain.com".*

DNS Names: 0 DNS names

Validity Period (in months): 12

Content Type: PKCS #12 | PEM

Lifetime Action Type: E-mail all contacts at a given percentage lifetime

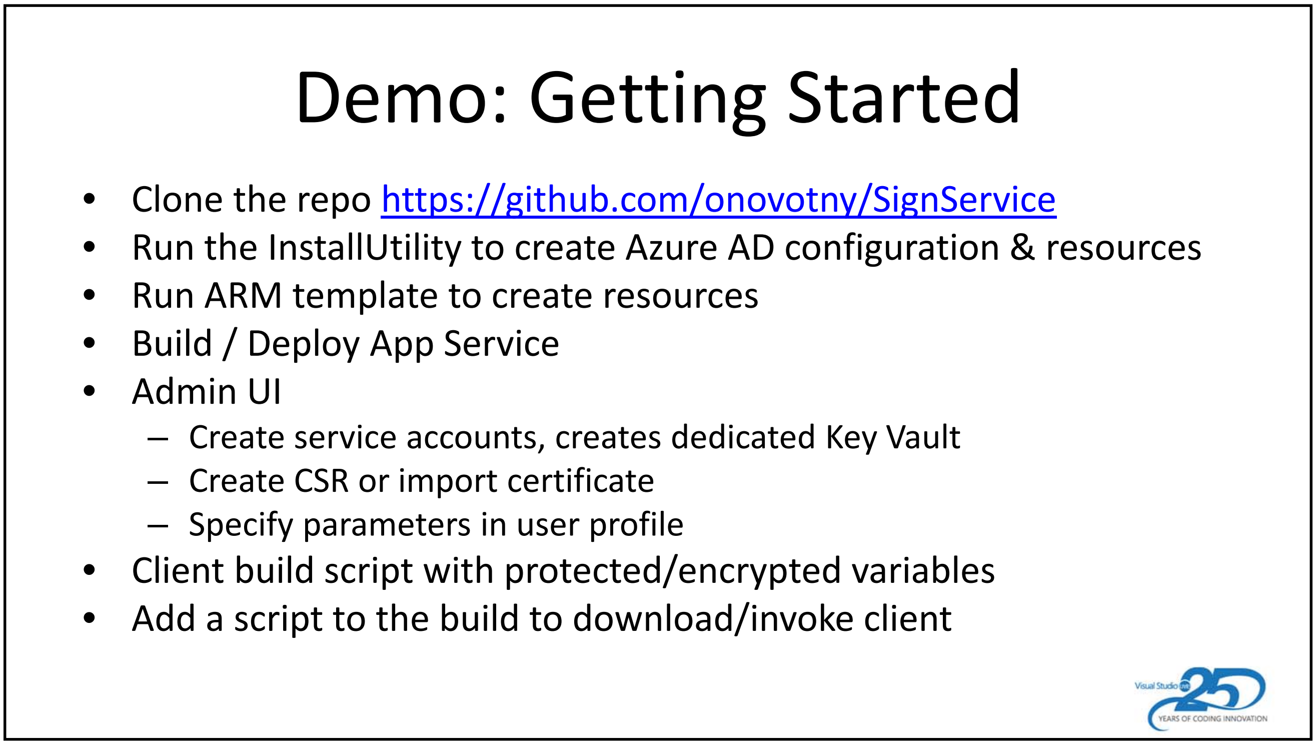
Percentage Lifetime: 80

Advanced Policy Configuration: Not configured

□ Pin to dashboard

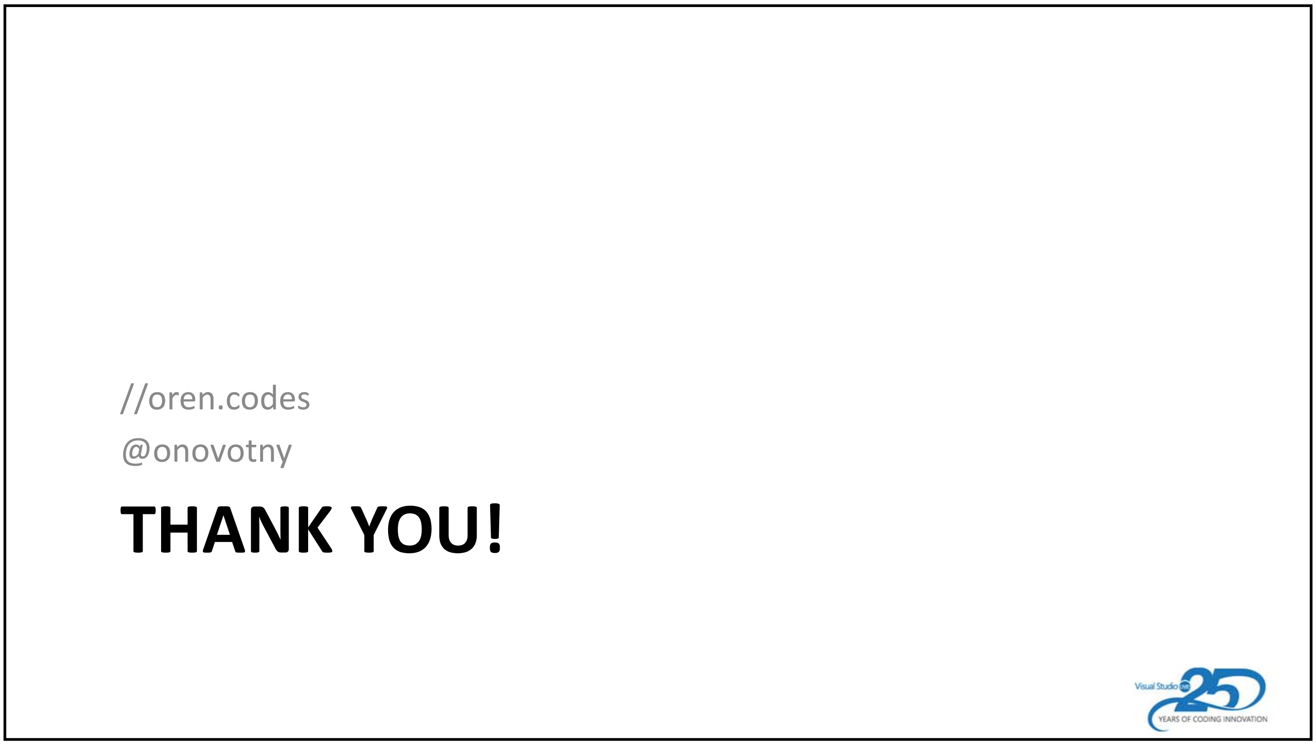
Configuration Requirements

- Azure Subscription and Global Admin of the backing AAD
  - Can use an a dedicated directory/subscription
- Recommended: Azure Pipelines CI/CD
- Resources created:
  - App Service Plan
  - App Service (Website), with site extensions
  - Application Insights
  - Key Vault for runtime config



Demo: Getting Started

- Clone the repo https://github.com/onovotny/SignService
- Run the InstallUtility to create Azure AD configuration & resources
- Run ARM template to create resources
- Build / Deploy App Service
- Admin UI
  - Create service accounts, creates dedicated Key Vault
  - Create CSR or import certificate
  - Specify parameters in user profile
- Client build script with protected/encrypted variables
- Add a script to the build to download/invoke client



//oren.codes

@onovotny

THANK YOU!