PowerShell Conference Europe

PowerShelling Active Directory far and wide

Evgenij Smirnov

MALMÖ 25

PSCONF.EU MALMÖ 25

Many thanks to our sponsors:

Chocolatey

Jane Street

IRONMAN SOFTWARE

KNOWLEDGE FACTORY

SynEdgy

@it-pro-berlin.de

Evgenij Smirnov

Building Modern Active Directory
Engineering, Building, and Running Active Directory for the Next 25 Years
Evgenij Smirnov
tiny PowerShell Projects
Bill Burns
Evgenij Smirnov

chance to win your copy at the end of the talk

Microsoft® MVP Most Valuable Professional

- I live in Berlin (*used to be @cj_berlin*)
- Principal Solutions Architect @ Semperis
- 30+ years of consulting and delivery
- PowerShell MVP since 2020
- Attended all PSConfEU since 2016
  - and spoke at six of them, including this one
- Ask Me Anything about Active Directory
  (but you might not like the answer)

@it-pro-berlin.de

In this session

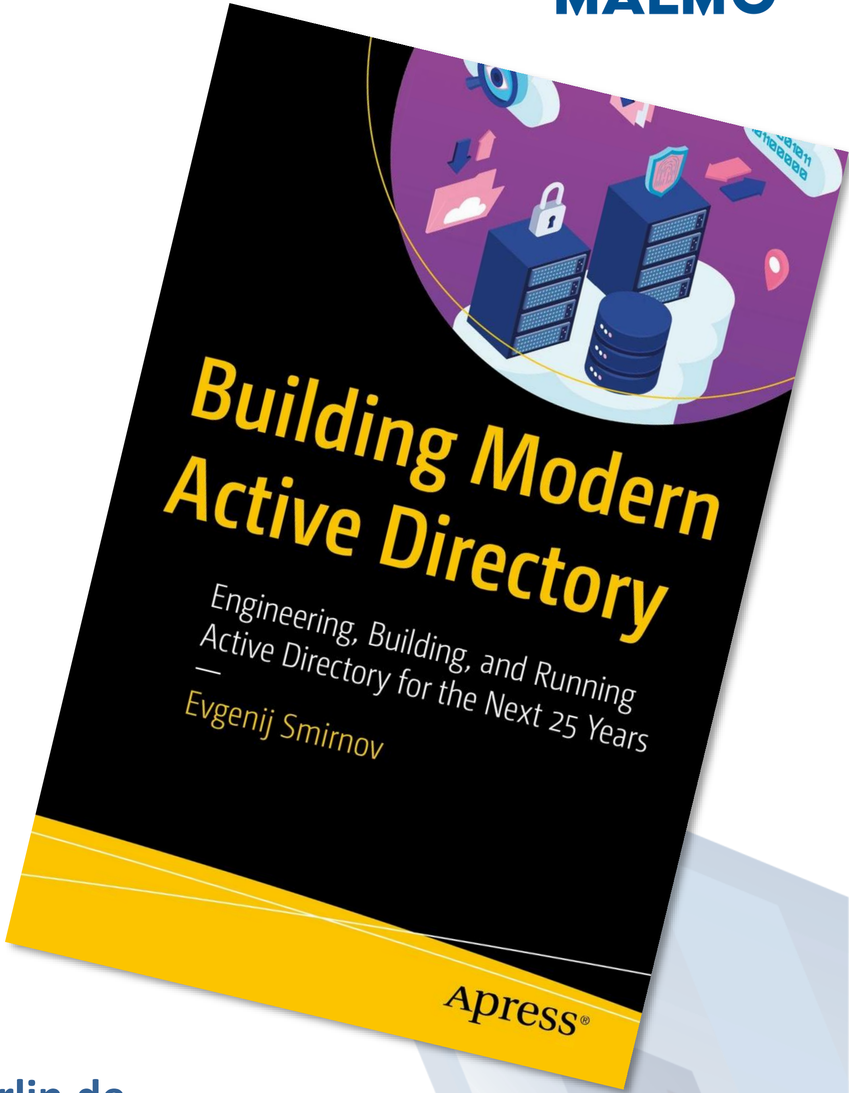
- Why this talk?
- What‘s wrong with `Get-ADUser -Filter * -Property *` ?
- Dealing with lots (and I mean LOTS!) of objects
- Dealing with lots of Domain(Controller)s
  - we will probably not have much time left for this ☹
- Things to take home

Why this talk?

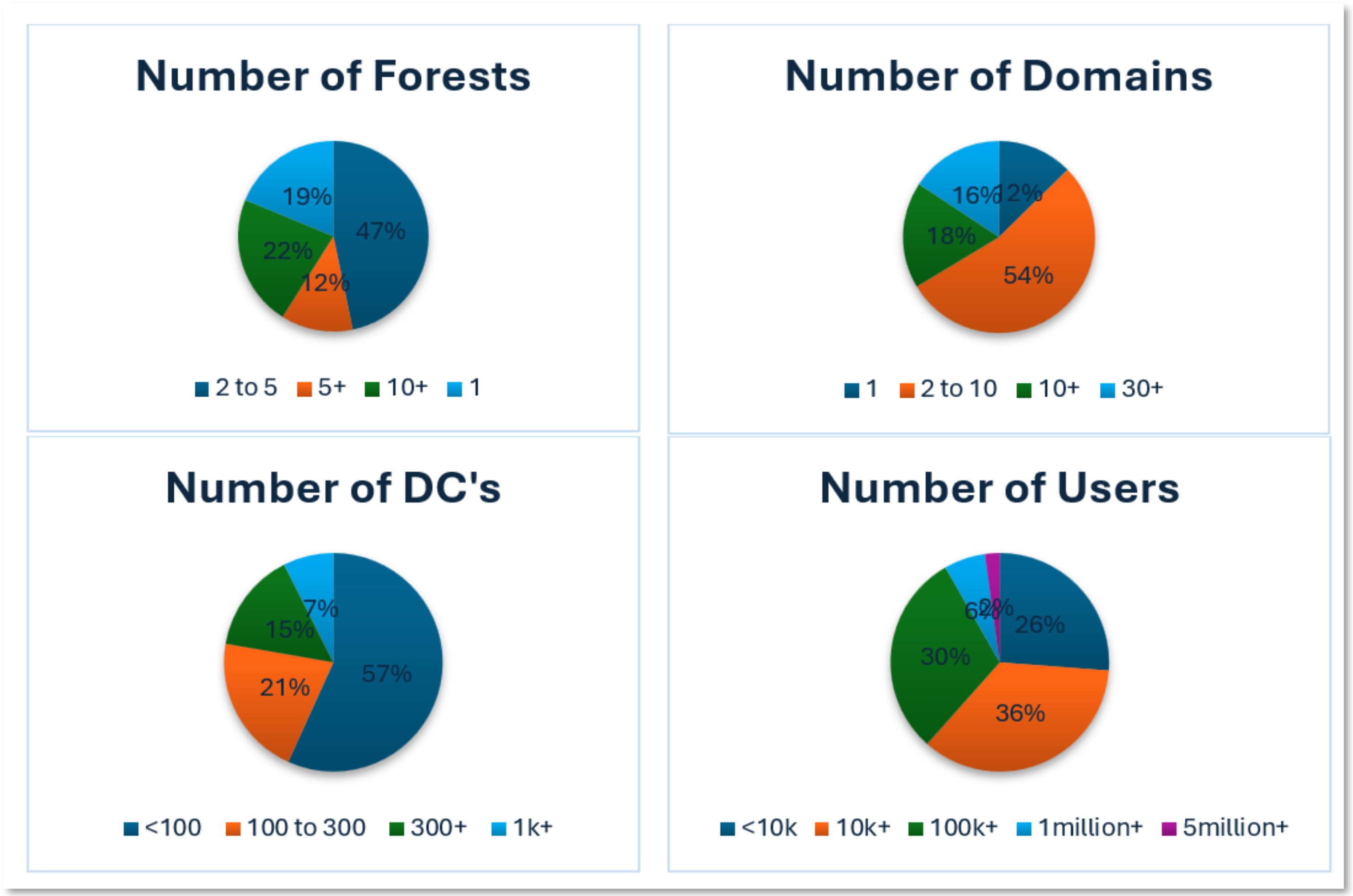
Isn‘t AD irrelevant already?

Why this talk?

- **Wager:** AD will still be foundational 24 years from now
- **Assumption:** Medium size AD forests will become more of an exception…
- …but the huge ones will remain, and so will the small ones, used as Red or application forests


Building Modern Active Directory
Engineering, Building, and Running Active Directory for the Next 25 Years
Evgenij Smirnov
Apress

From Linda Taylor‘s LinkedIn poll:



@it-pro-berlin.de

Mission defines Execution

- Code will run against known AD(s)
  - Addresses (DCs, domains, sites)
  - Structures (e.g. OUs or groups)
  - Policies (e.g. LDAP page size)
- Code must run in any environment
  - Locate everything automatically
  - Make decisions before executing expensive operations

PSCONF.EU
MALMÖ 25

What‘s wrong with AD module?

@it-pro-berlin.de

What‘s wrong with AD module?

- Uses ADWS
  - listens on a different port → may not always be reachable
  - has its own (*rather severe*) performance limitations
  - introduces overhead on two levels:
- Does expensive prep work that you don‘t need
  - like initializing the AD:\ drive
- Lots of post-processing whether you need it or not
  - like converting SIDs and GUIDs to strings

But what else is there?

- System.DirectoryServices **[S.DS]**
  - Ported to .NET but only on Windows!
- System.DirectoryServices.Protocols **[S.DS.P]**
  - The only first-party LDAP ported to Linux / macOS!
- 3rd party LDAP libraries and modules
  - May or may not work outside of Windows
  - e.g. PSOpenAD by Jordan Borean

Dealing with lots of objects

And by that, I do mean LOTS!

@it-pro-berlin.de

Reference Environment

- 1 Forest: Root + Child domain, Server 2025
  - Only 1x DC per domain so not dealing with replication
- Decently sized:
  - 2+ million users
  - 2+ million groups (DL + global + universal)
  - 15+ million group memberships (including cross-domain)
  - NTDS.DIT size > 45 GB
- DCs were not resource constrained during tests

Live Demo Environment

- 1 Forest: Root + Child domain, Server 2025
  - OU structures and naming are identical to reference
- Demoably sized:
  - ~15 thousand users
  - ~15 thousand groups
  - ~20 thousand group memberships
  - NTDS.DIT size ~ 300 MB
- DCs are running on this laptop, so there…

Execution times in the presentation were measured on the same VMs but running unconstrained on a different Hyper-V host

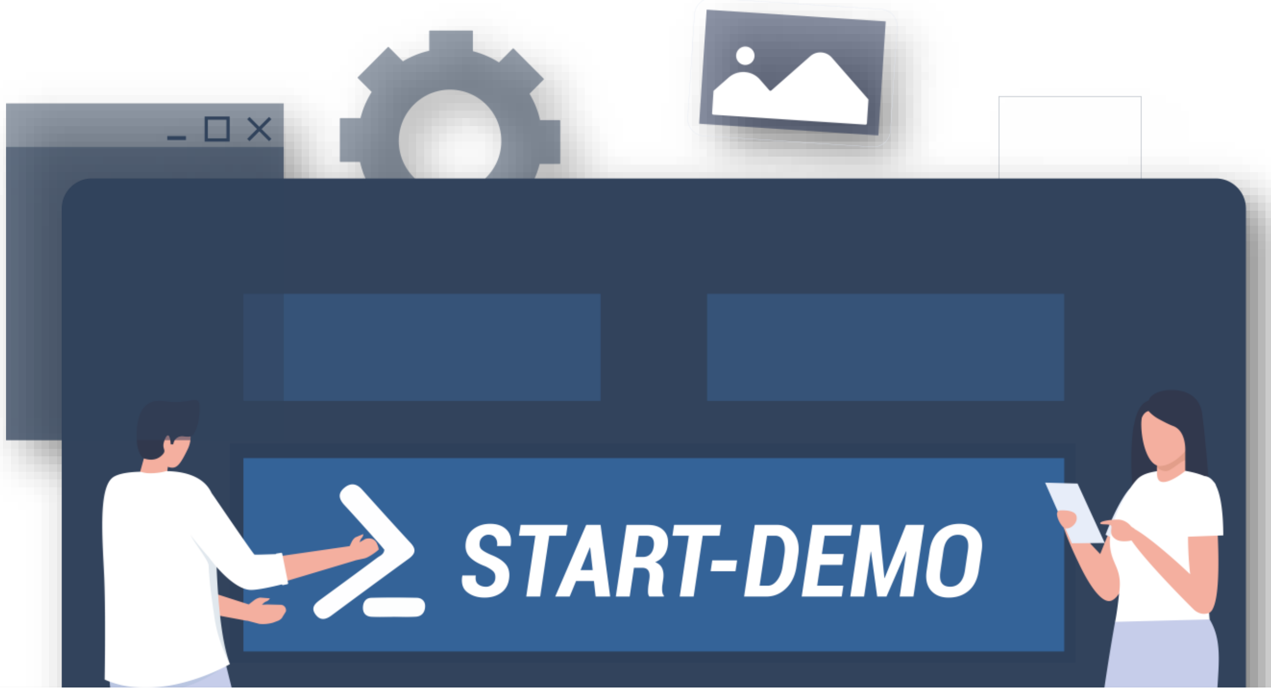
Let‘s establish some baselines

- (AD module) vs. SDS vs. SDS.P vs. PSopenAD
- PS5 vs. PS7 – not 100% fair because…
  - … PSOpenAD is 7 only
  - … ActiveDirectory is 5.1 and will run in implicit remoting

Demo

Module 00

Search



Test	Demo	Reference	Peak RAM
Count all groups by ActiveDirectory PowerShell module	1.4 seconds	229 seconds	2.8 GB
Count all groups by PSOpenAD	1.1 seconds	372 seconds	3.7 GB
Count all groups by System.DirectoryServices.DirectorySearcher	0.52 seconds	55 seconds	1.1 GB
Count all groups by System.DirectoryServices.Protocols in PS5.1	0.42 seconds	40 seconds	260 MB
Count all groups by System.DirectoryServices.Protocols in PS7.5.1	0.36 seconds	43 seconds	165 MB

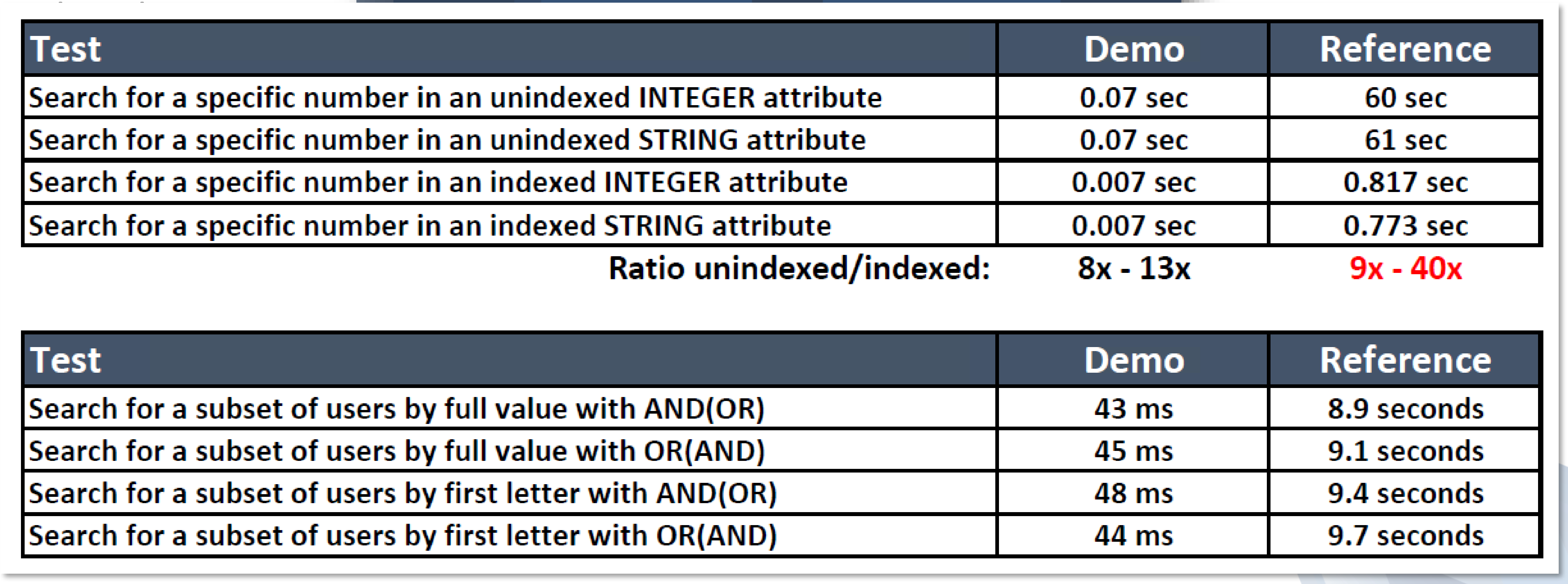
First things first

Before you start optimizing your PowerShell, make sure your LDAP is not slowing you down

- Far-flung searches by unindexed attributes
  - Most notably: `objectClass` vs. `objectCategory` *
- Too many attributes returned (unless they are needed)
- Clause nesting and AND/OR nesting
- Bitwise AND/OR on indexed attributes, logical NOT

Demo

Module 01



Test	Demo	Reference
Search for a specific number in an unindexed INTEGER attribute	0.07 sec	60 sec
Search for a specific number in an unindexed STRING attribute	0.07 sec	61 sec
Search for a specific number in an indexed INTEGER attribute	0.007 sec	0.817 sec
Search for a specific number in an indexed STRING attribute	0.007 sec	0.773 sec
Ratio unindexed/indexed:	**8x - 13x**	**9x - 40x**

Test	Demo	Reference
Search for a subset of users by full value with AND(OR)	43 ms	8.9 seconds
Search for a subset of users by full value with OR(AND)	45 ms	9.1 seconds
Search for a subset of users by first letter with AND(OR)	48 ms	9.4 seconds
Search for a subset of users by first letter with OR(AND)	44 ms	9.7 seconds

Dealing with paged results

- Effective page size is not reported in rootDSE ☹
- Page size is defined per DC
  - there is a Default Query Policy…
  - …but nothing stops you from creating individual ones

Be mindful of your RAM usage

- Dealing with huge result sets (and caching them locally) may lead to extreme RAM consumption!
- If you only need to iterate through the result set once, doing a paged search and not caching beyond one page solves this for you…
- If you *do* need the results long-term, consider SQLite or even SQL – your PowerShell session will thank you for that ☺

@it-pro-berlin.de

Processing constructed attributes

- Constructed attributes are useful:
  - tokenGroups* | msDS-memberOfTransitive
  - parentGUID
  - msDS-ResultantPSO
  - etc.
- However, they require a base search
  - Retrieving constructed attribute values from search results means looping through them!

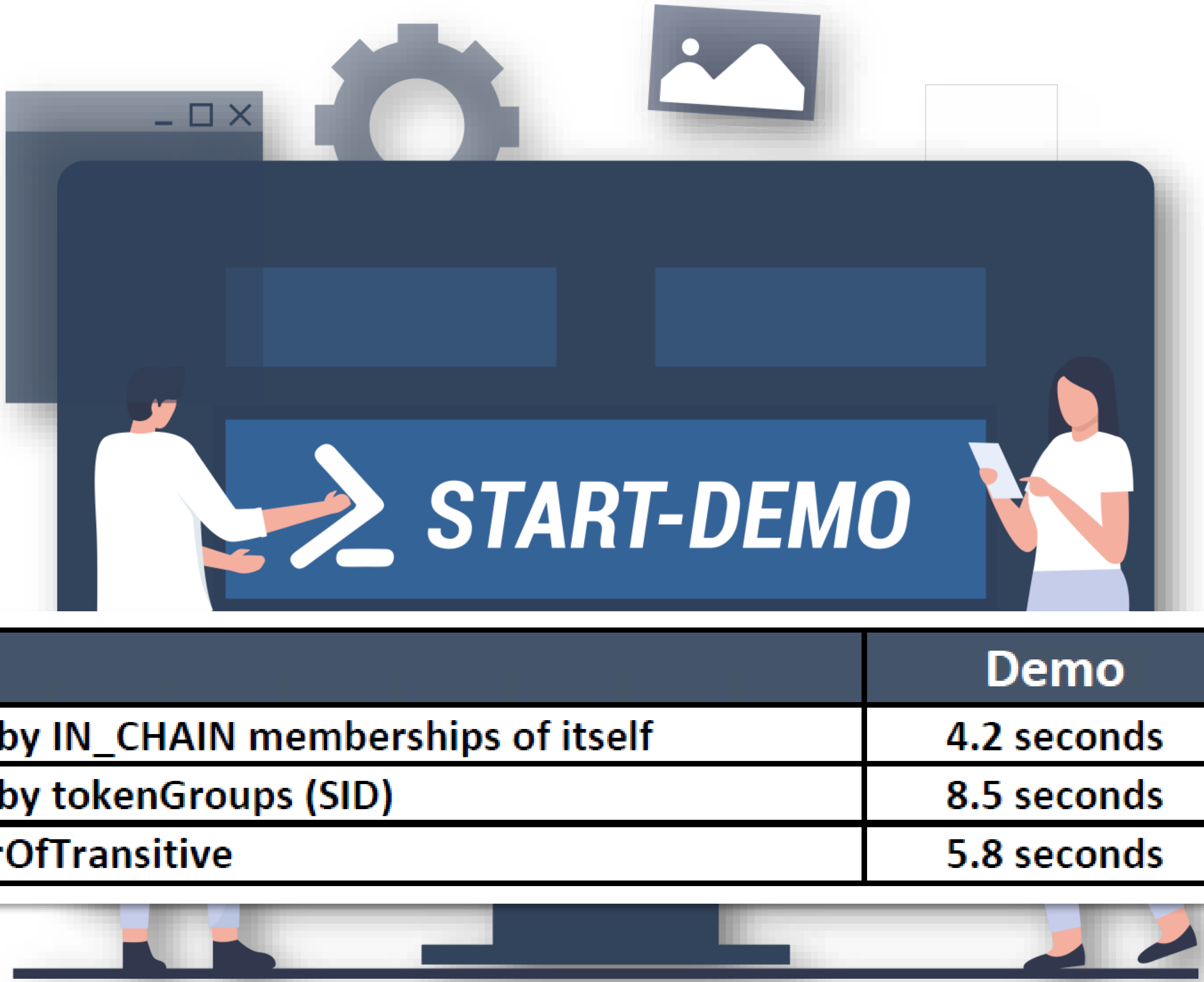
Example: Looped group chains

- We could start building trees locally from **`member`** …
- … or from **`memberOf`** which will be faster …
- …or use **`tokenGroups`** (yes, groups have that, too!) …
- …or use the **`IN_CHAIN`** LDAP control …
- …or use **`msDs-memberOfTransitive`** …

Demo

Module 02

Finding looped group nestings



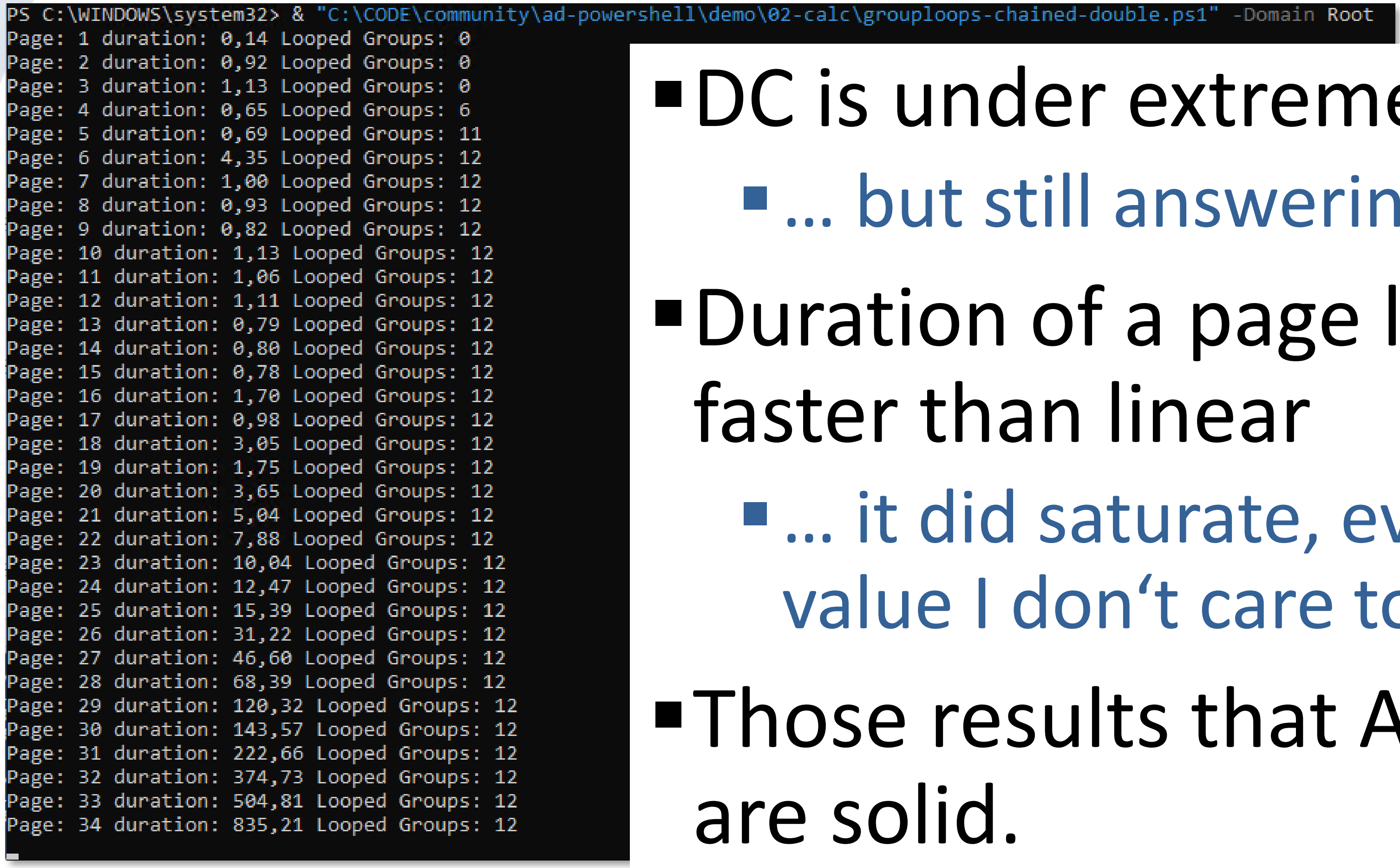
Test	Demo	Reference
Search for nesting loops by IN_CHAIN memberships of itself	4.2 seconds	Ages!
Search for nesting loops by tokenGroups (SID)	8.5 seconds	Ages!
Search by msDS-memberOfTransitive	5.8 seconds	Ages!

PSCONF.EU MALMÖ 25

Beware of member(Of)Transitive!

- There is a bug in NTDS that Microsoft will not fix:
- If msDS-member(Of)Transitive should contain more than 4,500 values…
- …the results past 4,500 are not returned ☹

That escalated quickly…



```
PS C:\WINDOWS\system32> & "C:\CODE\community\ad-powershell\demo\02-calc\grouploops-chained-double.ps1" -Domain Root
Page: 1 duration: 0,14 Looped Groups: 0
Page: 2 duration: 0,92 Looped Groups: 0
Page: 3 duration: 1,13 Looped Groups: 0
Page: 4 duration: 0,65 Looped Groups: 6
Page: 5 duration: 0,69 Looped Groups: 11
Page: 6 duration: 4,35 Looped Groups: 12
Page: 7 duration: 1,00 Looped Groups: 12
Page: 8 duration: 0,93 Looped Groups: 12
Page: 9 duration: 0,82 Looped Groups: 12
Page: 10 duration: 1,13 Looped Groups: 12
Page: 11 duration: 1,06 Looped Groups: 12
Page: 12 duration: 1,11 Looped Groups: 12
Page: 13 duration: 0,79 Looped Groups: 12
Page: 14 duration: 0,80 Looped Groups: 12
Page: 15 duration: 0,78 Looped Groups: 12
Page: 16 duration: 1,70 Looped Groups: 12
Page: 17 duration: 0,98 Looped Groups: 12
Page: 18 duration: 3,05 Looped Groups: 12
Page: 19 duration: 1,75 Looped Groups: 12
Page: 20 duration: 3,65 Looped Groups: 12
Page: 21 duration: 5,04 Looped Groups: 12
Page: 22 duration: 7,88 Looped Groups: 12
Page: 23 duration: 10,04 Looped Groups: 12
Page: 24 duration: 12,47 Looped Groups: 12
Page: 25 duration: 15,39 Looped Groups: 12
Page: 26 duration: 31,22 Looped Groups: 12
Page: 27 duration: 46,60 Looped Groups: 12
Page: 28 duration: 68,39 Looped Groups: 12
Page: 29 duration: 120,32 Looped Groups: 12
Page: 30 duration: 143,57 Looped Groups: 12
Page: 31 duration: 222,66 Looped Groups: 12
Page: 32 duration: 374,73 Looped Groups: 12
Page: 33 duration: 504,81 Looped Groups: 12
Page: 34 duration: 835,21 Looped Groups: 12
```

- DC is under extreme load
  - … but still answering queries
- Duration of a page loop is growing faster than linear
  - … it did saturate, eventually, but at a value I don‘t care to quote here
- Those results that ARE found, are solid.

What *did* work… sort of…

- Collect direct memberships into SQL (~ 4.5 hours)
  - almost zero load on DC
  - almost zero load on the client
  - significant write load on SQL (CPU, disk)
- Running an evaluation over that database (~ 3 days)
  - zero load on AD or client if using Stored Procedures
  - significant load on SQL but that‘s what it‘s for
- Idea: use graph database (SQL, neo4j, Aerospike, ???)

Small Changes in a Big Dataset

- Change notifications
  - Require async connection → not a good fit for scripting
- Cookies and DirSync
  - **NB: The cookie only exists on the same DC!**
- Replication Metadata
- USN
  - **NB: USNs are also maintained per DC!**
  - USNs are [int64] – PowerShell is not very good at this

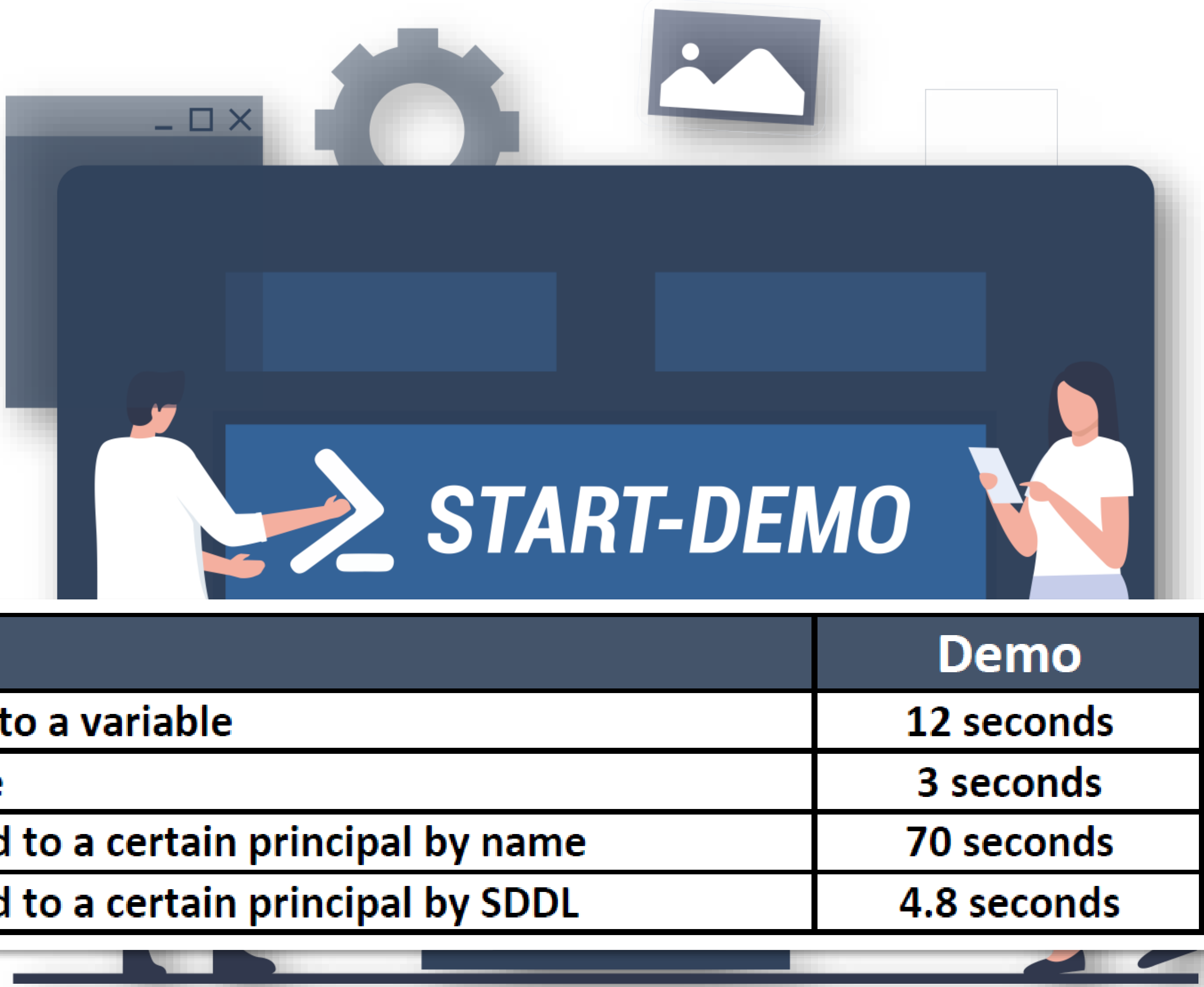
Processing permissions

- Every object's ACL is stored in `ntSecurityDescriptor`
  - Binary structure representing SDDL
  - SDDL is a very strictly structured string
- The usual way of dealing with ACLs is by using „security references"…
  - …which causes Windows to resolve the objects against AD
- There may, however, be a ~~better~~ faster way…

Demo

Module 03

Permissions & SDDL



Test	Demo	Reference
Read the Access object into a variable	12 seconds	1,706 seconds
Read SDDL into a variable	3 seconds	420-450 seconds
Find permissions assigned to a certain principal by name	70 seconds	6,300 seconds
Find permissions assigned to a certain principal by SDDL	4.8 seconds	420-450 seconds

Dealing with lots of endpoints

Because sometimes you simply have to…

@it-pro-berlin.de

Use cases for dealing with DCs

- Non-replicating attributes (LDAP)
    - lastLogon | logonCount (*remember the ID resolution?*)
    - badPasswordTime | badPwdCount
    - whenChanged
- Effective Audit policy or Security policy
- State of replication in DFS-R
- Event logs

Forestwide searches

- Normally, a GC search should be considered the default best practice
- In forests with many domains, a parallel search across individual domains can be faster
  - especially if you expect lots of results…
  - …and have to iterate through them!

Conclusion

Things to take home

Things to take home

- Think twice before you LDAP!
- For large enironments, use S.DS.P
  - Even if it means producing ugly code…
- Be mindful of local resource consumption
- Also look at…
  - ASQ searches (*only work within one domain, base search*)
  - Value range retrieval

More things to take home

- Computed attributes come with a hefty price tag
  - Base searches
  - LSASS load
- Learn SDDL – permissions are super important!
  - …have you caught that BadSuccessor yet? ;-)
- For regular searches, cache the USN watermark
  - and use the same DC every time

More things to take home

- In loops, optimize to the max
  - No extensive logging
  - No advanced functions
  - No output, maybe an „I‘m alive a doing stuff“ every page
  - No progress bar since that requires knowing the count!
- For many endpoints/partitions, work in parallel
  - Be mindful of the reachability, cache results locally

Q&A

Win a signed copy of “Building Modern Active Directory”!

@it-pro-berlin.de