How to get rid of NTLM?

SambaXP 2026

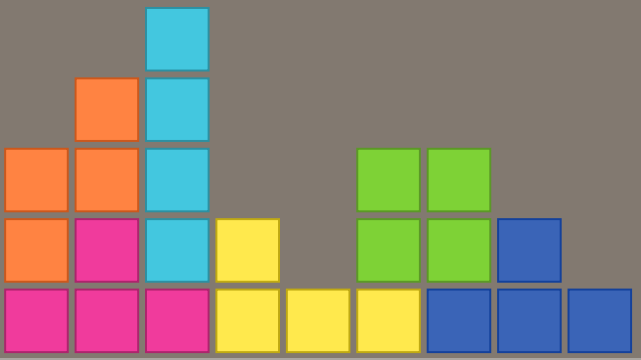
Alexander Bokovoy & Andreas Schneider

Senior & Principal Software Engineer | Red Hat | Samba Team

Who are we?

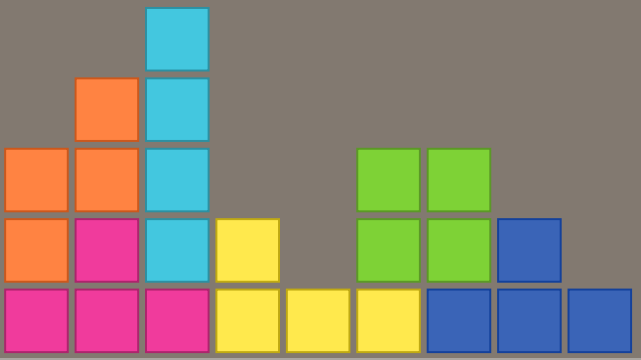
About Alexander

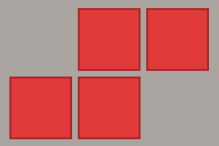
- FreeIPA core developer
- Samba Core Team member since 2003
- MIT Kerberos contributor



About Andreas

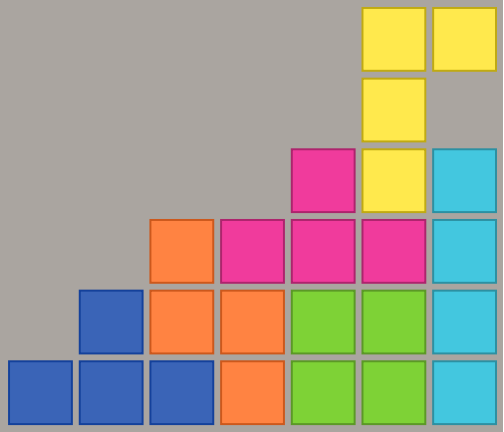
- FOSS Hacker (libssh, cmocka, ...)
- Samba Core Team member since 2010
- MIT Kerberos contributor

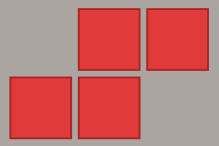




1

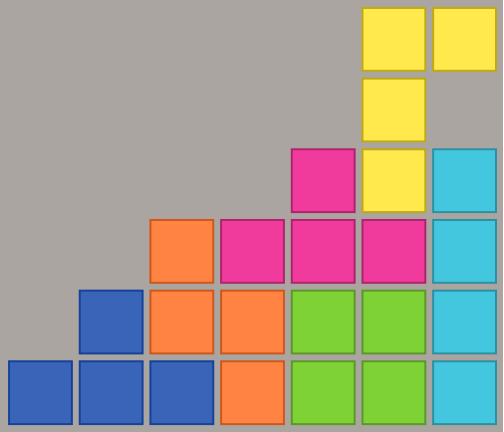
NTLM - What is it?

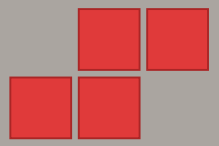




NTLM is Micrsoft's CRAP

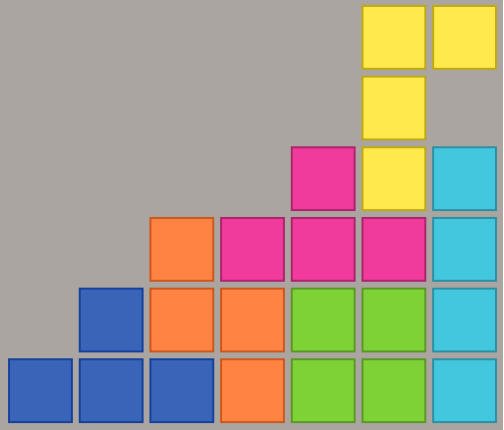
- **N**ew **T**echnology **L**an **M**anager is Microsoft's
- **C**hallenge **R**esponse **A**uthentication **P**rotocol





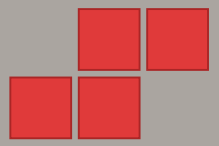
NTLM - A Brief History

- NT LAN Manager (NTLM, MS-NLMP) is a family of security protocols
  - First revisions date back to 1993 - more than 30 years old
  - LM → NTLM → NTLMv2



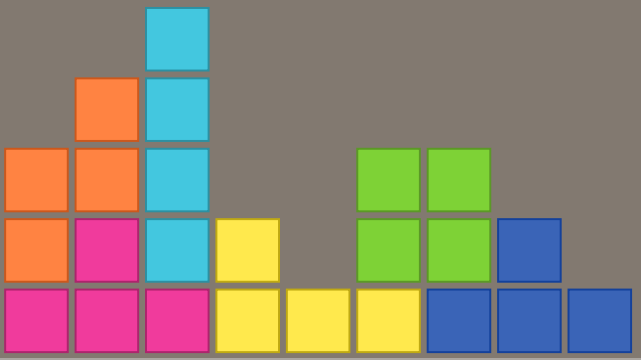
How NTLM Works

1. Client says "I want to authenticate"
2. Server sends a random challenge (nonce)
3. Client computes a response by hashing the NT password hash with the challenge
4. Server verifies the response (locally or by asking a DC)



NTLM in Practice

- Used across many Windows protocols:
  - SMB file sharing
  - HTTP/NTLMSSP (IIS, proxy auth)
  - LDAP
  - RPC
- Default fallback when Kerberos is not available

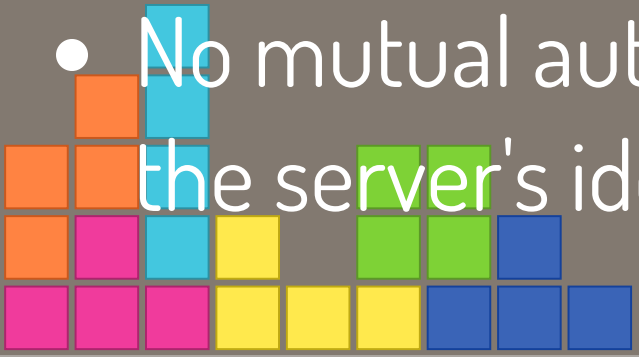


2

Why NTLM must go

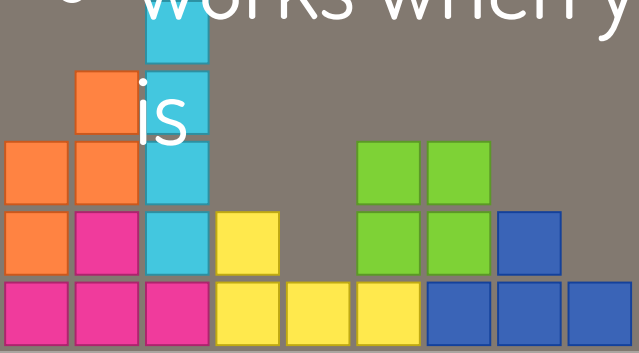
NTLM Security Weaknesses

- The NT hash is an unsalted MD4 checksum of the password
- **Pass-the-hash attacks**: stealing a hash is as good as stealing a password
- **Relay attacks**: captured NTLM exchanges can be forwarded to other servers
- No mutual authentication - client cannot verify the server's identity



Why is NTLM so sticky?

- doesn't require local network connection to a Domain Controller (DC)
- is the only protocol supported when using local accounts
- works when you don't know who the target server is

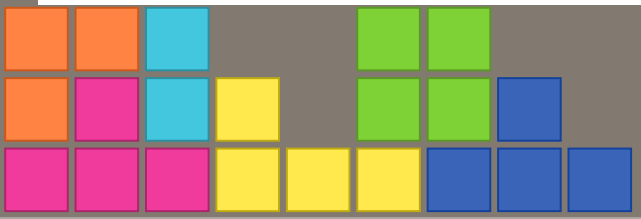


Full Circle In (Almost) 30 Years

- Microsoft (and everyone), since 2010 preaches:
  - Stop using NTLM!
- Microsoft, in 2023:

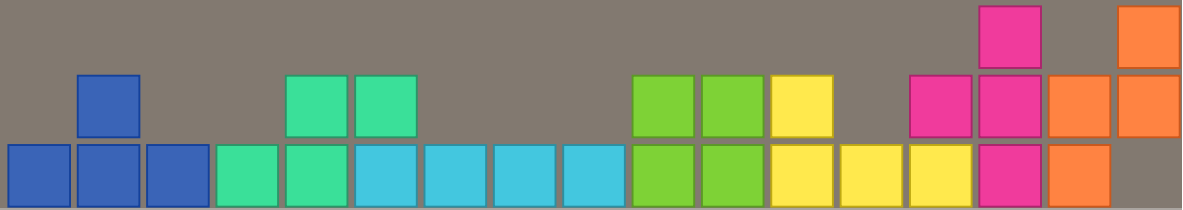
Kerberos, better than ever

For Windows 11, we are introducing two major features to Kerberos to expand when it can be used—addressing two of the biggest reasons why Kerberos falls back to NTLM today. The first, IAKerb, allows clients to authenticate with Kerberos in more diverse network topologies. The second, a local KDC for Kerberos, adds Kerberos support to local accounts.



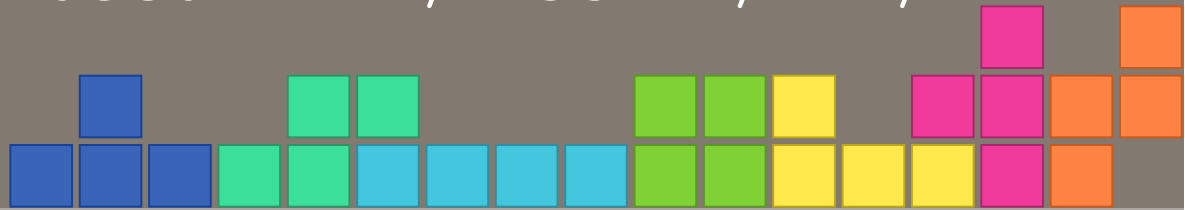
3

The future: Kerberos everywhere



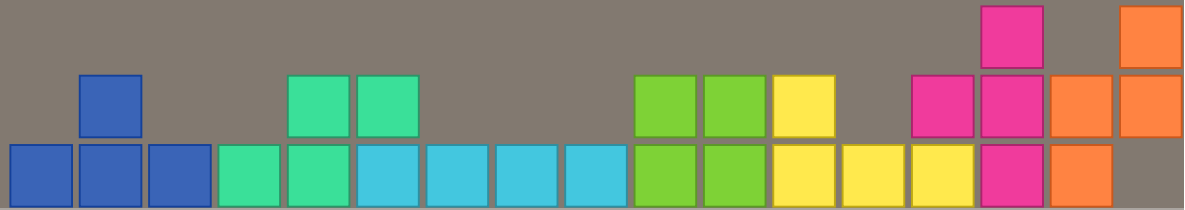
Why Kerberos?

- Mutual authentication - client and server verify each other
- Strong encryption - no password on the wire, only tickets
- Ticket-based - authenticate once, reuse ticket (Single Sign-On)
- Delegatable - service can act on behalf of a user
- Industry standard - used in AD, FreeIPA, MIT, Heimdal



The Missing Piece

- Kerberos requires **line-of-sight to a KDC**
  - Client must be able to reach the Key Distribution Center
- This is exactly the gap NTLM fills:
  - No DC reachable? NTLM takes over
  - Local accounts? No KDC at all
- **Solution: IAKerb** - proxy Kerberos through the service

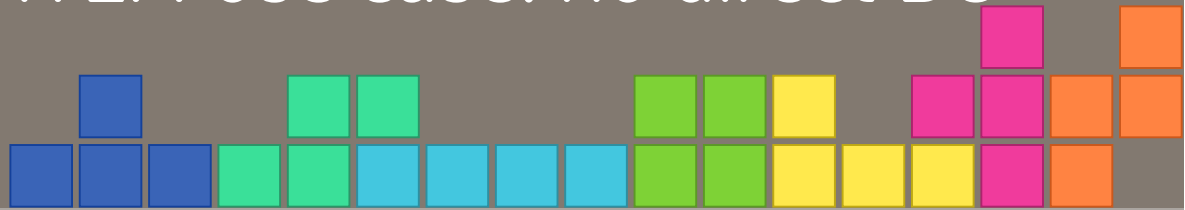


Initial and pass-through authentication using Kerberos (IAKerb)

- In 1997 MIT Kerberos folks designed a KRB5 proxy mechanism called IAKerb
  - Normally Kerberos requires three parties: a KDC, a client, and a service
  - What if the client can only have access to the service and not the KDC? (missing direct line of sight) => Solution: IAKerb

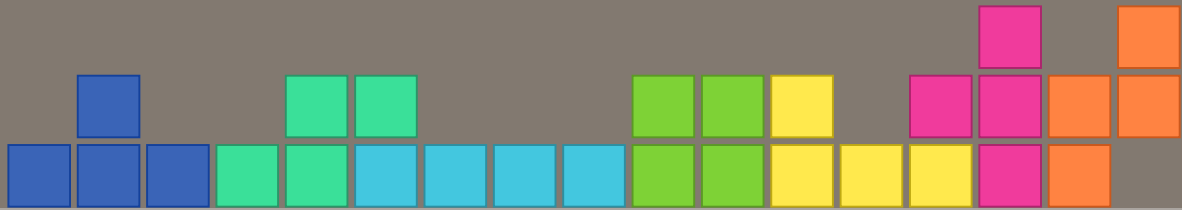
IAKerb in AD Domains

- In Active Directory, a service can act as a KDC proxy:
  - Client contacts the service → service forwards KRB-AS/TGS requests to the DC
  - Client gets a Kerberos ticket without ever directly reaching the DC
- Covers the classic NTLM use case: no direct DC line-of-sight



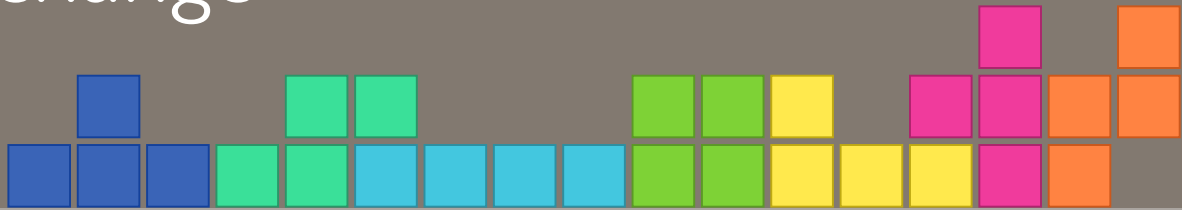
IAKerb Realm Discovery

- IAKerb has a way to ask the service for its **default realm**
- For a LocalKDC the realm is **random** - `LOCALKDC.<UUID>`
  - The client cannot know it in advance!



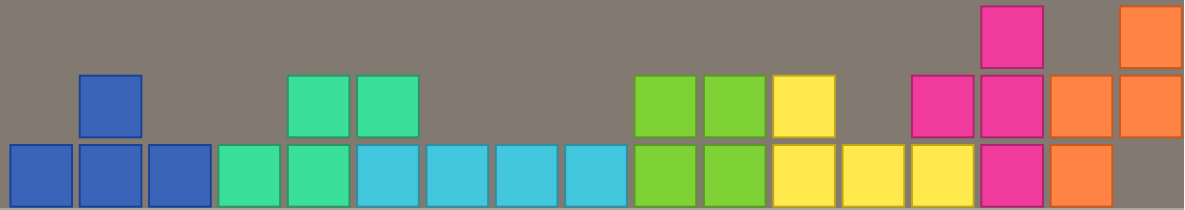
How does realm discovery work?

- Client sends an IAKerb token with an **empty realm**
- Service responds with a **realm referral** - its realm name and KDC info
- Client retries with the discovered realm; service proxies the KDC exchange



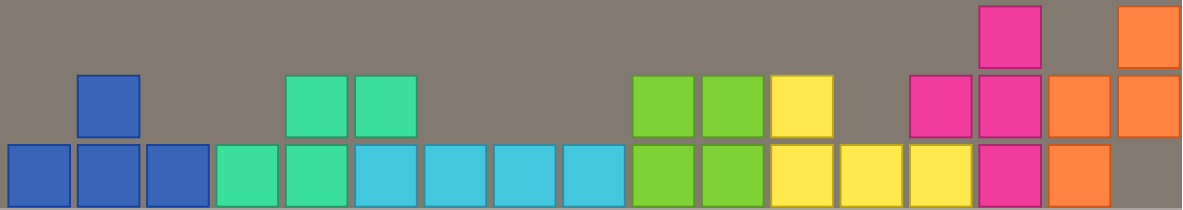
IAKerb in Kerberos implementations

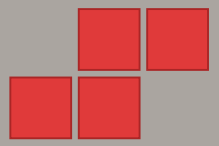
- MIT Kerberos
  - IAKerb draft spec support was updated to be on par with Microsoft
  - IAKerb realm discovery implemented
  - Available since version 1.22



IAKerb in Kerberos implementations

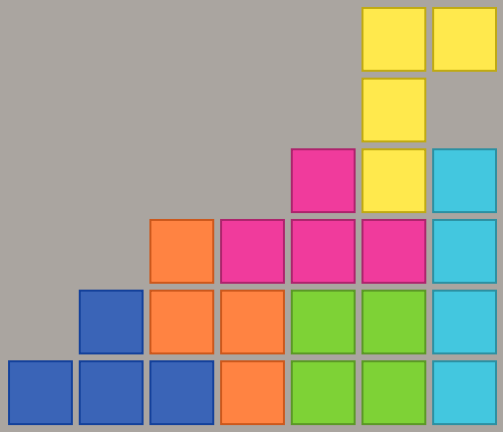
- Heimdal
  - no IAKerb support yet (despite Apple's integration)

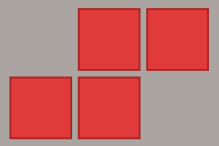




4

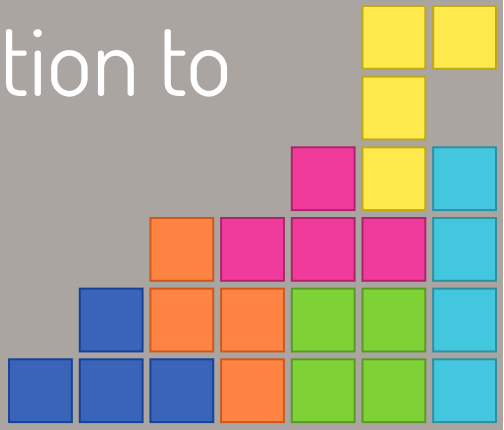
Standalone servers - the last NTLM holdout

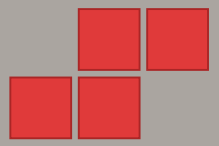




The Problem with Standalone Servers

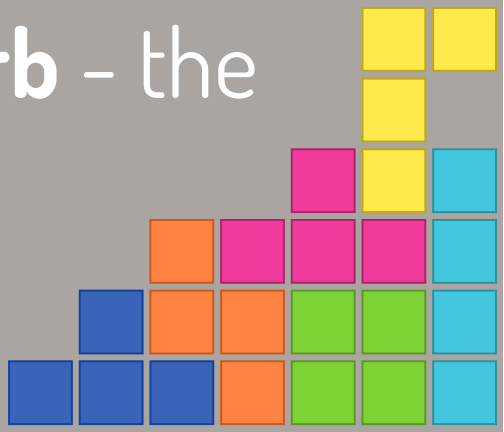
- Servers not joined to an AD domain have **no KDC**
- NTLM is the only authentication option for local accounts today
- Local user logs in → no Kerberos ticket → every service gets NTLM
- How do we bring Kerberos authentication to standalone machines?





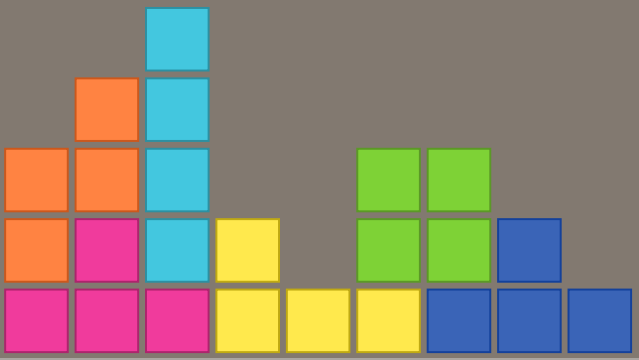
The Answer: A local KDC

- The answer is a Kerberos KDC only accessible over **local IPC**
  - Runs on the machine itself
  - runs on demand, not as a daemon
- Local services (Samba, SSH) talk to it directly
- Remote clients authenticate via **IAKerb** - the service proxies to the local KDC



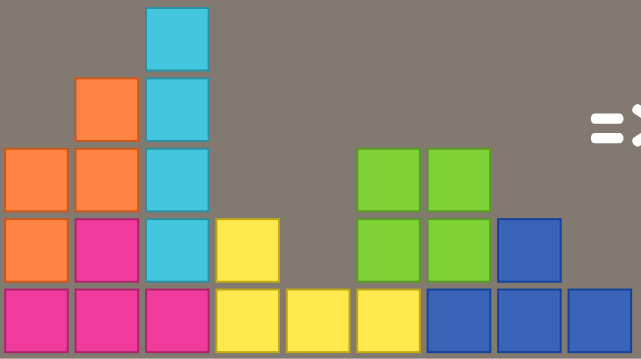
5

The LocalKDC



What do we need for a LocalKDC?

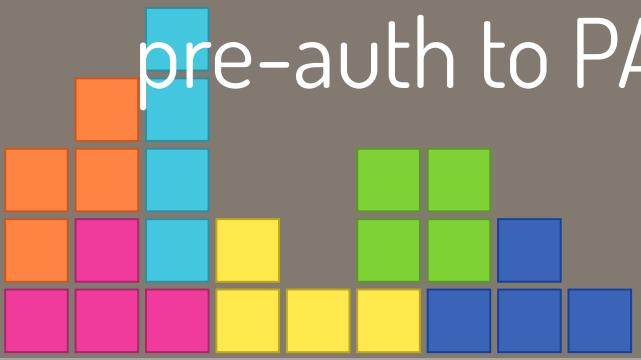
- A Kerberos implementation with:
  - Unix domain socket support and socket activation
  - Modules for DB, pre-auth, and auditing
  - IAKerb support



=> MIT Kerberos 1.22.1

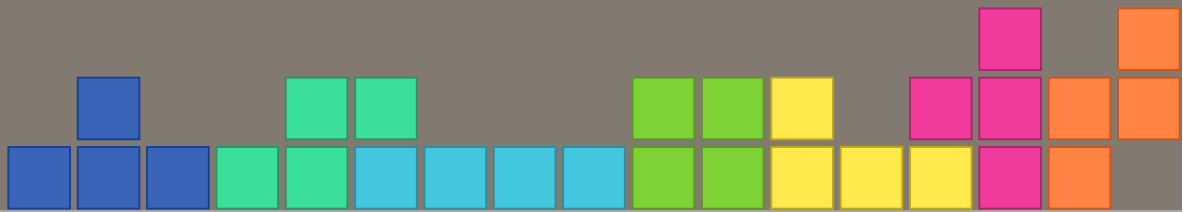
What else do we need?

- Management tools to set up and administer the local KDC
- KDC plugins (KDB, pre-auth, audit) to integrate with local accounts
- An authentication mechanism to bridge Kerberos pre-auth to PAM



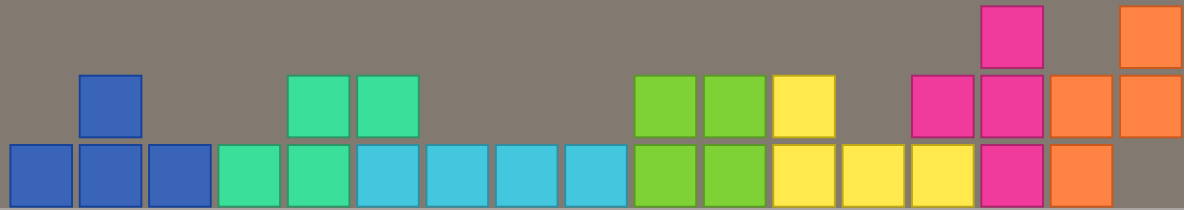
6

The Kirmes Project

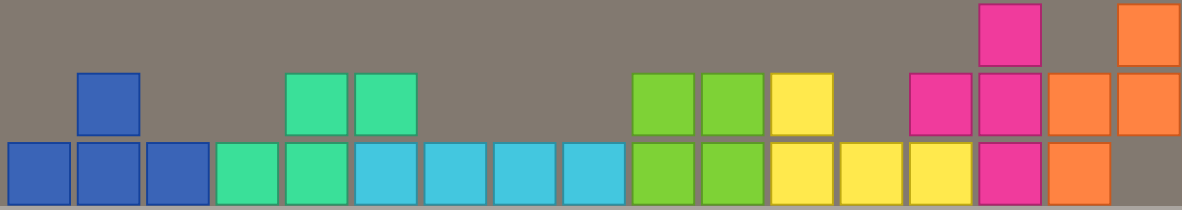


Management tools - lkdcctl

- `lkdcctl setup` - One-time setup of the local KDC
  - Generates realm `LOCALKDC.<UUID>`, installs KDC config
  - Requests PKINIT certificates via certmonger
  - Creates KDB, service principals, and keytabs
  - Configures PAM, Samba, SSSD; enables systemd units

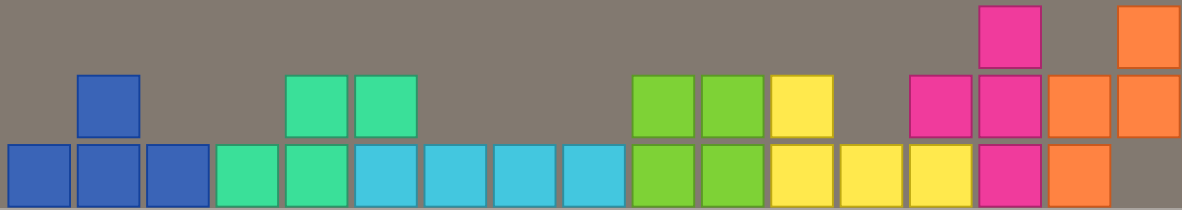


Demo: Setup and service start



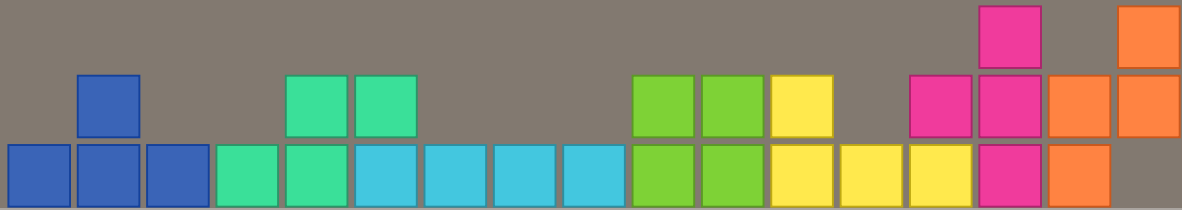
Management tools - lkdcctl

- `lkdcctl kadmin` - Direct access to kadmin.local
- `lkdcctl kinit <user>` - Obtains a Kerberos ticket for a local user



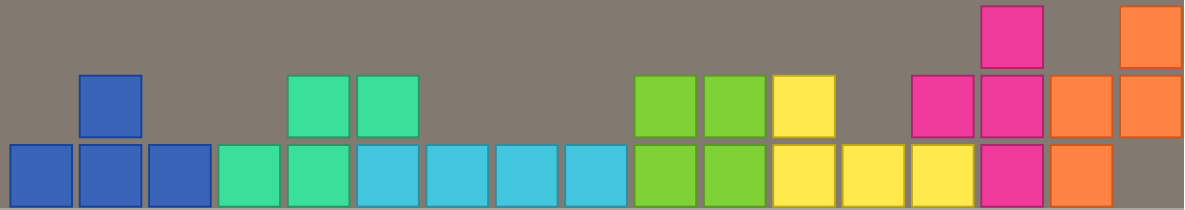
KDC plugins in Rust!

- **kurbu5** - library for writing MIT KDC plugins in Rust
  - Safe, modern systems language for security-critical plugin code



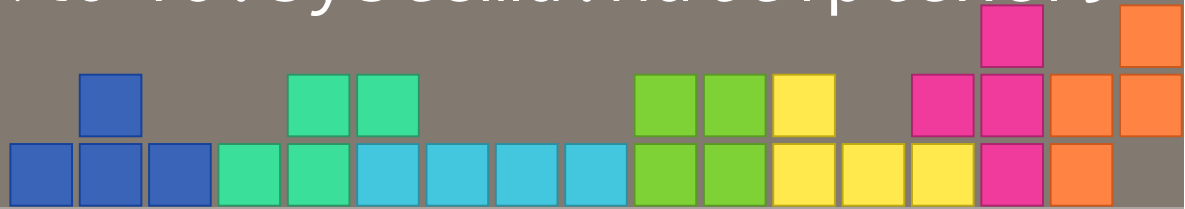
Kirmes KDC plugins

- Three plugins implemented:
  - **KDB plugin** - database backend integrating systemd-userdb
  - **Pre-auth plugin** - OTP pre-authentication (RFC 6560)
  - **Audit plugin** - JSON audit logging of all KDC events



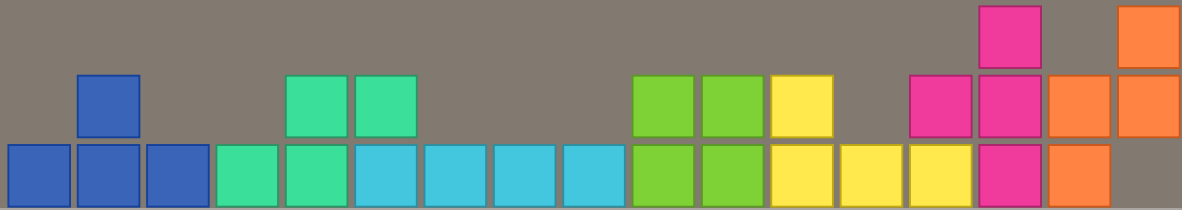
KDB plugin - `kdb_userdb`

- Overlay plugin wrapping the **klmdb** backend
- Principal lookup strategy:
  - Try klmdb first (persistent KDB storage)
  - Fall back to **service alias resolution** (DNS-based hostname lookup for service principals)
  - Fall back to **systemd-userdb** (via libkirmes/varlink to `io.systemd.Multiplexer`)



KDB plugin - `kdb_userdb`

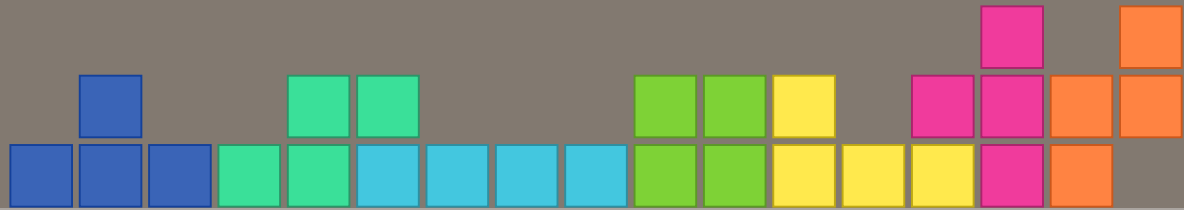
- Synthesizes Kerberos principal entries on-the-fly from userdb records
- Injects OTP pre-auth requirement for user principals



libkirmes - systemd-userdb client library

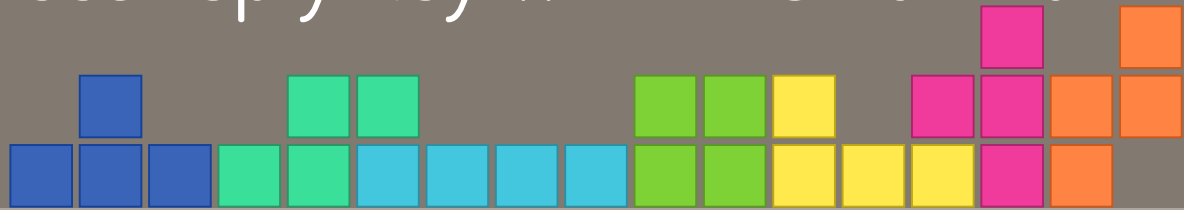
- Provides an async Rust and C API
- Makes it simple to talk to systemd-userdb over Varlink =>

  `/run/systemd/userdb/io.systemd.Multiplexer`
- https://docs.rs/kirmes/



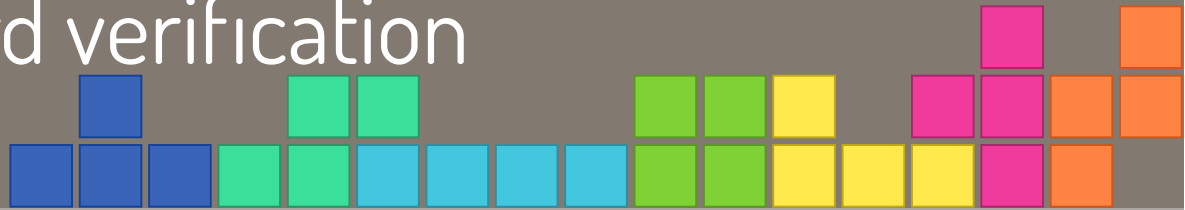
Pre-auth plugin - OTP

- Implements **RFC 6560** - OTP pre-authentication
- Authentication flow:
  - `kinit` -> FAST armor (anonymous PKINIT) -> OTP challenge
  - Client collects password, builds OTP request
  - KDC verifies via **RADIUS over Unix domain socket**
  - On success, replaces reply key with FAST armor key

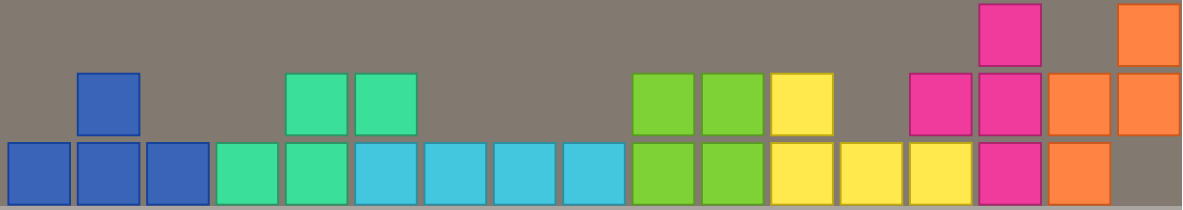


PAM RADIUS tool - localkdc-pam-auth

- Unix-socket RADIUS server bridging KDC pre-auth to **PAM**
- systemd socket-activated (per-connection instances)
- Authenticates via PAM service (default: `system-auth`)
- Completes the chain: KDC OTP -> RADIUS -> PAM -> system password verification



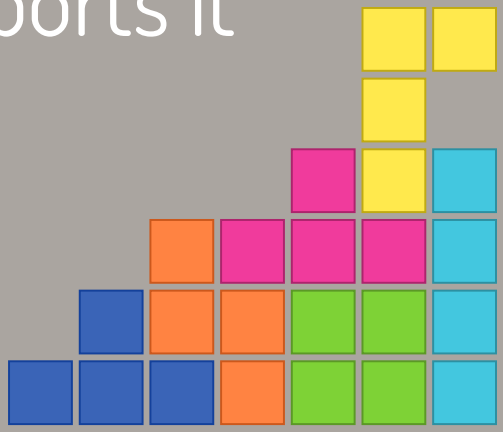
Demo: Local Authentication



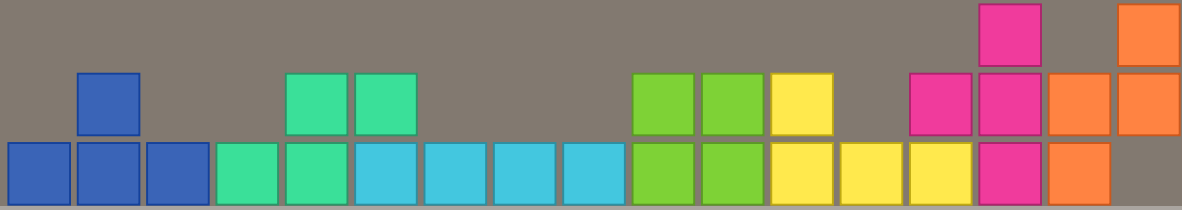
7

Samba and IAKerb

- We have an initial implementation in Samba
- File server and clients already supports it

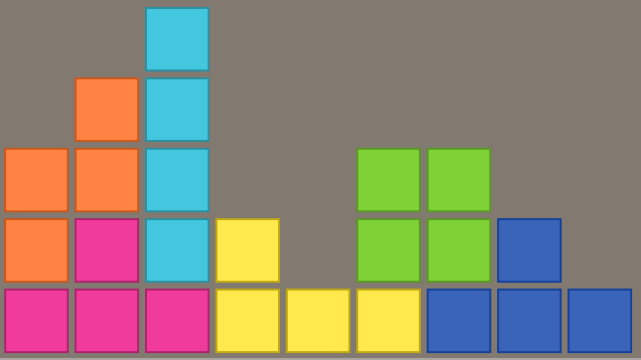


Demo: Kinit over SMB



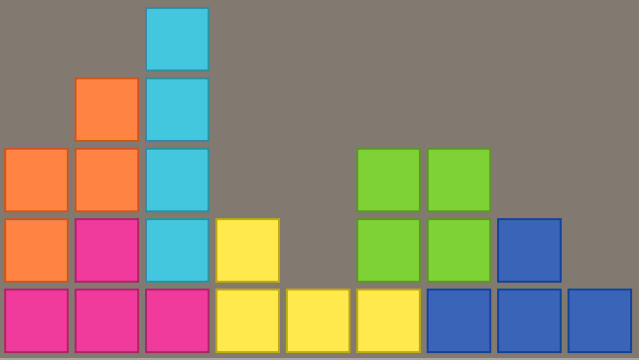
8

Future work



Future work in Kerberos

- Async GSSAPI for S4U extensions MIT PR#1500
- Async Anonymous PKINIT

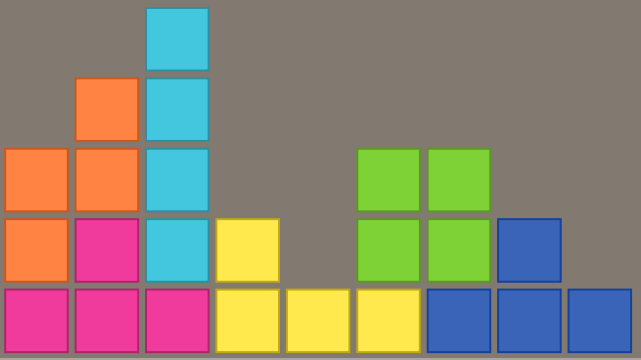


Future work Samba

- cifs.ko/smb3.ko IAKerb support
- Windows interop testing as soon as we there is a build enabling it
- smbpasswd database (passdb) - migration or read it directly?

Future work LocalKDC

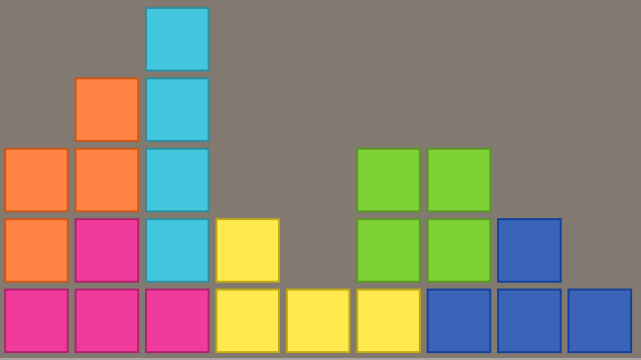
- Password changes through Kerberos
- Multifactor authentication
- PAM stack optimizations
- MS-PAC support (PIDL in Rust)



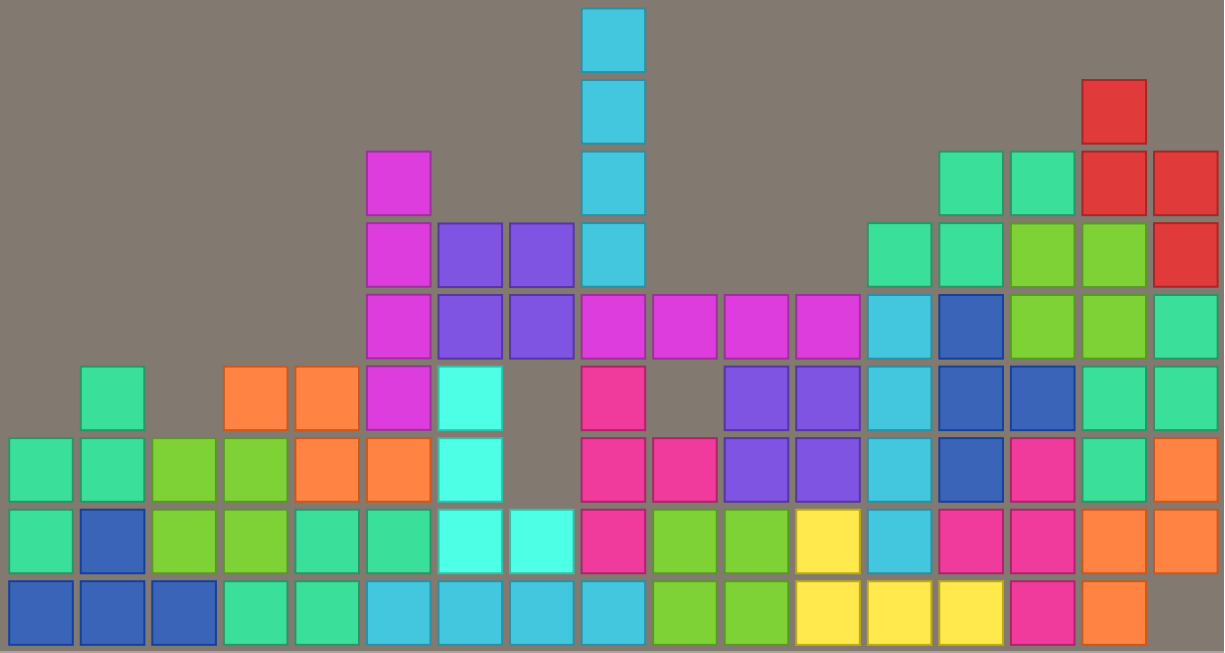
9

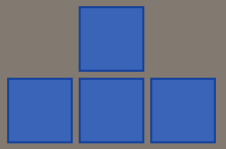
Lonterm future work

We want to open windows to different worlds. One idea is making OAuth2 usable on the desktop.



GAME OVER





Questions?

Mastodon (Alexander): @abbra:mastodon.social

Mastodon (Andreas): @cryptomilk:mastodon.social

Blog (Alexander): vda.li/en/ Blog (Andreas): blog.cryptomilk.org