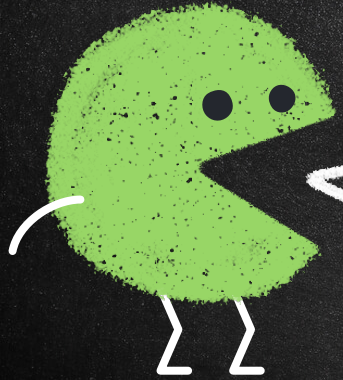
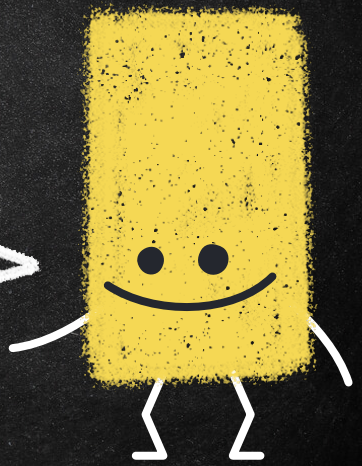
Lock Picking the macOS keychain



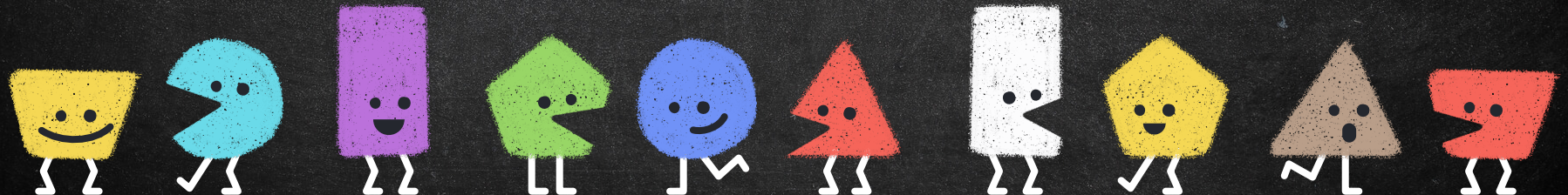


OBTS v5 Oct 2022

Whoami?

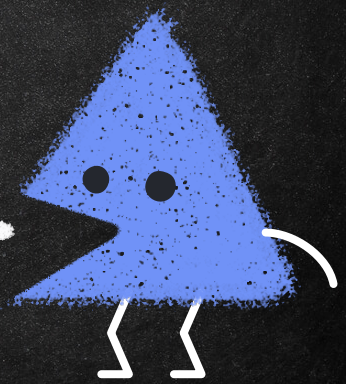
Cody Thomas @its_a_feature_

- ➜ Sr. Software Dev. at SpecterOps
- ➜ Created Mythic C2 Framework
- ➜ macOS Red Teamer
- ➜ Instructor for Mac Tradecraft
- ➜ Love-hate relationship with JXA



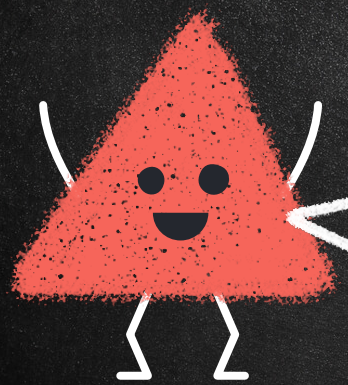
Overview

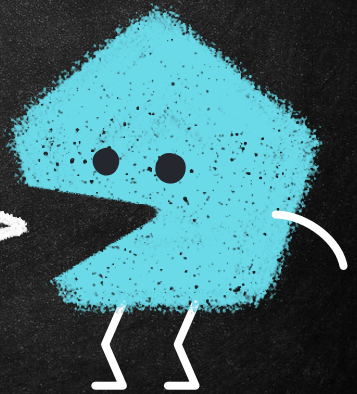
- ➜ What is the macOS Keychain
- ➜ Why go after the macOS Keychain
- ➜ What are Keychain Items
- ➜ Accessing the Keychain
- ➜ 3 Keychain Goodies



1.

What is the macOS Keychain?





API Collection

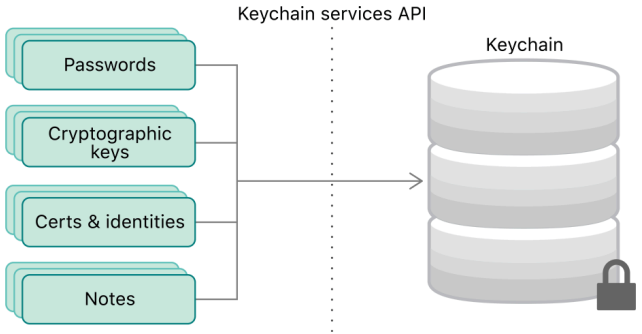
Keychain Services

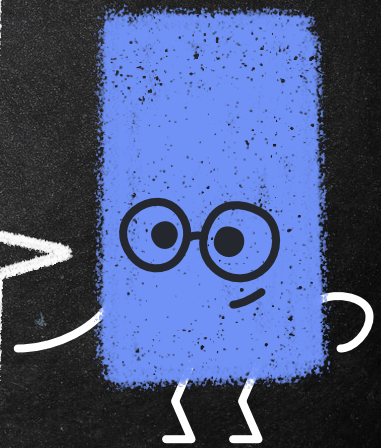
Securely store small chunks of data on behalf of the user.

Overview

Computer users often have small secrets that they need to store securely. For example, most people manage numerous online accounts. Remembering a complex, unique password for each is impossible, but writing them down is both insecure and tedious. Users typically respond to this situation by recycling simple passwords across many accounts, which is also insecure.

The keychain services API helps you solve this problem by giving your app a mechanism to store small bits of user data in an encrypted database called a keychain. When you securely remember the password for them, you free the user to choose a complicated one.





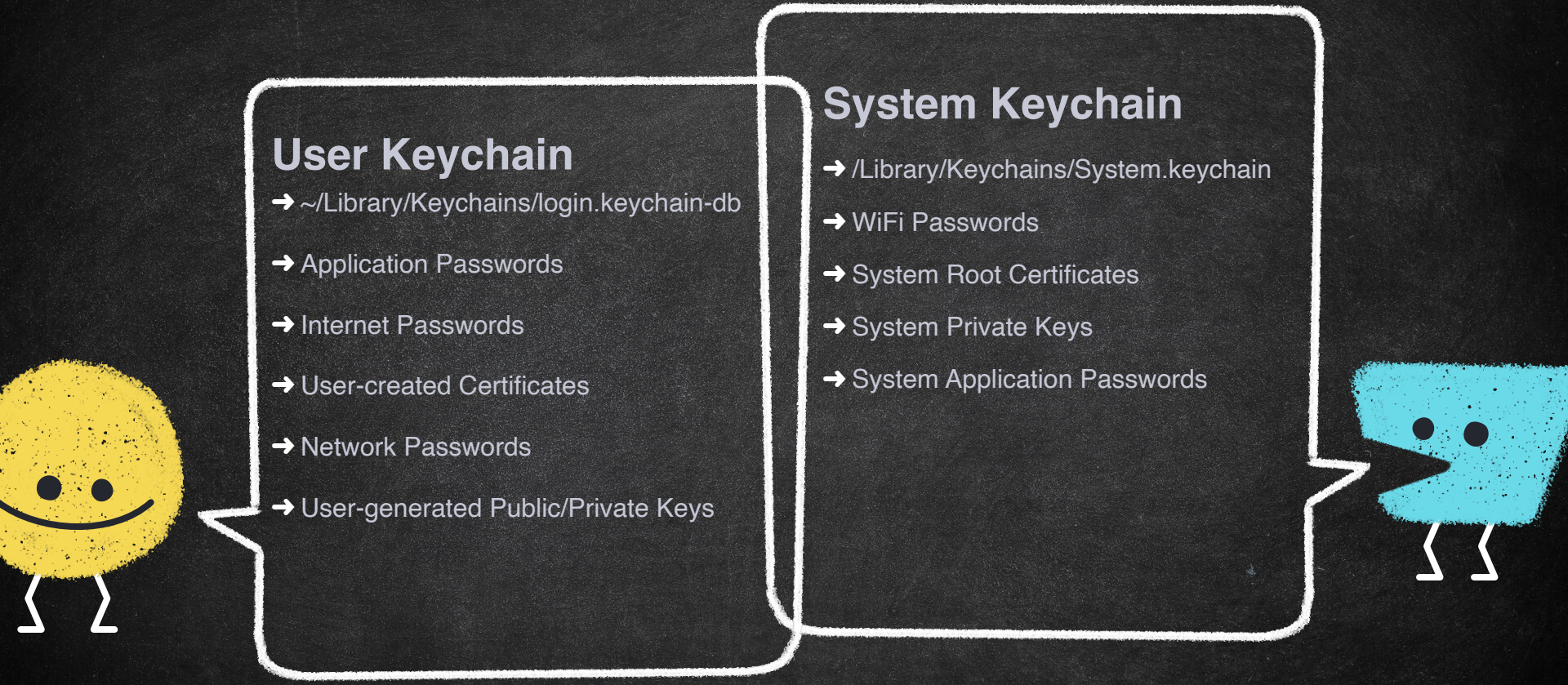
https://developer.apple.com/documentation/security/keychain_services

Keychain Types

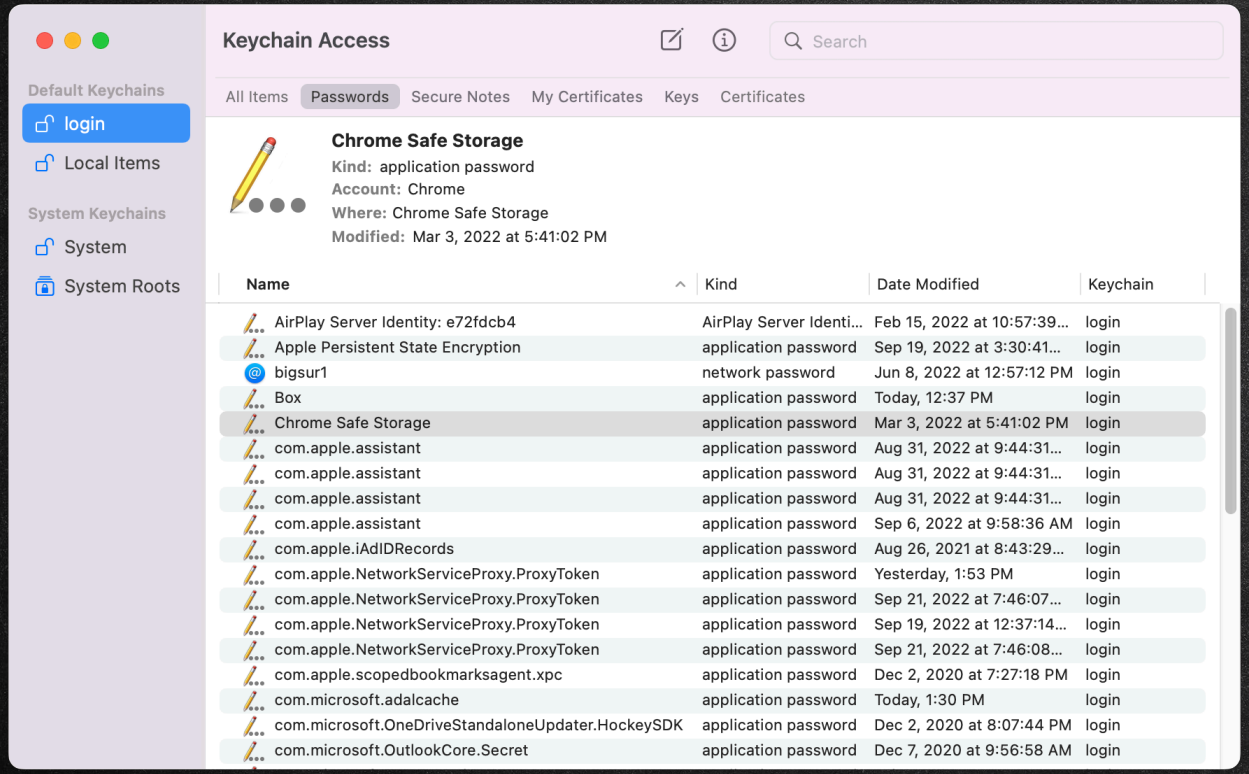
➜Two different “kinds” of Keychains:

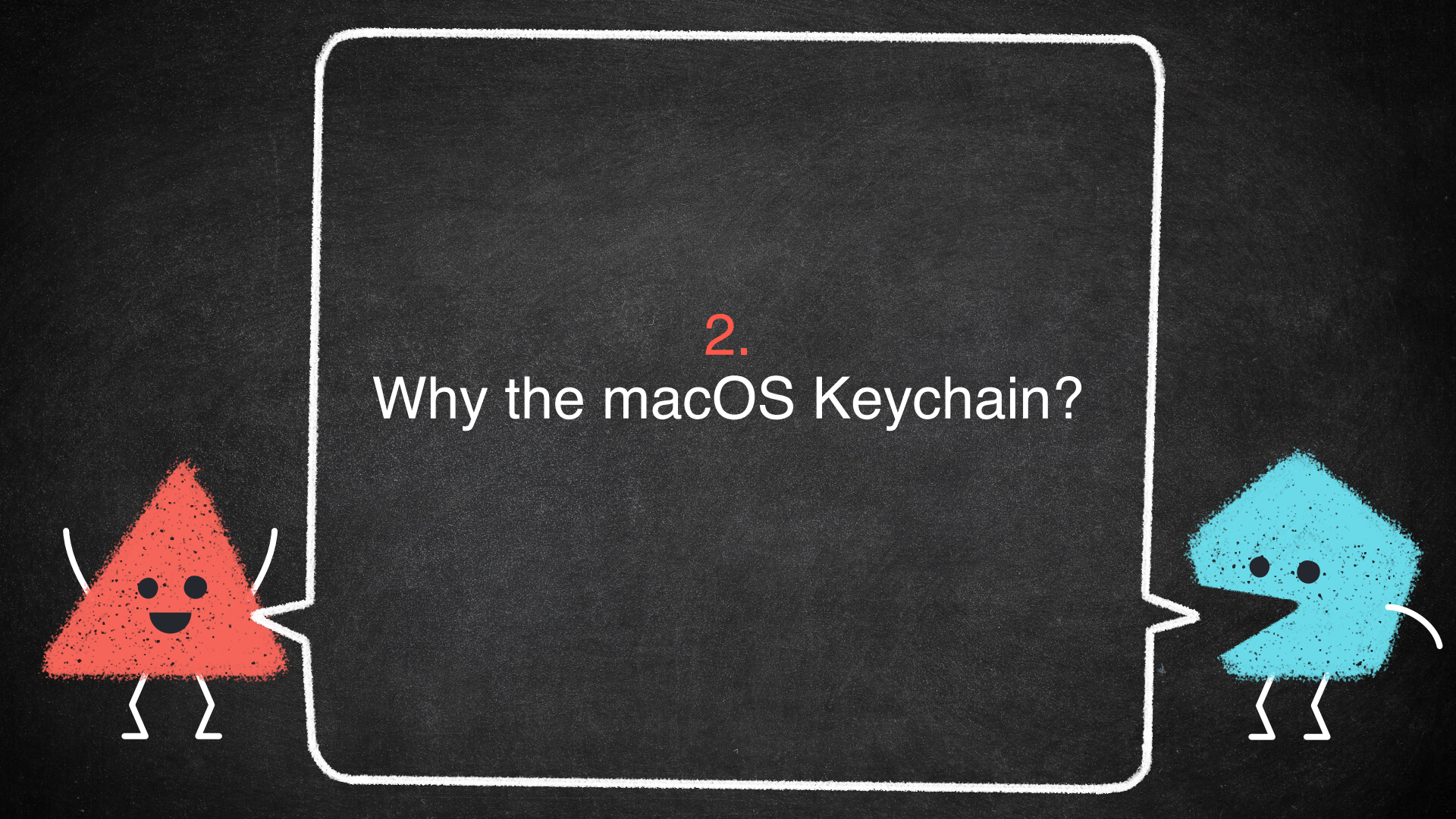
- File-based keychain (what we’re talking about here)
  - macOS based
- Database ”Data Protection” keychain
  - iOS and iCloud based

Two Keychains by Default



Keychain Access.app




2.
Why the macOS Keychain?

Offensive Tradecraft - current

➜No Apple protections around the keychain files

- Download for later analysis
- Metadata around entries is plaintext, but passwords are encrypted

➜Need the user's plaintext password to decrypt offline file

- Can be very tough to get user's password
- Chainbreaker Python GitHub Project

➜Want keychain entries for additional techniques

- Chrome Safe Storage, Slack Safe Storage, Developer signing certificates, etc

Offensive Tradecraft - desire

➜Retrieve decrypted secrets without prompting

- Even if we're wrong about what we can get without prompting, no prompting
  - `SecKeychainSetUserInteractionAllowed`

➜Don't have to know the user's plaintext password

- If we had this, we'd just do it all offline

➜Find misconfigurations (and abuse them)

➜Avoid command-line based detections

- Avoid anomalous file opens on the keychain

Offensive Tradecraft - Tooling

➜Created new Objective C tool – LockSmith

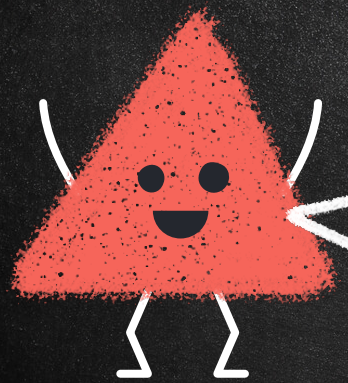
- Open Source on GitHub:
  - https://github.com/its-a-feature/LockSmith
  - Generates Mach-O and Dylib

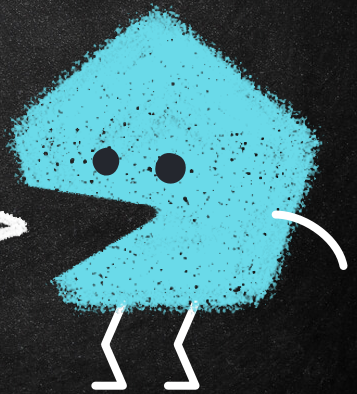
➜Complementary LockSmithLiteJXA

- Open Source on same GitHub

3.

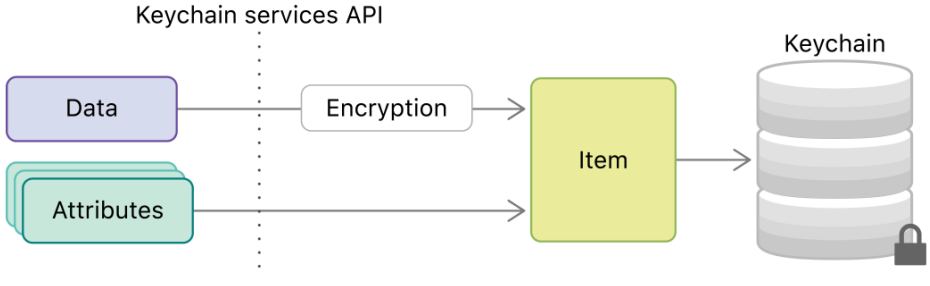
What are Keychain Items?



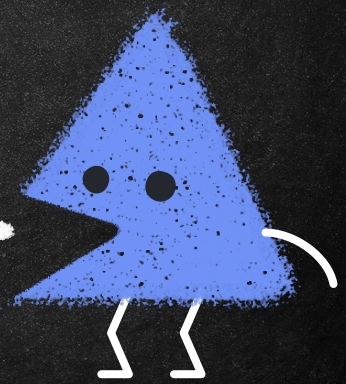


Keychain Items – 2 components

- ➜ Encrypted secret
- ➜ Attributes about the secret (not encrypted)




```
==================================
---------Keychain Entry----------
==================================
Account:          Slack
Label:            Slack Safe Storage
Service:          Slack Safe Storage
Creation Date:    2021-11-12 22:27:45
Modify Date:      2021-11-12 22:27:45
Class:            genp
```

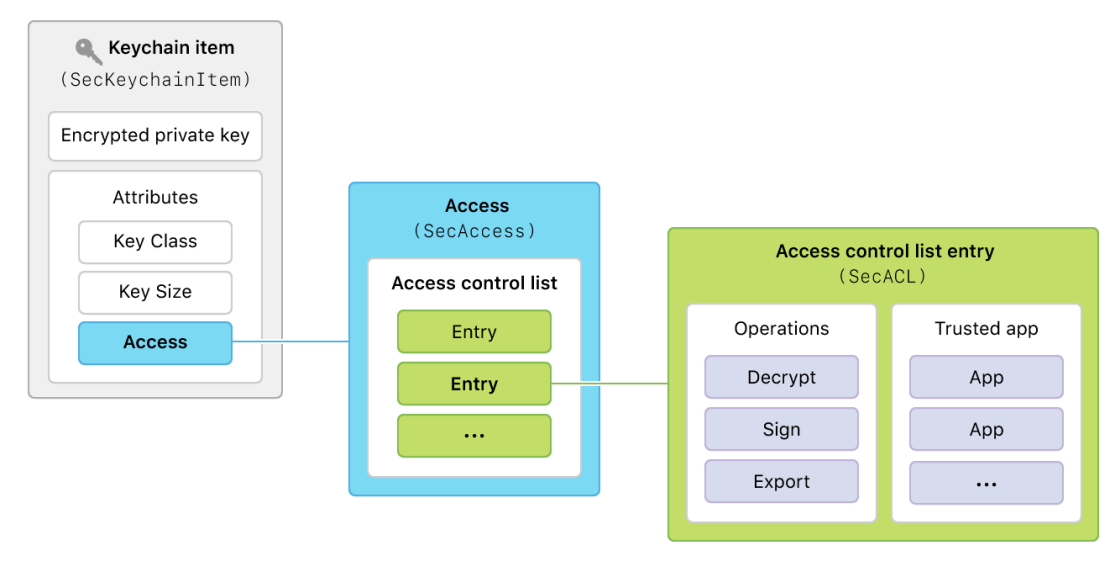


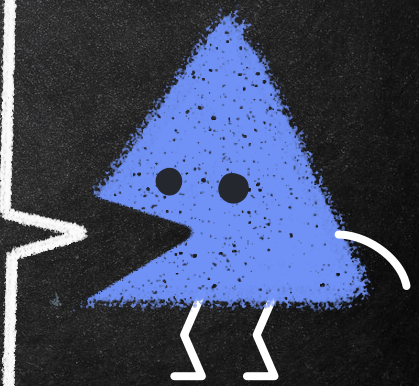
https://developer.apple.com/documentation/security/keychain_services/keychain_items?language=objc

Keychain Items Access

➜ Access Control Lists (ACLs)

- Authorizations (operations) and who can do them

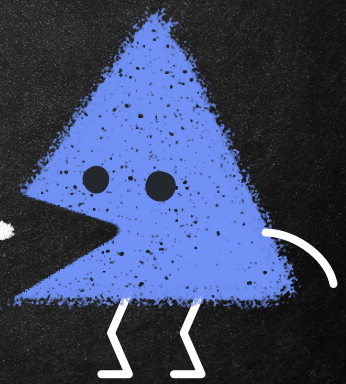




https://developer.apple.com/documentation/security/keychain_services/access_control_lists?language=objc

Keychain ACL Entries

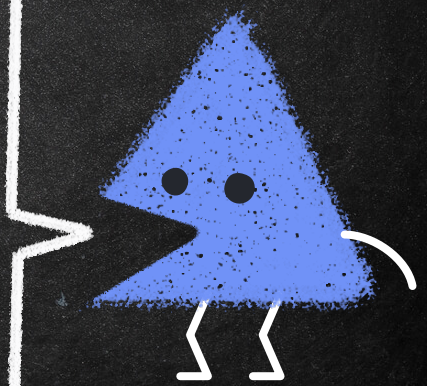
- Offensively interesting authorizations
  - ACLAuthorizationExportClear
  - ACLAuthorizationExportWrapped
  - ACLAuthorizationAny
- Trusted Applications
  - List* of applications that can do the action without prompting
    - Can be Nil, empty list, or a list of applications



Keychain ACL Entries

- ACLAuthorizationPartitionID
  - Super weird description

```
        --- Entry 2
                Description: 67622e03a0b87c7394d39ce76ae4c10d92f2bcecbad8038e51c37be0786d893f
                All applications are trusted
                Authorizations:
                        ACLAuthorizationIntegrity
        --- Entry 3
                Description: 3c3f786d6c2076657273696f6e3d22312e302220656e636f64696e673d225554462d38223f3e0a3c21444f43545950
4520706c697374205055424c494320222d2f2f4170706c652f2f44544420504c49535420312e302f2f454e222022687474703a2f2f7777772e6170706c6
52e636f6d2f445444732f50726f70657274794c6973742d312e302e647464223e0a3c706c69737420766572273696f6e3d22312e30223e0a3c646963743e
0a093c6b65793e506172746974696f6e733c2f6b65793e0a093c61727261793e0a09093c737472696e673e6170706c653a3c2f737472696e673e0a093c2
f61727261793e0a3c2f646963743e0a3c2f706c6973743e0a
                All applications are trusted
                Authorizations:
                        ACLAuthorizationPartitionID
```



Keychain ACL Entries

➜ ACLAuthorizationPartitionID

`kSecACLAuthorizationKeychainItemInsert`

Insert an item into a keychain.

`kSecACLAuthorizationKeychainItemModify`

Modify an item in a keychain.

`kSecACLAuthorizationKeychainItemDelete`

Delete an item from a keychain.

`kSecACLAuthorizationChangeACL`

Change an access control list entry.

`kSecACLAuthorizationChangeOwner`

For internal system use only. Use the `CSSM_ACL_AUTHORIZATION_CHANGE_ACL` tag for changes to owner ACL entries.

`kSecACLAuthorizationIntegrity`

`kSecACLAuthorizationPartitionID`

See Also

Keychain ACL Entries

- ACLAuthorizationPartitionID

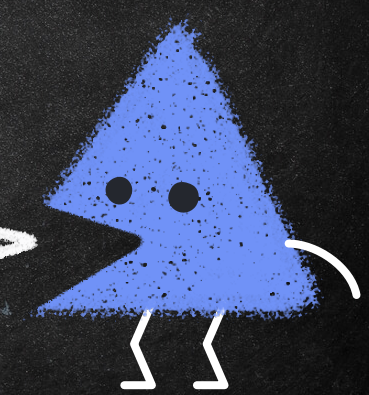
Global Variable

kSecACLAuthorizationPartitionID

macOS 10.11+

Declaration

```
const CFStringRef kSecACLAuthorizationPartitionID;
```



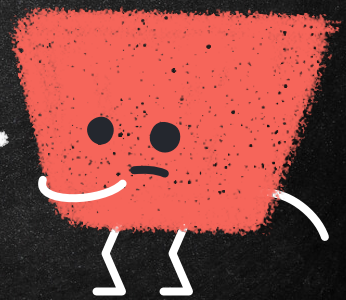
ACLAuthorizationPartitionID

➜Not always present (ex: not on System keychain entries)

➜Description is hex encoded plist with one key – Partitions

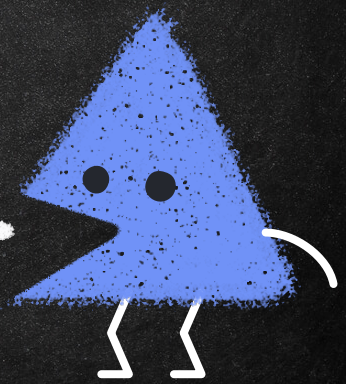
- teamid: - must be signed by a certain teamid
- apple: - must be signed by apple
- cdhash: - must have a specific CDHash

```
        --- Entry 3
                Allowed Code Signatures: teamid:BQR82RBBHL
                        TeamID doesn't match current application: nil
                Description: 3c3f786d6c2076657273696f6e3d22312e302220656e636f64696e673d225554462d38223f3e0a3c21444f435459
504520706c697374205055424c494320222d2f2f4170706c652f2f44544420504c49535420312e302f2f454e222022687474703a2f2f7777772e61707
06c652e636f6d2f445444732f50726f70657274794c6973742d312e302e647464223e0a3c706c69737420766572273696f6e3d22312e30223e0a3c6469
63743e0a093c6b65793e506172746974696f6e733c2f6b65793e0a093c61727261793e0a09093c737472696e673e7465616d69643a425152383252424
2484c3c2f737472696e673e0a093c2f61727261793e0a3c2f646963743e0a3c2f706c6973743e0a
                All applications are trusted
                Authorizations:
                        ACLAuthorizationPartitionID
```



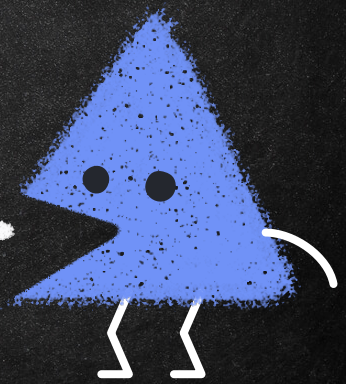
Keychain Basic ACLs

- New Keychain items via Keychain Access get 5 ACLs:
  - All applications can Encrypt
  - No applications can Export/Decrypt
  - All applications can see the Integrity Check
  - No applications can change ACLs
  - PartitionID set to **apple:**
- Set this way to all applications can do “non dangerous” things and no applications can do “dangerous” things
- “No application” means “without prompting for user consent”

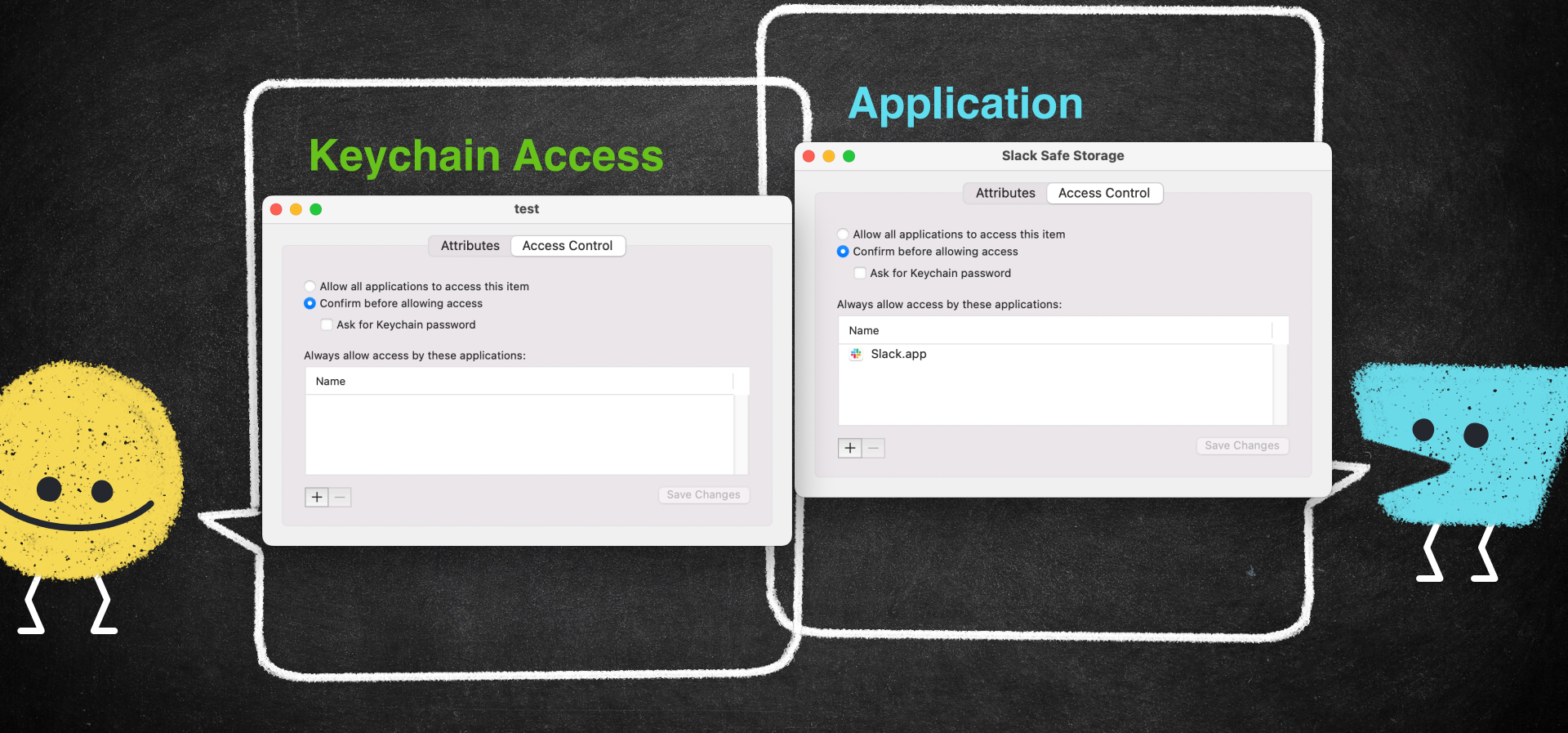


Keychain Standard ACLs

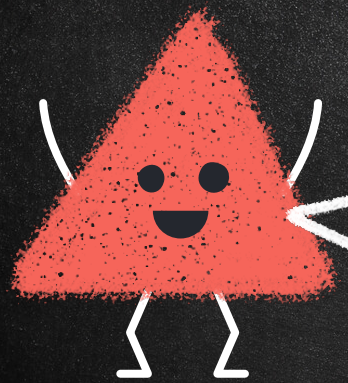
- New Keychain items **from applications** get 5 ACLs:
  - All applications can Encrypt
  - **1+** applications can Export/Decrypt
  - All applications can see the Integrity Check
  - No applications can change ACLs
  - PartitionID set to **teamid:[teamID here]**
- Set this way to all applications can do “non dangerous” things and few applications can do “dangerous” things
- “No application” means “without prompting for user consent”

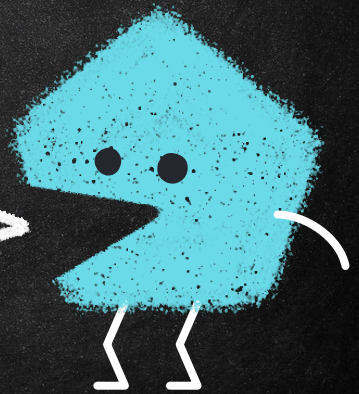


GUI Access Control Perspective



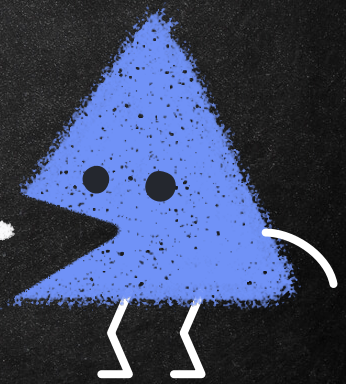
4.
Accessing the Keychain





Security built-in tool

- **security dump-keychain –a –d**
  - Dump all metadata and decrypted secrets from the keychain
- **security find-generic-password -a “Slack” –g**
  - Find generic passwords for the “Slack” account and print the secrets
- **security set-generic-password-partition-list –s “test service” –a “test account” –S “cdhash:ac48c1600ffe453258c156478a9b31af922c53a5”**
  - Change the specified entry’s PartitionID entry



API - SecItemCopyMatching

➜Gives the generic attributes about entries

➜CFDictionary of search parameters

- kSecReturnData – try to decrypt automatically or not
  - Set to **False**
- kSecReturnRef – get reference to keychain item
  - Set to **True**
- kSecReturnAttributes – get metadata about entries
- kSecMatchLimit – how many results to return
- kSecClass – what kind of keychain entry

Keychain Item Attributes

➜ Keys

```
================================
---------Keychain Entry----------
================================
Account:       (null)
Label:         com.apple.kerberos.kdc
Service:       (null)
Creation Date: (null)
Modify Date:   (null)
Class:         keys
Key details:
        Signable:   YES
        kcls:       1
        Encrypt:    NO
        Decrypt:    YES
        Wrap Key:   NO
        Unwrap Key: YES
        Verify Key: NO
        Key Type:   42
        Permanent:  YES
        Derive Key: NO
        klbl:       uxG3rmNFPl4glO1cs1ZdQ7BxYxs=
        esiz:       2048
        bsiz:       2048
```

Certs

```
================================
---------Keychain Entry----------
================================
Account:       (null)
Label:         com.apple.kerberos.kdc
Service:       (null)
Creation Date: (null)
Modify Date:   (null)
Class:         cert
Certificate details:
        Cert Type:  0
        Cert Subj:  MDsxHzAdBgNVBAMMFm
        Cert pkhh:  uxG3rmNFPl4glO1cs1
        Cert issr:  MDsxHzAdBgNVBAMMFm
        Cert slnr:  cnc5Ww==
        Cert cenc:  3
```

API - SecAccessCopyACLList

➜Get access control lists for keychain item reference
➜Returns list of ACLs where each has:

- Description
- Trusted Application List
    - /Applications/Slack.app
    - /usr/libexec/airportd
    - group://AirPort
        - Application Group
- Prompt Selector
    - Largely ignored

API - Exporting Passwords

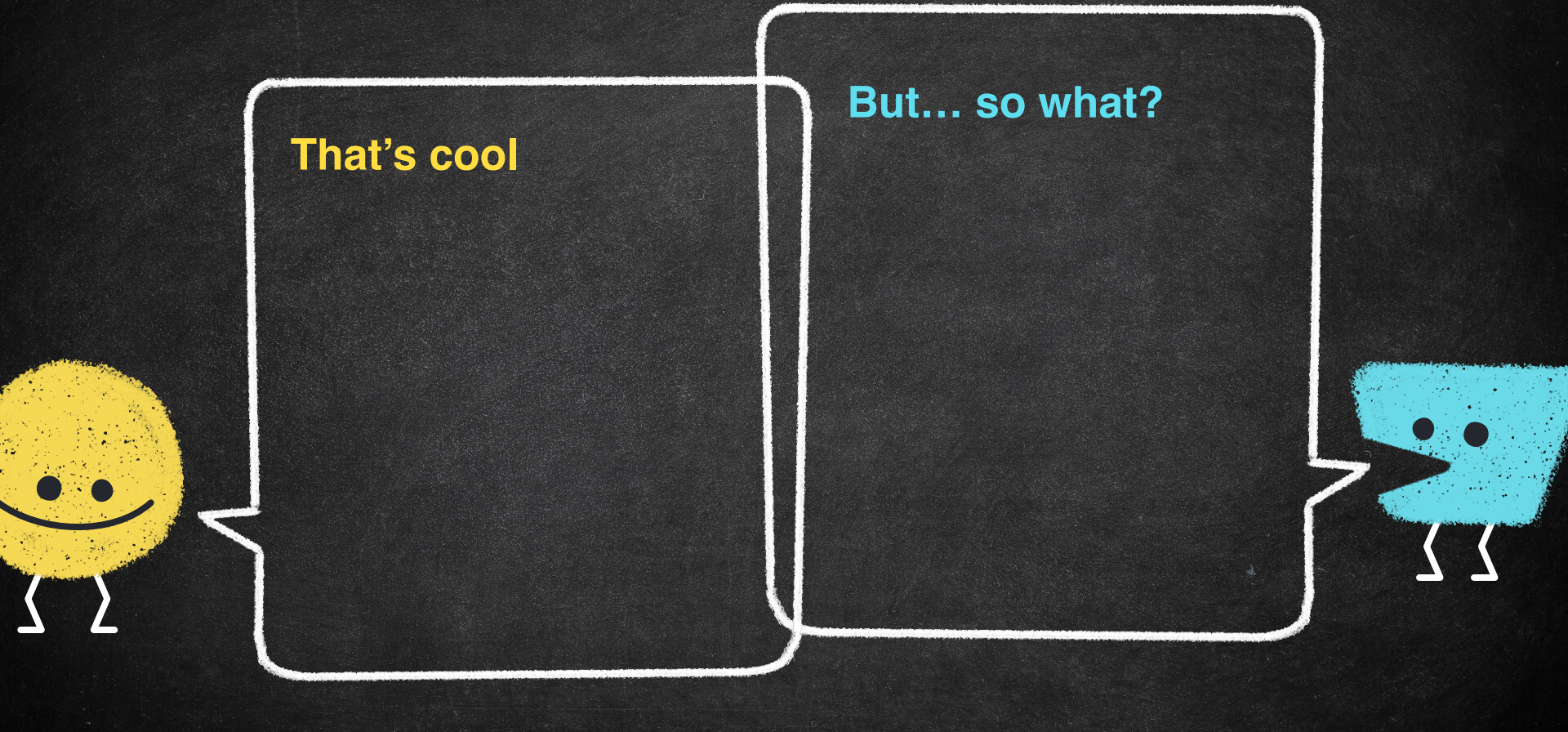
➜SecKeychainItemCopyContent

- Gets the plaintext secret data

➜SecItemExport

- Exports Keys and Certificates
- Might have to set passwords and export encrypted

Operationally though…



Exporting Without Prompts

➜If 1+ trusted applications listed:

- Need appropriate Authorizations
- Need code signature to match PartitionID
- Need code signature to match that of one trusted App
    - Or be a member of the right KeychainAccessGroup

➜If **all** applications trusted:

- Need appropriate Authorizations
- Need code signature to match PartitionIDs
    - If no PartitionID then truly is **ALL** aplications

Matching Apps and PartitionIDs

➜If 1+ trusted applications are listed:

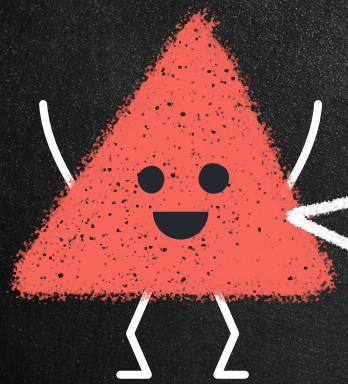
- Need to get app to load up our code somehow
- DYLD_INSERT_LIBRARIES, Dylib Hijacking, etc.

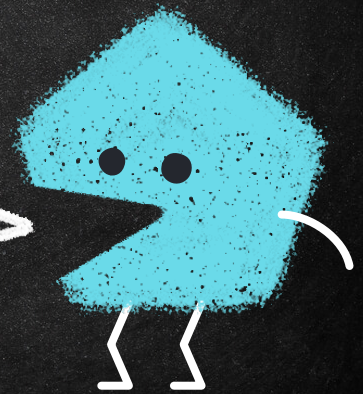
➜What about **apple:** partition ID?

- **osascript** is an apple platform binary that loads up our arbitrary code
  - **JXA** for the win!
- **python** also works, but isn't included by default ☹

5.
Keychain Goodies

Microsoft





Two Additional Attributes

➜Invisible

- Boolean flag to hide Keychain entry from Keychain Access application
- Only affects users trying to browse through the UI

➜General

- Free-form string of additional **metadata**
- Because it’s metadata, it’s not encrypted

Microsoft Office Ticket Cache

➜Programmatically we can fetch all entries and all their metadata properties

- MASSIVE General Attribute blob

```
================================
---------Keychain Entry----------
================================
Account:          Microsoft Office Ticket Cache
Label:            Microsoft Office Ticket Cache
Service:          (null)
Creation Date:    2020-12-03 02:11:43
Modify Date:      2022-09-23 19:18:21
Class:            genp
Invisible:        YES
General:          YnBsaXN0MDDRAQJfECk0YmZjMzM3MS0
5ZWZjLTJhNWYwNjY0YjEzNNEFBltUaWNrZXRDYWNoZd8Q
ZW50Lm9zaS5vZmZpY2UubmV0L3xfEGBodHRwczovL291d
HJ1ZSwgInZhbHVlIjoiMTY2MjIyNjQ3NCJ9fX1fEClodH
ZpY2UubzM2NWZpbHRlcmluZy5jb218XxAfaHR0cHM6Ly9
```

Microsoft Office Ticket Cache

➜Turns out to be a large binary plist

- plutil –convert xml1

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>4bf...4b134_ADAL</key>
    <dict>
        <key>4bf...4b134</key>
        <dict>
            <key>TicketCache</key>
            <dict>
                <key>394...be2a|</key>
                <dict>
                    <key>Expires</key>
                    <string>2022-09-23T20:48:22Z</string>
                    <key>Ticket</key>
                    <string>eyJhbGci...</string>
                </dict>
                <key>https://api.office.net|</key>
                <dict>
                    <key>Expires</key>
                    <string>2022-09-24T13:50:04Z</string>
                    <key>Ticket</key>
                    <string>eyJ0eXAi...</string>
```

Microsoft Office Ticket Cache

➜JWT “tickets” for:

- https://api.office.net
- https://augloop.office.com/v2
- https://clients.config.office.net/
- https://dataservice.o365filtering.com/
- https://enrichment.osi.office.net/
- https://graph.microsoft.com/
- https://messaging.engagement.office.com/*
- https://outlook.office.com
- https://outlook.office365.com/
- https://pptsgs.officeapps.live.com
- https://presence.teams.microsoft.com/

Office365.com JWT Scope

- Calendars.ReadWrite
- Calendars.ReadWrite.Shared
- Contacts.ReadWrite
- Contacts.ReadWrite.Shared
- EAS.AccessAsUser.All
- EopPolicySync.AccessAsUser.All
- EopPsorWs.AccessAsUser.All
- EWS.AccessAsUser.All
- Files.ReadWrite.All
- Files.ReadWrite.Shared
- Group.ReadWrite.All
- Mail.ReadWrite
- Mail.ReadWrite.Shared
- Mail.Send
- Mail.Send.Shared
- MailboxSettings.ReadWrite
- MapiHttp.AccessAsUser.All
- MessageReaction-Internal.Update
- Notes.Read
- Notes.ReadWrite
- Oab.AccessAsUser.All
- OutlookService.AccessAsUser.All
- OWA.AccessAsUser.All
- People.Read
- People.ReadWrite
- Place.Read.All
- Privilege.ELT
- Signals.Read
- Signals.ReadWrite
- SubstrateSearch-Internal.ReadWrite
- Tags.ReadWrite
- Tasks.ReadWrite
- Tasks.ReadWrite.Shared
- Todo-Internal.ReadWrite
- User.ReadBasic
- User.ReadBasic.All
- user_impersonation
- User-Internal.ReadWrite

Microsoft Office Ticket Cache

➜ACLs – Locked down to Microsoft Outlook

- What's stored in the keychain entry then?

```
        --- Entry 0
                Description: Microsoft Office Ticket Cache
                Trusted App: /Applications/Microsoft Outlook.app
                        Requirement String: identifier "com.microsoft.Outlook" and anchor apple generic and certificate 1
[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and cert
ificate leaf[subject.OU] = UBF8T346G9
                        Application is valid
                Authorizations:
                         ACLAuthorizationDecrypt
                         ACLAuthorizationDerive
                         ACLAuthorizationExportClear
                         ACLAuthorizationExportWrapped
                         ACLAuthorizationMAC
                         ACLAuthorizationSign
        --- Entry 1
                Description: Microsoft Office Ticket Cache
                All applications are trusted
                Authorizations:
                         ACLAuthorizationEncrypt
        --- Entry 2
                Description: 5360d82434bf62a5a90b59e7f62723992cba2f2fb692bce080a613a313d46983
                All applications are trusted
                Authorizations:
                         ACLAuthorizationIntegrity
        --- Entry 3
                Allowed Code Signatures: teamid:UBF8T346G9
```

Microsoft Office Ticket Cache

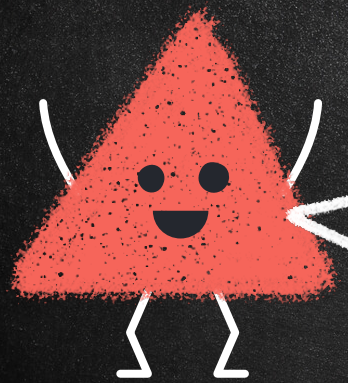
➜Dump the entry with **security** tooling

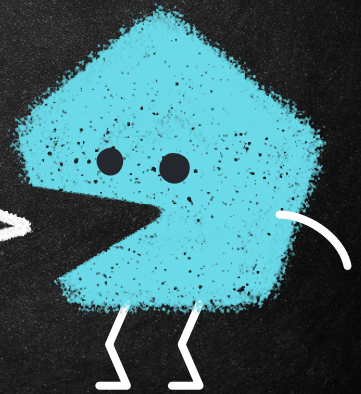
```
itsafeature@spooky Debug % security find-generic-password -l "Microsoft Office Ticket Cache" -g
keychain: "/Users/itsafeature/Library/Keychains/login.keychain-db"
version: 512
class: "genp"
attributes:
    0x00000007 <blob>="Microsoft Office Ticket Cache"
    0x00000008 <blob>=<NULL>
    "acct"<blob>="Microsoft Office Ticket Cache"
    "cdat"<timedate>=0x323032303132303330323131343 35A00  "20201203021143Z\000"
    "crtr"<uint32>=<NULL>
    "cusi"<sint32>=<NULL>
    "desc"<blob>=<NULL>
    "gena"<blob>=0x62706C...<snip>
    bplist00\321\001\002_\020)4bfc...<snip>
    "icmt"<blob>=<NULL>
    "invi"<sint32>=0x00000001
    "mdat"<timedate>=0x323032323039323331393138323 15A00  "20220923191821Z\000"
    "nega"<sint32>=<NULL>
    "prot"<blob>=<NULL>
    "scrp"<sint32>=<NULL>
    "svce"<blob>=<NULL>
    "type"<uint32>=<NULL>
password:
```

IT'S EMPTY!!

5.
Keychain Goodies

Slack





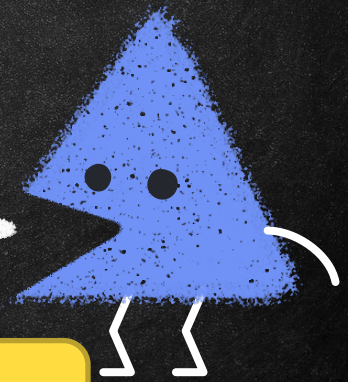
Slack App Keychain Entries

```
================================
---------Keychain Entry----------
================================
Account:       Slack
Label:         Slack Safe Storage
Service:       Slack Safe Storage
Creation Date: 2021-11-12 22:27:45
Modify Date:   2021-11-12 22:27:45
Class:         genp
Owner Authorizations based on ACLAuthorizationPartitionID
Owner Authorizations:
        ACLAuthorizationEncrypt
        ACLAuthorizationDecrypt
        ACLAuthorizationDerive
        ACLAuthorizationExportClear
        ACLAuthorizationExportWrapped
        ACLAuthorizationMAC
        ACLAuthorizationSign
        ACLAuthorizationIntegrity
        ACLAuthorizationPartitionID
------------ACLS------------
        --- Entry 0
                Description: Slack Safe Storage
                All applications are trusted
                Authorizations:
                        ACLAuthorizationEncrypt
        --- Entry 1
                Description: Slack Safe Storage
                Trusted App: /Applications/Slack.app
                        Requirement String: identifier "com.tinyspeck.slackmacgap" and anchor a
pple generic and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate lea
f[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] = BQR82RBBHL
                        Application is valid
                Authorizations:
                        ACLAuthorizationDecrypt
                        ACLAuthorizationDerive
                        ACLAuthorizationExportClear
                        ACLAuthorizationExportWrapped
                        ACLAuthorizationMAC
                        ACLAuthorizationSign
        --- Entry 2
                Description: 9c7d3204702cae4e374379f65059d1d695ada67bb1efd6f69cb34bdb6127ab29
                All applications are trusted
                Authorizations:
                        ACLAuthorizationIntegrity
        --- Entry 3
                Allowed Code Signatures: teamid:BQR82RBBHL
```

Trusted Application with requirement string

Authorizations we want

PartitionID with teamID requirements



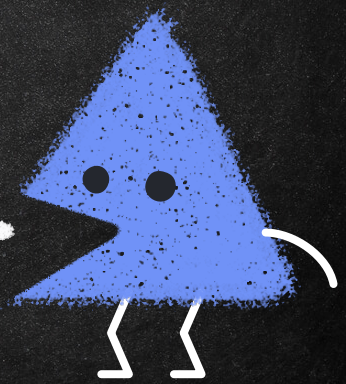
Slack App Hijacks

➜ Current application is hardened

```
itsafeature@spooky Debug % codesign -d -vvv /Applications/Slack.app
Executable=/Applications/Slack.app/Contents/MacOS/Slack
Identifier=com.tinyspeck.slackmacgap
Format=app bundle with Mach-O thin (x86_64)
CodeDirectory v=20500 size=485 flags=0x12000(library-validation,runtime) hashes=4+7 location=em
bedded
```

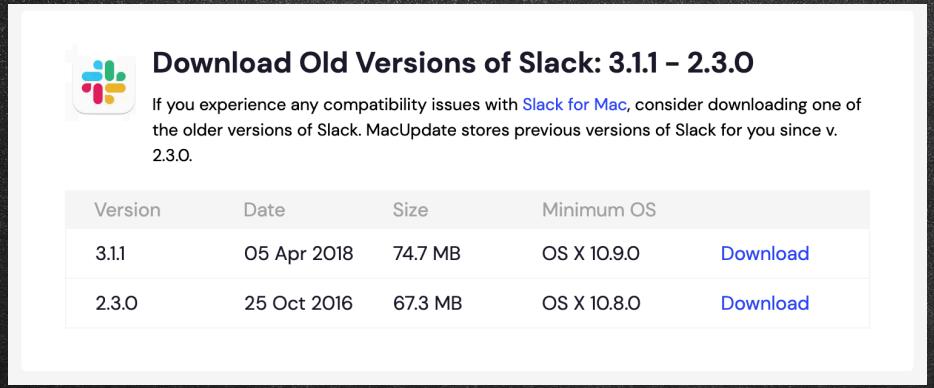
```
itsafeature@spooky Debug % codesign -d --entitlements - /Applications/Slack.app
Executable=/Applications/Slack.app/Contents/MacOS/Slack
[Dict]
        [Key] com.apple.security.device.usb
        [Value]
                [Bool] true
        [Key] com.apple.security.cs.allow-jit
        [Value]
                [Bool] true
        [Key] com.apple.security.device.print
        [Value]
                [Bool] true
        [Key] com.apple.security.device.camera
        [Value]
                [Bool] true
        [Key] com.apple.security.device.bluetooth
        [Value]
                [Bool] true
        [Key] com.apple.security.device.audio-input
        [Value]
                [Bool] true
        [Key] com.apple.security.personal-information.location
        [Value]
                [Bool] true
```

No vulnerable entitlements for malicious dylibs



Slack App Hijacks

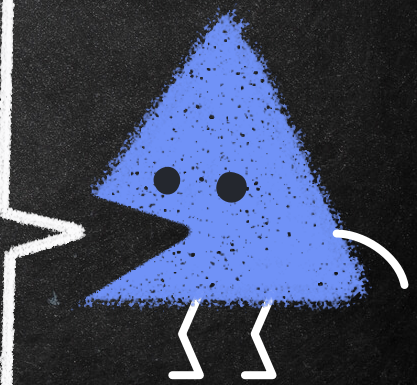
- ➜ Hijack execution of existing app
  - Not easy ☹
- ➜ Download older version of app
  - https://www.macupdate.com/app/mac/50617/slack/old-versions



Download Old Versions of Slack: 3.1.1 – 2.3.0

If you experience any compatibility issues with Slack for Mac, consider downloading one of the older versions of Slack. MacUpdate stores previous versions of Slack for you since v. 2.3.0.

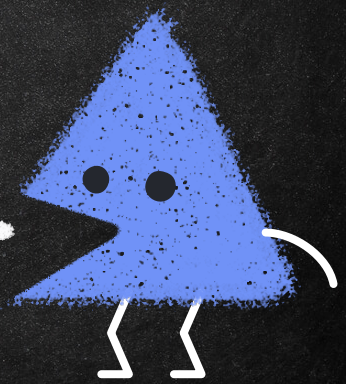
Version	Date	Size	Minimum OS	
3.1.1	05 Apr 2018	74.7 MB	OS X 10.9.0	Download
2.3.0	25 Oct 2016	67.3 MB	OS X 10.8.0	Download



Slack App Hijacks

- Bad signing / entitlements - abusable
  - DYLD_INSERT_LIBRARIES FTW

```
itsafeature@spooky Debug % codesign -d -vvv /Volumes/Slack.app/Slack.app
Executable=/Volumes/Slack.app/Slack.app/Contents/MacOS/Slack
Identifier=com.tinyspeck.slackmacgap
Format=app bundle with Mach-O thin (x86_64)
CodeDirectory v=20200 size=281 flags=0x0(none) hashes=3+3 location=embedded
Hash type=sha256 size=32
CandidateCDHash sha1=d5fbdfcff3d5f68207f7ab33cfea873fc7393d56
CandidateCDHashFull sha1=d5fbdfcff3d5f68207f7ab33cfea873fc7393d56
CandidateCDHash sha256=b6fd4445e810dd2ef4d1b1809831f03b5fb035a0
CandidateCDHashFull sha256=b6fd4445e810dd2ef4d1b1809831f03b5fb035a02e1405c230a344026998f69e
Hash choices=sha1,sha256
CMSDigest=2167f33311be94371a0715e544ed12fc1ec2408d933df373e12c5d66d56536af
CMSDigestType=2
CDHash=b6fd4445e810dd2ef4d1b1809831f03b5fb035a0
Signature size=8936
Authority=Developer ID Application: Slack Technologies, Inc. (BQR82RBBHL)
Authority=Developer ID Certification Authority
Authority=Apple Root CA
Timestamp=Mar 30, 2018 at 5:04:18 PM
Info.plist entries=19
TeamIdentifier=BQR82RBBHL
Sealed Resources version=2 rules=13 files=86
Internal requirements count=1 size=188
```



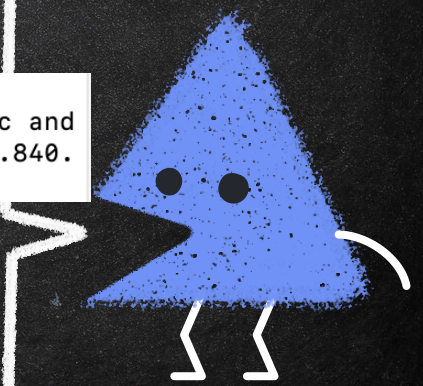
Slack App Hijacks

- Code Requirements from new and old applications match
  - Current Slack

```
Trusted App: /Applications/Slack.app
        Requirement String: identifier "com.tinyspeck.slackmacgap" and anchor a
pple generic and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate lea
f[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] = BQR82RBBHL
```

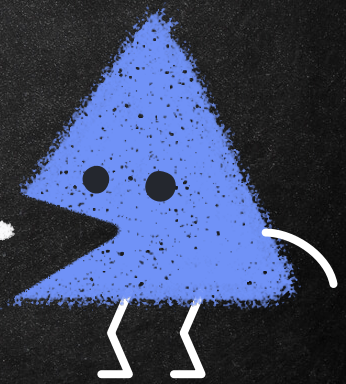
  - Old Slack

```
Current Process: /Volumes/Slack.app/Slack.app/Contents/MacOS/Slack
        Requirement String: identifier "com.tinyspeck.slackmacgap" and anchor apple generic and
 certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.
113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] = BQR82RBBHL
```



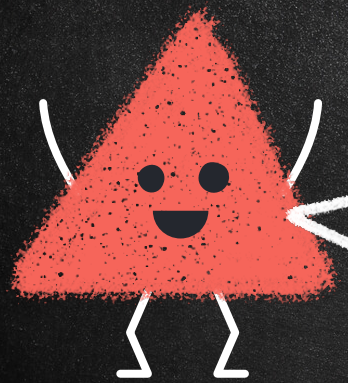
Slack App Hijacks

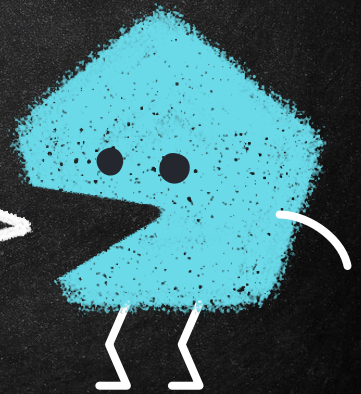
- ➜ Don’t need to run from /Applications/Slack.app
  - Just need to match requirements string and PartitionID
- ➜ Mount old Slack in /Volumes/Slack
- ➜ Use DYLD_INSERT_LIBRARIES to load our Dylib
- ➜ Execute our LockSmith code in the constructor



5.
Keychain Goodies

computer$ & com.apple.kerberos.kdc



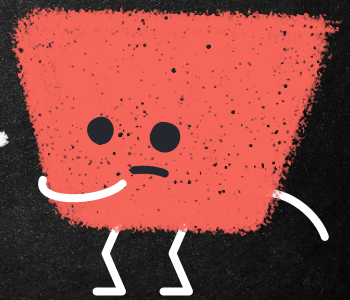


Computer$ when AD joined

- Can’t always get contents of ACL entries

```
==================================
---------Keychain Entry----------
==================================
Account:        gala$
Label:          /Active Directory/ORCHARD
Service:        /Active Directory/ORCHARD
Creation Date: 2022-08-02 18:37:17
Modify Date:    2022-09-13 18:39:16
Class:          genp
OwnerID: 0
GroupID: 0
OwnerType: 0x101
        Owner Authorizations don't match our user context
Owner Authorizations:
        ACLAuthorizationAny
------------ACLs------------
        --- Entry 0
                Failed to copy contents with error: -25240
                Failed to copy simple contents with error: -25240
                Authorizations:
                        ACLAuthorizationAny
        --- Entry 1
                Failed to copy contents with error: -25240
                Failed to copy simple contents with error: -25240
                Authorizations:
                        ACLAuthorizationChangeACL
```

Suspiciously broad

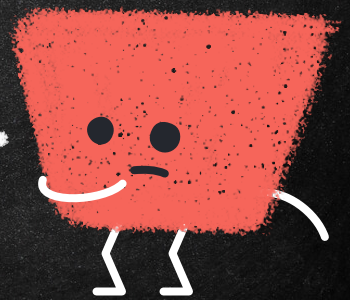


Computer$ when AD joined

- As a standard user, can't get the plaintext password without prompting
- As root, can get the plaintext password without prompting

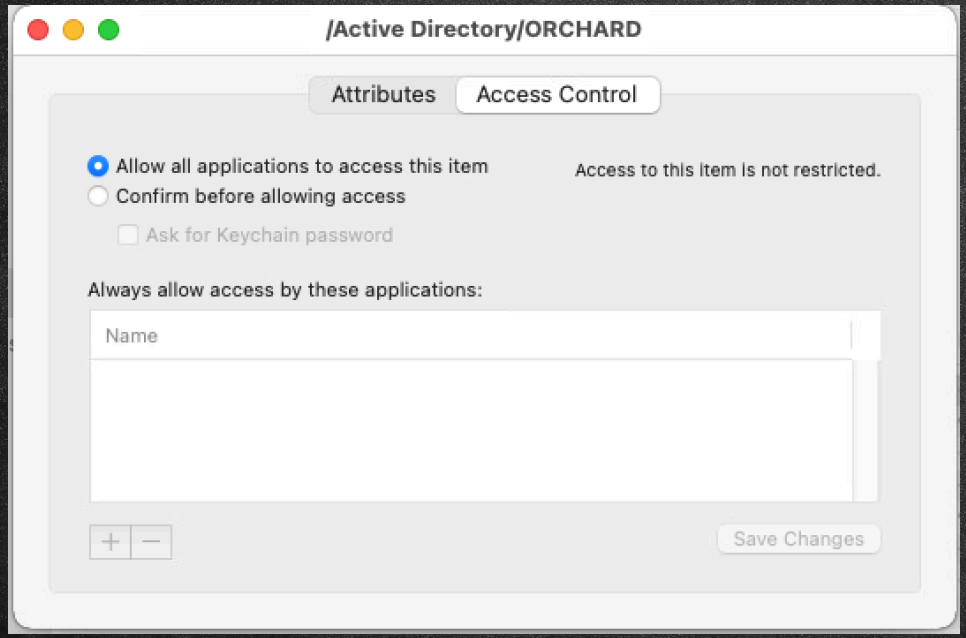
```
OwnerID: 0
GroupID: 0
OwnerType: 0x101
        Owner Authorizations don't match our user context
```

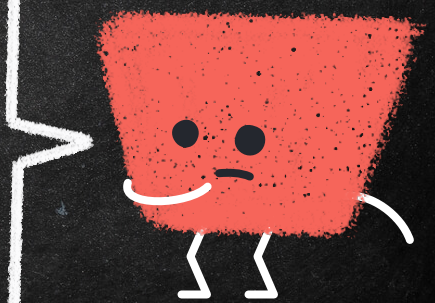
- Owner type 0x101 means the userID must match (0 in this case) and that root should be treated as a normal user account
- This is expected* – normal users shouldn't get computer$



Computer$ when AD joined

- Looking at the same data in the Keychain Access application shows a different story



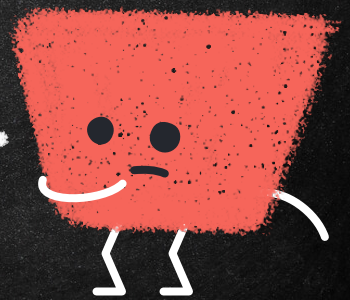


Computer$ when AD joined

- If we toggle the “All applications” default access to “no applications” and back again, then save the entry, the ACLs change

```
==================================
---------Keychain Entry----------
==================================
Account:       gala$
Label:         /Active Directory/ORCHARD
Service:       /Active Directory/ORCHARD
Creation Date: 2022-08-02 18:37:17
Modify Date:   2022-09-24 02:41:13
Class:         genp
OwnerID: 0
GroupID: 0
OwnerType: 0x101
        Owner Authorizations don't match our user context
Owner Authorizations:
        ACLAuthorizationAny
------------ACLS------------
        --- Entry 0
                Description: /Active Directory/ORCHARD
                All applications are trusted
                Authorizations:
                        ACLAuthorizationAny
        --- Entry 1
                Failed to copy contents with error: -25240
                Failed to copy simple contents with error: -25240
                Authorizations:
                        ACLAuthorizationChangeACL
```

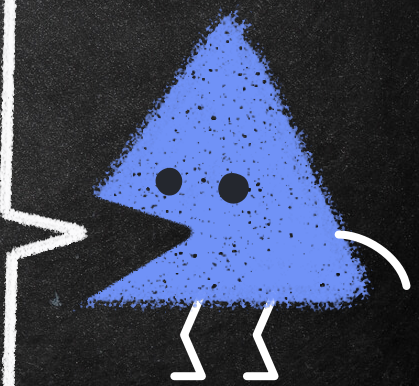
Also, now standard users can export secret without prompting!



com.apple.kerberos.kdc

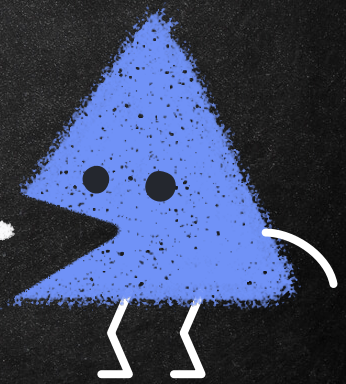
- com.apple.kerberos.kdc PrivateKey has the same issue as computer$

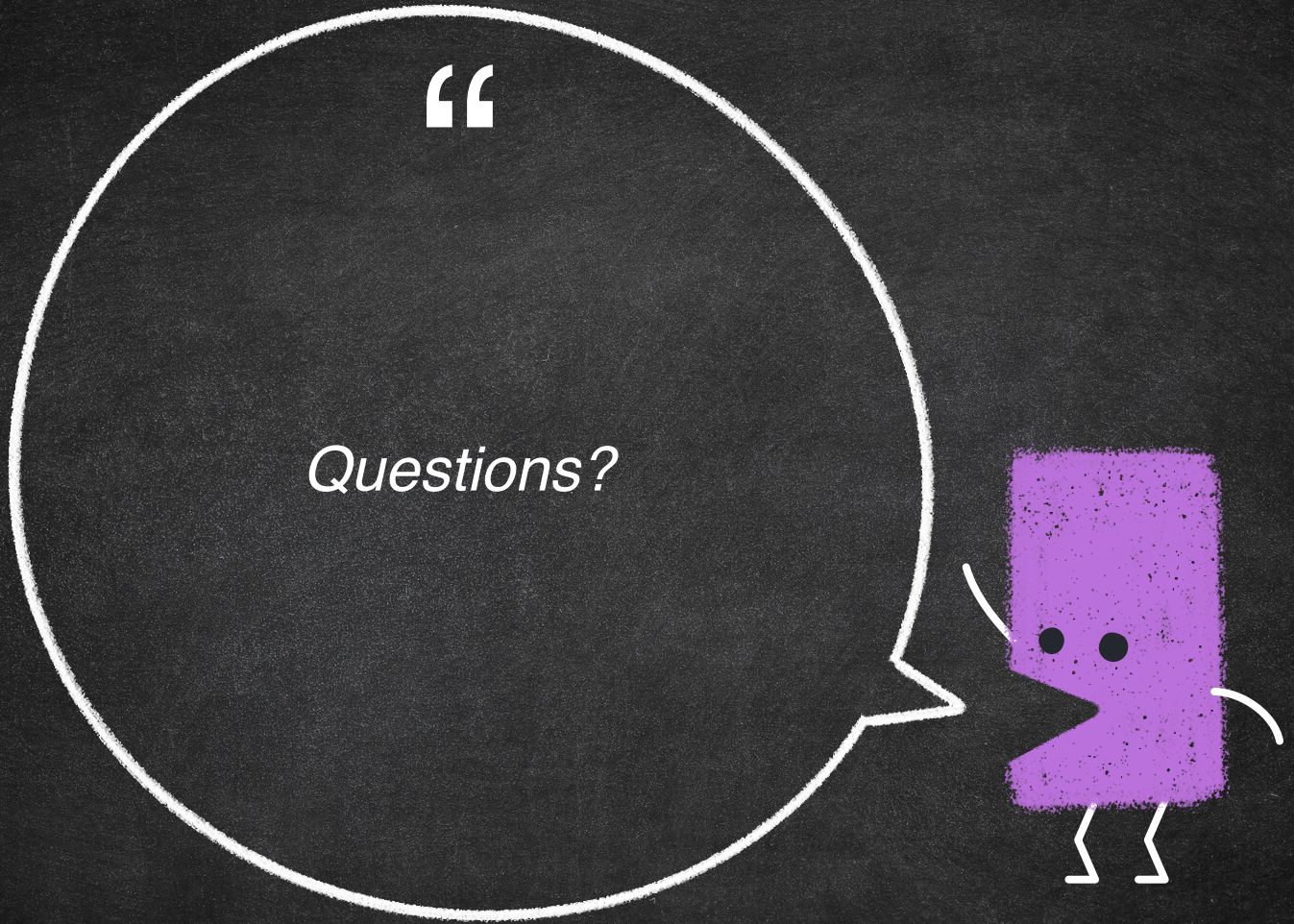
```
OwnerID: 0
GroupID: 0
OwnerType: 0x1
        Owner Authorizations don't match our user context
Owner Authorizations:
        ACLAuthorizationDecrypt
        ACLAuthorizationDerive
        ACLAuthorizationExportClear
        ACLAuthorizationExportWrapped
        ACLAuthorizationMAC
        ACLAuthorizationSign
        ACLAuthorizationAny
------------ACLS------------
        --- Entry 0
                Description: lkdc-acl
                Trusted App: /System/Library/PrivateFrameworks/Heimdal.framework/Helpers/kdc
                        Requirement String: identifier "com.apple.kdc" and anchor apple
                Authorizations:
                         ACLAuthorizationDecrypt
                         ACLAuthorizationDerive
                         ACLAuthorizationExportClear
                         ACLAuthorizationExportWrapped
                         ACLAuthorizationMAC
                         ACLAuthorizationSign
        --- Entry 1
                Description: com.apple.kerberos.kdc
                All applications are trusted
                Authorizations:
                         ACLAuthorizationAny
```



com.apple.kerberos.kdc

- com.apple.kerberos.kdc PrivateKey has the same issue as computer$
- Root can export com.apple.kerberos.kdc private key, public key, and certificate without prompting
  - These together allow you to be any user to LKDC
  - LKDC handles authentication for:
    - Screen Sharing (VNC)
    - File Sharing (SMB / APFS)
    - Remote Management
  - Who resets LKDC on each mac after compromise?




“
Questions?