



GLOBAL CATALOG SERVICE IMPLEMENTATION IN FREEIPA

Alexander Bokovoy

Red Hat Inc.

May 4th, 2017

ABOUT:ME

Sr. Principal Software Engineer at Red Hat

- Samba Team member since 2003
- Core FreeIPA developer since 2011

WHAT IS THIS TALK ABOUT?

WHAT IS THIS ABOUT?

FreeIPA is a directory service for Linux and POSIX clients:

- 389-ds - The LDAP directory server (and a lot of plugins)
- Samba - as a traditional (NT4-style) domain controller with a twist (`smbd` and `winbindd`)
- MIT Kerberos - Kerberos KDC
- MS-KKDCP proxy
- Dogtag Certificate Authority
- Custodia (secrets management)
- SSSD - client side identity (nss, PAM, D-Bus, ...)
- FreeIPA management framework written in Python and running under Apache

WHAT IS THIS ABOUT?

FreeIPA supports forest trust to Active Directory:

- Active Directory sees FreeIPA as a "native Active Directory" deployment
- Since Samba 4.5 it is possible to establish a trust between Samba AD and FreeIPA
- Active Directory users can access resources on FreeIPA clients
- FreeIPA users cannot natively access resources in Active Directory

WHAT IS THIS ABOUT?

FreeIPA users cannot access resources in Active Directory:

- Access control in Active Directory uses SIDs of users/groups in ACLs
- "Security" tab in UI deals with user and group names
- Windows performs user or group SID lookup
- **FreeIPA does not provide interfaces expected by Active Directory to perform name to SID lookups**

ANATOMY OF A NAME RESOLUTION

FOUR STYLES OF CONVERSATION

Active Directory has four ways of discovering SIDs of users/groups:

- Domain controller LDAP ping allows user name validation ([MS-ADTS] 6.3.3.2 Domain Controller Response to an LDAP Ping) but doesn't allow to discover SIDs
- DsCrackNames is part of DRSU API, Directory Replication Service of Active Directory. It is not implemented in `smbd`, only in Samba AD
- LsaLookupNames and SamLogon families of RPC calls
- LDAP queries to Global Catalog

DOMAIN CONTROLLER LOCATOR REQUEST

LDAP ping is used by all Windows clients to discover closest domain controller. As part of it clients may request a user name validation to avoid hitting domain controllers that don't have that user replicated

DOMAIN CONTROLLER LOCATOR REQUEST

Windows UI seems to trigger CLDAP ping requests with random user names instead of the one you entered:

The screenshot displays a Wireshark capture of LDAP traffic. The left pane shows a list of packets, including search requests and responses. The right pane shows the details of a selected search request (packet 392), including filters for domain and host names. Below the details pane is a hex dump of the packet data.

Protocol	Length	Info
eb77	CLDAP	392 searchRequest(389) "cR00T*" baseObject
:1a9e	CLDAP	223 searchResEntry(389) "cR00T*" searchResDone(389) succes
eb77	CLDAP	392 searchRequest(390) "cR00T*" baseObject
:1a9e	CLDAP	223 searchResEntry(390) "cR00T*" searchResDone(390) succes
eb77	CLDAP	392 searchRequest(391) "cR00T*" baseObject
:1a9e	CLDAP	223 searchResEntry(391) "cR00T*" searchResDone(391) succes
eb77	CLDAP	392 searchRequest(392) "cR00T*" baseObject
:1a9e	CLDAP	223 searchResEntry(392) "cR00T*" searchResDone(392) succes
eb77	CLDAP	392 searchRequest(393) "cR00T*" baseObject
:1a9e	CLDAP	223 searchResEntry(393) "cR00T*" searchResDone(393) succes
eb77	CLDAP	392 searchRequest(394) "cR00T*" baseObject
:1a9e	CLDAP	223 searchResEntry(394) "cR00T*" searchResDone(394) succes
eb77	CLDAP	392 searchRequest(395) "cR00T*" baseObject
:1a9e	CLDAP	223 searchResEntry(395) "cR00T*" searchResDone(395) succes
eb77	CLDAP	392 searchRequest(396) "cR00T*" baseObject
:1a9e	CLDAP	223 searchResEntry(396) "cR00T*" searchResDone(396) succes
eb77	CLDAP	392 searchRequest(397) "cR00T*" baseObject
:1a9e	CLDAP	223 searchResEntry(397) "cR00T*" searchResDone(397) succes
eb77	CLDAP	392 searchRequest(398) "cR00T*" baseObject
:1a9e	CLDAP	223 searchResEntry(398) "cR00T*" searchResDone(398) succes
eb77	CLDAP	392 searchRequest(399) "cR00T*" baseObject
:1a9e	CLDAP	223 searchResEntry(399) "cR00T*" searchResDone(399) succes
eb77	CLDAP	392 searchRequest(400) "cR00T*" baseObject
:1a9e	CLDAP	223 searchResEntry(400) "cR00T*" searchResDone(400) succes

```
> and: {&{&{&{&{&{DnsDomain=xs.ipa.cool}|(Host=METIS)}|(User=ADMIN)}|(AAC=10:00:00:00)}|(DomainSid=S-1-5-21-570121326-3336757064-115732047 (Domain SID))}
  > and: 7 items
  > Filter: {DnsDomain=xs.ipa.cool}
    > and item: equalityMatch (3)
      > equalityMatch
  > Filter: {Host=METIS}
    > and item: equalityMatch (3)
      > equalityMatch
  > Filter: {User=ADMIN}
    > and item: equalityMatch (3)
      > equalityMatch
  > Filter: {AAC=10:00:00:00}
    > and item: equalityMatch (3)
      > equalityMatch
  > Filter: {DomainSid=S-1-5-21-570121326-3336757064-115732047 (Domain SID)}
    > and item: equalityMatch (3)
      > equalityMatch
  > Filter: {NTLDR=0x01000016}
    > and item: equalityMatch (3)
      > equalityMatch
  > Filter: {DnsHostName=metis.ad.ipa.cool}
    > and item: equalityMatch (3)
```

```
0000 00 1a 4a 62 eb 77 5c 5e ab 7c ea 81 86 dd 60 00  ..Jb.w^..|.....
0010 00 00 00 f8 11 3d 2a 02 27 70 00 17 06 00 02 1a  ....*..p.....
0020 4a ff fe 05 1a 9e 25 02 27 70 00 09 00 00 02 1a  J...*.p.....
0030 4a ff fe 62 eb 77 e4 06 01 85 00 f8 64 2d 30 84  J..b.w.....d-0.
0040 00 00 00 ea 02 02 01 86 63 84 00 00 00 ea 04 00  ....C.....
0050 0a 01 00 0a 01 00 02 01 00 02 01 00 01 01 00 00  .....Dns
0060 84 00 00 00 b9 a3 84 00 00 00 18 04 09 44 6e 73  .....
0070 44 6f 6d 61 69 6e 04 0b 78 73 2e 69 70 61 2e 63  Domain..xs.ipa.c
0080 6f 6f 6c a3 84 00 00 00 04 04 48 6f 73 74 04  col.....Host.
0090 05 4d 45 54 49 53 a3 84 00 00 00 0d 04 04 55 73  .METIS.....Us
00a0 65 72 04 05 41 44 4d 49 4e a3 84 00 00 00 0b 04  er..ADMIN.....
00b0 03 41 41 43 04 04 10 00 00 00 a3 84 00 00 00 25  .AAC.....%
00c0 04 09 44 6f 6d 61 69 6e 53 69 64 04 18 01 04 00  ..Domain Sid....
00d0 00 00 00 00 05 15 00 00 00 6e 5c fb 21 48 df e2  .....n\..M..
00e0 c6 4f 7c fb 44 a3 84 00 00 00 04 05 4e 74 56  .01.D.....NTV
00f0 65 72 04 16 00 00 01 83 84 00 00 20 04 00 00  ..et.....
0100 44 6e 73 48 6f 73 74 4e 61 6d 65 04 11 6d 65 74  DnsHostN ame..met
0110 69 73 2e 61 64 2e 69 70 61 2e 63 6f 6f 6c 30 84  is.ad.ip a.cool0.
0120 00 00 00 0a 04 08 4e 65 74 6c 6f 6f 6e 6e  ....Ne tlogon
```



WHO IS ALICE?

DOMAIN CONTROLLER LOCATOR REQUEST

Alice is a fine random name, along with Marvin, Heather, Student, User2, Test, and many others seen on the wire

The state of user inquiries in Windows is ... interesting

GLOBAL CATALOG SERVICE

If Global Catalog service is available, Windows will attempt to connect to it:

The screenshot displays a network traffic capture in Wireshark. The left pane shows a list of captured packets, including LDAP search requests and responses, and NBSS continuation messages. The right pane shows a detailed view of an LDAP message (packet 62:eb77) with the following structure:

- name-string: 2 items
 - SNameString: krbtgt
 - SNameString: XS.IPA.COOL
- enc-part
- authenticator
 - etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
 - cipher: bfbed6f2c28f01f632810862bc9f42ebca06aa555aab0e...
- req-body
 - padding: 0
 - kdc-options: 46810000 (forwardable, renewable, canonicalize)
 - realm: XS.IPA.COOL
 - sname
 - name-type: KRBS-NT-SRV-INST (2)
 - sname-string: 3 items
 - SNameString: ldap
 - SNameString: nyx.xs.ipa.cool
 - SNameString: xs.ipa.cool
 - till: 2037-09-13 02:48:05 (UTC)
 - nonce: 1668017033
 - etype: 5 items
 - enc-authorization-data
 - etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)

GLOBAL CATALOG SERVICE SEARCH RESULTS

If search in a Global Catalog returns no results, Windows falls back to `netr_LogonSamLogonWithFlags` RPC call:

The screenshot displays a Wireshark capture of network traffic. The left pane shows the packet list with the following entries:

- 62:eb77 TCP 74 3268 → 58826 [FIN, ACK] Seq=1955 Ack=206 Win=2167904 Len=0
- ee5:1a9e TCP 74 3268 → 58826 [RST] Seq=206 Win=0 Len=0
- 62:eb77 TCP 86 58827 → 3268 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=1440 MS=256 S
- ee5:1a9e TCP 86 3268 → 58827 [SYN, ACK] Seq=0 Ack=1 Win=28800 Len=0 MSS=1440 SACK.P
- 62:eb77 TCP 74 58827 → 3268 [ACK] Seq=1 Ack=1 Win=263424 Len=0
- 62:eb77 LDAP 1755 bindRequest(47) "c-ROOT*" sasl
- ee5:1a9e TCP 74 3268 → 58827 [ACK] Seq=1 Ack=1682 Win=32256 Len=0
- ee5:1a9e LDAP 278 bindResponse(47) saslBindInProgress
- 62:eb77 LDAP 271 SASL GSS-API Integrity:
- ee5:1a9e TCP 74 3268 → 58827 [FIN, ACK] Seq=205 Ack=1879 Win=35872 Len=0
- 62:eb77 TCP 74 58827 → 3268 [ACK] Seq=1879 Ack=206 Win=263168 Len=0
- 62:eb77 TCP 74 58827 → 3268 [FIN, ACK] Seq=1879 Ack=206 Win=263168 Len=0
- ee5:1a9e TCP 74 3268 → 58827 [ACK] Seq=206 Ack=1880 Win=35872 Len=0
- 62:eb77 TCP 86 58828 → 3268 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=1440 MS=256 S
- ee5:1a9e TCP 86 3268 → 58828 [SYN, ACK] Seq=0 Ack=1 Win=28800 Len=0 MSS=1440 SACK.P
- 62:eb77 TCP 74 58828 → 3268 [ACK] Seq=1 Ack=1 Win=263424 Len=0
- 62:eb77 LDAP 1755 bindRequest(51) "c-ROOT*" sasl
- ee5:1a9e TCP 74 3268 → 58828 [ACK] Seq=1 Ack=1682 Win=32256 Len=0
- ee5:1a9e LDAP 278 bindResponse(51) saslBindInProgress
- 62:eb77 LDAP 271 SASL GSS-API Integrity:
- ee5:1a9e TCP 74 3268 → 58828 [FIN, ACK] Seq=205 Ack=1879 Win=35872 Len=0
- 62:eb77 TCP 74 58828 → 3268 [ACK] Seq=1879 Ack=206 Win=263168 Len=0
- 62:eb77 LDAP 150 SASL GSS-API Integrity:
- ee5:1a9e TCP 74 3268 → 58828 [RST] Seq=206 Win=0 Len=0
- CLDAP 281 searchRequest(3492) "c-ROOT*" baseObject
- CLDAP 288 searchResEntry(3492) "c-ROOT*" searchResDone(3492) success [1 resul
- 62:eb77 TCP 86 58829 → 1824 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=1440 MS=256 S
- ee5:1a9e TCP 86 1824 → 58829 [SYN, ACK] Seq=0 Ack=1 Win=28800 Len=0 MSS=1440 SACK.P
- 62:eb77 TCP 74 58829 → 1824 [ACK] Seq=1 Ack=1 Win=263424 Len=0
- ee5:1a9e TCP 74 1824 → 58829 [ACK] Seq=1 Ack=206 Win=29952 Len=0
- ee5:1a9e DCERPC 166 Bind_ack: call_id: 109, Fragment: Single, max_xmit: 4280 max_recv:
- 62:eb77 RPC_NETU 690 NetrLogonSamLogonWithFlags request
- ee5:1a9e RPC_NETU 498 NetrLogonSamLogonWithFlags response
- 62:eb77 TCP 74 58829 → 1824 [ACK] Seq=822 Ack=517 Win=262912 Len=0
- CLDAP 282 searchRequest(3493) "c-ROOT*" baseObject
- CLDAP 288 searchResEntry(3493) "c-ROOT*" searchResDone(3493) success [1 resul

The right pane shows the details of the selected packet (62:eb77), which is a NetrLogonSamLogonWithFlags request. The details include:

- Frag Length: 205
- Auth Length: 37
- Call ID: 109
- Max Xmit Frag: 5840
- Max Recv Frag: 5840
- Assoc Group: 0x00000000
- Num Ctx Items: 3
- Ctx Item[1]: Context ID:0, RPC_NETLOGON, 32bit NDR
- Ctx Item[2]: Context ID:1, RPC_NETLOGON, 64bit NDR
- Ctx Item[3]: Context ID:2, RPC_NETLOGON, Bind Time Feature Negotiation
- Auth type: NETLOGON Secure Channel (08)
- Auth level: Packet privacy (6)
- Auth pad len: 0
- Auth Rsvrd: 0
- Auth Context ID: 0
- Secure Channel _AUTH_MESSAGE
- Message Type: Request (0x00000000)
- Message Flags: 0x00000017, NetBios Domain, NetBios Host, DNS Domain, NetBios Host(UTF8)
- NetBios Domain: XS
- NetBios Host: METIS
- DNS Domain: xs.ipa.cool
- NetBios Host(UTF8): METIS

The packet bytes pane shows the raw data of the request, including the NetBios Host name 'METIS' and the DNS Domain 'xs.ipa.cool'.

NETLOGON RESULTS

But the name passed to `netr_LogonSamLogonWithFlags` is totally different from what is entered in Windows UI

```
[2017/05/03 13:58:05.818919, 5, pid=17774, effective(0, 0), real(0, 0)] ../source3/lib/smbldap.c:1249(smbld
smbldap_search_ext: base => [dc=xs,dc=ipa,dc=cool], filter => [(&(objectClass=ipaNTUserAttrs)(uid=USER))],
[2017/05/03 13:58:05.858058, 4, pid=17774, effective(99, 99), real(99, 0)] ../source3/smbd/sec_ctx.c:439(po
pop_sec_ctx (99, 99) - sec_ctx_stack_ndx = 1
[2017/05/03 13:58:05.858094, 3, pid=17774, effective(99, 99), real(99, 0), class=auth] ../source3/auth/chech
check_sam_security: Couldn't find user 'USER' in passdb.
[2017/05/03 13:58:05.858119, 5, pid=17774, effective(99, 99), real(99, 0), class=auth] ../source3/auth/auth
check_ntlm_password: sam authentication for user [USER] FAILED with error NT_STATUS_NO_SUCH_USER
[2017/05/03 13:58:05.858171, 10, pid=17774, effective(99, 99), real(99, 0), class=auth] ../source3/auth/auth
Check auth for: [USER]
[2017/05/03 13:58:05.858185, 3, pid=17774, effective(99, 99), real(99, 0), class=auth] ../source3/auth/auth
check_winbind_security: Not using winbind, requested domain [XS] was for this SAM.
[2017/05/03 13:58:05.858195, 10, pid=17774, effective(99, 99), real(99, 0), class=auth] ../source3/auth/auth
check_ntlm_password: winbind had nothing to say
[2017/05/03 13:58:05.858204, 2, pid=17774, effective(99, 99), real(99, 0), class=auth] ../source3/auth/auth
check_ntlm_password: Authentication for user [USER] -> [USER] FAILED with error NT_STATUS_NO_SUCH_USER
[2017/05/03 13:58:05.858217, 5, pid=17774, effective(99, 99), real(99, 0), class=rpc_srv] ../source3/rpc_se
netr_LogonSamLogonWithFlags: check password returned status NT_STATUS_NO_SUCH_USER
[2017/05/03 13:58:05.858229, 4, pid=17774, effective(99, 99), real(99, 0)] ../source3/smbd/sec_ctx.c:217(pu
```

NAME RESOLUTION ORDER

NAME RESOLUTION ORDER

- CLDAP ping is an important operation but actual name resolution is done by querying Global Catalog
- If Global Catalog available, Windows will try to use that
- If search in Global Catalog does return no results, Windows will fall back to RPC calls
- **FreeIPA does not provide a Global Catalog service**
- As result, Windows does not even try to fall back to RPC calls

FREEIPA CHALLENGES

SASL GSS-SPNEGO

Samba implements own SASL code, FreeIPA components rely on Cyrus-SASL

- Cyrus-SASL GSS-SPNEGO implementation is compatible with itself, not Windows
- GSS-SPNEGO negotiates SSF based on GSSAPI flags, not separately
- This was fixed by Simo Sorce in February 2017:
<https://github.com/cyrusimap/cyrus-sasl/commit/67ca66685e11acc0f69d5ff8013107d4b172e67f>
- No Cyrus-SASL release with the fix yet but Fedora 26 has it backported

KERBEROS TARGET PRINCIPALS

Windows TGS-REQ requests use three-component principal names:

- `ldap/host.example.com/example.com@EXAMPLE.COM`
- **Real service name is** `ldap/host.example.com@EXAMPLE.COM`
- **FreeIPA 4.4+ added support for Kerberos principal aliases:**
`ipa service-add-principal ldap/host.example.com@EXAMPLE.COM`
`ldap/host.example.com/example.com@EXAMPLE.COM`

LDAP SASL BIND MAPPING

Windows always uses SASL GSS-SPNEGO for LDAP bind authentication

- Successful authentication means LDAP server needs to map authenticated identity to existing LDAP object
- There are no users or machines accounts from a trusted Active Directory in FreeIPA LDAP store
- Luckily, Global Catalog access for out-of-domain accounts is read-only
- We can map all authenticated but unknown identities to a single LDAP object with read-only rights

LDAP SCHEMA

- FreeIPA has its own LDAP schema and LDAP tree structure
- Active Directory LDAP schema is not compatible with FreeIPA LDAP schema
- Attributes and objects can be re-mapped but direct access is useless

389-DS LIMITATIONS

- 389-ds LDAP server only allows to listen on a single port per protocol
- TCP/389 for LDAP, TCP/636 for LDAPS
- Global Catalog is always TCP/3268 for LDAP access

FREEIPA GLOBAL CATALOG SERVICE

GLOBAL CATALOG SERVICE

- Runs as a separate 389-ds instance to serve port TCP/3268
- Transforms user and group data from primary FreeIPA LDAP instance to AD schema
- Access is read-only with SASL GSS-SPNEGO authentication

DATA TRANSFORMATION

- LDAP SYNCREPL is used to pick up changes from the primary FreeIPA LDAP instance running on the same IPA master
- Schema Compatibility plugin code is used to transform the changes to AD-compatible schema and DIT
- <https://pagure.io/slapi-nis/>

CURRENT STATE

- Design documents are available at https://www.freeipa.org/page/V4/Global_Catalog_Support
- SYNCREPL plugin almost ready
- Schema Compatibility plugin refactoring has started
- We plan to have working prototype ready for Redmond IO Lab in June 2017



Questions & Answers

THANK YOU

<https://samba.org/>

<https://freeipa.org/>