PSCONF.EU

PowerShell Conference Europe

Keeping Secrets: State of the Union

Evgenij Smirnov

PRAGUE23

Many thanks to our sponsors:

Evgenij Smirnov

- Born in Riga 1972, lives in Berlin since 1994
- 25+ years of infrastructure consulting
- Senior Solutions Architect @ **Semperis**
- User Group Lead: WSUGB, PSUGB, EXUSG
- MVP Cloud & Datacenter Management
- Blog: it-pro-berlin.de
- Speaker: PSConfEU, CIM, PSDAY.UK, SS2022

Microsoft Most Valuable Professional

@cj_berlin

A strong personal opinion

PSCONF.EU
PRAGUE23

[stuff] that is not secure by design is not worth doing in the first place

a.k.a.

you do not want to be the @$$h013 whose script leaked the VMware admin password to the attacker

Microsoft Most Valuable Professional

@cj_berlin

In this session

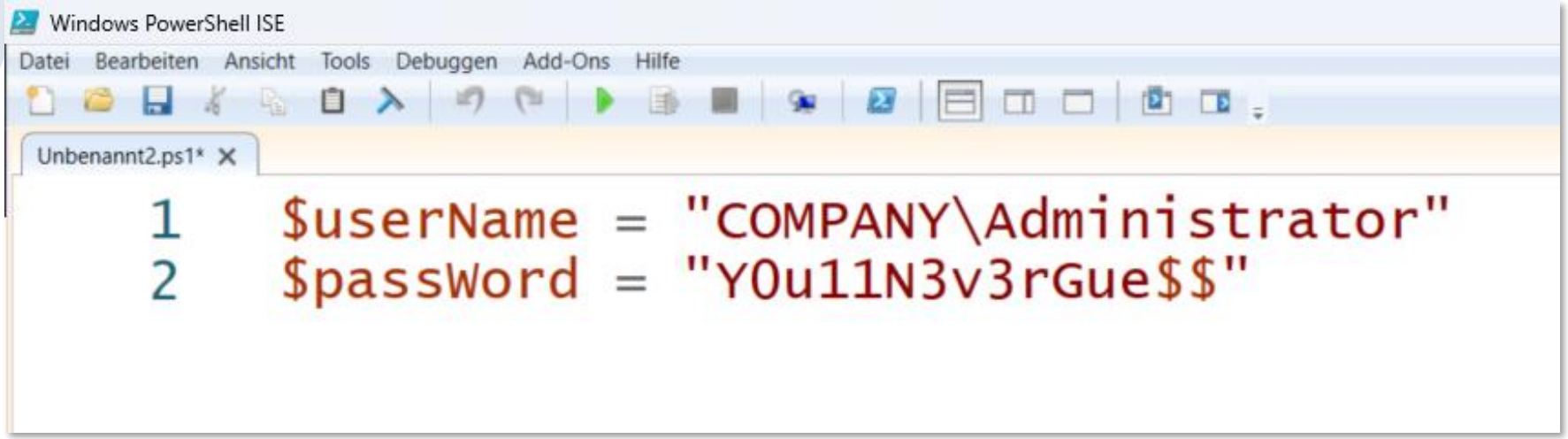
- Why is it (still) such a big deal?
- Managing Expectations
- Secret Management vs. Secret Distribution
- Tying it all together

Why is it such a big deal?

Keeping secrets is important

- Secrets are often the only layer of protection between
  - the big, bad world and
  - the crown jewels
- Secrets are:
  - Credentials (UserName + Password / PSCredential)
  - General-use string Scalars (API keys, application secrets)
  - Cryptographic scalars (private keys)
  - Security by obscurity scalars: port numbers, URIs

Reference Architectures (i)



Reference Architectures (ii)

Windows PowerShell ISE

Datei Bearbeiten Ansicht Tools Debuggen Add-Ons Hilfe

Save-Creds.ps1 | Use-Creds.ps1

```
1   $cred = Get-Credential -Message "Enter Creds"
2   $cred | Export-Clixml -Path "$PSScriptRoot\creds.xml"
```

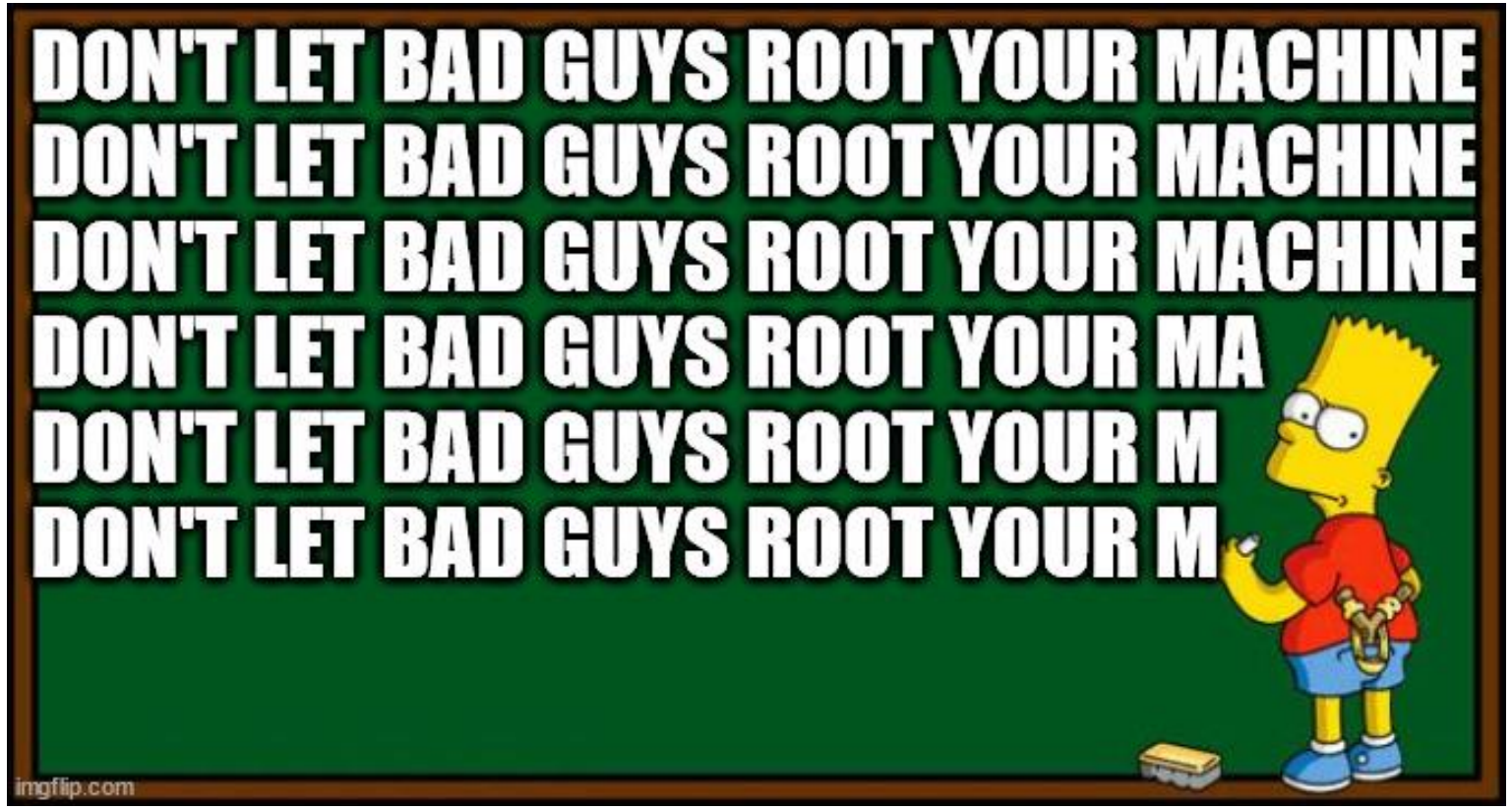
Windows PowerShell ISE

Datei Bearbeiten Ansicht Tools Debuggen Add-Ons Hilfe

Save-Creds.ps1 | Use-Creds.ps1

```
1   $cred = Import-Clixml -Path "$PSScriptRoot\creds.xml"
```

First Rule of Keeping Secrets



Managing Expectations

Use Cases (i)


IF YOU'RE THE MASTER OF ALL SECRETS,
KEEPING THEM SAFE BECOMES
BOTH EASIER AND HARDER!

Use Cases (i)

- If your workstation is yours and yours alone, → use DPAPI and SecureString representation
- If you only need secrets when at your workstation → use SecretManagement and a Vault of your choosing…
  - https://www.powershellgallery.com/packages?q=secretmanagement
  - CredMan, Chromium, MacOS KeyChain, LAPS, …
  - KeePass, 1Password, LastPass, BitWarden, Keeper, …
  - AzKeyVault, HashiCorp, Devolutions, Pleasant, …

Demo

Using KeePass as a vault

Use Cases (ii)

- I need to give a bunch of users a script…
  - …that authenticates against a system those users wouldn't normally have access to…
  - …and then the secret this script is using changes!
- I have to run a script on a bunch of machines…
  - … and then it turns out that it must run on another bunch of machines…
  - … and then the secret the script is using changes!

What can we wish for?

- Scripting breaks trustworthy computing
  - a process can trust another process (known code)
  - scripting host → unknown code by design!
- Objective: Provide a PSCredential object to a script
  - For API keys or other scalar secrets, ignore the user part
- Restriction of visibility with a generic script host:
  - certain user context (or SYSTEM) on a certain machine…
  - …but no further!

A „trusted script host“ design

- [**secret vault**] trusts [**script host**] to:
  - validate the script‘s integrity → *possible with signing*
  - validate the script‘s identity → *compare to known hash*
  - validate the script‘s caller → user
  - …only provide secrets to intended recipients!
- For centralised execution, that‘s exactly what we expect our friends (ScriptRunner & co.) to provide
  - usually host and vault are tied together into one process

Demo

A mock-up of a trusted PowerShell host

Trusted script host possibilities

- Protected Service →intended for ELAM
  - register with Microsoft
  - have a known antimalware vendor provide functionality

- SYSTEM spawning a user-mode shell using impersonation
  - not the same level of security but easier to implement

Management vs. Distribution

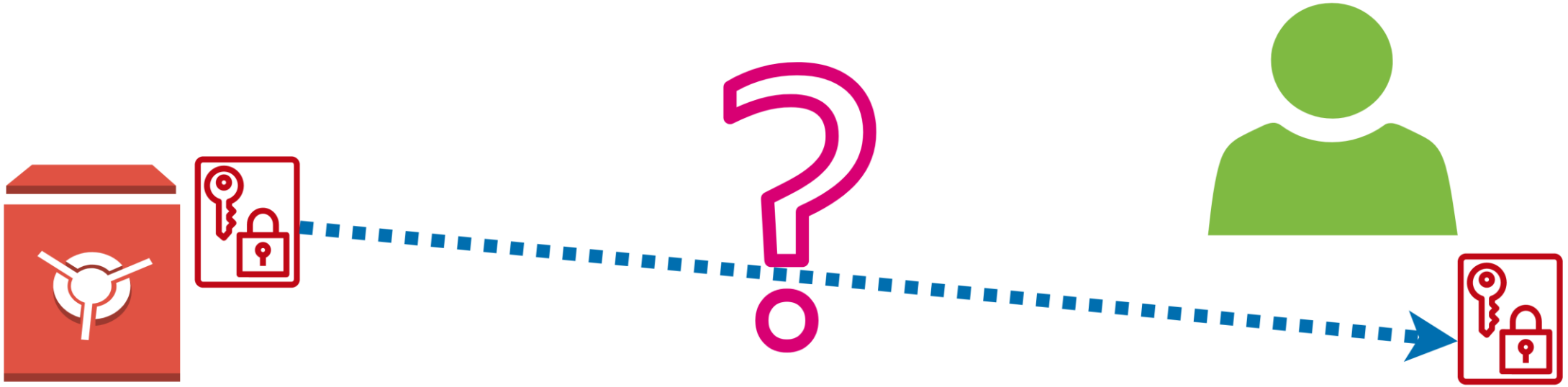
There‘s a Cmdlet for that!

PSCONF.EU PRAGUE23

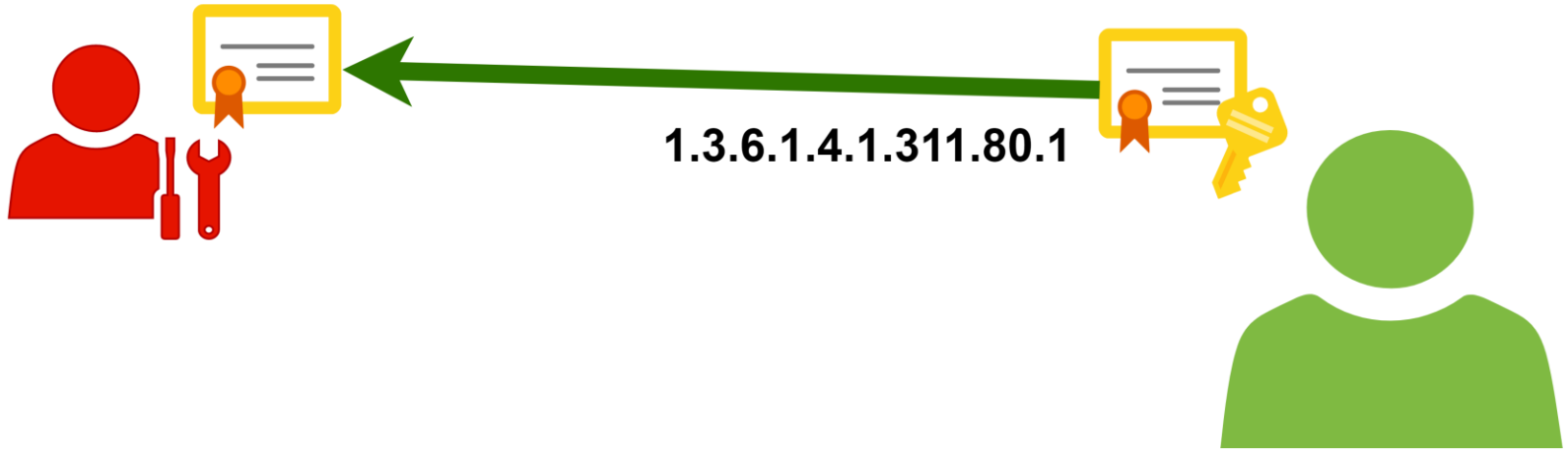
- Storing a secret (**only needs the public part**):
  `Protect-CMSMessage -To <Cert> -Content <Cleartext>`
- Retrieving a secret (**needs the private key**):
  `Unprotect-CMSMessage -To <Cert> -Content <Ciphertext>`
- Making a selfsigned Document Encryption certificate:

```
New-SelfSignedCertificate -Subject "CMS01" -KeyLength 2048
-KeyUsage KeyEncipherment,DataEncipherment
-TextExtension @("2.5.29.37={text}1.3.6.1.4.1.311.80.1")
-KeyExportPolicy Exportable
-CertStoreLocation "Cert:\CurrentUser\My"
-NotAfter ((Get-Date).AddDays(1000))
```

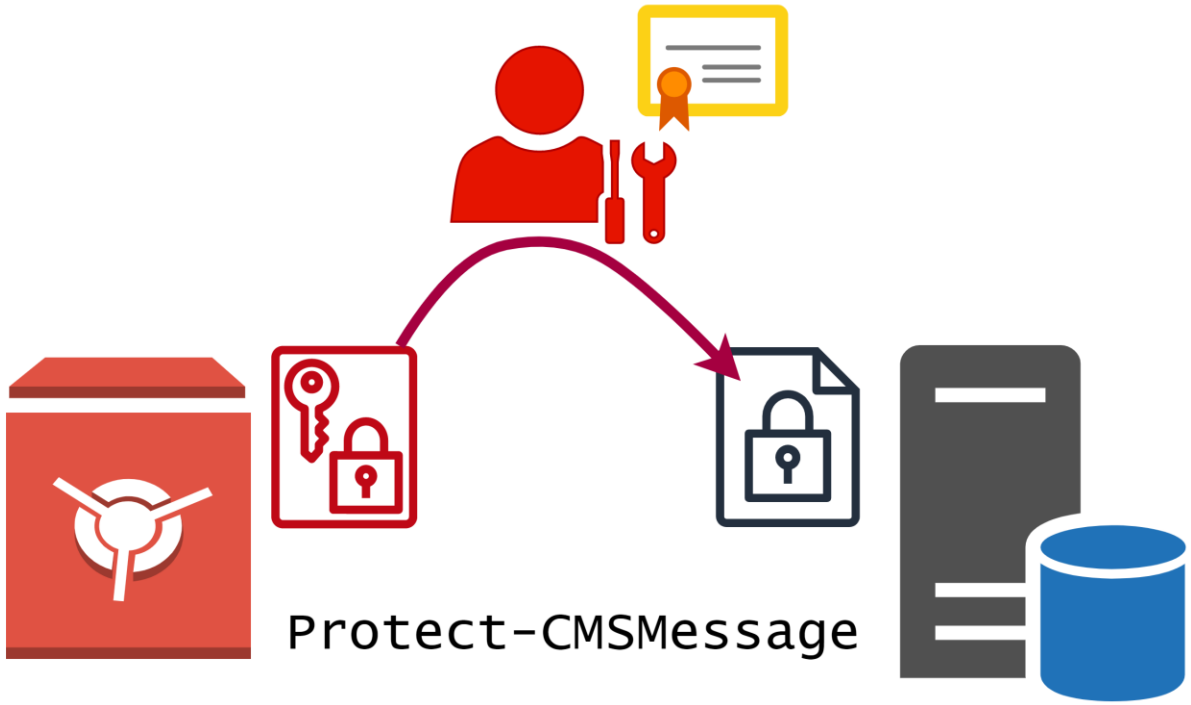
Secret Delivery System



Secret Delivery System

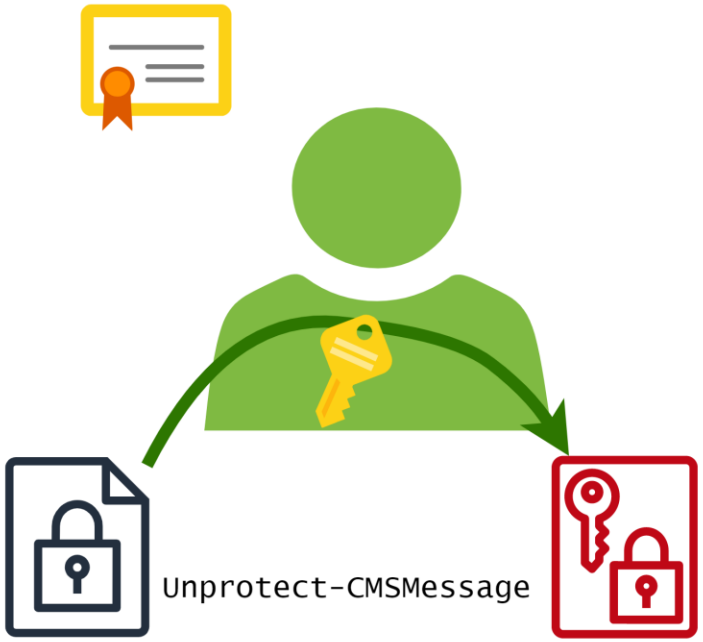


Secret Delivery System



Secret Delivery System

Secret Delivery System



Demo

Secret Delivery System in action

SDE Cross-Platform Blues

- PowerShell 6 didn‘t have the *-CMSMessage cmdlets on *X
  - [X509*] namespaces were all there
  - Had to roll your own
- *X has no „generic“ certificate stores
  - But file system access rights are good enough ☺
- PowerShell on *X has no Resolve-DNSName cmdlet
  - Can‘t use SRV records to locate SDE webservice

Tying it all together

Keeping Screts – In a nutshell

1. Do not let the bad guys root your machine
2. DPAPI is the (potentially) weakest link, on Windows
3. Separating secret management from secret delivery is a major step in securing the whole process
4. Cryptography helps ☺
5. Machine using the secret may need more love, security-wise, than the one providing it!

Q&A

15 minutes, they said…

PSCONF.EU PRAGUE23

Microsoft Most Valuable Professional

@cj_berlin

Thank you!

Code mentioned and shown in this session:

https://github.com/metabpa/secret-delivery
https://github.com/metabpa/trusty-pshost

both will go public on 2023-06-26!

Demo code & Slides: https://github.com/psconfeu/2023

@cj_berlin