Hello!

LINUS HENZE

Key Steal

www.pinauten.de

Objective by the Sea

ABOUT ME

- Linus Henze (@LinusHenze)
- Independent iOS and macOS security researcher from Germany
- CS student at Universität Koblenz
- Website: pinauten.de
- Exploits can be found on GitHub: github.com/LinusHenze

WHOAMI

AGENDA

- Let's talk about the Keychain
- Keychain Internals
- Exploiting the Keychain
- Apple's fix
- Demonstration

AGENDA

LET'S TALK ABOUT THE KEYCHAIN

HIGH LEVEL VIEW ON THE KEYCHAIN

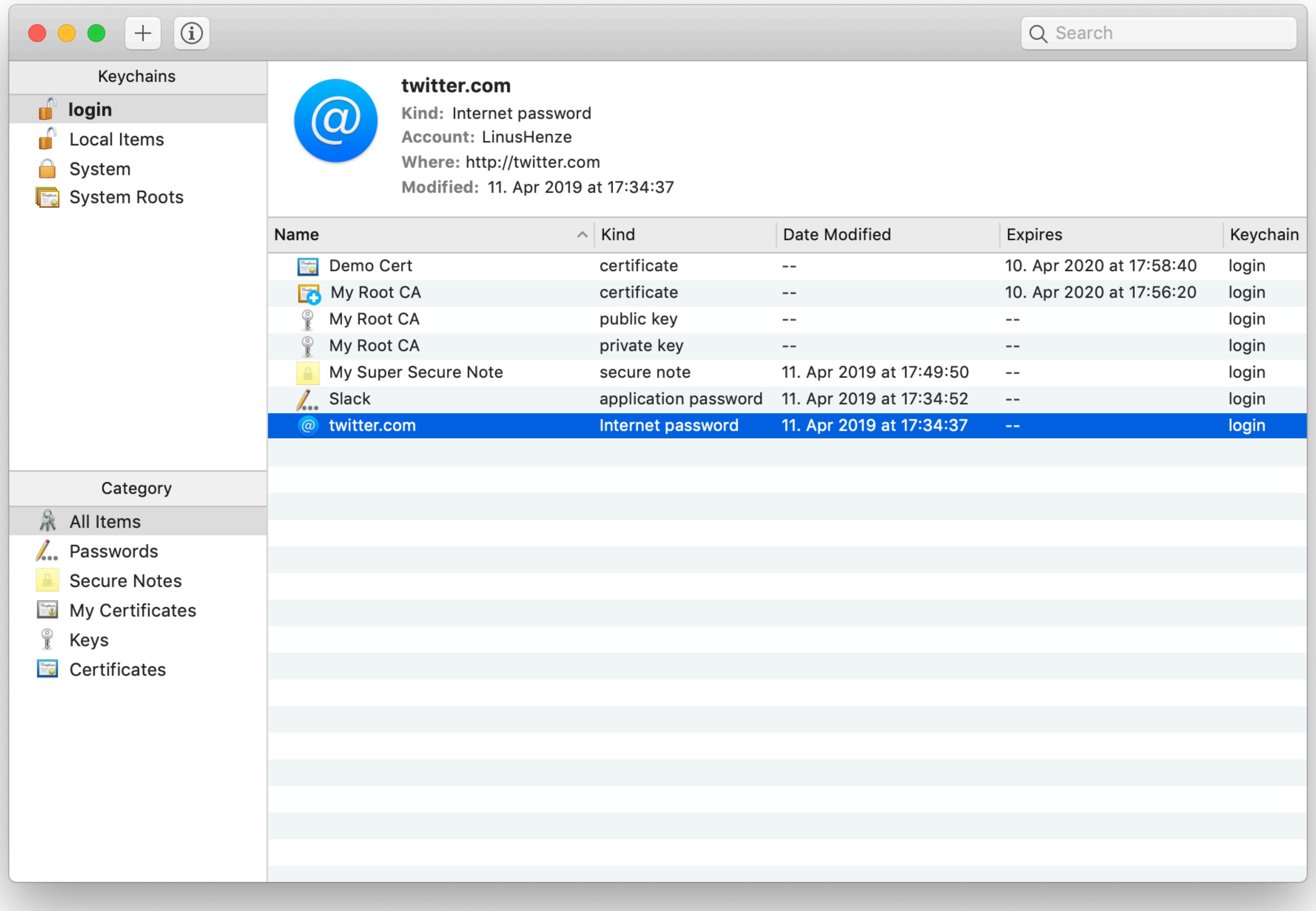
WHAT IS THE KEYCHAIN?

- Central place for your passwords/certificates/...
- One Keychain per user + System Keychain
- Additionally, each user has an iCloud Keychain
  - Not a normal Keychain: different implementation and APIs
  - Not in scope of this talk

KEYCHAIN

LOGIN KEYCHAIN

- Login Keychain
  - Located in ~/Library/Keychains/login.keychain-db
- Usually encrypted using your login password
  - Automatically unlocked on login
- Used by many Apps and system services
- Contains all your personal passwords



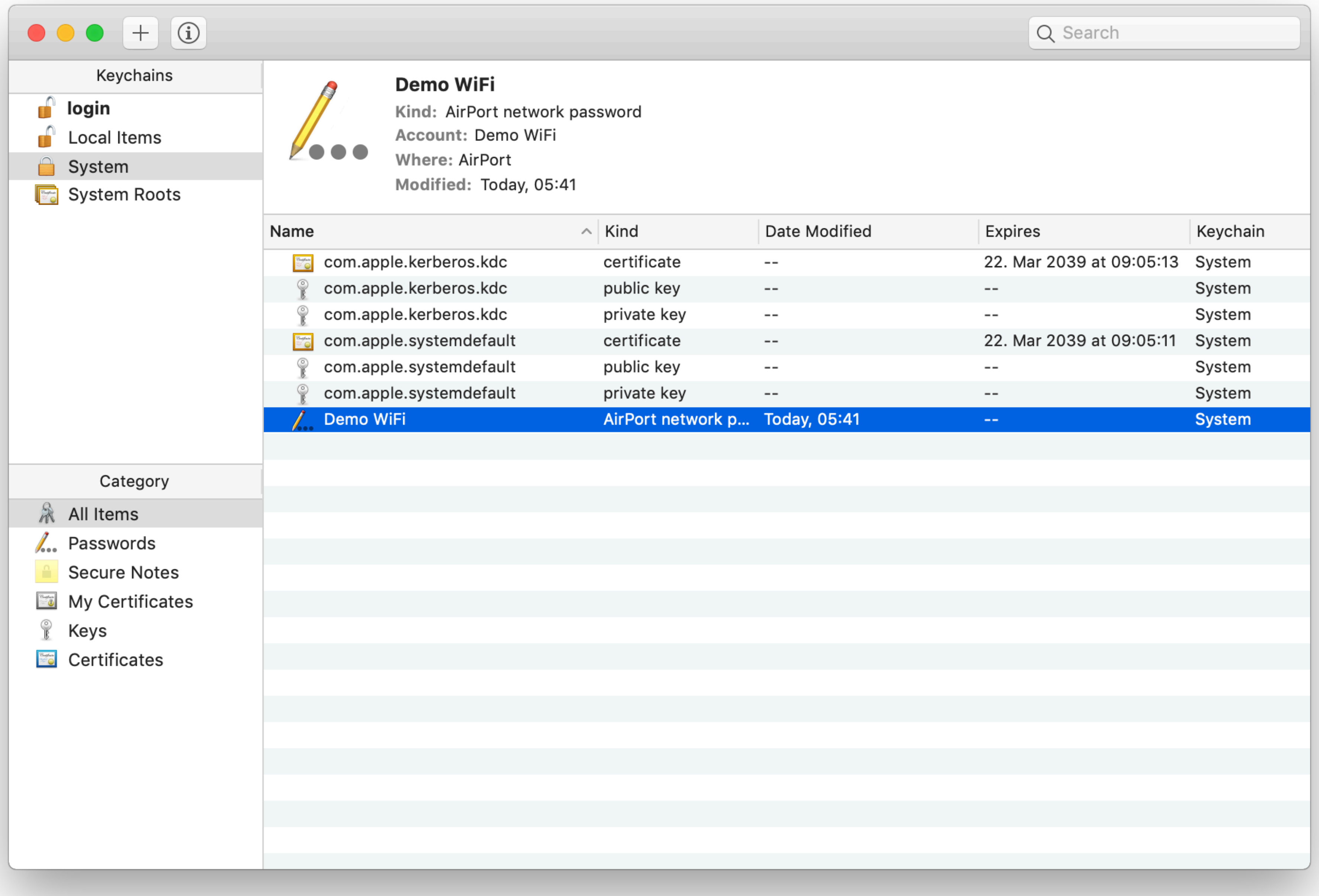
twitter.com
Kind: Internet password
Account: LinusHenze
Where: http://twitter.com
Modified: 11. Apr 2019 at 17:34:37

Name	Kind	Date Modified	Expires	Keychain
Demo Cert	certificate	--	10. Apr 2020 at 17:58:40	login
My Root CA	certificate	--	10. Apr 2020 at 17:56:20	login
My Root CA	public key	--	--	login
My Root CA	private key	--	--	login
My Super Secure Note	secure note	11. Apr 2019 at 17:49:50	--	login
Slack	application password	11. Apr 2019 at 17:34:52	--	login
twitter.com	Internet password	11. Apr 2019 at 17:34:37	--	login

LOGIN KEYCHAIN

SYSTEM KEYCHAIN

- System Keychain
  - Located in /Library/Keychains/System.keychain
- Encrypted using a per-device key
  - Key stored in /var/db/SystemKey, can only be read by root
- Mainly stores WiFi passwords and certificates
- Only accessible by administrators



Demo WiFi
Kind: AirPort network password
Account: Demo WiFi
Where: AirPort
Modified: Today, 05:41

Name	Kind	Date Modified	Expires	Keychain
com.apple.kerberos.kdc	certificate	--	22. Mar 2039 at 09:05:13	System
com.apple.kerberos.kdc	public key	--	--	System
com.apple.kerberos.kdc	private key	--	--	System
com.apple.systemdefault	certificate	--	22. Mar 2039 at 09:05:11	System
com.apple.systemdefault	public key	--	--	System
com.apple.systemdefault	private key	--	--	System
Demo WiFi	AirPort network p...	Today, 05:41	--	System

SYSTEM KEYCHAIN

ADVANTAGES/DISADVANTAGES

- Simple (and safe) way to store credentials
- Safe way to share credentials with other Apps
- Only need to remember the login password

- Single point of failure
- Large attack surface
  - Process responsible for the Keychain is doing a lot of things
- Metadata (e.g. usernames) stored unencrypted, only passwords/keys/secure notes are encrypted

KEYCHAIN

Accessing the Keychain

High Level API

KEYCHAIN ITEMS

- Every entry in the Keychain is a Keychain Item
- Keychain Items have an associated "class"
  - Internet Password, Generic Password
  - Key (public/private)
  - Certificate, Identity (Certificate + private Key)
- Depending on their class, Keychain Items may have multiple attributes
  - e.g. the Username and Server for Internet Passwords or a Type (used for Secure Notes, which are Generic Passwords)

ITEMS

USEFUL APIS

- SecItemCopyMatching: Allows you to search the keychain for items having certain attributes (e.g. class, username, server etc.)
- SecItemAdd: Create a new item with attributes
- SecItemDelete: Delete an item
- SecItemUpdate: Search for items and update them

USEFUL APIS

ACCESSING THE KEYCHAIN

```
import Foundation
import Security

/*
 * Setup our query
 * We want to get every Internet Password Item (without requesting the actual password as the user would need to allow
that)
 *
 * Class: Internet Password
 * Limit: None (return all Items that are of the Internet Password class)
 * Return Attributes: True so that we get the Account Names
 * Return Data: False because that would show a Keychain Prompt
 */
let query: [CFString: Any] = [kSecClass: kSecClassInternetPassword,
                              kSecMatchLimit: kSecMatchLimitAll,
                              kSecReturnAttributes: true,
                              kSecReturnData: false]
var items: CFTypeRef!
let status = SecItemCopyMatching(query as CFDictionary, &items) // Query the Keychain

guard status == errSecSuccess else {
    /* Proper error handling goes here... */
    fatalError("Failed to get Keychain Items")
}

print("Found the following Internet Accounts in your keychain:")

for item in items as! [[String: Any]] {
    let username = item["acct"] as? String ?? "<No username>"
    let server = item["srvr"] as? String ?? "<No server>"
    print("Username: \(username) - Server: \(server)")
}
```

ACCESSING THE KEYCHAIN

```
/*
 * Setup our query
 * We want to get every Internet Password Item (without requesting the actual password
 * as the
 * user would need to allow that)
 *
 * Class: Internet Password
 * Limit: None (return all Items that are of the Internet Password class)
 * Return Attributes: True so that we get the Account Names
 * Return Data: False because that would show a Keychain Prompt
 */


let query: [CFString: Any] = [kSecClass: kSecClassInternetPassword,
                              kSecMatchLimit: kSecMatchLimitAll,
                              kSecReturnAttributes: true,
                              kSecReturnData: false]
var items: CFTypeRef!
let status = SecItemCopyMatching(query as CFDictionary, &items) // Query the Keychain
```

ACCESSING THE KEYCHAIN

```
import Foundation
import Security

/*
 * Setup our query
 * We want to get every Internet Password Item (without requesting the actual password as the user would need to allow
that)
 *
 * Class: Internet Password
 * Limit: None (return all Items that are of the Internet Password class)
 * Return Attributes: True so that we get the Account Names
 * Return Data: False because that would show a Keychain Prompt
 */
let query: [CFString: Any] = [kSecClass: kSecClassInternetPassword,
                              kSecMatchLimit: kSecMatchLimitAll,
                              kSecReturnAttributes: true,
                              kSecReturnData: false]
var items: CFTypeRef!
let status = SecItemCopyMatching(query as CFDictionary, &items) // Query the Keychain

guard status == errSecSuccess else {
    /* Proper error handling goes here... */
    fatalError("Failed to get Keychain Items")
}

print("Found the following Internet Accounts in your keychain:")

for item in items as! [[String: Any]] {
    let username = item["acct"] as? String ?? "<No username>"
    let server = item["srvr"] as? String ?? "<No server>"
    print("Username: \(username) - Server: \(server)")
}
```

KEYCHAIN INTERNALS

HOW IT WORKS

KEYCHAIN INTERNALS

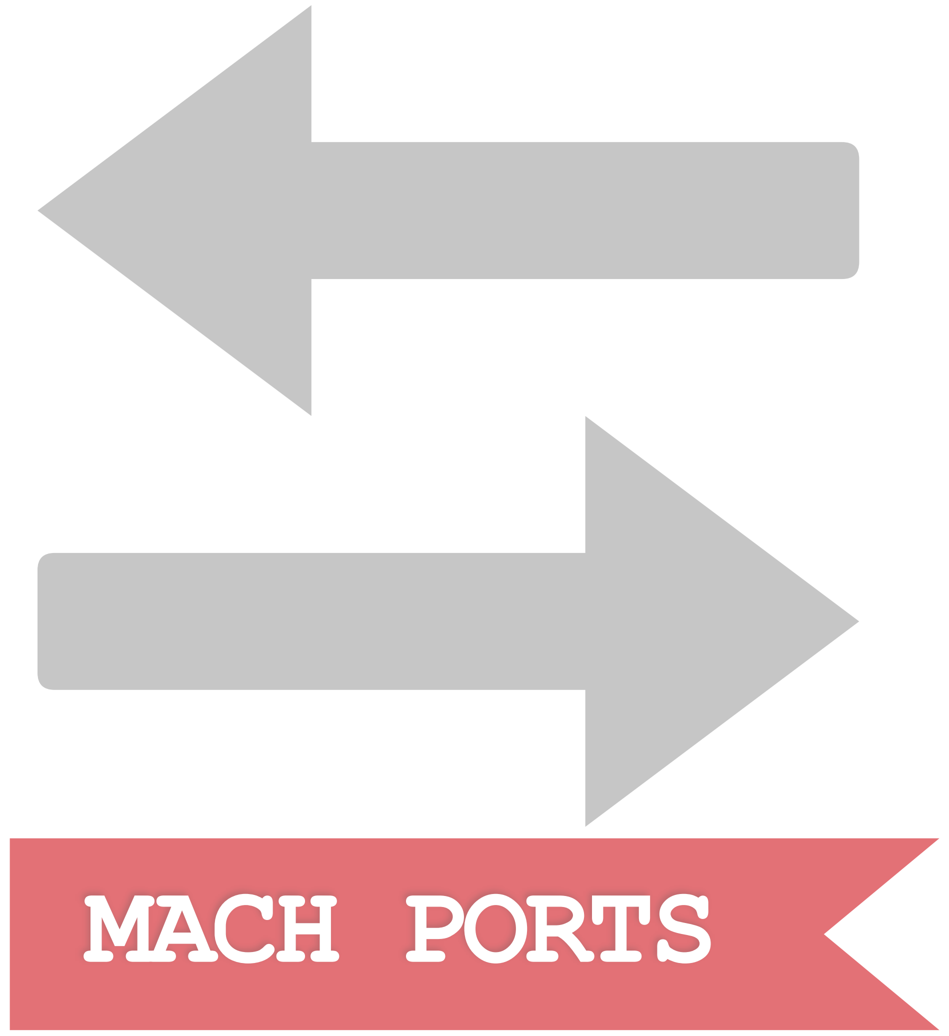
- securityd is the daemon responsible for the keychain
  - Holds encryption keys for all unlocked keychains
  - Performs access control
- Security Framework (implementing the high level keychain APIs) communicates with securityd through low level MIG APIs
  - MIG (Mach Interface Generator): Implements RPC over mach messages

MR. KEYCHAIN

Communication between Apps and securityd

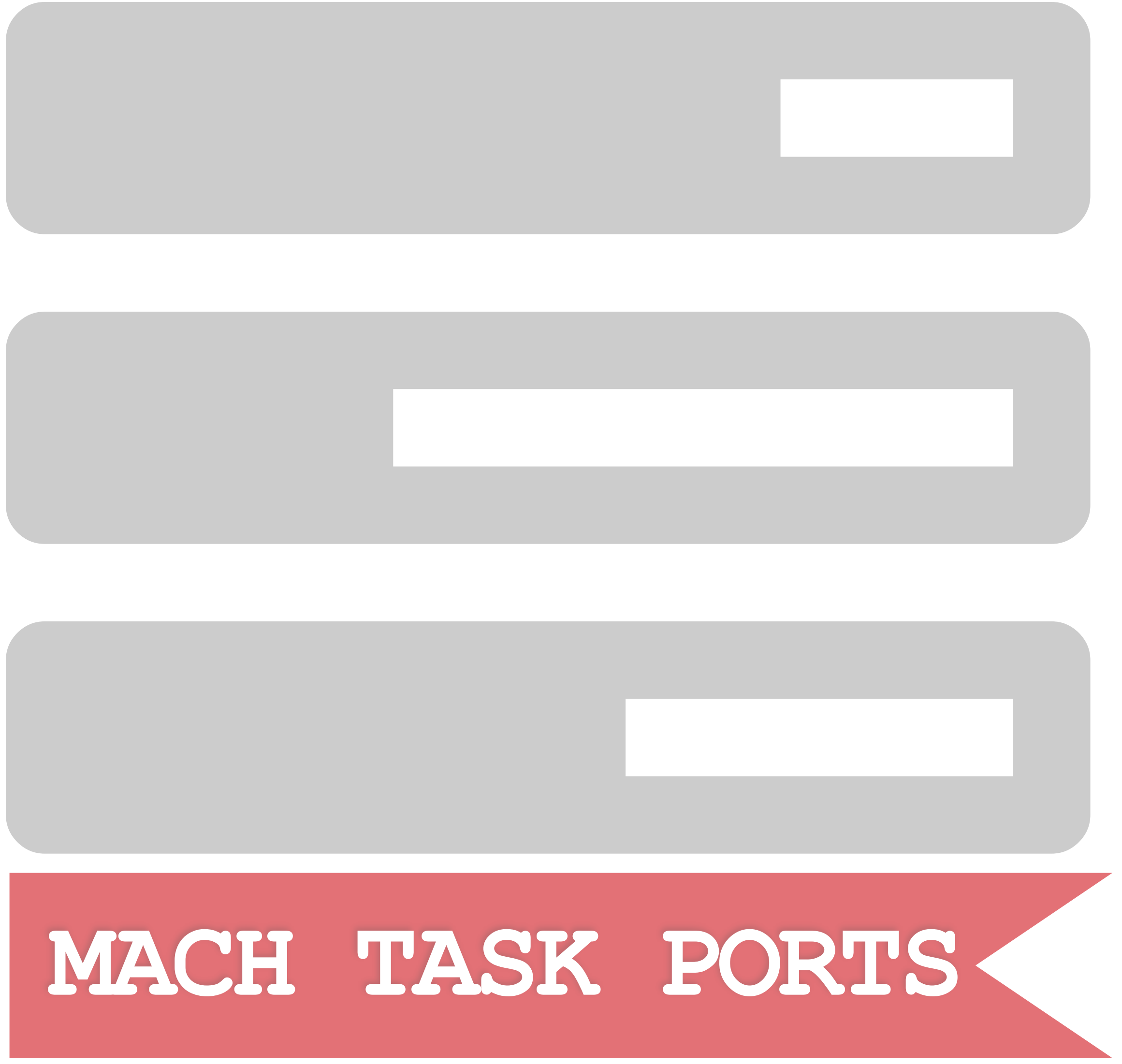
MACH PORTS

- Message queues, implemented by the kernel
- Works like a mailbox
  - Many senders, each holding a send right
  - Exactly one receiver, holding the receive right
    - Receiver has to tell other processes where they should send their messages to
- Referenced by mach port "names"
  - Integers, at least in userspace
  - Unique for very process



MACH TASK PORTS

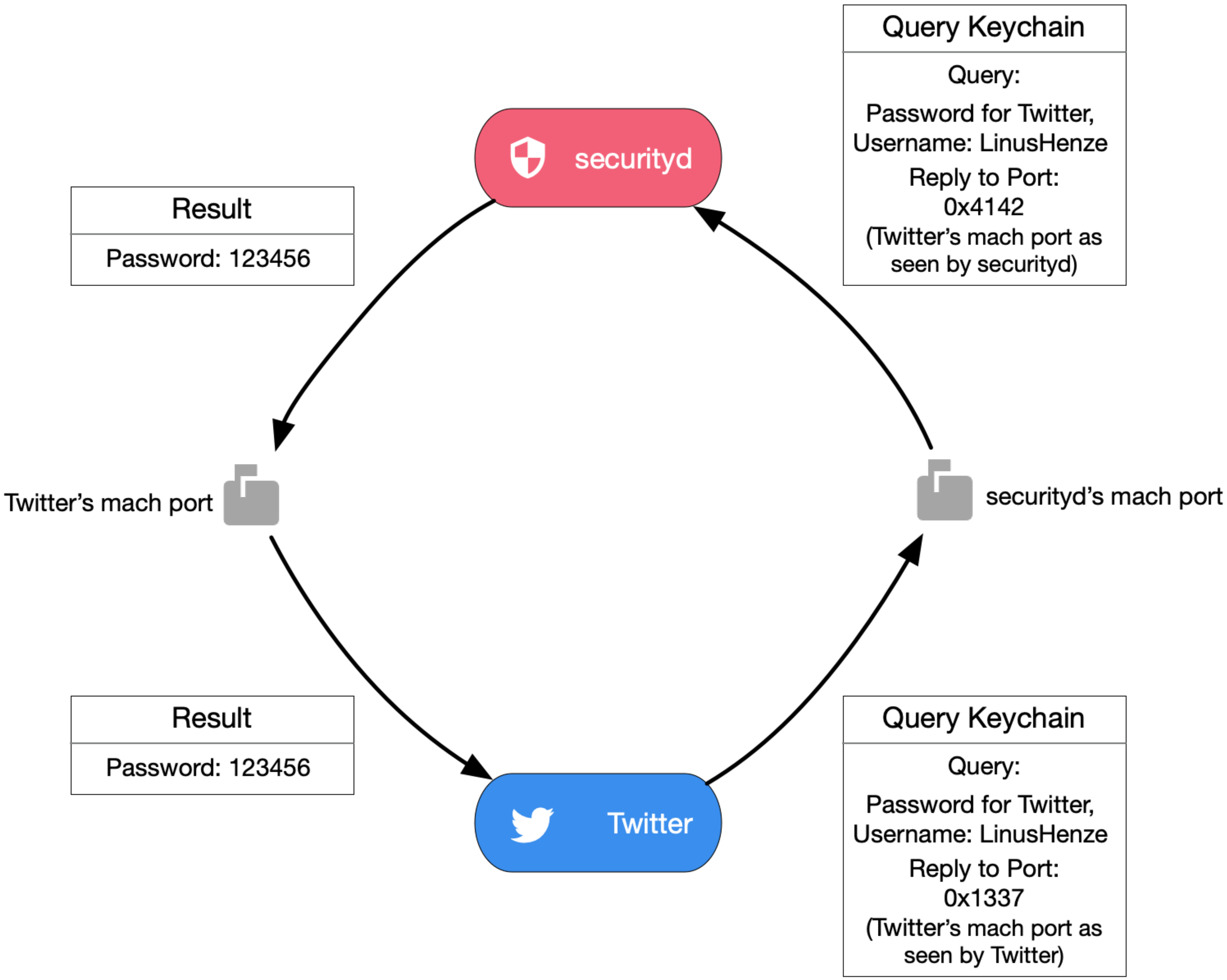
- Special type of mach port
- Every process has one
- Kernel listens on these ports
- Allows to modify the process
  - i.e. map/unmap/modify memory and other stuff
- Can be used to identify a process
- Automatically deleted once the process dies



MACH MESSAGES

- Structured data sent to a mach port
  - Header: Basic information like where to send the message to, size, message ID and an optional reply port
  - Body: May contain send/receive rights for mach ports and arbitrary data
- Queued in the kernel until retrieved by the receiver
  - Unless there are already too many messages in the queue...

MESSAGES



Query Keychain

Query:

Password for Twitter,
Username: LinusHenze

Reply to Port:
0x1337
(Twitter’s mach port as
seen by Twitter)

Query Keychain

Query:

Password for Twitter,
Username: LinusHenze

Reply to Port:
0x4142
(Twitter’s mach port as
seen by securityd)

Result

Password: 123456

Result

Password: 123456

SECURITYD EXAMPLE*

*simplified, in reality it's not that easy...

SECURITYD - MACH PORT

- How do we get a send right to securityd's mach port?

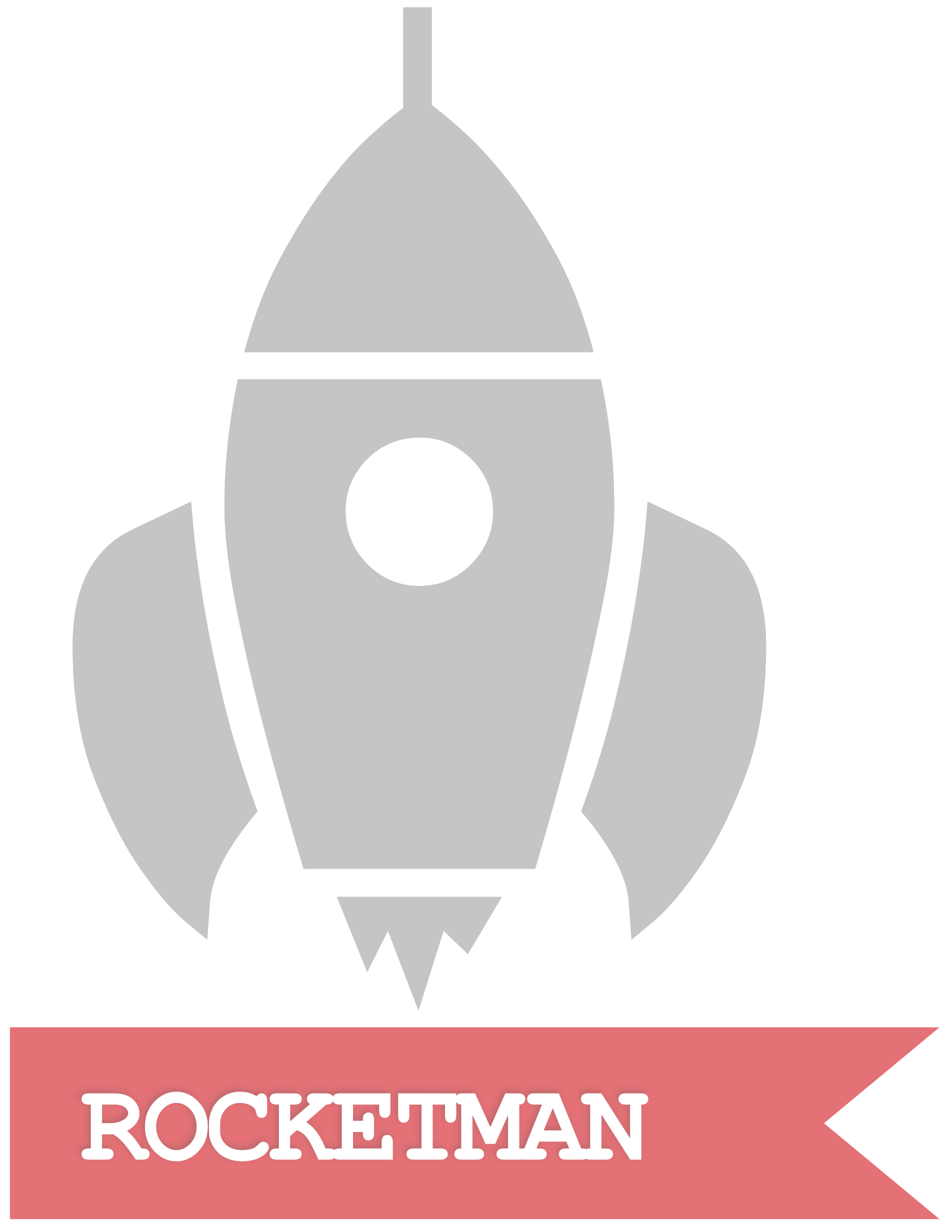
SECURITYD

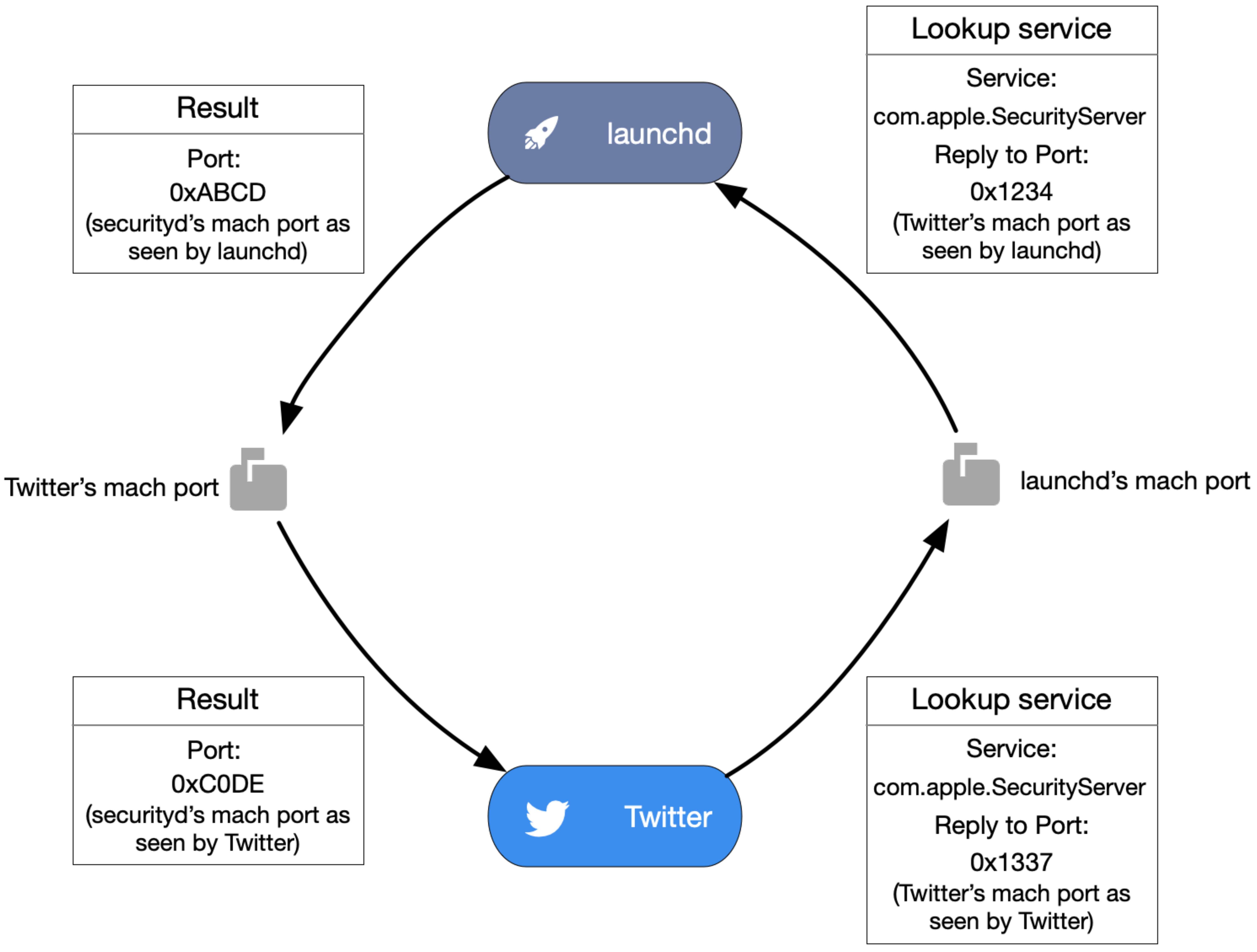
SECURITYD - MACH PORT

- How do we get a send right to securityd's mach port?
    - Through launchd!

LAUNCHD

- Init process on macOS
- Every process inherits a "bootstrap port" from it's parent
  - This is almost always launchd's mach port
- All services register with launchd
  - Just need to ask launchd to give us a send right to securityd's mach port





Lookup service

Service:
com.apple.SecurityServer
Reply to Port:
0x1337
(Twitter's mach port as
seen by Twitter)

Lookup service

Service:
com.apple.SecurityServer
Reply to Port:
0x1234
(Twitter's mach port as
seen by launchd)

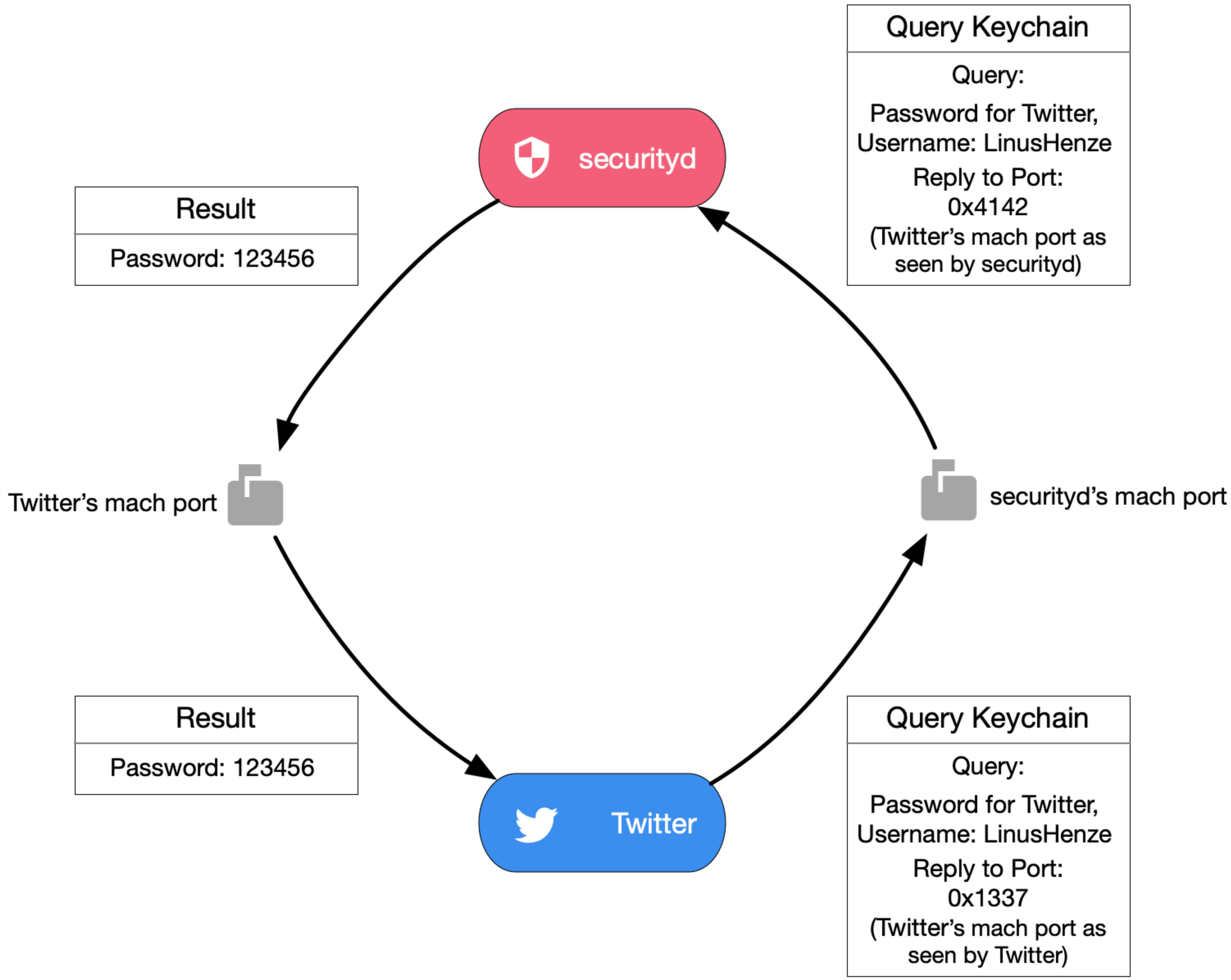
Result

Port:
0xABCD
(securityd's mach port as
seen by launchd)

Result

Port:
0xC0DE
(securityd's mach port as
seen by Twitter)

LAUNCHD EXAMPLE



Query Keychain

Query:

Password for Twitter,
Username: LinusHenze

Reply to Port:
0x1337
(Twitter's mach port as seen by Twitter)

Query Keychain

Query:

Password for Twitter,
Username: LinusHenze

Reply to Port:
0x4142
(Twitter's mach port as seen by securityd)

Result

Password: 123456

Result

Password: 123456

SECURITYD EXAMPLE*

*simplified, in reality it's not that easy...

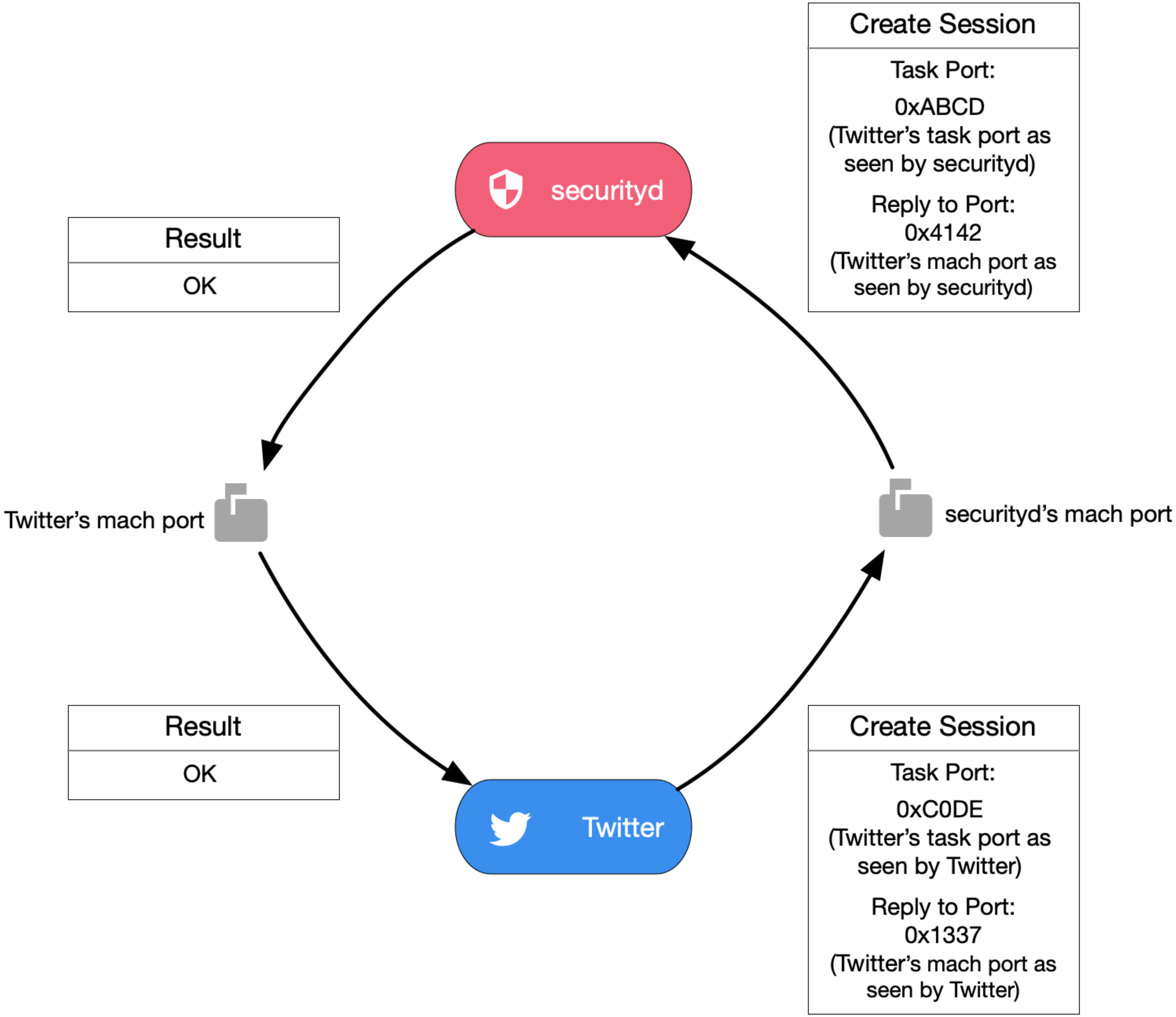
Keychain Access Control

KEYCHAIN ACCESS CONTROL

- Each Keychain Item has an ACL (Access Control List)
  - List of applications that may access the item without a password prompt
- Can only be changed by the user or Apps already in the item's ACL
- But how is it enforced?

KEYCHAIN ACCESS CONTROL

- Each Keychain Item has an ACL (Access Control List)
    - List of applications that may access the item without a password prompt
- Can only be changed by the user or Apps already in the item's ACL
- But how is it enforced?
    - By requiring Apps to submit their task port before being allowed to do anything else



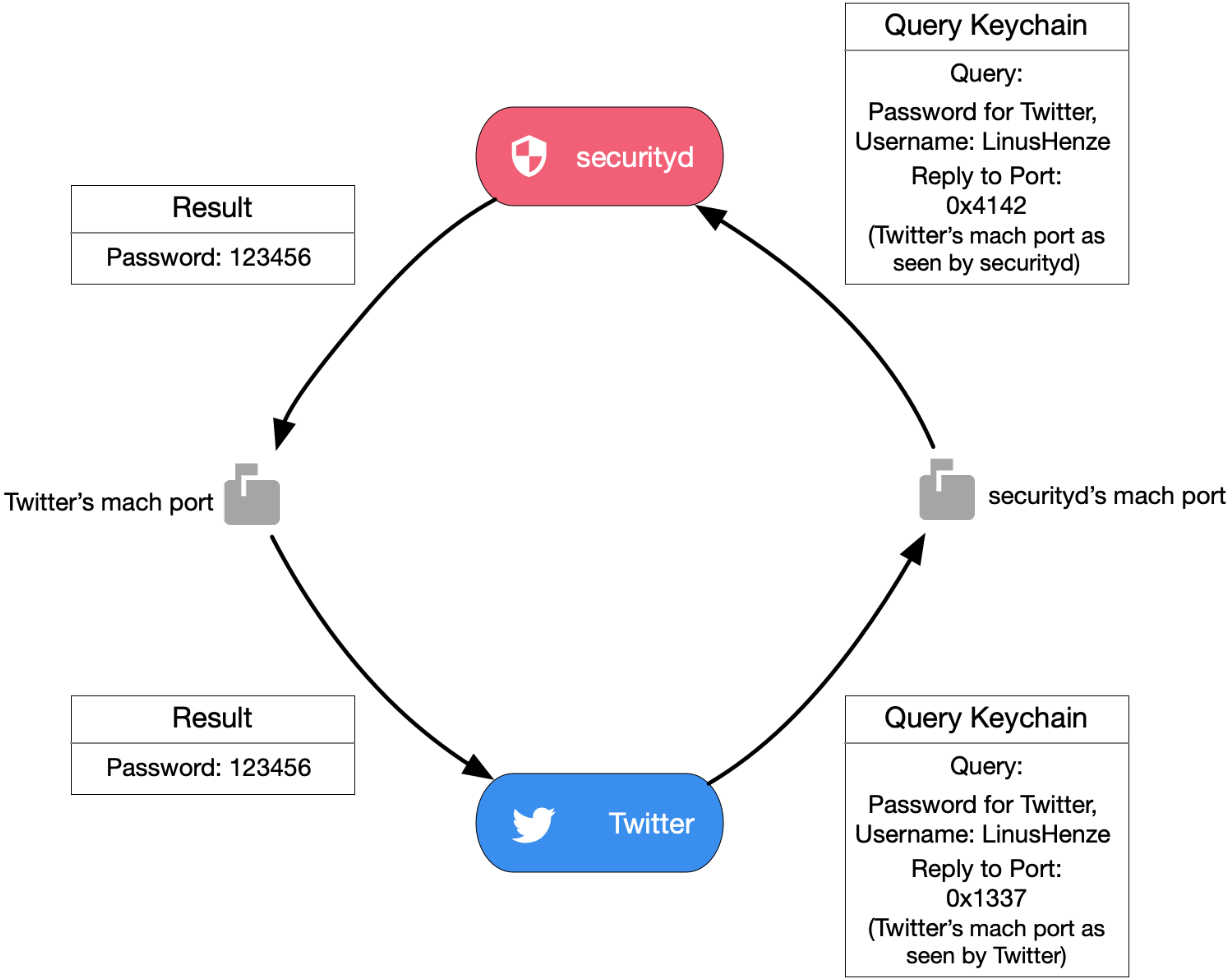
Create Session
Task Port: 0xABCD (Twitter's task port as seen by securityd) Reply to Port: 0x4142 (Twitter's mach port as seen by securityd)

Result
OK

Result
OK

Create Session
Task Port: 0xC0DE (Twitter's task port as seen by Twitter) Reply to Port: 0x1337 (Twitter's mach port as seen by Twitter)

SECURITYD EXAMPLE



Query Keychain

Query:

Password for Twitter,
Username: LinusHenze

Reply to Port:
0x4142
(Twitter's mach port as seen by securityd)

Result

Password: 123456

Result

Password: 123456

Query Keychain

Query:

Password for Twitter,
Username: LinusHenze

Reply to Port:
0x1337
(Twitter's mach port as seen by Twitter)

SECURITYD EXAMPLE*

*simplified, in reality it's not that easy...

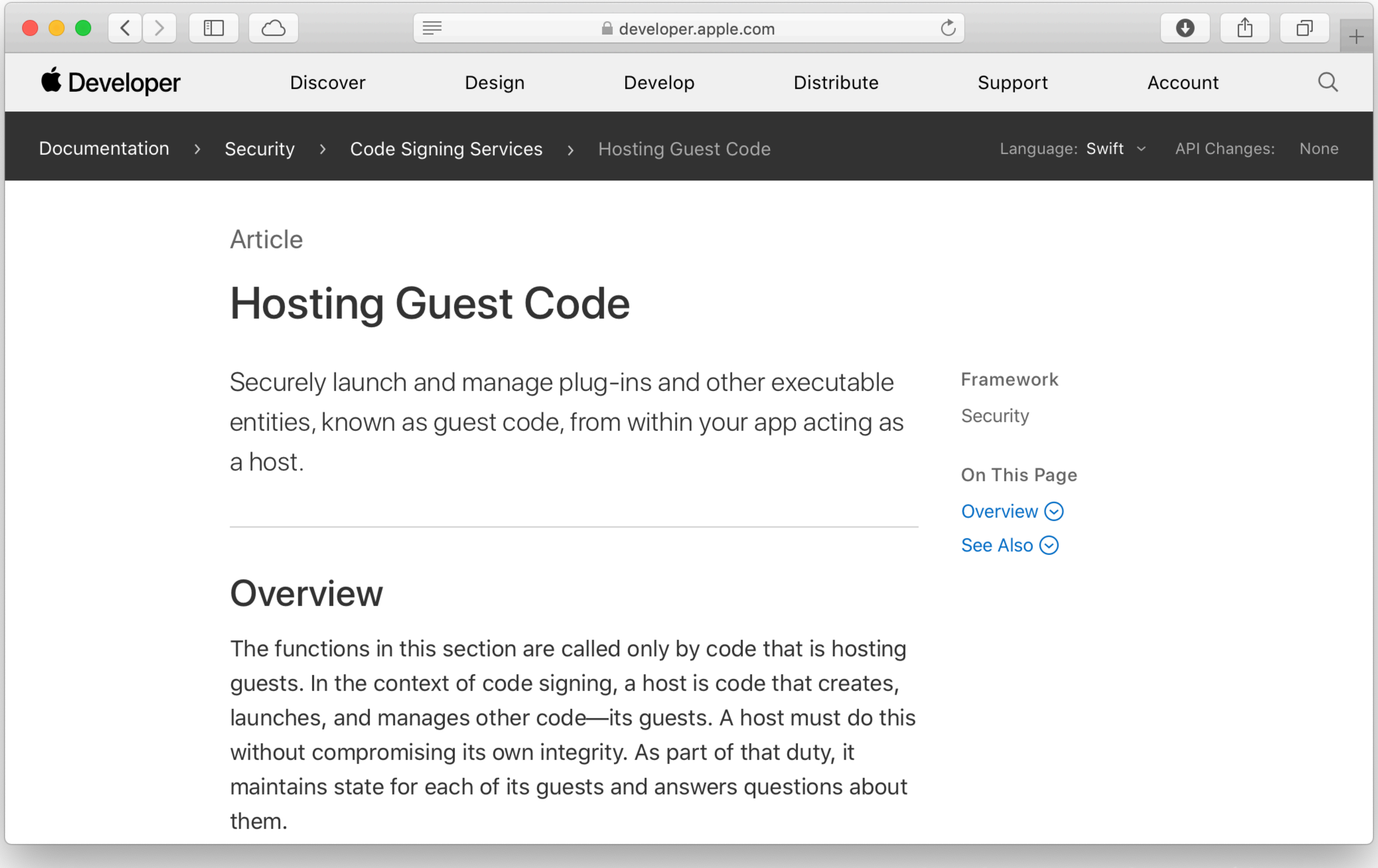
TIME TO EXPLOIT THE KEYCHAIN

KEYSTEAL VS KEYCHAIN

HOW I FOUND THE BUG

- Needed a sandbox escape (so I can do something cool when I find my next WebKit vulnerability)
- Looked into WebContent's sandbox profile
  - Is allowed to access the "com.apple.SecurityServer" service (securityd)
    - I just had to look into this (because of the name)
- It's not what I hoped for, but without this bug I wouldn't be here ;)

WHERE IS THE BUG



Article

Hosting Guest Code

Securely launch and manage plug-ins and other executable entities, known as guest code, from within your app acting as a host.

Framework

Security

On This Page

Overview

See Also

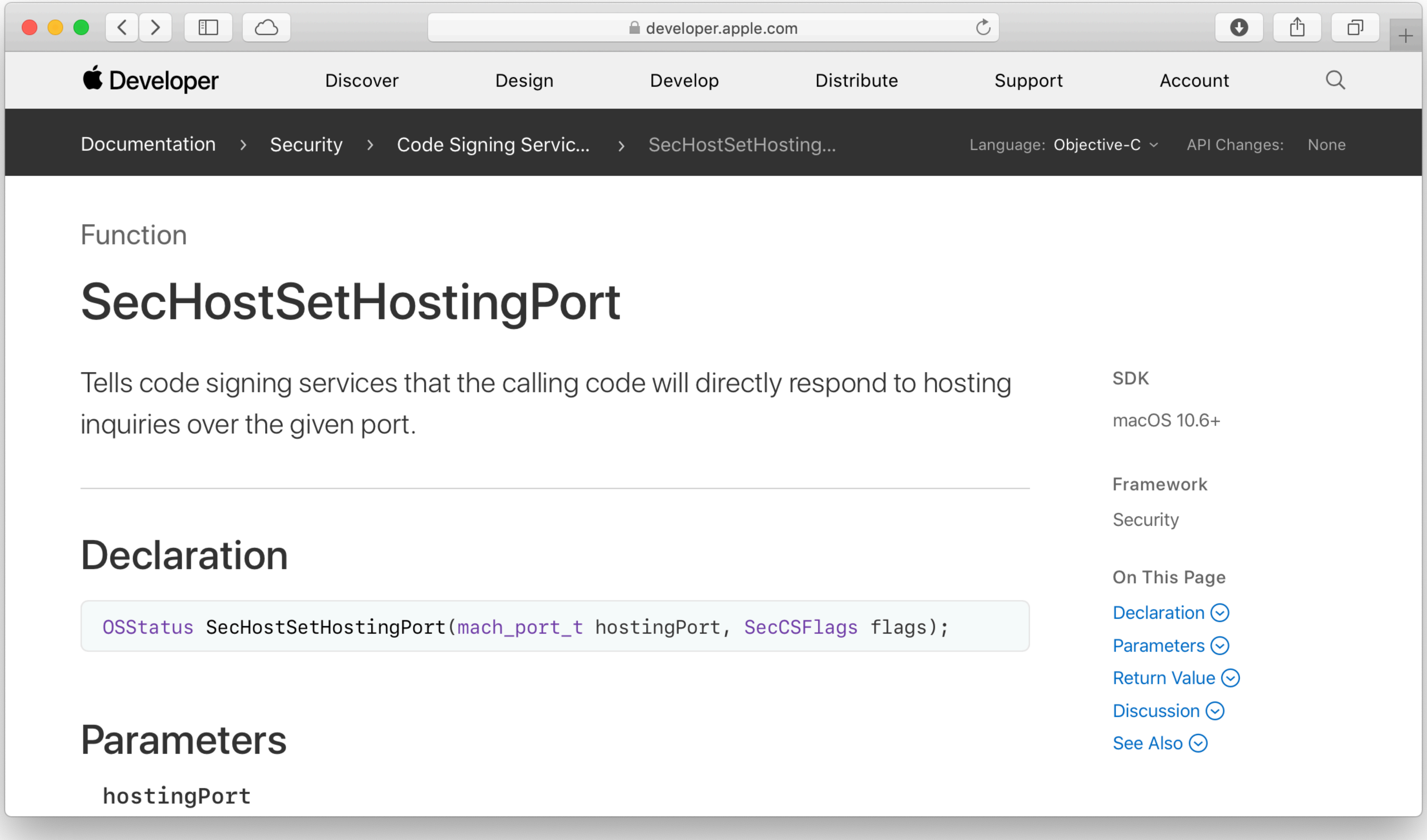
Overview

The functions in this section are called only by code that is hosting guests. In the context of code signing, a host is code that creates, launches, and manages other code—its guests. A host must do this without compromising its own integrity. As part of that duty, it maintains state for each of its guests and answers questions about them.

Source: developer.apple.com/documentation/security/code_signing_services/hosting_guest_code

HOSTING GUEST CODE

- Never heard of this feature?
  - Me neither!
- Implemented in securityd
- Apparently, you should be able to use it to host guest code and tell the system about it
  - But it's completely broken...
    - And also has a nice vulnerability



Function

SecHostSetHostingPort

Tells code signing services that the calling code will directly respond to hosting inquiries over the given port.

SDK

macOS 10.6+

Framework

Security

On This Page

- Declaration
- Parameters
- Return Value
- Discussion
- See Also

Declaration

```
OSStatus SecHostSetHostingPort(mach_port_t hostingPort, SecCSFlags flags);
```

Parameters

hostingPort

Source: developer.apple.com/documentation/security/1392127-sechostsethostingport

IMPLEMENTATION (SECURITYD)

```
//
// Register a hosting API service port where the host will dynamically
// answer hosting queries from interested parties. This switches the process
// to dynamic hosting mode, and is incompatible with proxy hosting.
//
void CodeSigningHost::registerCodeSigning(mach_port_t hostingPort, SecCSFlags flags)
{
  StLock<Mutex> _(mLock);
  switch (mHostingState) {
  case noHosting:
    mHostingPort = hostingPort;
    mHostingState = dynamicHosting;
    secnotice("SecServer", "%d host register: %d", mHostingPort.port(), mHostingPort.port());
    break;
  default:
    MacOSError::throwMe(errSecCSHostProtocolContradiction);
  }
}
```

Security-58286.220.15/securityd/src/csproxy.cpp

THE BUG

```
//
// Reset Code Signing Hosting state.
// This turns hosting off and clears all children.
//
void CodeSigningHost::reset()
{
  StLock<Mutex> _(mLock);
  switch (mHostingState) {
  case noHosting:
    break; // nothing to do
  case dynamicHosting:
    mHostingPort.destroy();
    mHostingPort = MACH_PORT_NULL;
    secnotice("SecServer", "%d host unregister", mHostingPort.port());
    break;
  case proxyHosting:
    Server::active().remove(*this);// unhook service handler
    mHostingPort.destroy(); // destroy receive right
    mHostingState = noHosting;
    mHostingPort = MACH_PORT_NULL;
    mGuests.erase(mGuests.begin(), mGuests.end());
    secnotice("SecServer", "%d host unregister", mHostingPort.port());
    break;
  }
}
```

Security-58286.220.15/securityd/src/csproxy.cpp

THE BUG

```
//
// Reset Code Signing Hosting state.
// This turns hosting off and clears all children.
//
void CodeSigningHost::reset()
{
  StLock<Mutex> _(mLock);
  switch (mHostingState) {
  case noHosting:
    break; // nothing to do
  case dynamicHosting:
    mHostingPort.destroy();
    mHostingPort = MACH_PORT_NULL;
    secnotice("SecServer", "%d host unregister", mHostingPort.port());
    break;
  case proxyHosting:
    Server::active().remove(*this);// unhook service handler
    mHostingPort.destroy(); // destroy receive right
    mHostingState = noHosting;
    mHostingPort = MACH_PORT_NULL;
    mGuests.erase(mGuests.begin(), mGuests.end());
    secnotice("SecServer", "%d host unregister", mHostingPort.port());
    break;
  }
}
```

Calls mach_port_destroy on our port!!!

Security-58286.220.15/securityd/src/csproxy.cpp

THE BUG

- We can give securityd a send right to an arbitrary port
- When our session is destroyed, mach_port_destroy is called on the port
    - Should have been mach_port_deallocate ...
- Causes ALL references to the port being destroyed instead of just one
    - Can be used to free an arbitrary port in securityd
        - And replace it afterwards...

THE BUG

ATTACK PLAN

- Free an arbitrary port in securityd
- ???

SECURITYD SESSIONS

- As I've already said, before being able to talk to securityd, we need to create a session
- Session is tied to the task port of your process
  - Free the task port -> Interesting stuff happens

SESSIONS

ATTACK PLAN

- Free task port of a process in securityd
- Force session to have a dangling task port
- ???

ATTACK PLAN

```
//
// Set up a new Connection. This establishes the environment (process et al) as needed
// and registers a properly initialized Connection object to run with.
// Type indicates how "deep" we need to initialize (new session, process, or connection).
// Everything at and below that level is constructed. This is straight-forward except
// in the case of session re-initialization (see below).
//
void Server::setupConnection(ConnectLevel type, Port replyPort, Port taskPort,
    const audit_token_t &auditToken, const ClientSetupInfo *info)
{
  Security::CommonCriteria::AuditToken audit(auditToken);

  // first, make or find the process based on task port
  RefPointer<Process> &proc = mProcesses[taskPort];
  if (proc && type == connectNewProcess) {
    // the client has amnesia - reset it
    proc->reset(taskPort, info, audit);
    proc->changeSession(audit.sessionId());
  }
  if (!proc) {
    if (type == connectNewThread) // client error (or attack)
      CssmError::throwMe(CSSM_ERRCODE_INTERNAL_ERROR);
    proc = new Process(taskPort, info, audit);
    notifyIfDead(taskPort);
    mPids[proc->pid()] = proc;
  }

  // now, establish a connection and register it in the server
  Connection *connection = new Connection(*proc, replyPort);
  if (mConnections.contains(replyPort))   // malicious re-entry attempt?
    CssmError::throwMe(CSSM_ERRCODE_INTERNAL_ERROR); //@@@ error code? (client error)
  mConnections[replyPort] = connection;
  notifyIfDead(replyPort);
}
```

Security-58286.220.15/securityd/src/server.cpp

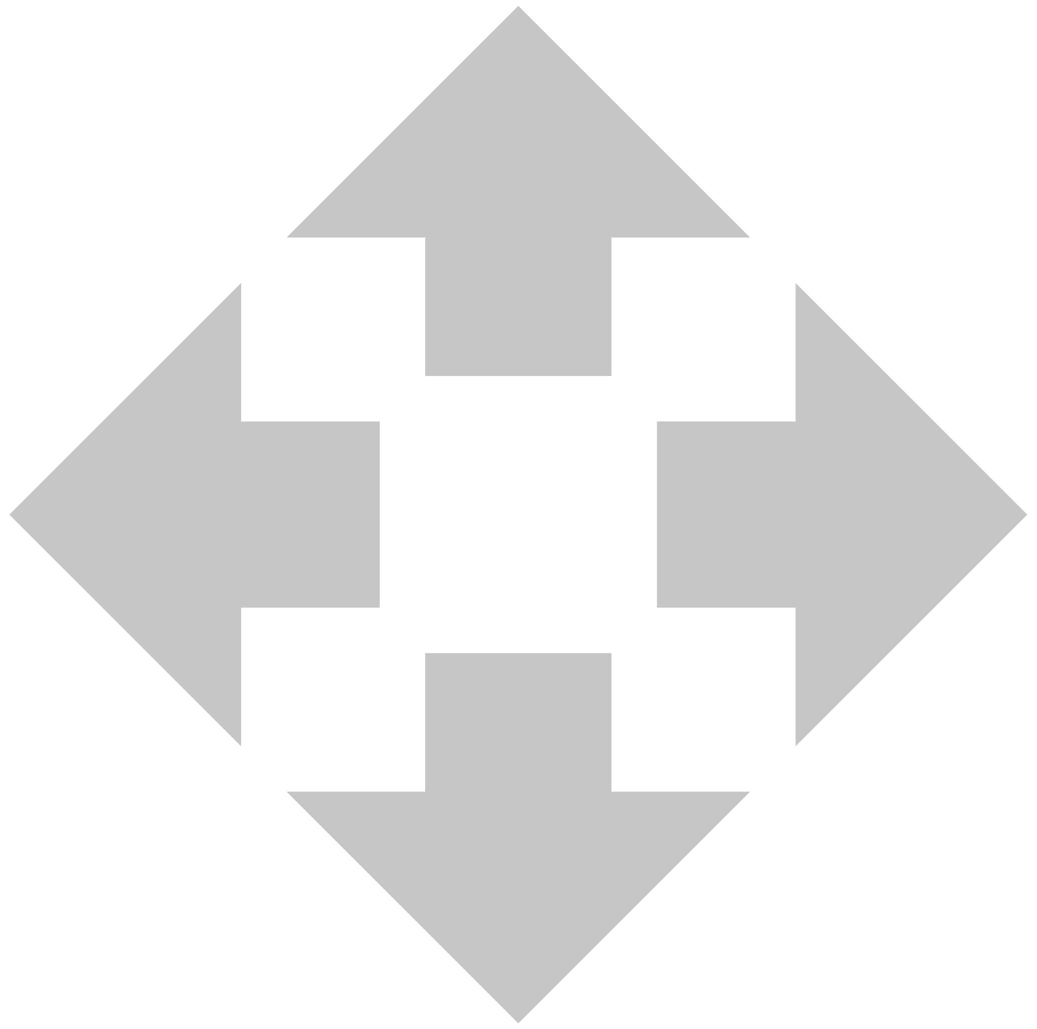
```
//
// Screen a process setup request for an existing process.
// This means the client has requested intialization even though we remember having
// talked to it in the past. This could either be an exec(2), or the client could just
// have forgotten all about its securityd client state. Or it could be an attack...
//
void Process::reset(TaskPort taskPort, const ClientSetupInfo *info, const CommonCriteria::AuditToken &audit)
{
  StLock<Mutex> _(*this);
  if (taskPort != mTaskPort) {
    secnotice("SecServer", "Process %p(%d) reset mismatch (tp %d-%d)",
      this, pid(), taskPort.port(), mTaskPort.port());
    //@@@ CssmError::throwMe(CSSM_ERRCODE_VERIFICATION_FAILURE);   // liar
  }
  setup(info);
  CFCopyRef<SecCodeRef> oldCode = processCode();

  // Note: The following will reload the code signature of the process
  // including all entitlements
  // HOWEVER, IT IS USING THE SAVED PID, NOT THE ONE OF THE PROCESS ASKING FOR REINITIALIZATION
  ClientIdentification::setup(this->pid());// re-constructs processCode()
  if (CFEqual(oldCode, processCode())) {
        secnotice("SecServer", "%p Client reset amnesia", this);
  } else {
        secnotice("SecServer", "%p Client reset full", this);
    CodeSigningHost::reset();
  }
}
```

Security-58286.220.15/securityd/src/process.cpp

ATTACK PLAN

- Free task port of a process in securityd
- Force session to have a dangling task port
- Reinitialize session, making sure it's PID is reused by a process allowed to access the Keychain without a password prompt
  - Must have the "com.apple.private.security.allow-migration" entitlement



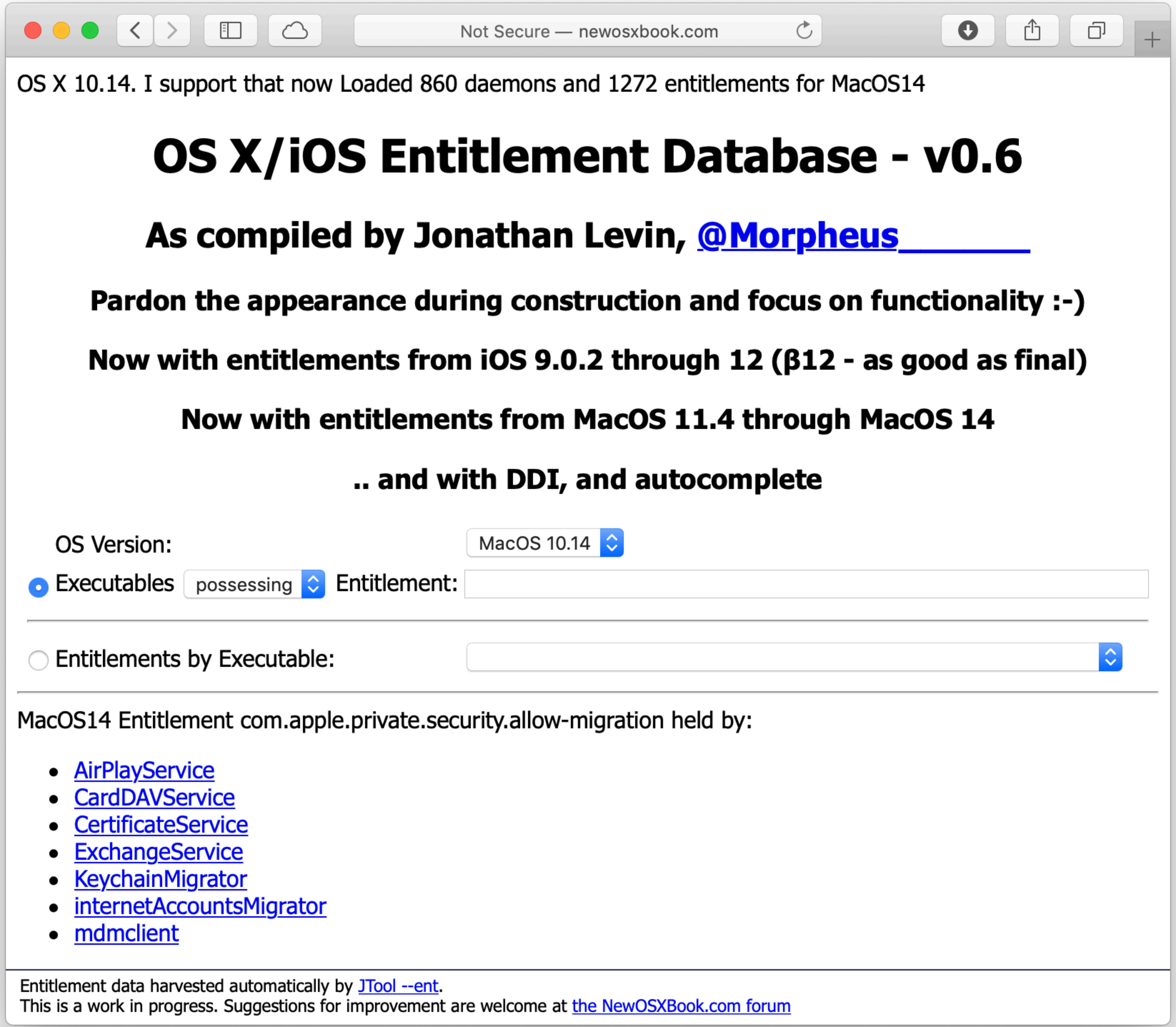
ATTACK PLAN

```
//
// If we have a KeychainPromptAclSubject, we want KeychainMigrator to have
// access even if we don't have the "pop ui" credential. Do the code signing
// check first, then process this ACL as normal.
//
bool KeychainPromptAclSubject::validates(const AclValidationContext &ctx) const
{
    Process &process = Server::process();
    if (process.checkAppleSigned() && process.hasEntitlement(migrationEntitlement)) {
        Syslog::info("bypassing keychain prompt for keychain migrator");
        secnotice("kcacl", "bypassing keychain prompt for keychain migrator");
        return true;   // migrator client -> automatic win
    }

    // Also, mark down that we evaluated a prompt ACL. We want to record this for testing even if the
client did not pass credentials for UI
    // (so that tests can disable prompts but still detect if one would have popped)
    promptsValidated++;

    return SimpleAclSubject::validates(ctx);
}
```

Security-58286.220.15/securityd/src/acl_keychain.cpp



OS X 10.14. I support that now Loaded 860 daemons and 1272 entitlements for MacOS14

OS X/iOS Entitlement Database - v0.6

As compiled by Jonathan Levin, @Morpheus______

Pardon the appearance during construction and focus on functionality :-)

Now with entitlements from iOS 9.0.2 through 12 (β12 - as good as final)

Now with entitlements from MacOS 11.4 through MacOS 14

.. and with DDI, and autocomplete

OS Version: MacOS 10.14

● Executables possessing Entitlement:

○ Entitlements by Executable:

MacOS14 Entitlement com.apple.private.security.allow-migration held by:

- AirPlayService
- CardDAVService
- CertificateService
- ExchangeService
- KeychainMigrator
- internetAccountsMigrator
- mdmclient

Entitlement data harvested automatically by JTool --ent.
This is a work in progress. Suggestions for improvement are welcome at the NewOSXBook.com forum

newosxbook.com/ent.jl?ent=com.apple.private.security.allow-migration&osVer=MacOS14

ATTACK PLAN

- Free task port of a process in securityd
- Force session to have a dangling task port
- Reinitialize session, making sure it's PID is reused by a process allowed to access the Keychain without a password prompt
  - Must have the "com.apple.private.security.allow-migration" entitlement
  - e.g. /System/Library/InternetAccounts/internetAccountsMigrator
- Access Keychain without password prompt!

FULL ACCESS

RECLAIM SESSION

- After freeing the task port, we won't have access to our session anymore
  - Need to reclaim our session
- Can be done by sending securityd a huge number of ports, hoping one of them gets the same number as our task port had
  - Use this new fake task port to access our session

RECLAIM SESSION

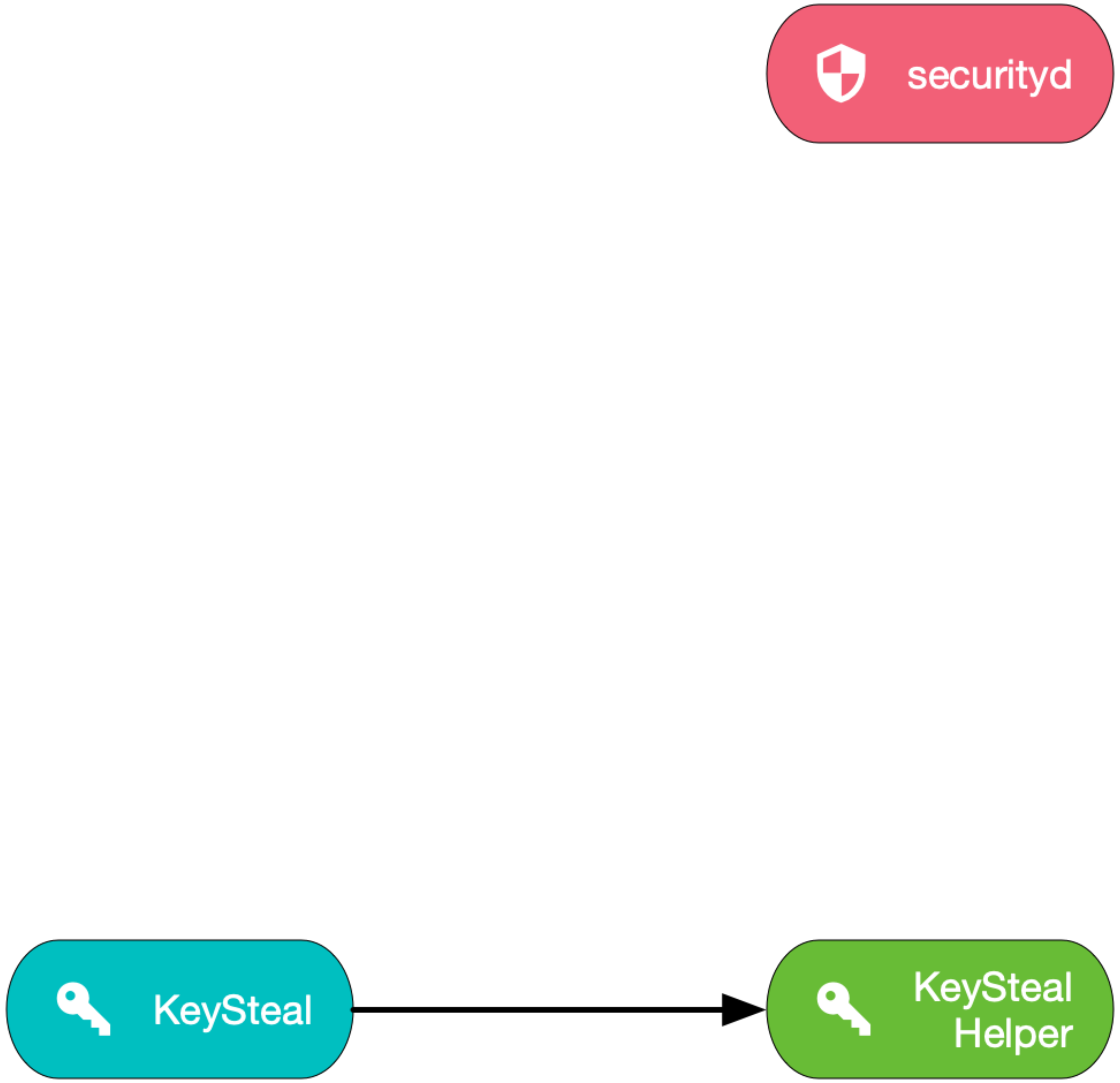
ATTACK PLAN

1. Create three processes: A, B and C
2. B should create a session with securityd
3. Send task port of B to C
4. Let C free B's task port in securityd
5. B should now reclaim it's session by sending securityd many ports, hoping one of them will get the same number as B's task port had
6. Send this fake task port to A (receive right!)
7. B should exec internetAccountsMigrator
    7.1. Reclaimed session won't be deleted as A now owns the fake task port which therefore won't be deleted
8. A can now reset B's session using the fake task port
    8.1. Causes the entitlements of internetAccounts migrator to be loaded
9. Use fake task port to access keychain!!!

securityd

KeySteal

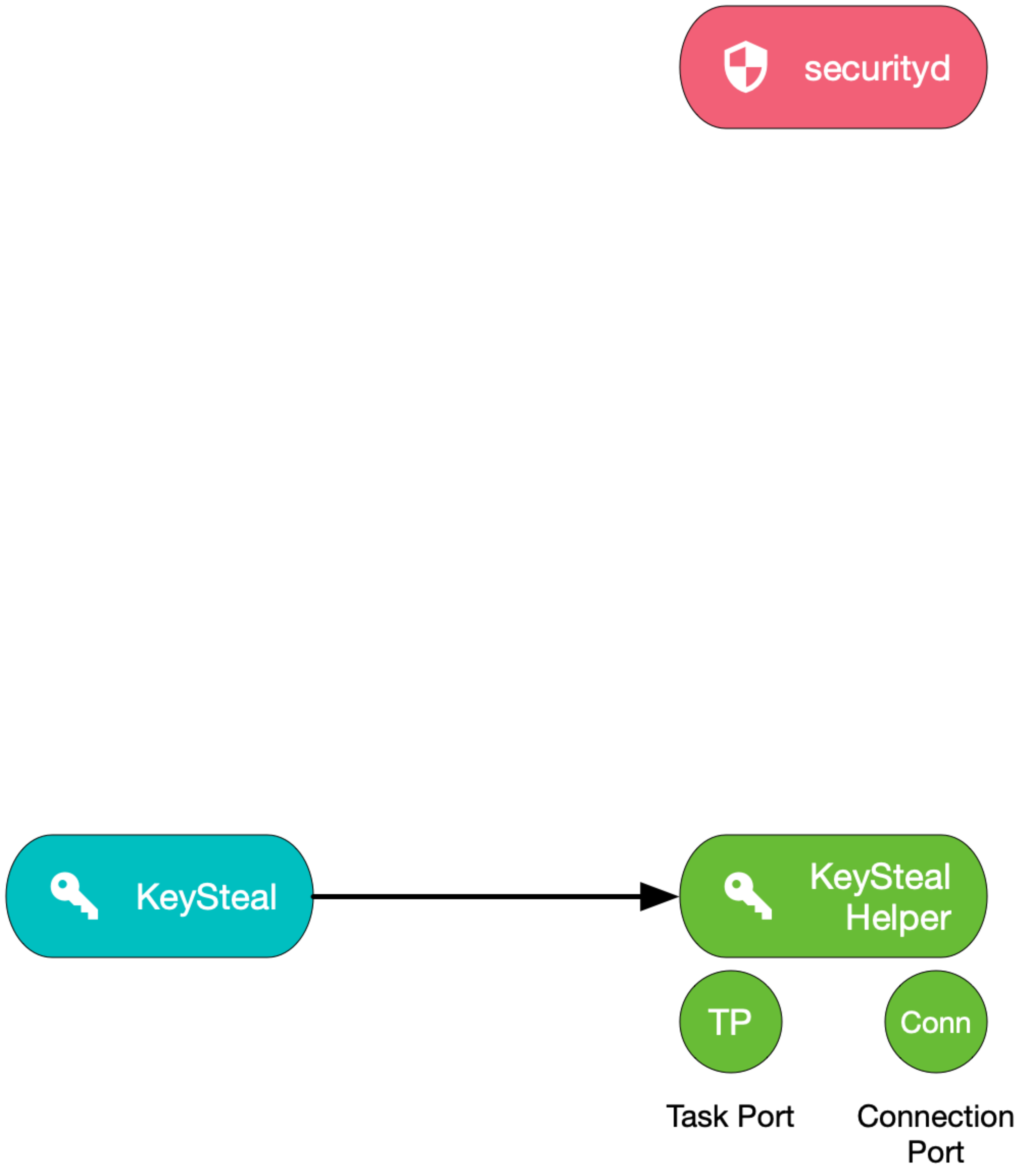
PREPARATION



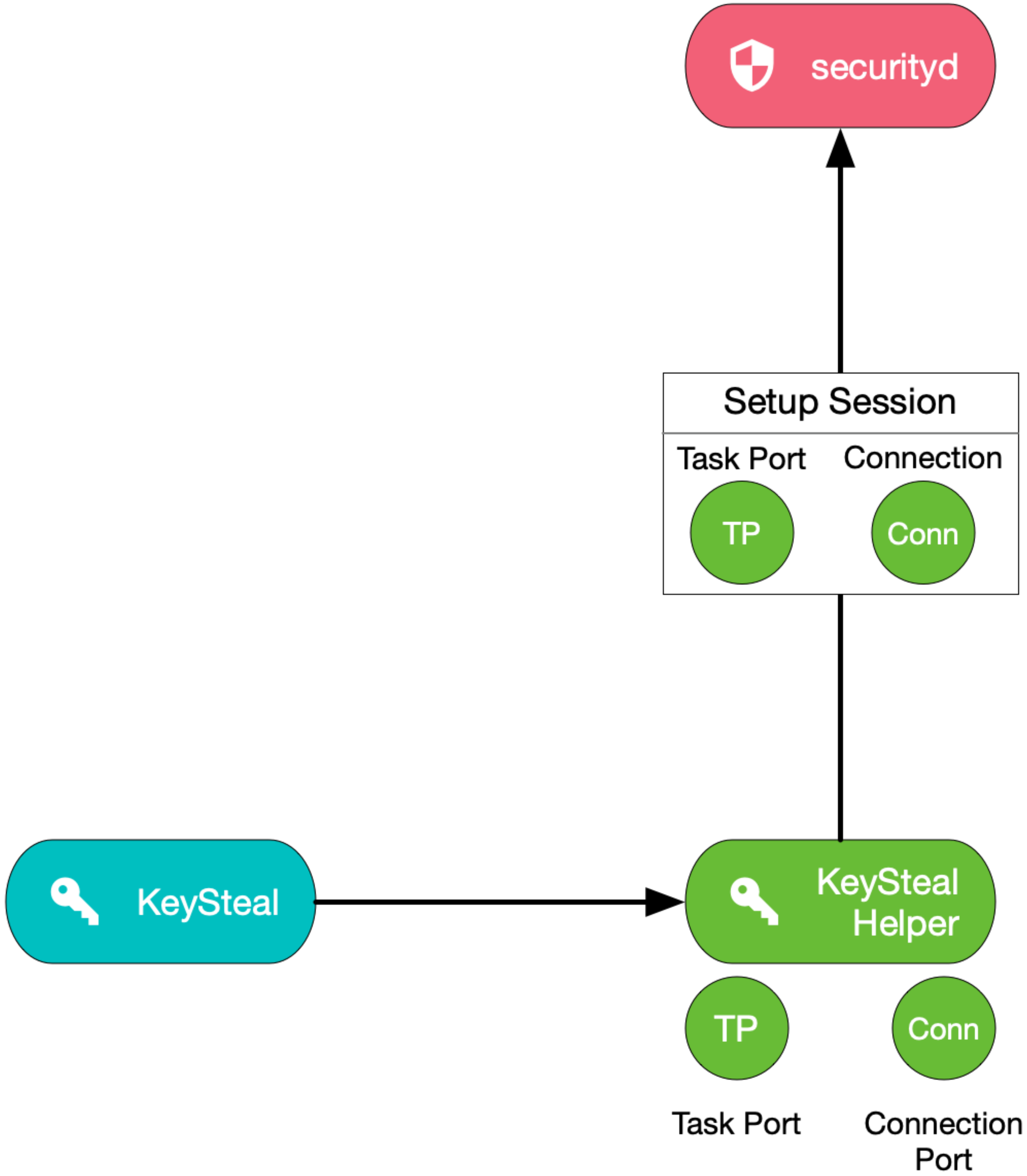
PREPARATION

ATTACK PLAN

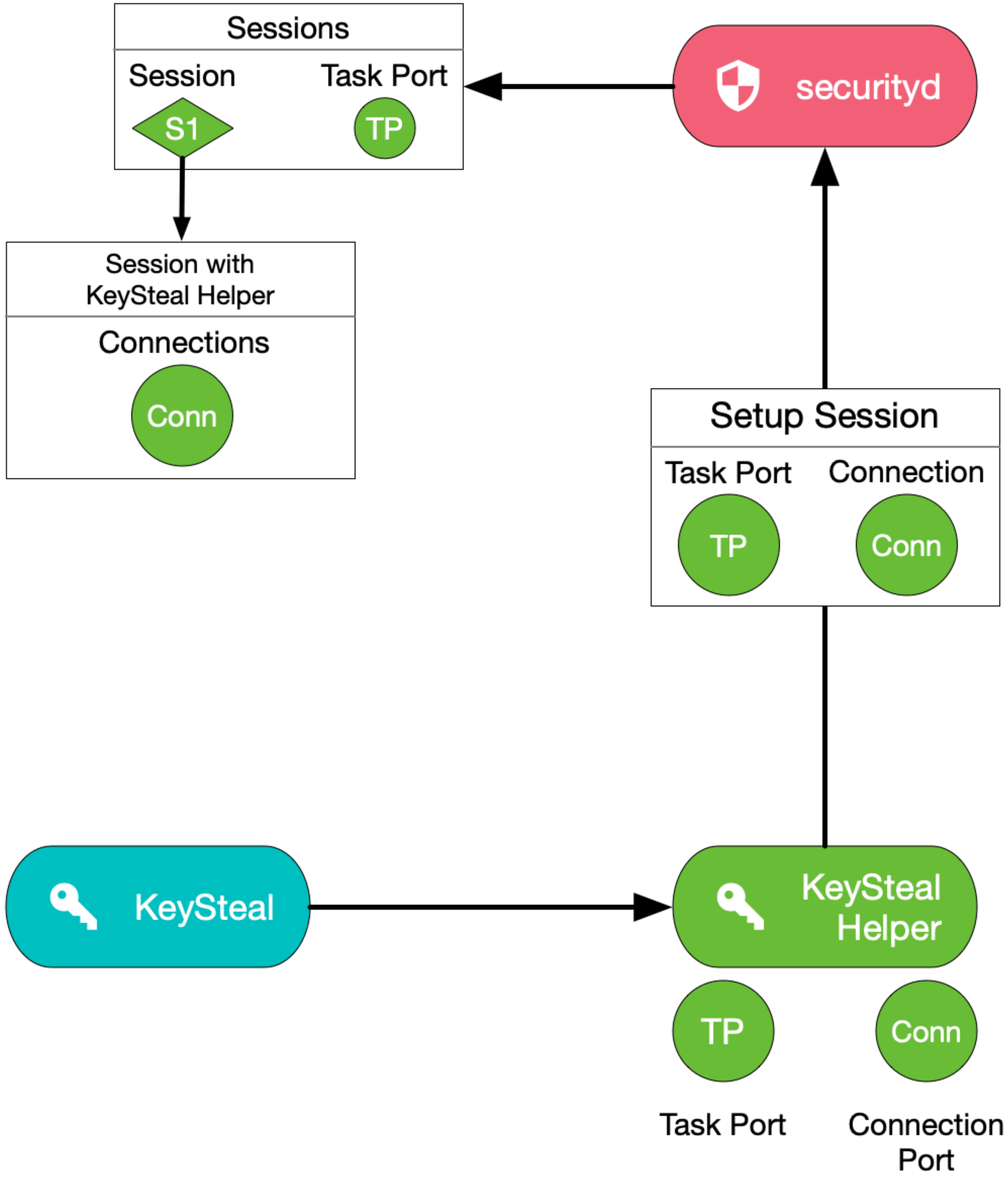
1. Create three processes: A, B and C
2. B should create a session with securityd ←


securityd
KeySteal
KeySteal Helper
TP
Conn
Task Port
Connection Port

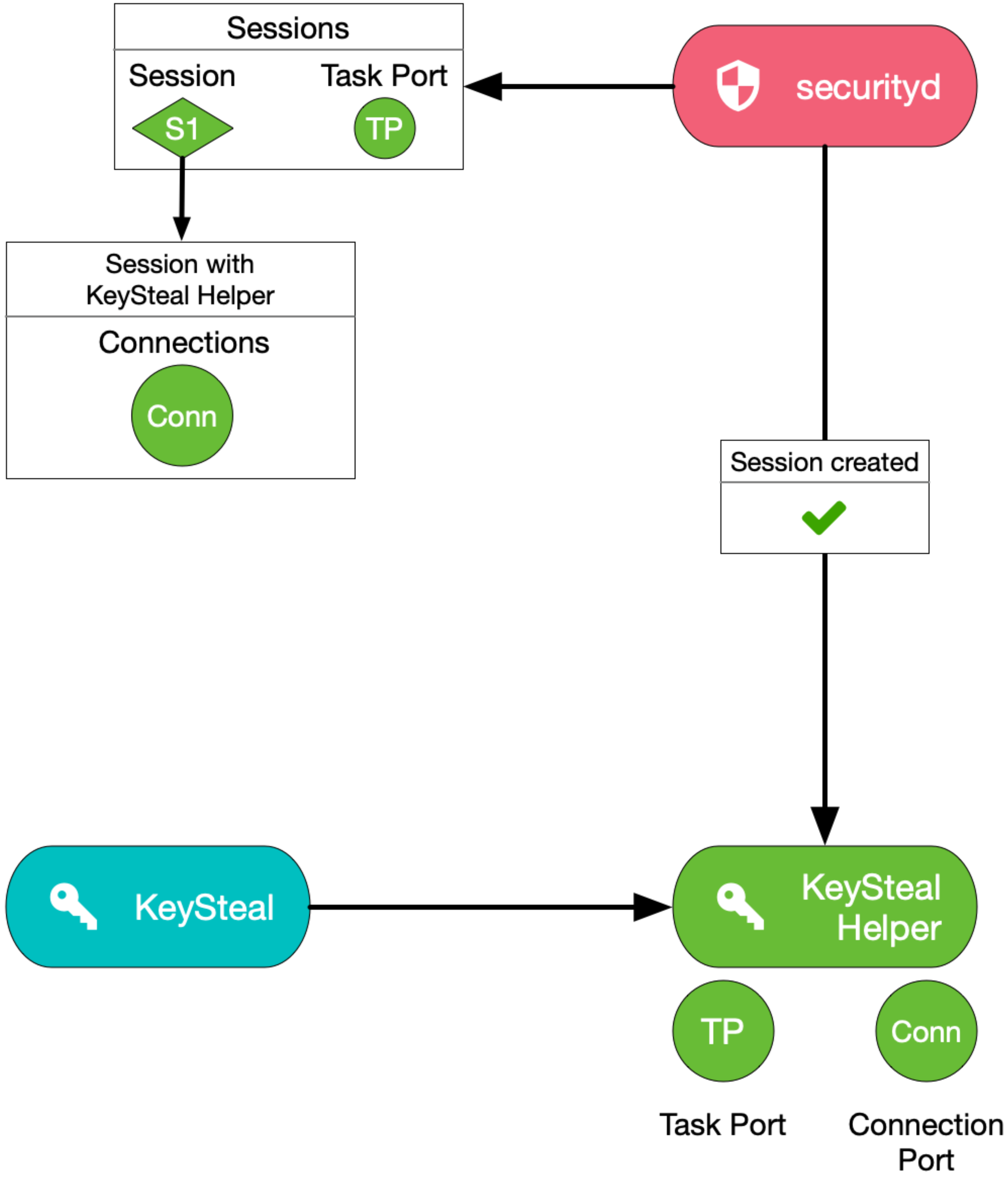
NEW SESSION


securityd
Setup Session
Task Port
Connection
TP
Conn
KeySteal
KeySteal Helper
TP
Conn
Task Port
Connection Port

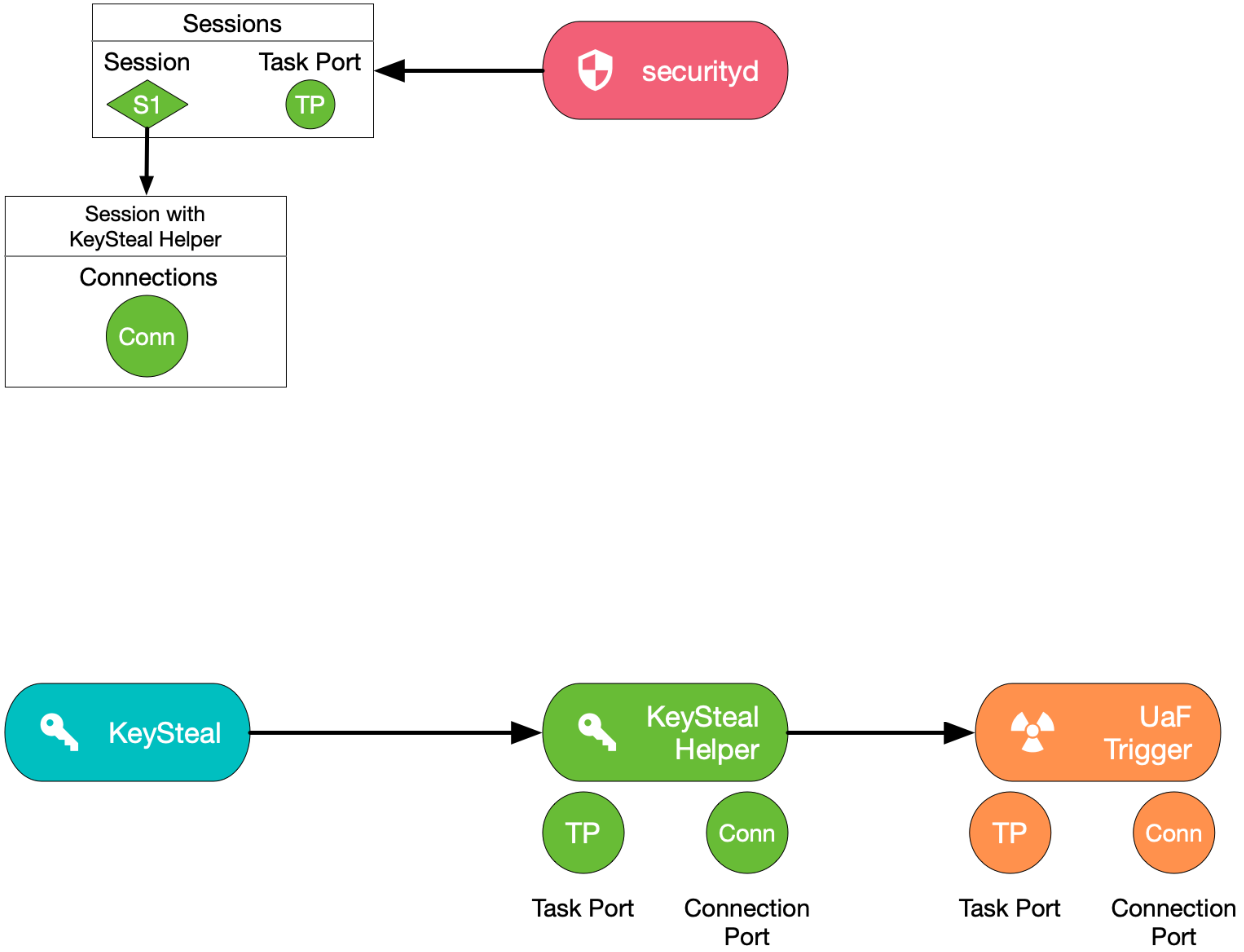
NEW SESSION


Sessions
Session
Task Port
S1
TP
securityd
Session with KeySteal Helper
Connections
Conn
Setup Session
Task Port
Connection
TP
Conn
KeySteal
KeySteal Helper
TP
Conn
Task Port
Connection Port

NEW SESSION


Sessions
Session
Task Port
S1
TP
securityd
Session with
KeySteal Helper
Connections
Conn
Session created
✓
KeySteal
KeySteal
Helper
TP
Conn
Task Port
Connection
Port

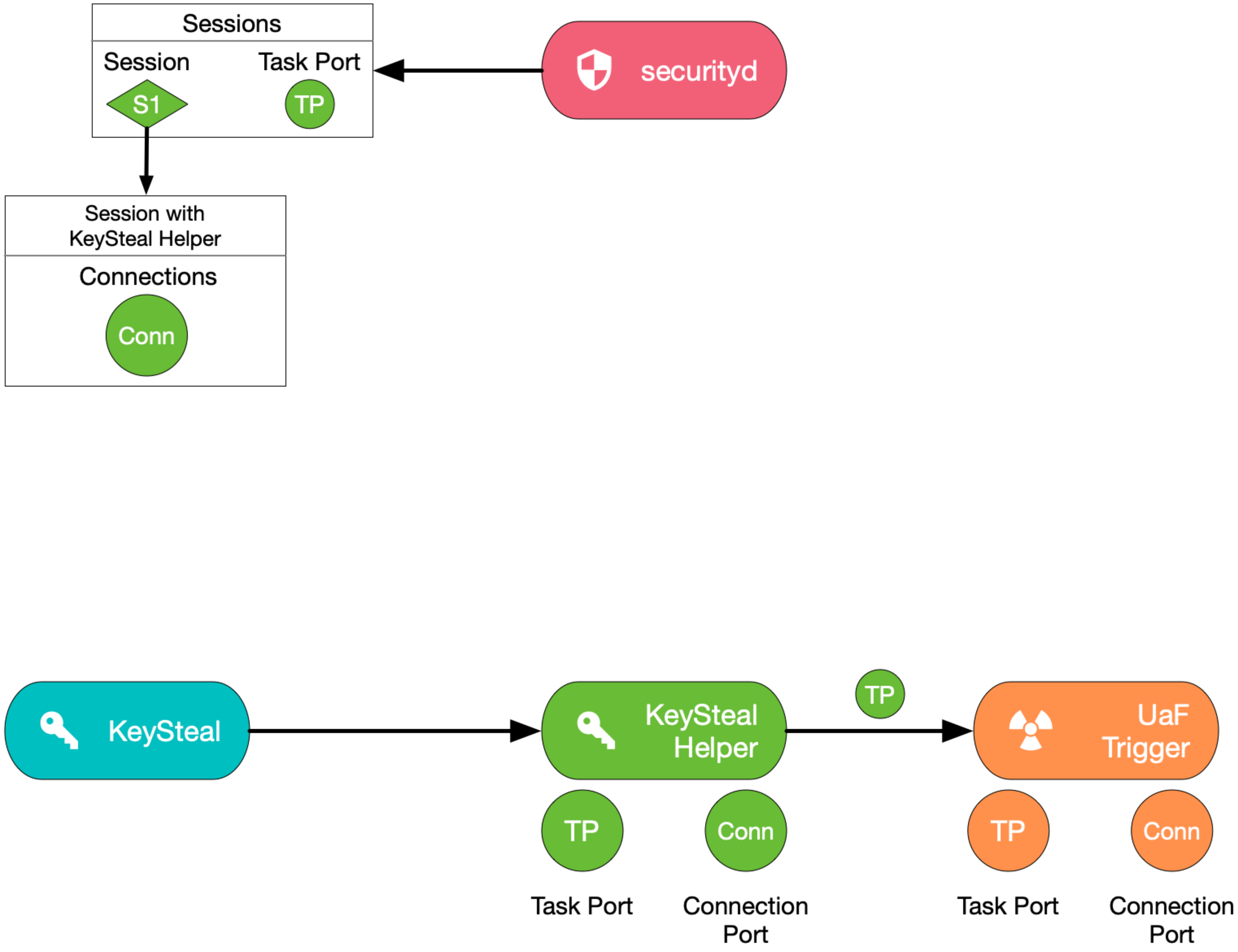
NEW SESSION


Sessions
Session
Task Port
S1
TP
securityd
Session with
KeySteal Helper
Connections
Conn
KeySteal
KeySteal
Helper
UaF
Trigger
TP
Conn
Task Port
Connection
Port
TP
Conn
Task Port
Connection
Port

TRIGGER

ATTACK PLAN

1. Create three processes: A, B and C ✓
2. B should create a session with securityd ✓
3. Send task port of B to C ←



Sessions

Session

Task Port

S1

TP

securityd

Session with KeySteal Helper

Connections

Conn

KeySteal

KeySteal Helper

TP

UaF Trigger

TP

Conn

Task Port

Connection Port

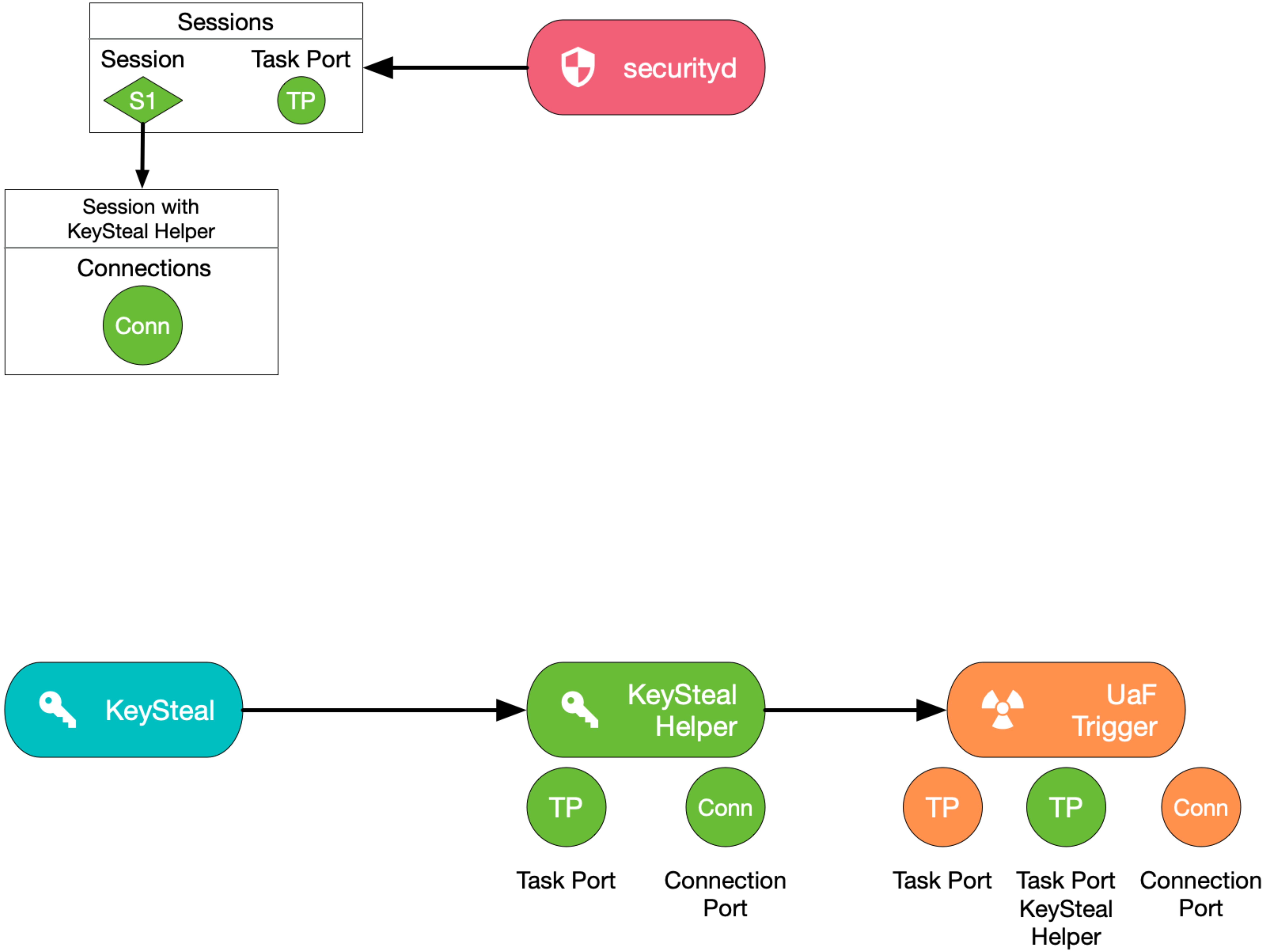
TP

Conn

Task Port

Connection Port

TRIGGER



Sessions

Session

S1

Task Port

TP

securityd

Session with
KeySteal Helper

Connections

Conn

KeySteal

KeySteal
Helper

TP

Task Port

Conn

Connection
Port

UaF
Trigger

TP

Task Port

TP

Task Port
KeySteal
Helper

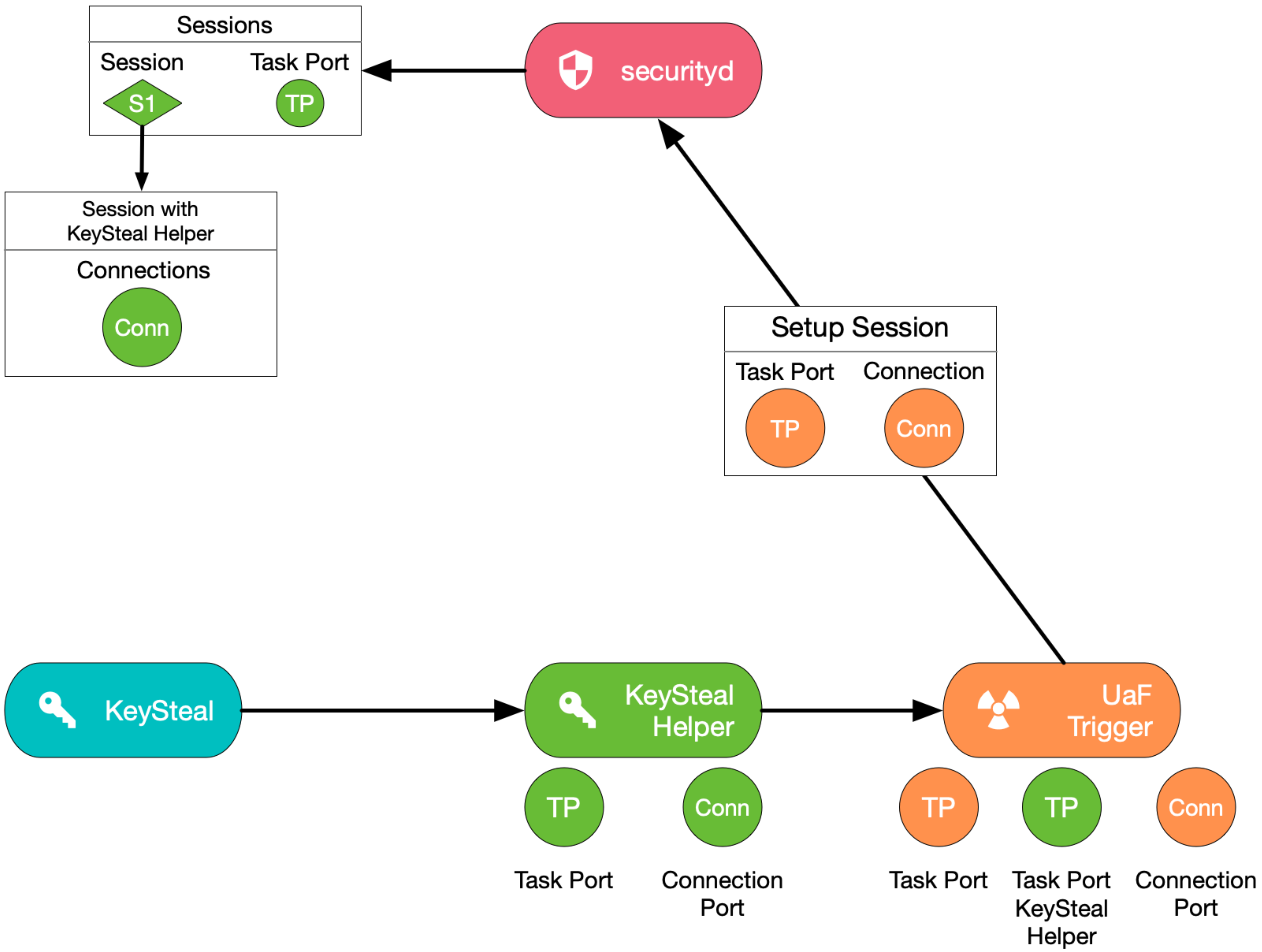
Conn

Connection
Port

TRIGGER

ATTACK PLAN

1. Create three processes: A, B and C ✓
2. B should create a session with securityd ✓
3. Send task port of B to C ✓
4. Let C free B's task port in securityd ←



Sessions

Session

S1

Task Port

TP

securityd

Session with KeySteal Helper

Connections

Conn

Setup Session

Task Port

TP

Connection

Conn

KeySteal

KeySteal Helper

TP

Task Port

Conn

Connection Port

UaF Trigger

TP

Task Port

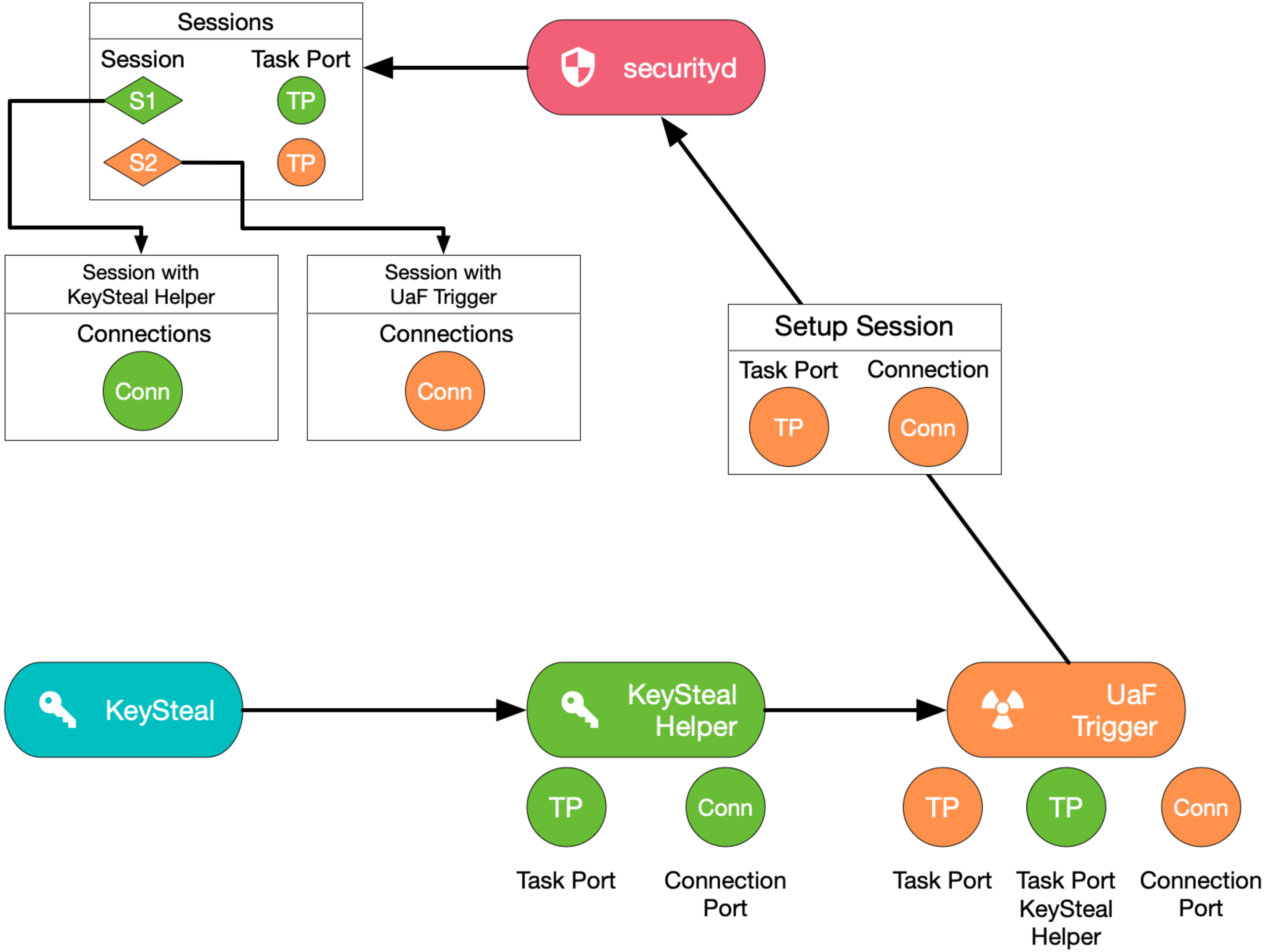
TP

Task Port KeySteal Helper

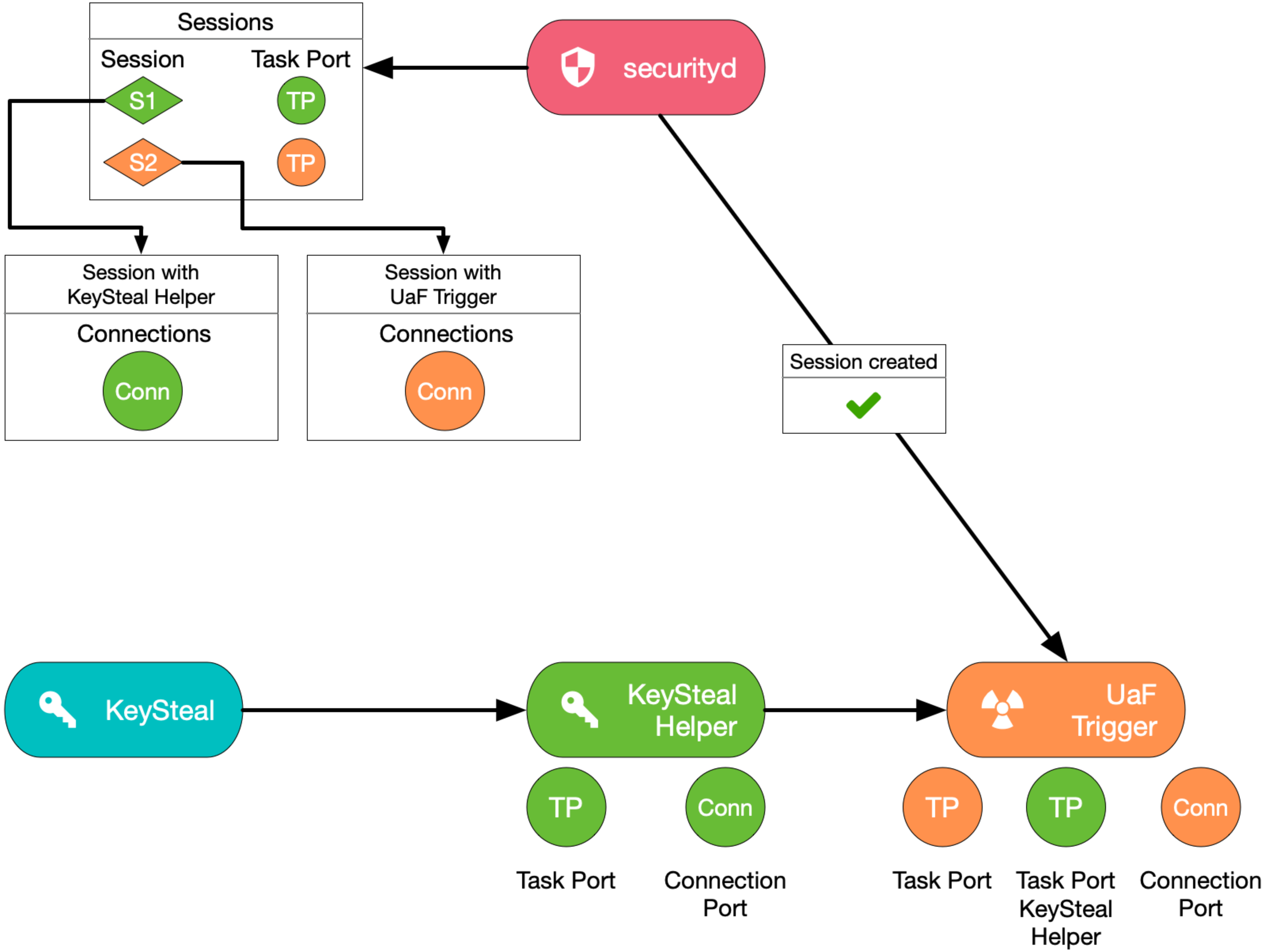
Conn

Connection Port

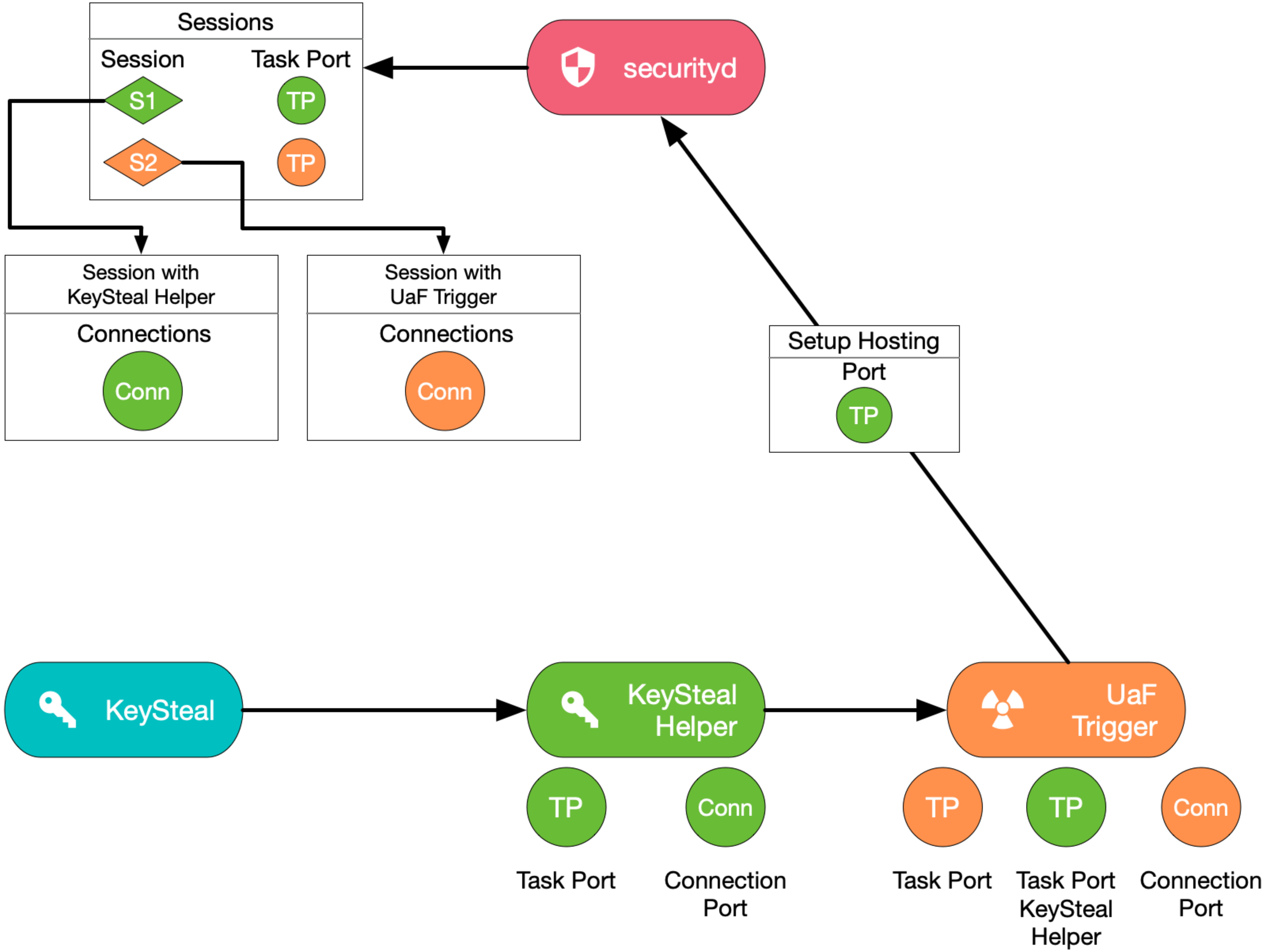
TRIGGER


Sessions
Session
Task Port
S1
TP
S2
TP
securityd
Session with KeySteal Helper
Connections
Conn
Session with UaF Trigger
Connections
Conn
Setup Session
Task Port
Connection
TP
Conn
KeySteal
KeySteal Helper
UaF Trigger
TP
Conn
TP
TP
Conn
Task Port
Connection Port
Task Port
Task Port KeySteal Helper
Connection Port

TRIGGER


Sessions
Session
Task Port
S1
TP
S2
TP
Session with
KeySteal Helper
Connections
Conn
Session with
UaF Trigger
Connections
Conn
securityd
Session created
✓
KeySteal
KeySteal
Helper
UaF
Trigger
TP
Task Port
Conn
Connection
Port
TP
Task Port
TP
Task Port
KeySteal
Helper
Conn
Connection
Port

TRIGGER



Sessions

Session

Task Port

S1

TP

S2

TP

securityd

Session with KeySteal Helper

Connections

Conn

Session with UaF Trigger

Connections

Conn

Setup Hosting Port

TP

KeySteal

KeySteal Helper

UaF Trigger

TP

Task Port

Conn

Connection Port

TP

Task Port

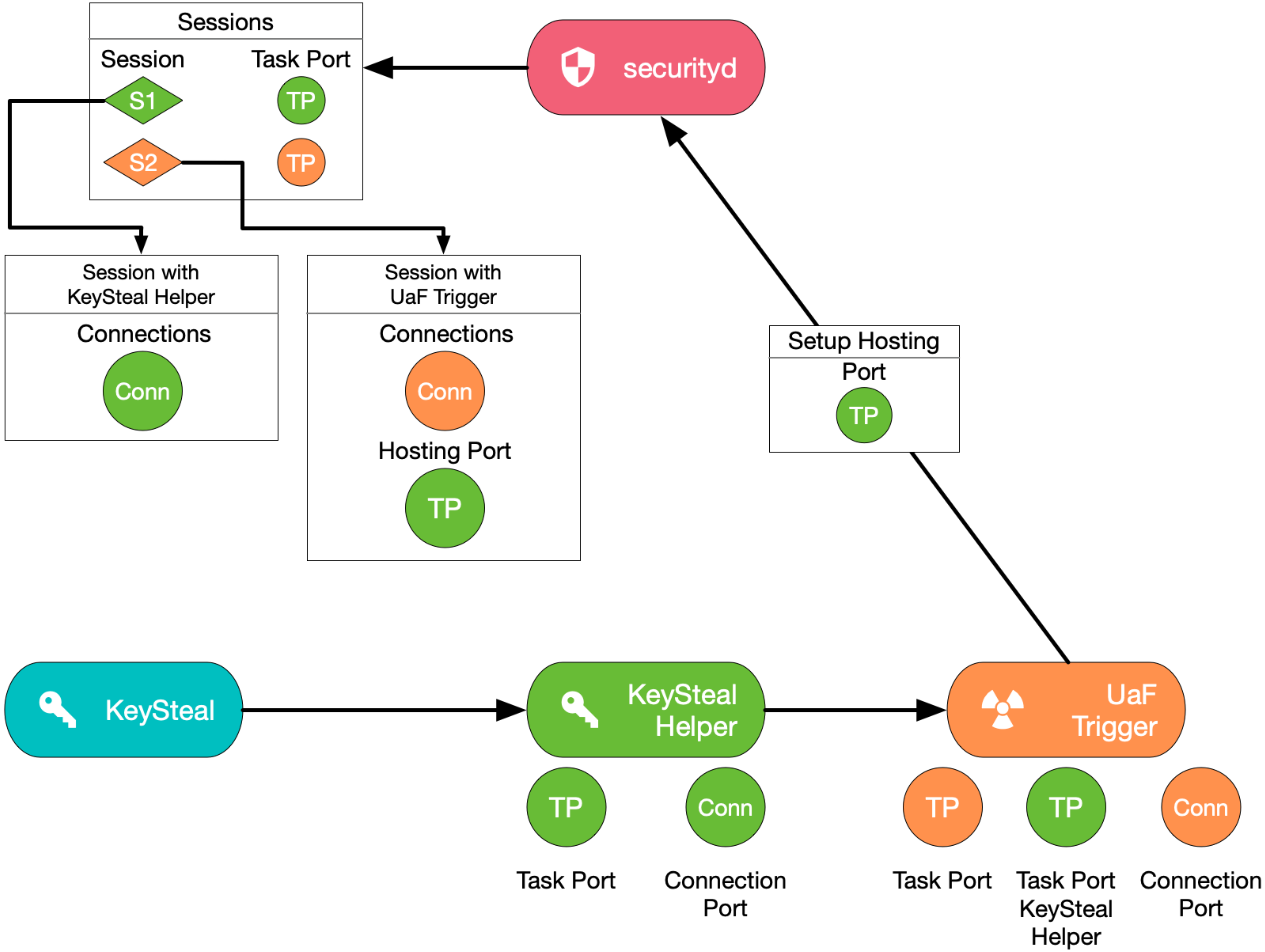
TP

Task Port KeySteal Helper

Conn

Connection Port

TRIGGER



Sessions

Session

Task Port

S1

TP

S2

TP

securityd

Session with
KeySteal Helper

Connections

Conn

Session with
UaF Trigger

Connections

Conn

Hosting Port

TP

Setup Hosting
Port

TP

KeySteal

KeySteal
Helper

UaF
Trigger

TP

Task Port

Conn

Connection
Port

TP

Task Port

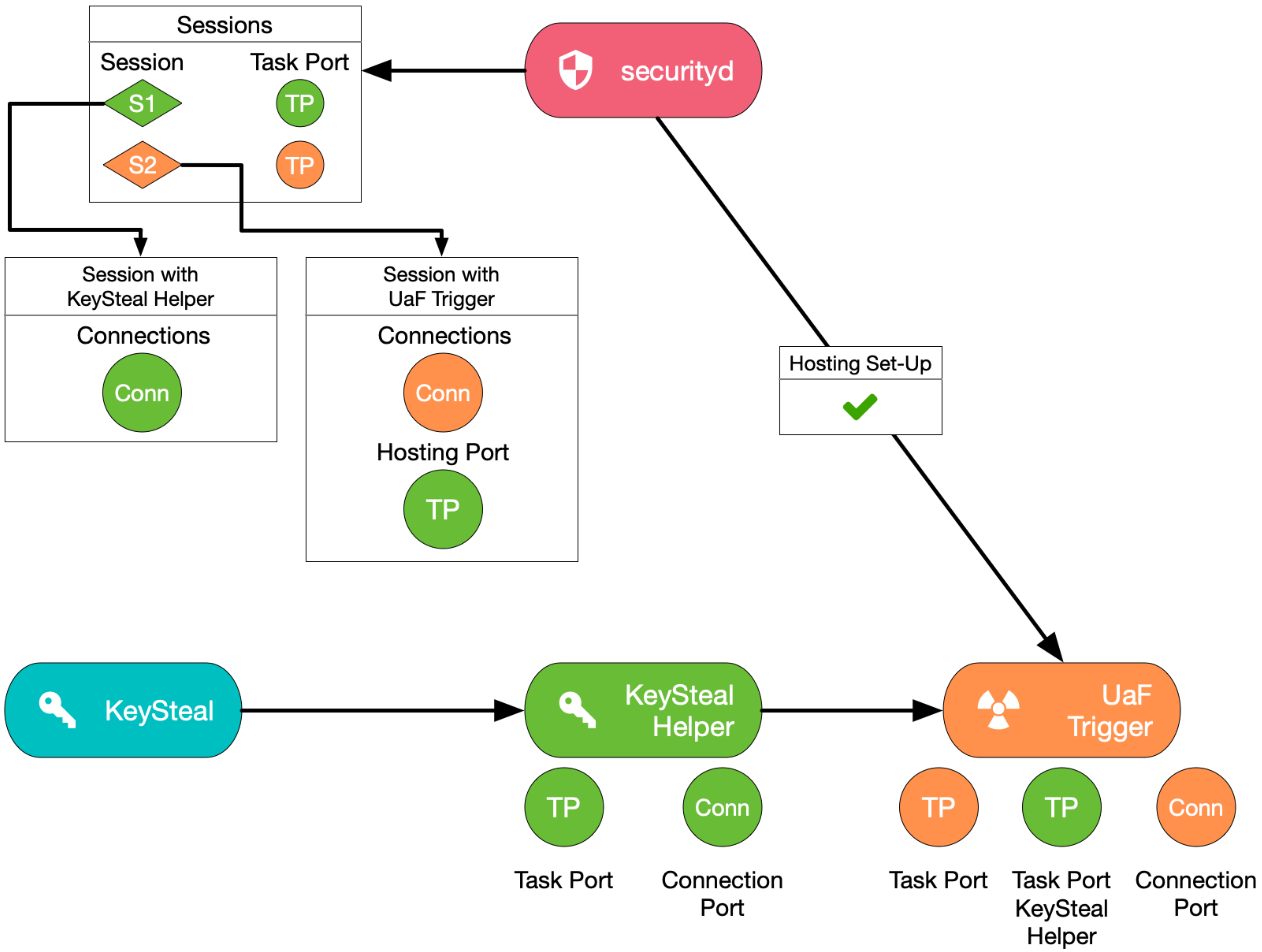
TP

Task Port
KeySteal
Helper

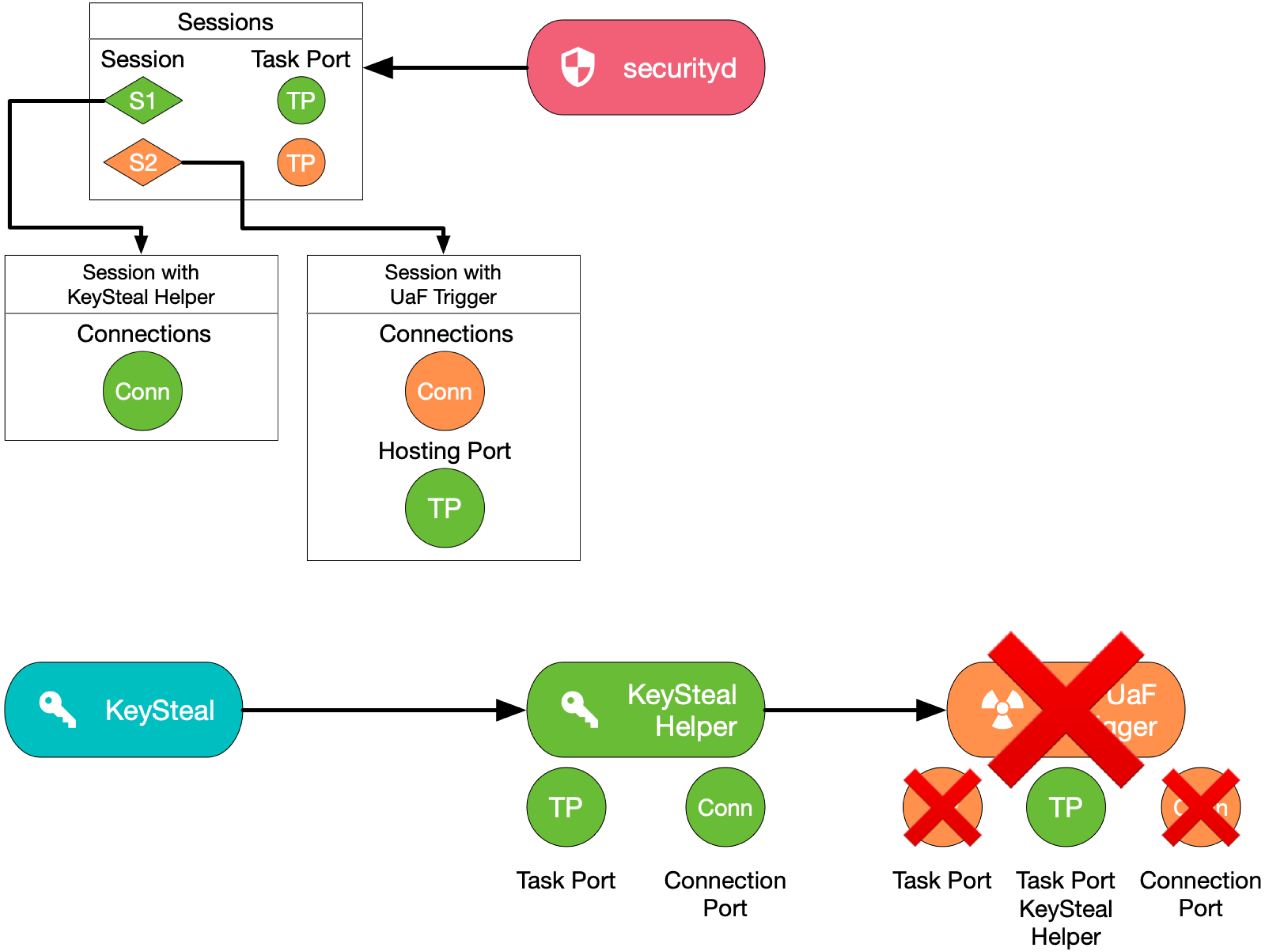
Conn

Connection
Port

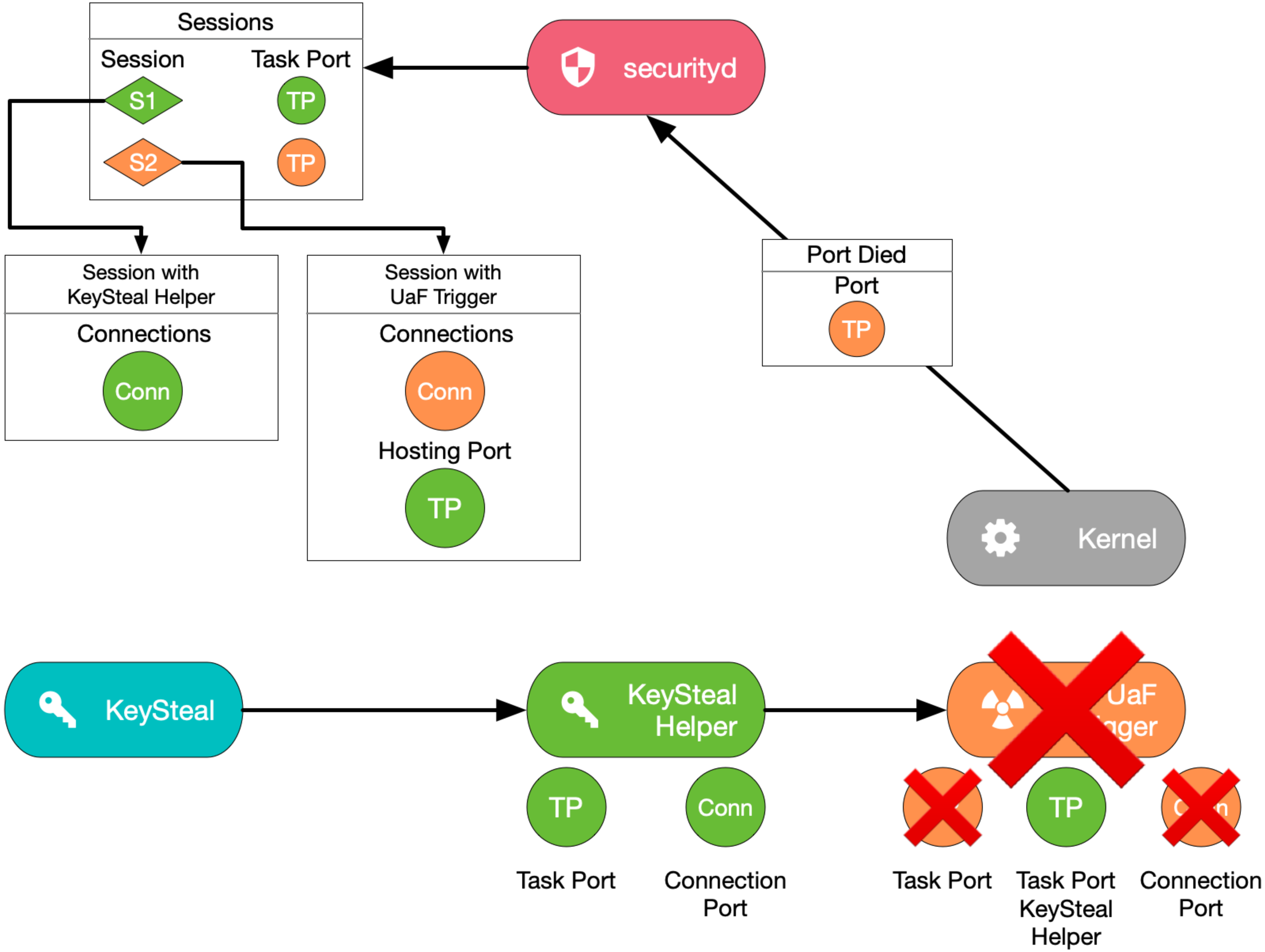
TRIGGER


Sessions
Session
Task Port
S1
TP
S2
TP
securityd
Session with
KeySteal Helper
Connections
Conn
Session with
UaF Trigger
Connections
Conn
Hosting Port
TP
Hosting Set-Up
✓
KeySteal
KeySteal
Helper
UaF
Trigger
TP
Conn
TP
TP
Conn
Task Port
Connection
Port
Task Port
Task Port
KeySteal
Helper
Connection
Port

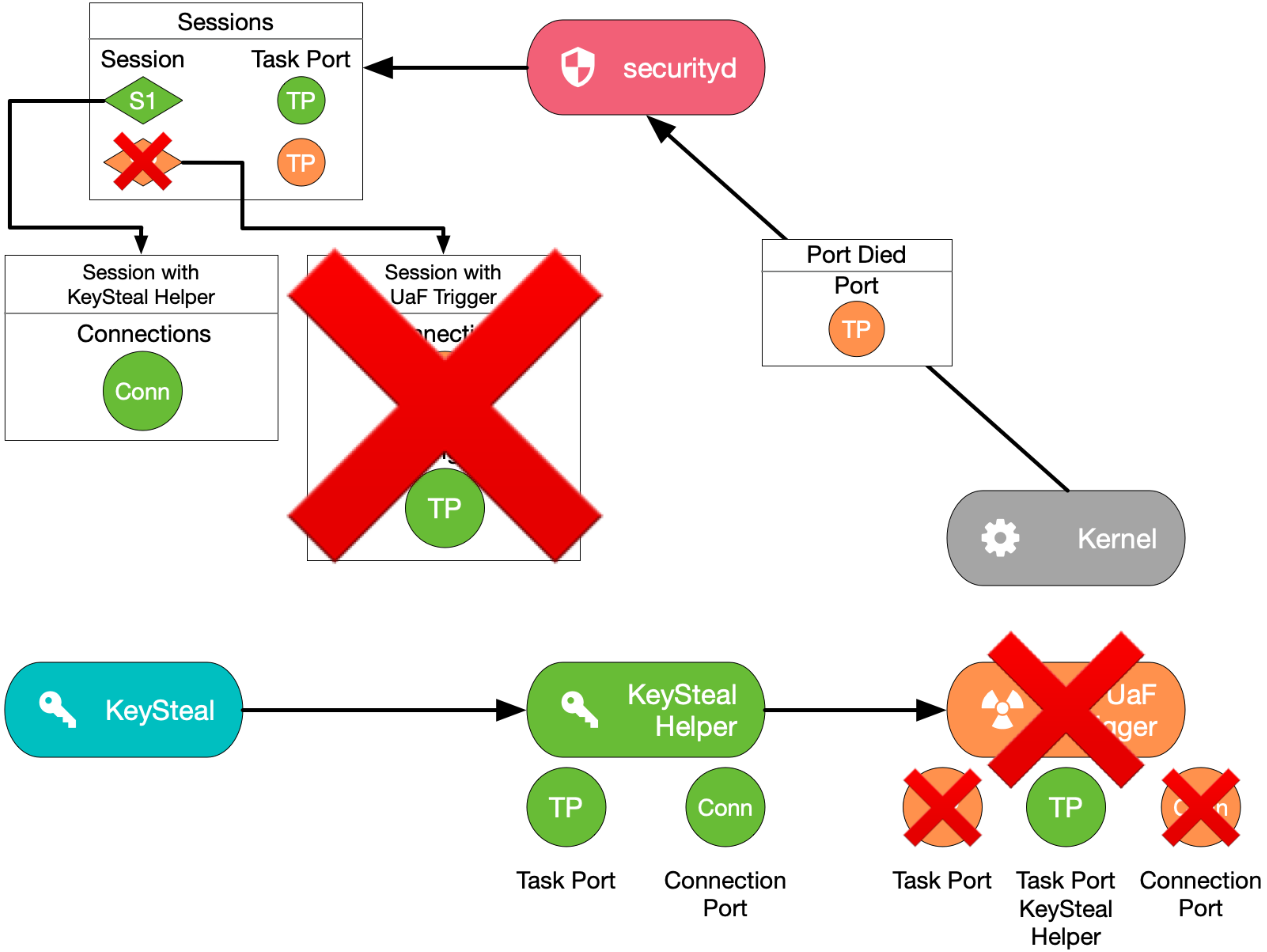
TRIGGER


Sessions
Session
Task Port
S1
TP
S2
TP
securityd
Session with KeySteal Helper
Connections
Conn
Session with UaF Trigger
Connections
Conn
Hosting Port
TP
KeySteal
KeySteal Helper
UaF Trigger
TP
Conn
Task Port
Connection Port
TP
Task Port
Task Port KeySteal Helper
Connection Port

TRIGGER


Sessions
Session
Task Port
S1
TP
S2
TP
securityd
Session with KeySteal Helper
Connections
Conn
Session with UaF Trigger
Connections
Conn
Hosting Port
TP
Port Died
Port
TP
Kernel
KeySteal
KeySteal Helper
UaF Trigger
TP
Conn
Task Port
Connection Port
TP
Task Port
Task Port KeySteal Helper
Connection Port

TRIGGER



Sessions

Session

Task Port

S1

TP

TP

Session with KeySteal Helper

Connections

Conn

Session with UaF Trigger

nnecti

TP

securityd

Port Died

Port

TP

Kernel

KeySteal

KeySteal Helper

UaF

gger

TP

Conn

Task Port

Connection Port

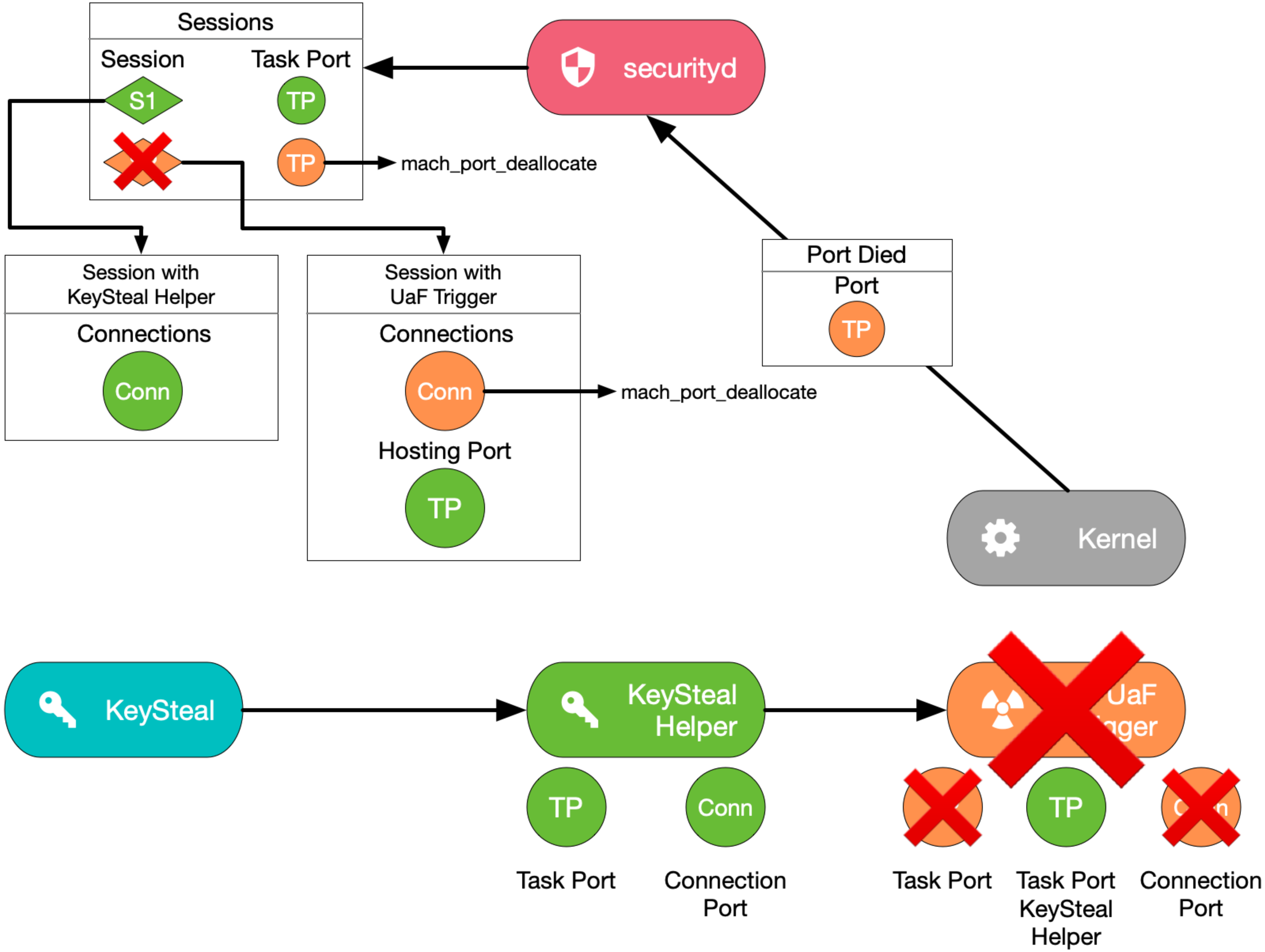
Task Port

TP

Task Port KeySteal Helper

Connection Port

TRIGGER



Sessions

Session

Task Port

S1

TP

TP

mach_port_deallocate

securityd

Session with
KeySteal Helper

Connections

Conn

Session with
UaF Trigger

Connections

Conn

mach_port_deallocate

Hosting Port

TP

Port Died

Port

TP

Kernel

KeySteal

KeySteal
Helper

UaF
…gger

TP

Conn

Task Port

Connection
Port

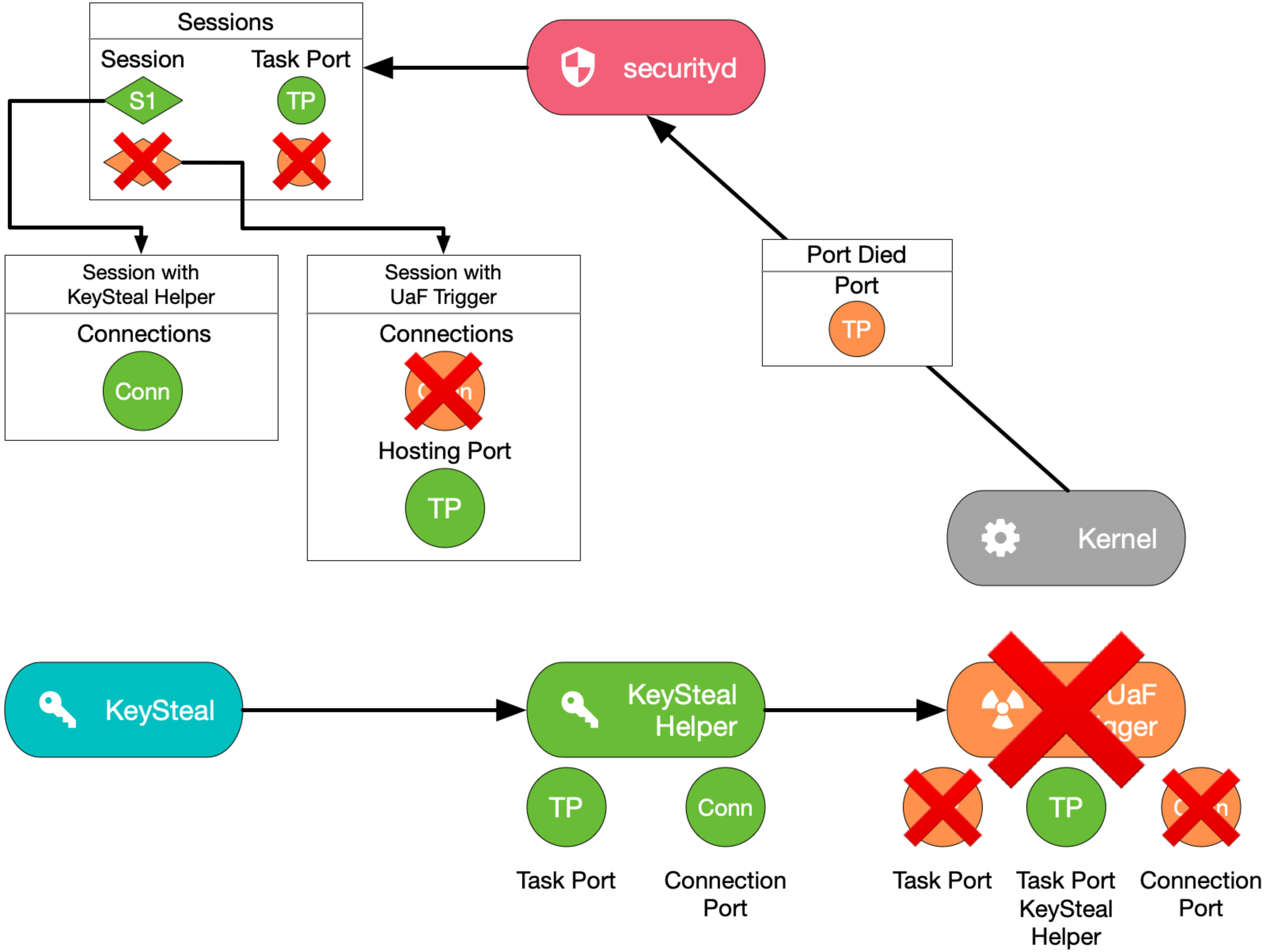
TP

Task Port

Task Port
KeySteal
Helper

Connection
Port

TRIGGER



Sessions

Session

Task Port

S1

TP

~~TP~~

Session with
KeySteal Helper

Connections

Conn

Session with
UaF Trigger

Connections

~~Conn~~

Hosting Port

TP

securityd

Port Died

Port

TP

Kernel

KeySteal

KeySteal
Helper

TP

Conn

Task Port

Connection
Port

~~UaF
Trigger~~

TP

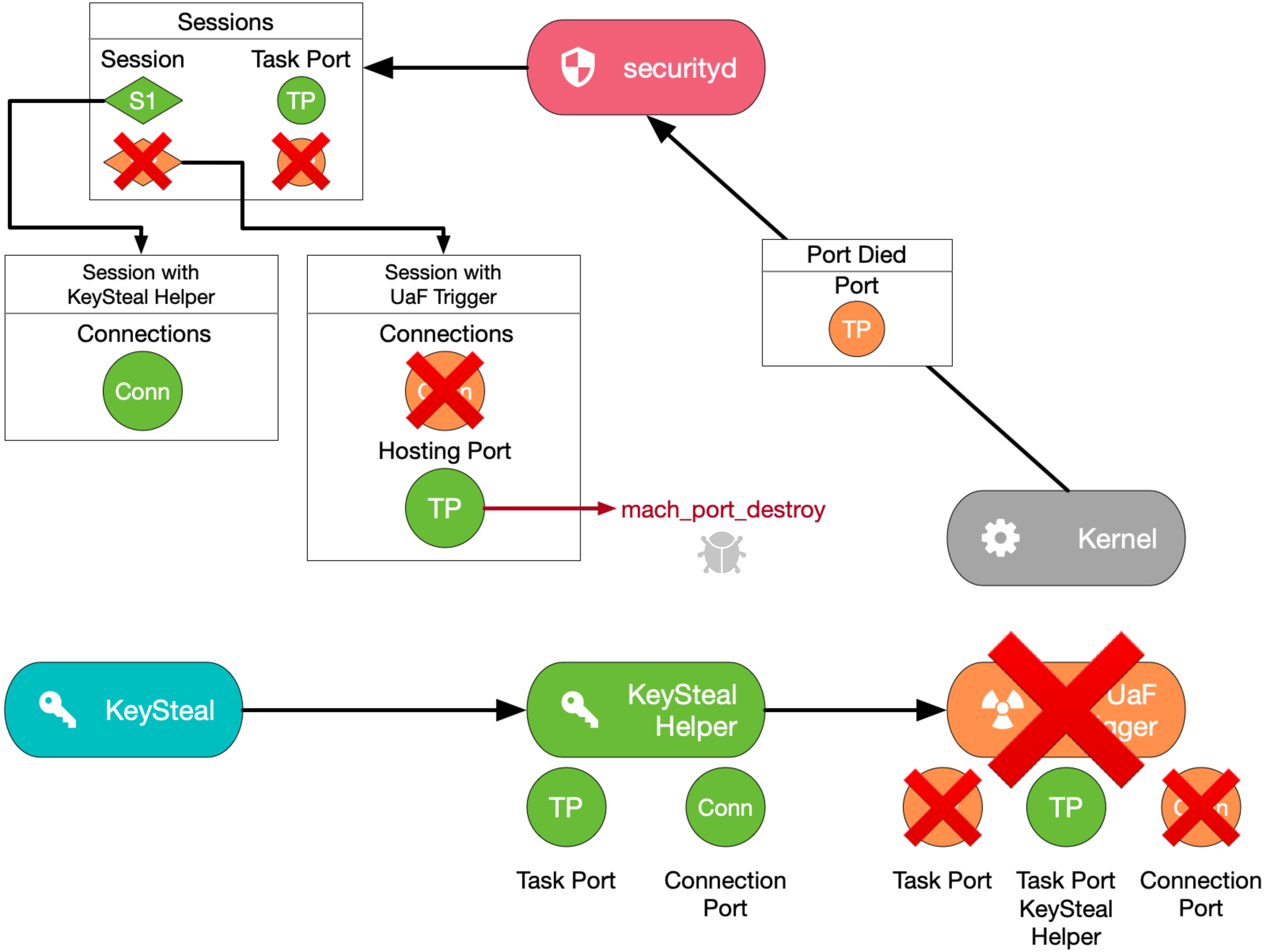
Task Port

Task Port
KeySteal
Helper

~~Conn~~

Connection
Port

TRIGGER



Sessions

Session

Task Port

S1

TP

Session with KeySteal Helper

Connections

Conn

Session with UaF Trigger

Connections

C n

Hosting Port

TP

mach_port_destroy

securityd

Port Died

Port

TP

Kernel

KeySteal

KeySteal Helper

UaF gger

TP

Task Port

Conn

Connection Port

Task Port

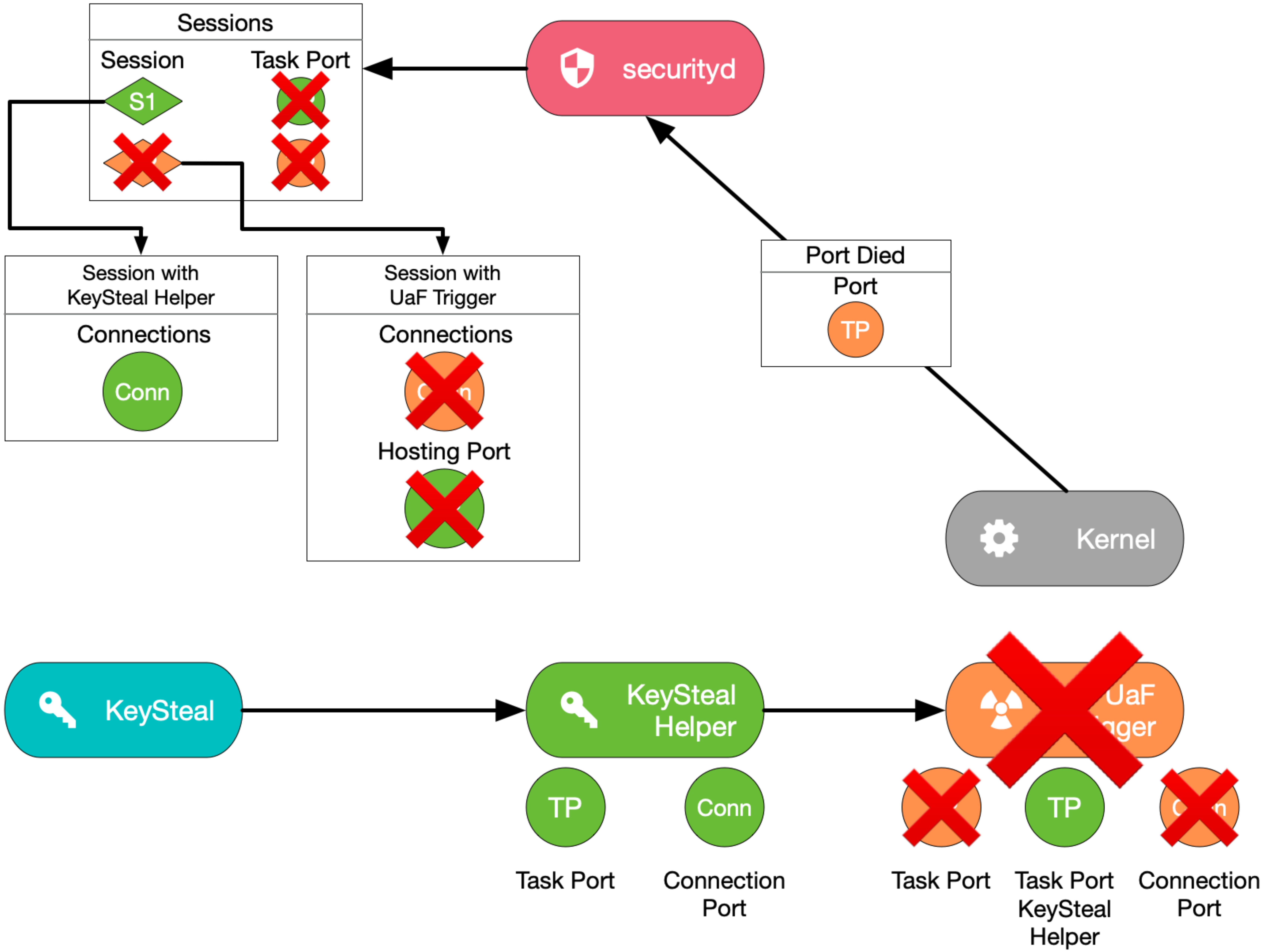
TP

Task Port KeySteal Helper

C n

Connection Port

TRIGGER



Sessions

Session

Task Port

S1

securityd

Session with KeySteal Helper

Connections

Conn

Session with UaF Trigger

Connections

Conn

Hosting Port

Port Died

Port

TP

Kernel

KeySteal

KeySteal Helper

UaF Trigger

TP

Conn

TP

Task Port

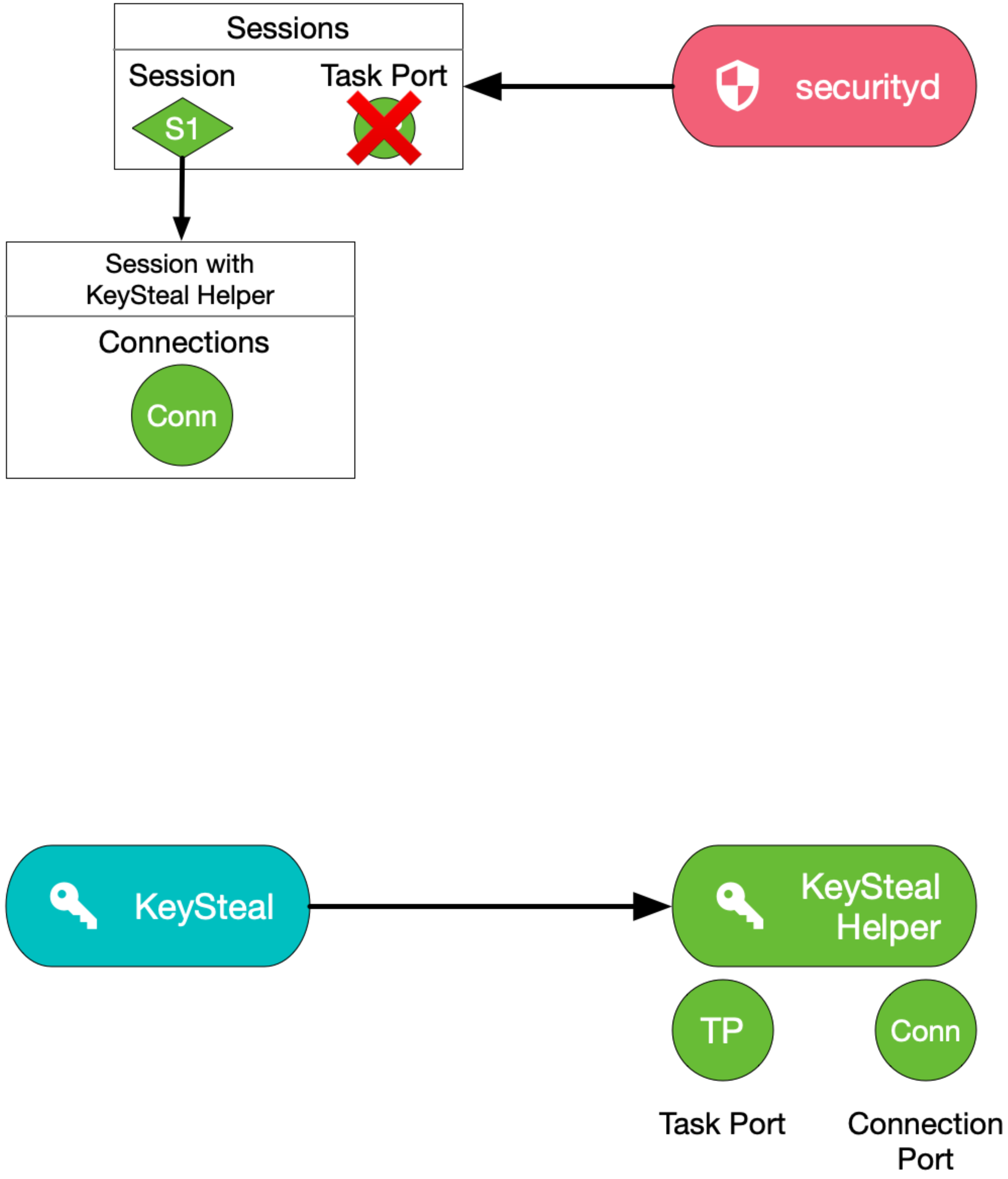
Connection Port

Task Port

Task Port KeySteal Helper

Connection Port

TRIGGER



Sessions

Session

Task Port

S1

securityd

Session with
KeySteal Helper

Connections

Conn

KeySteal

KeySteal
Helper

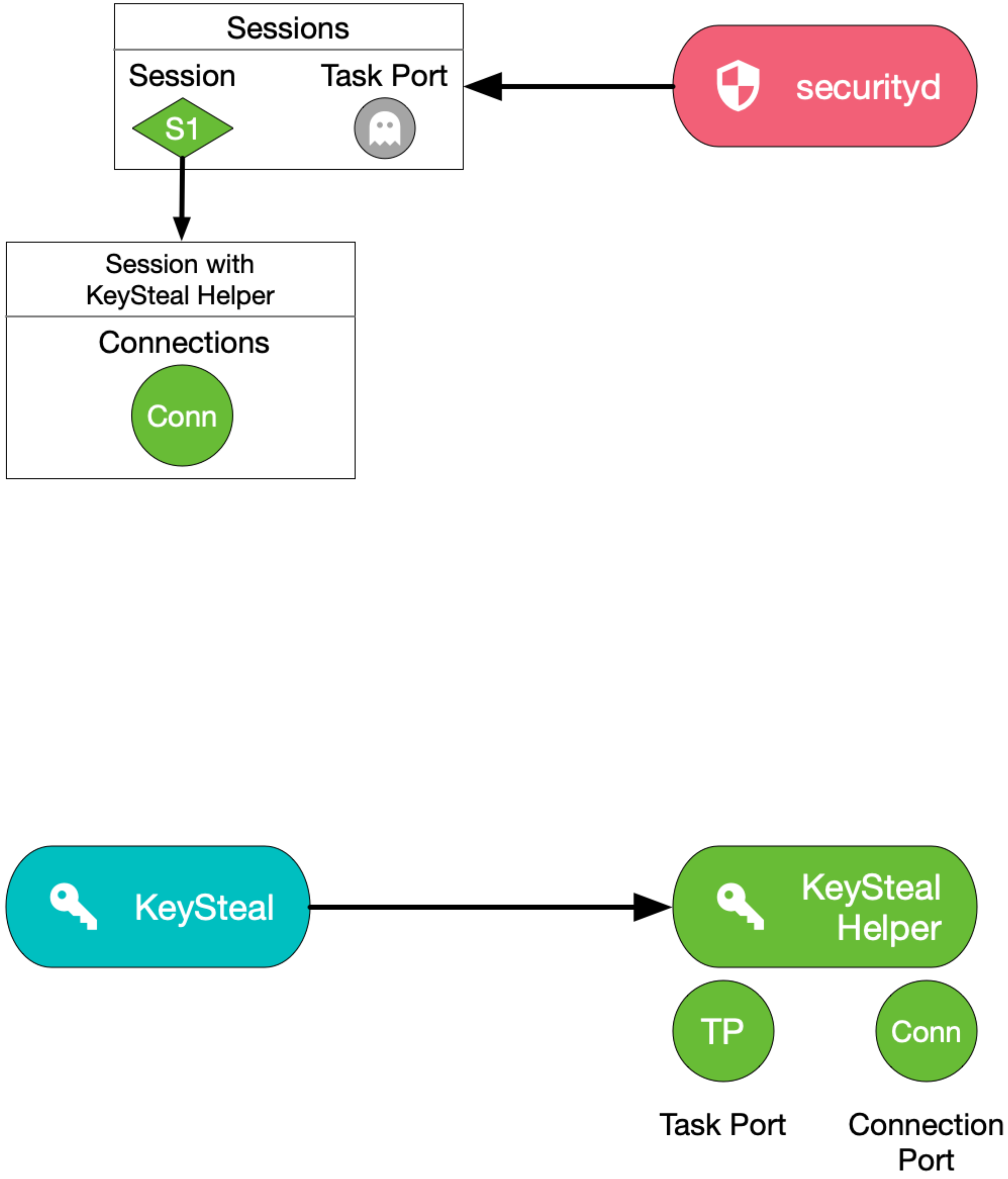
TP

Conn

Task Port

Connection
Port

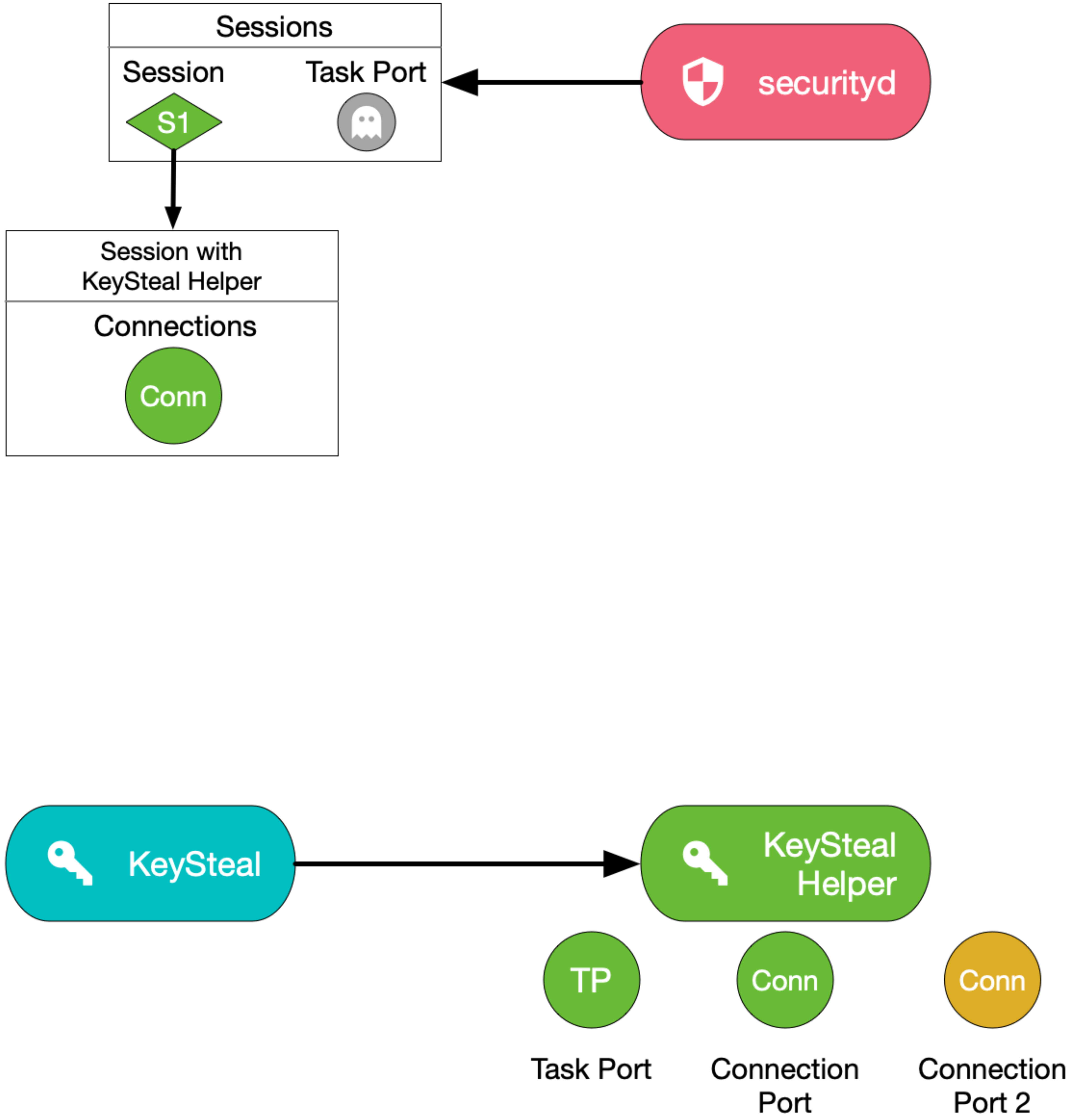
FREED PORT


Sessions
Session
Task Port
S1
securityd
Session with
KeySteal Helper
Connections
Conn
KeySteal
KeySteal
Helper
TP
Conn
Task Port
Connection
Port

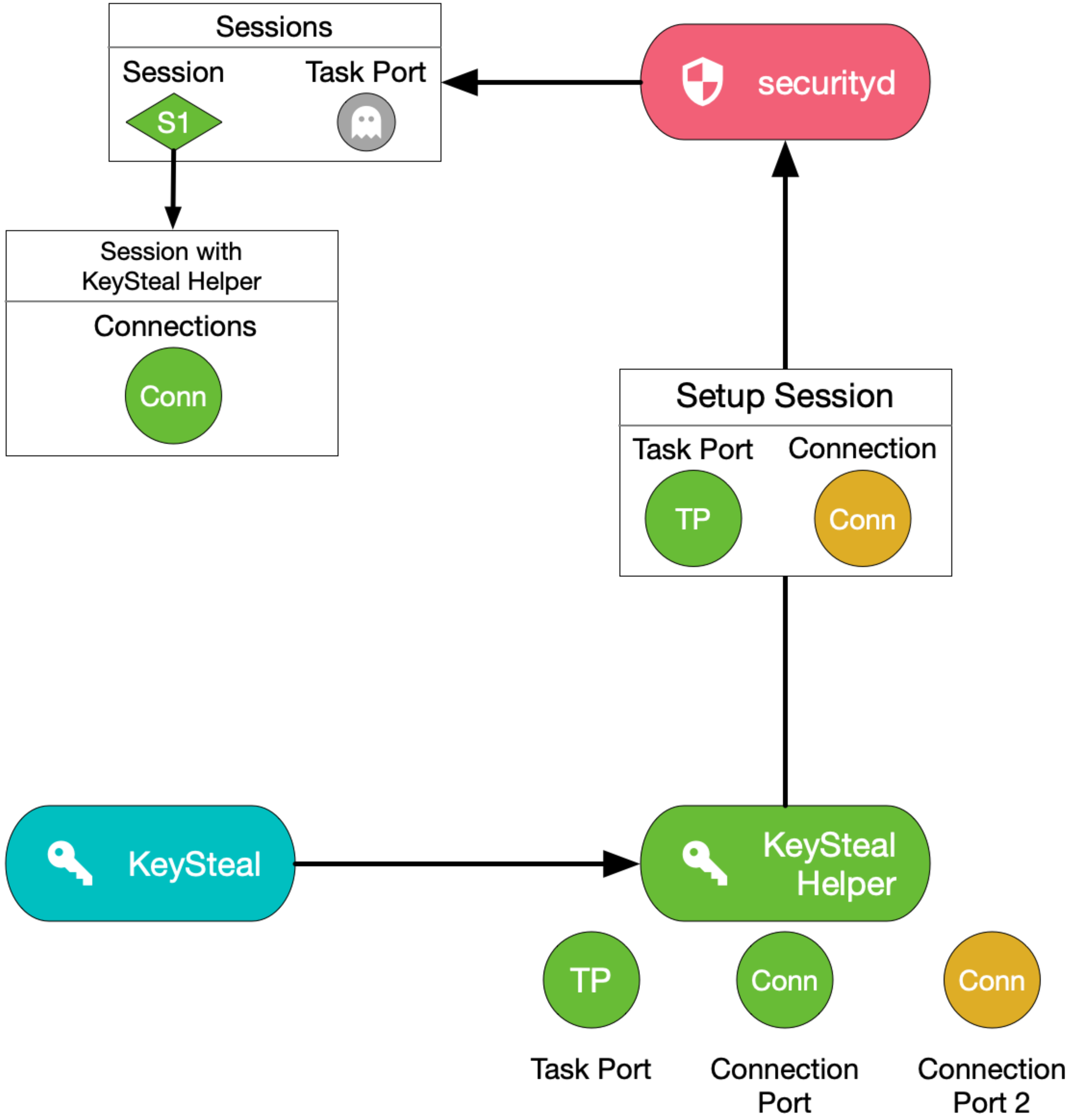
FREED PORT

ATTACK PLAN

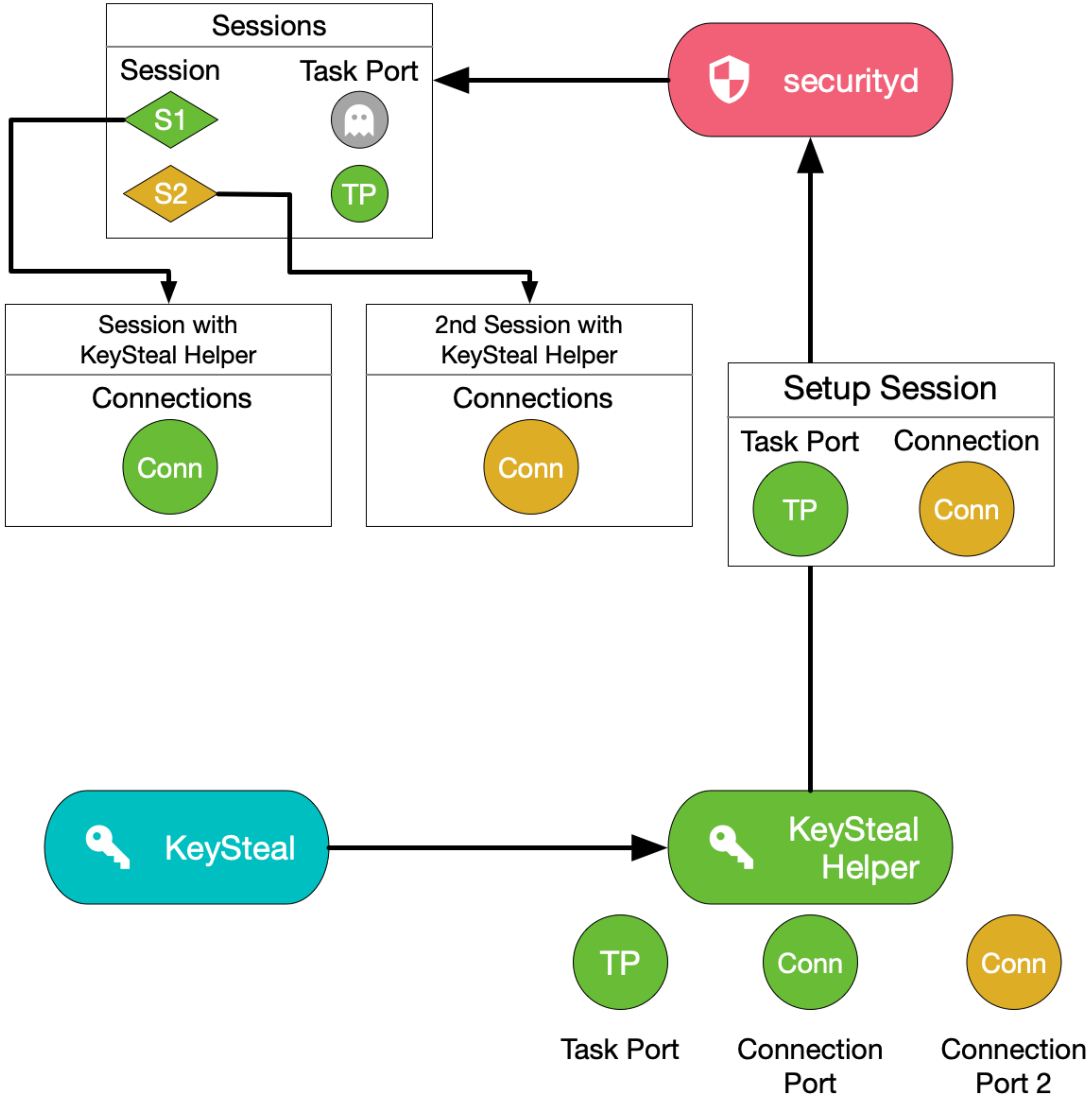
1. Create three processes: A, B and C ✓
2. B should create a session with securityd ✓
3. Send task port of B to C ✓
4. Let C free B's task port in securityd ✓
5. B should now reclaim it's session by sending securityd many ports, hoping one of them will get the same number as B's task port had ←



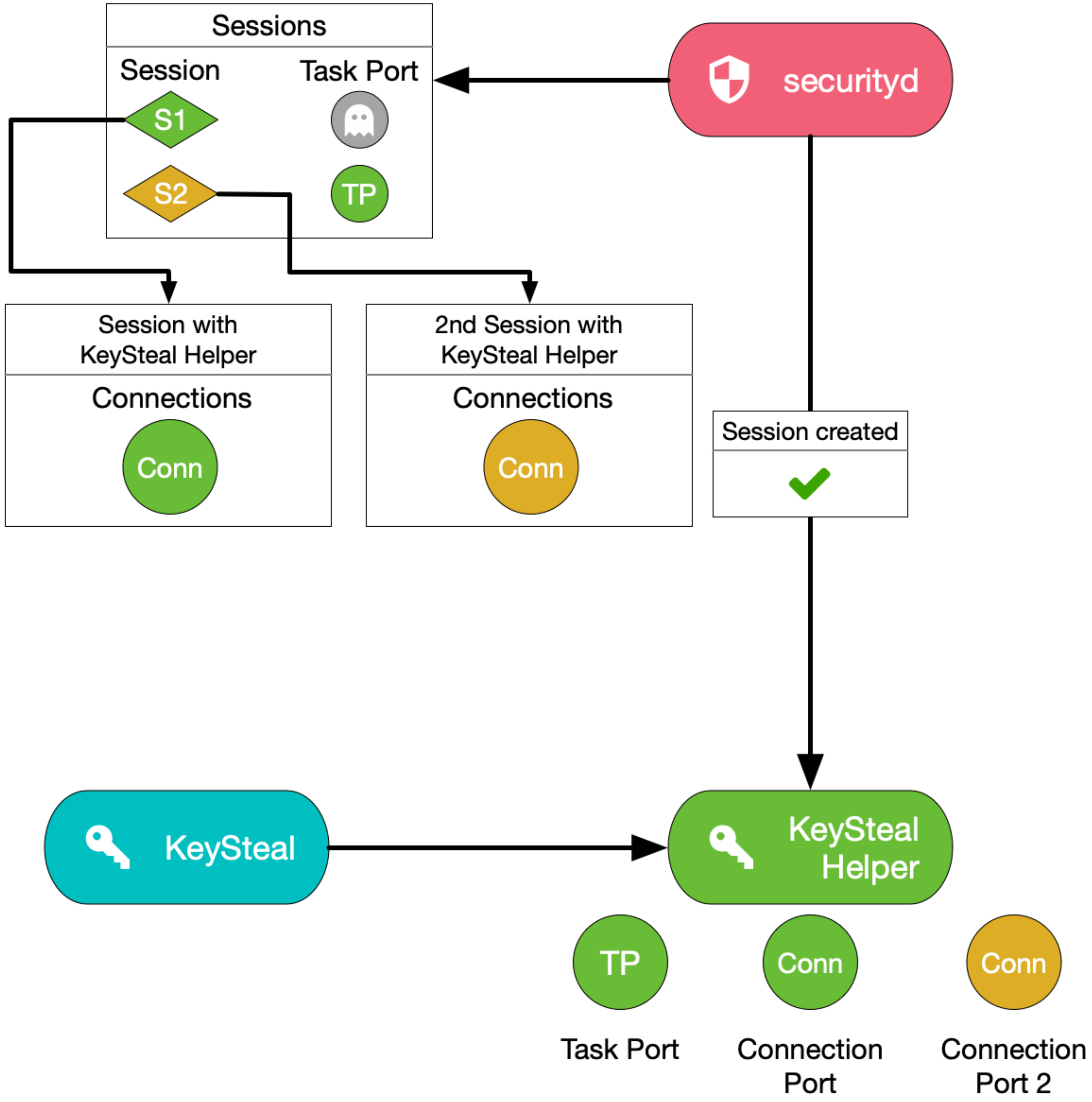
RECLAIMING – SETUP



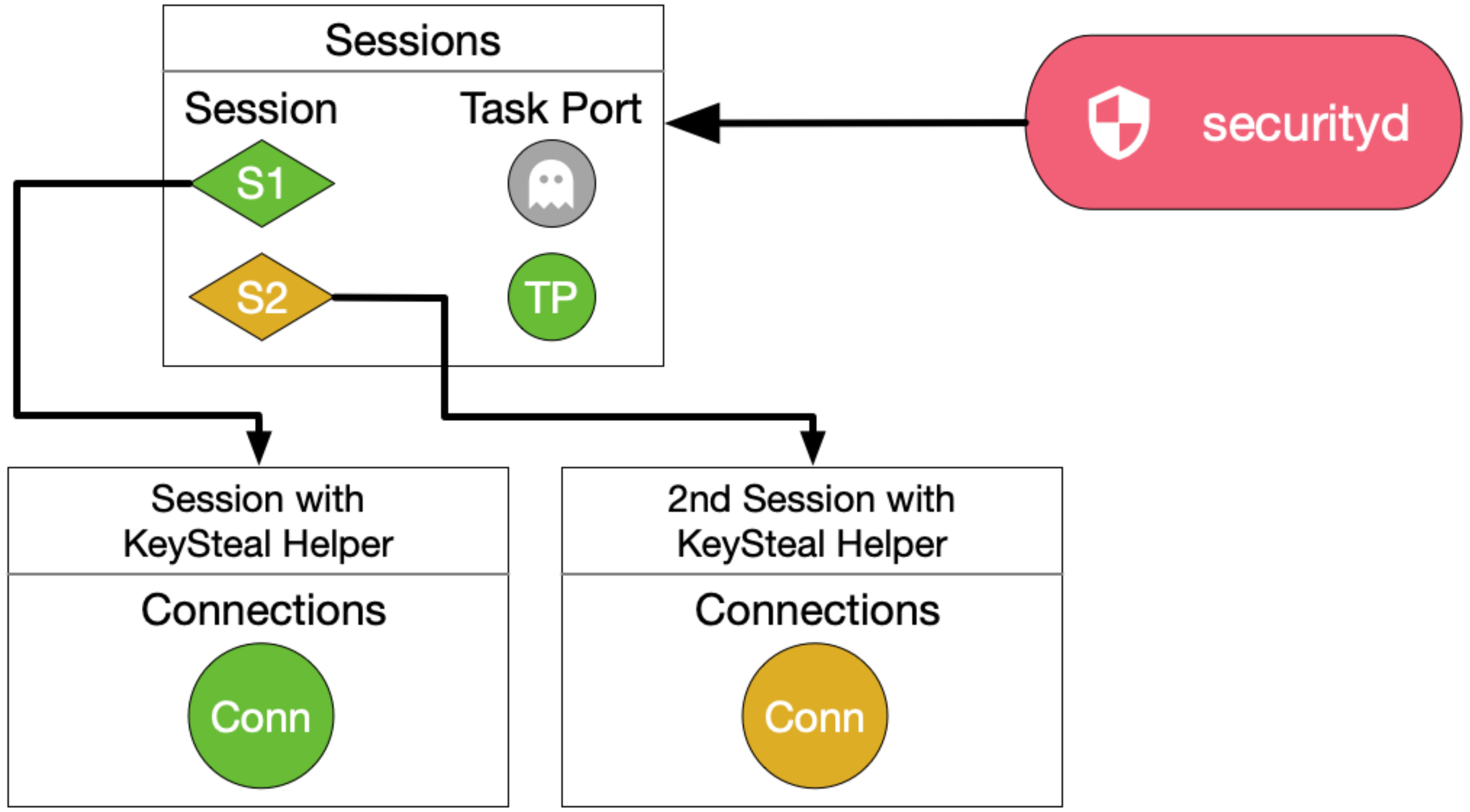
RECLAIMING – SETUP



RECLAIMING – SETUP



RECLAIMING – SETUP



Sessions

Session

Task Port

S1

S2

TP

securityd

Session with KeySteal Helper

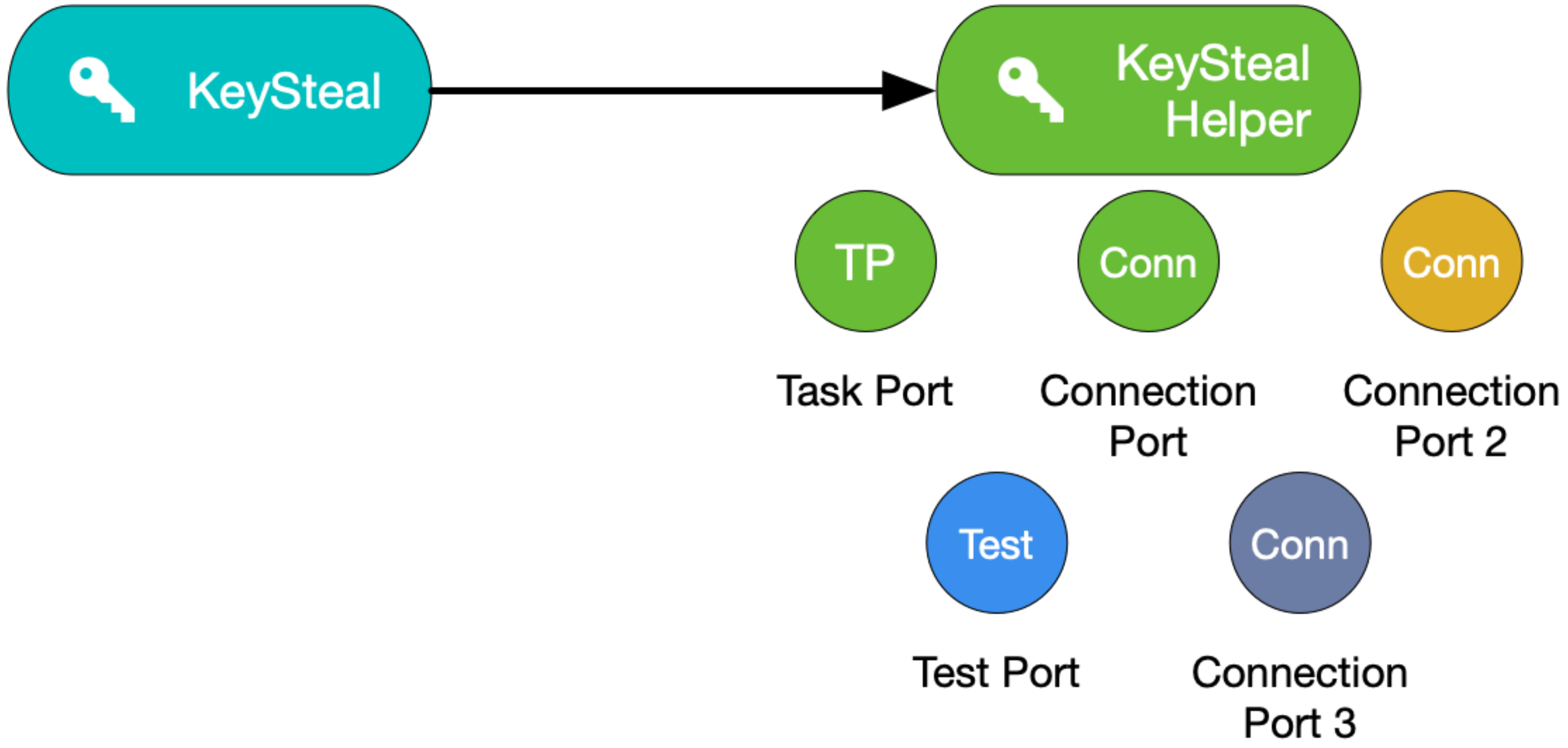
Connections

Conn

2nd Session with KeySteal Helper

Connections

Conn



KeySteal

KeySteal Helper

TP

Task Port

Conn

Connection Port

Conn

Connection Port 2

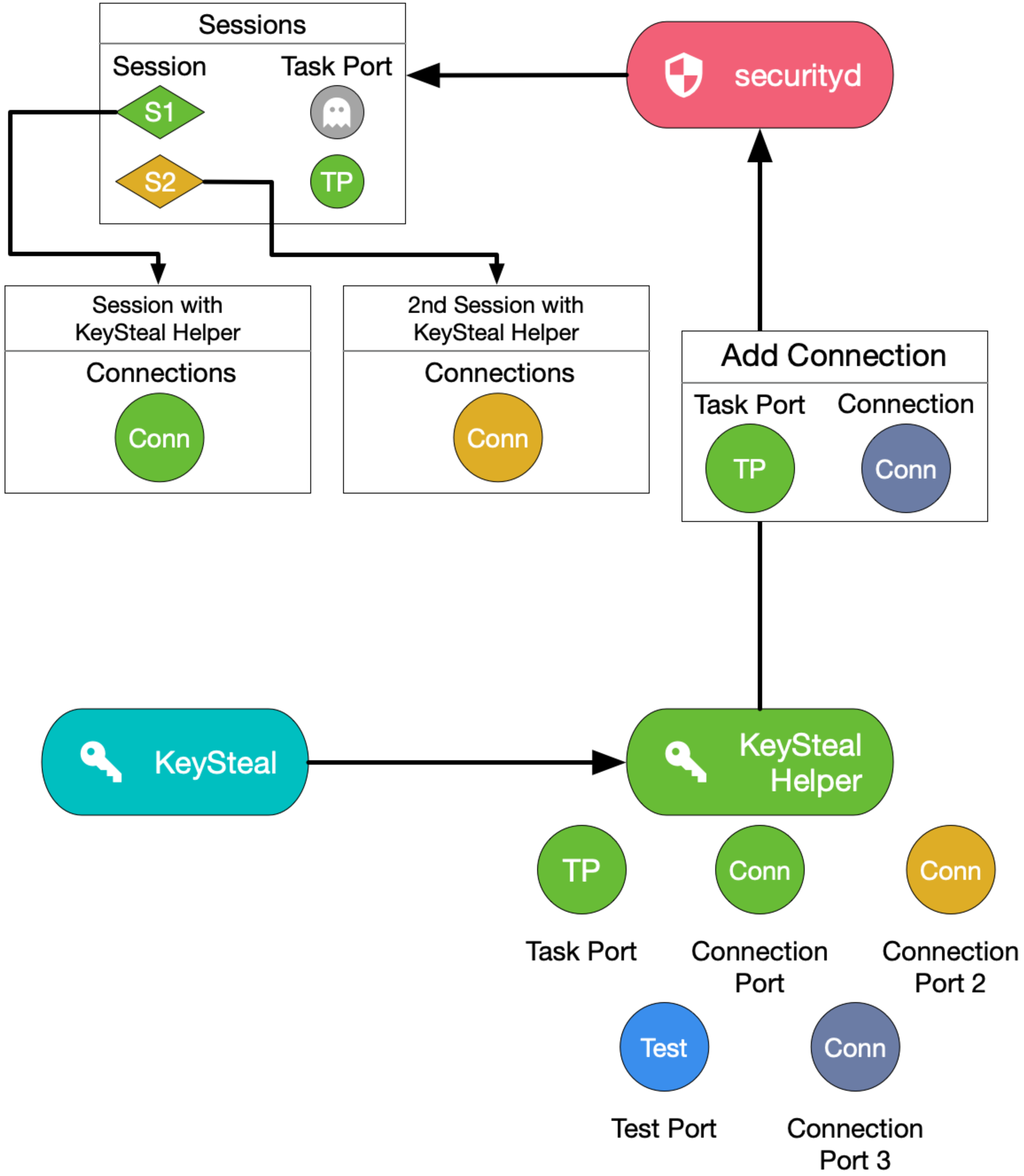
Test

Test Port

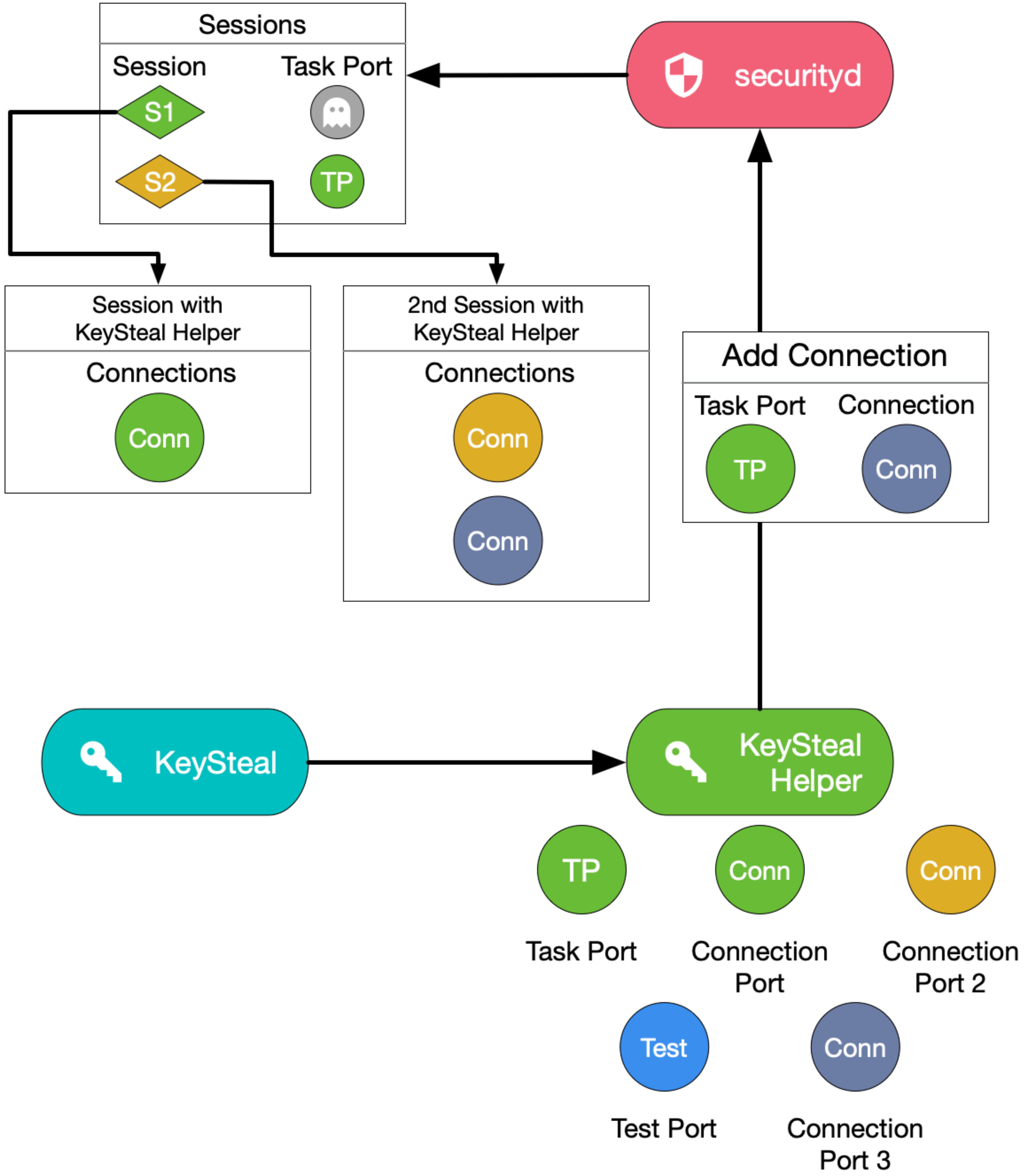
Conn

Connection Port 3

RECLAIMING


Sessions
Session
Task Port
S1
S2
TP
securityd
Session with KeySteal Helper
Connections
Conn
2nd Session with KeySteal Helper
Connections
Conn
Add Connection
Task Port
Connection
TP
Conn
KeySteal
KeySteal Helper
TP
Task Port
Conn
Connection Port
Conn
Connection Port 2
Test
Test Port
Conn
Connection Port 3

RECLAIMING



Sessions

Session

Task Port

S1

S2

TP

securityd

Session with KeySteal Helper

Connections

Conn

2nd Session with KeySteal Helper

Connections

Conn

Conn

Add Connection

Task Port

Connection

TP

Conn

KeySteal

KeySteal Helper

TP

Task Port

Conn

Connection Port

Conn

Connection Port 2

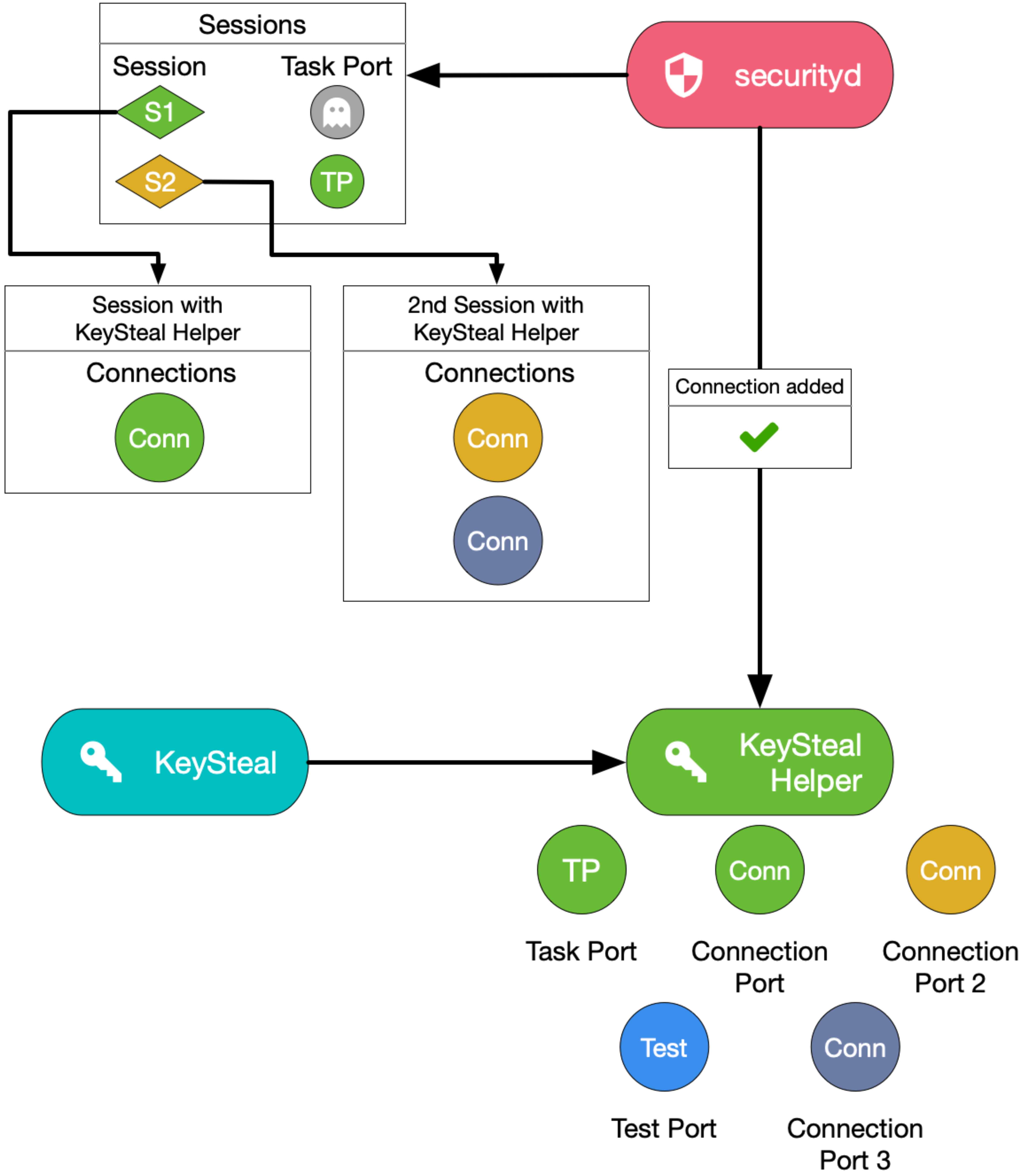
Test

Test Port

Conn

Connection Port 3

RECLAIMING



Sessions

Session

Task Port

S1

S2

TP

securityd

Session with KeySteal Helper

Connections

Conn

2nd Session with KeySteal Helper

Connections

Conn

Conn

Connection added

✓

KeySteal

KeySteal Helper

TP

Task Port

Conn

Connection Port

Conn

Connection Port 2

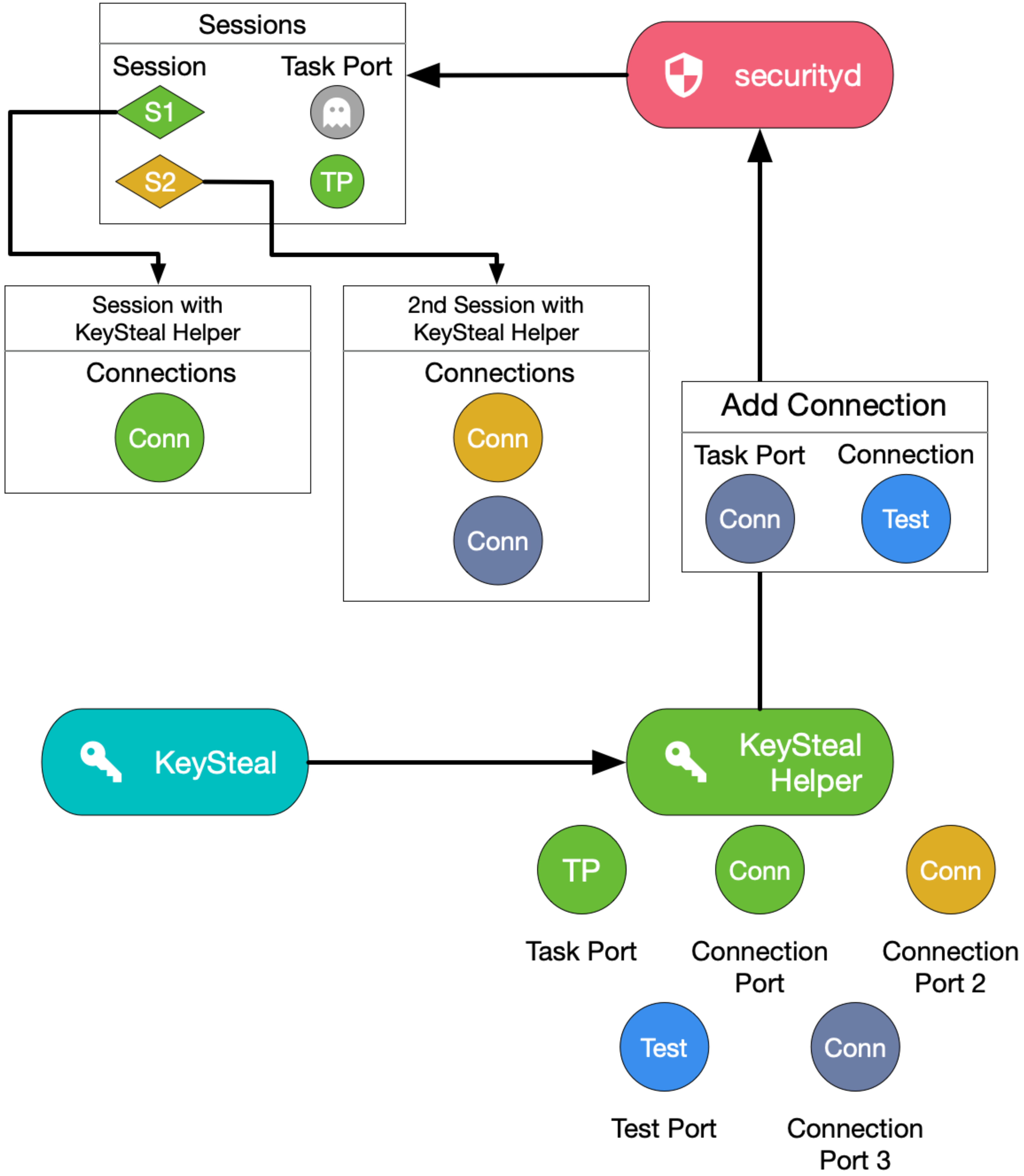
Test

Test Port

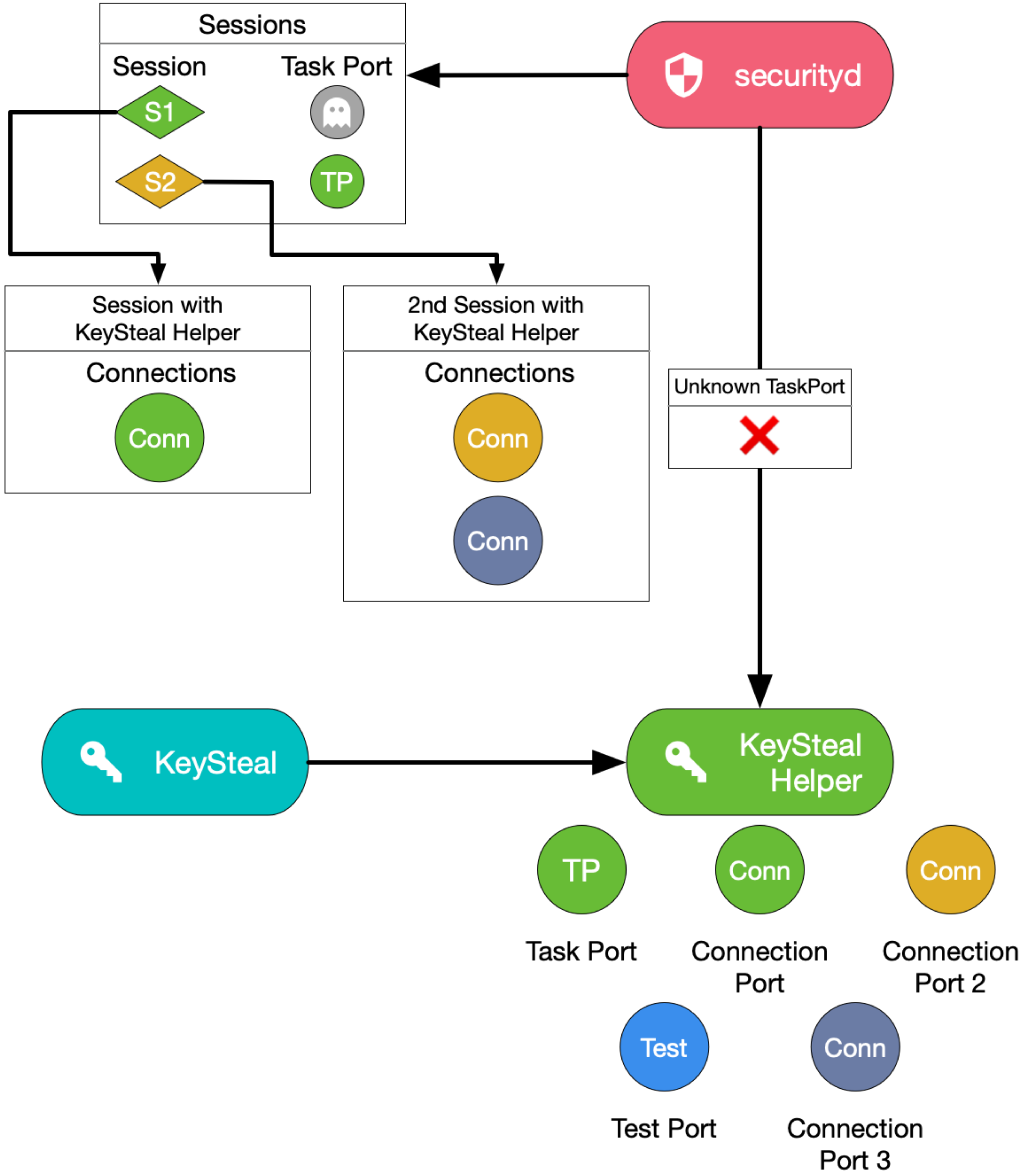
Conn

Connection Port 3

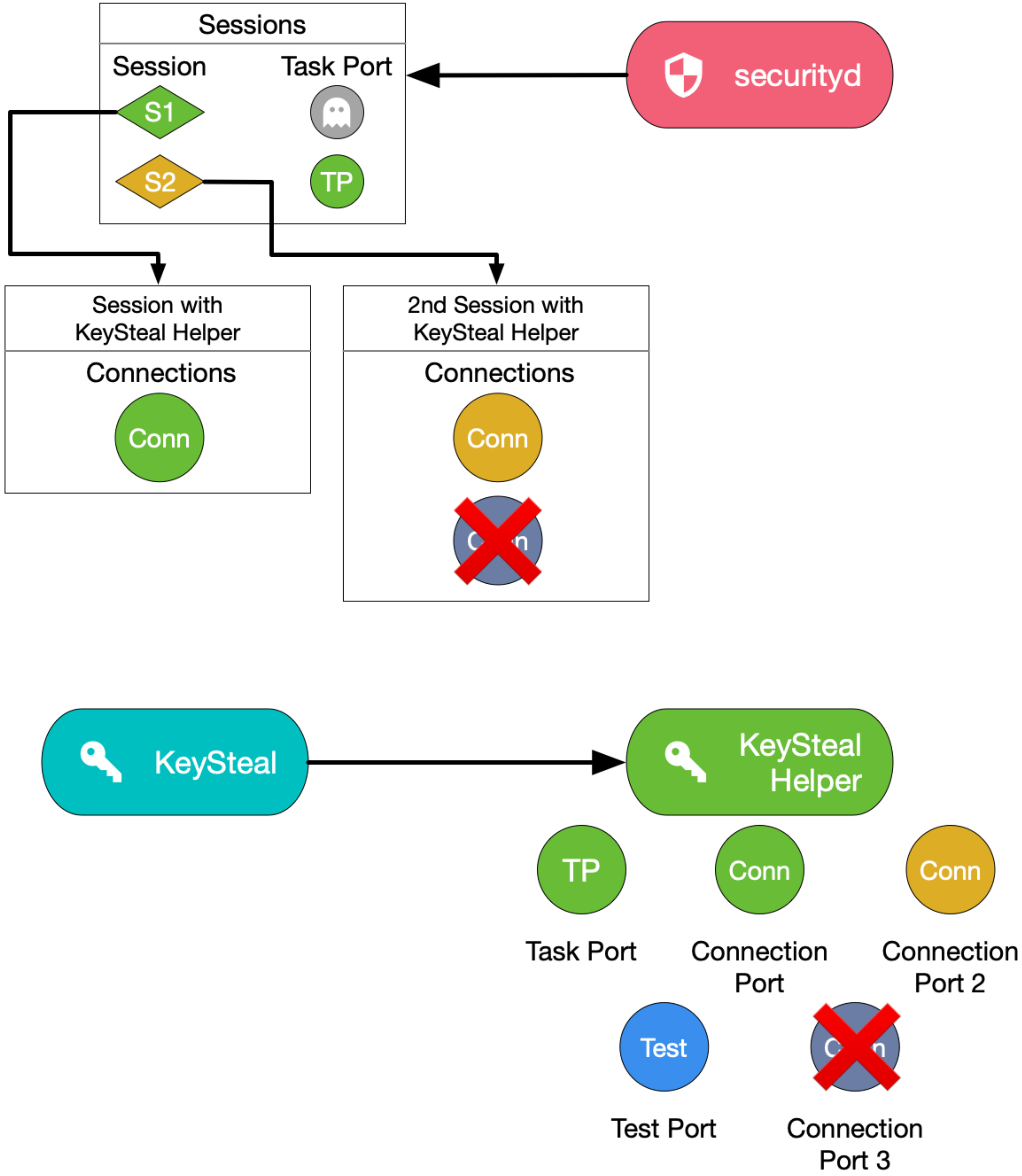
RECLAIMING


Sessions
Session
Task Port
S1
S2
TP
securityd
Session with KeySteal Helper
Connections
Conn
2nd Session with KeySteal Helper
Connections
Conn
Conn
Add Connection
Task Port
Connection
Conn
Test
KeySteal
KeySteal Helper
TP
Task Port
Conn
Connection Port
Conn
Connection Port 2
Test
Test Port
Conn
Connection Port 3

RECLAIMING


Sessions
Session
Task Port
S1
S2
TP
securityd
Session with KeySteal Helper
Connections
Conn
2nd Session with KeySteal Helper
Connections
Conn
Conn
Unknown TaskPort
KeySteal
KeySteal Helper
TP
Task Port
Conn
Connection Port
Conn
Connection Port 2
Test
Test Port
Conn
Connection Port 3

RECLAIMING



Sessions

Session

Task Port

S1

S2

TP

securityd

Session with KeySteal Helper

Connections

Conn

2nd Session with KeySteal Helper

Connections

Conn

Conn

KeySteal

KeySteal Helper

TP

Task Port

Conn

Connection Port

Conn

Connection Port 2

Test

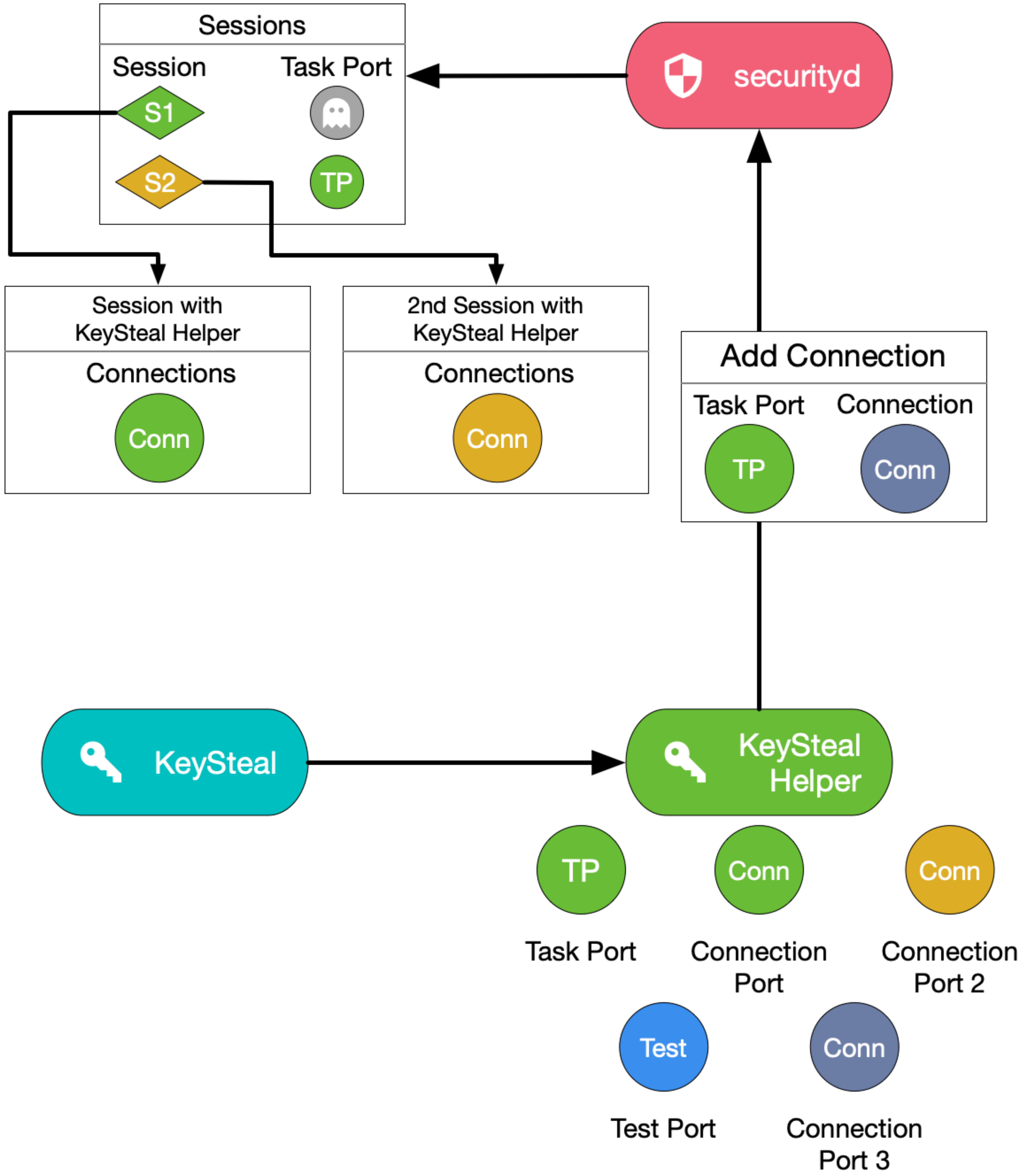
Test Port

Conn

Connection Port 3

RECLAIMING

AFTER SOME TIME...



Sessions

Session

Task Port

S1

S2

TP

securityd

Session with KeySteal Helper

Connections

Conn

2nd Session with KeySteal Helper

Connections

Conn

Add Connection

Task Port

Connection

TP

Conn

KeySteal

KeySteal Helper

TP

Task Port

Conn

Connection Port

Conn

Connection Port 2

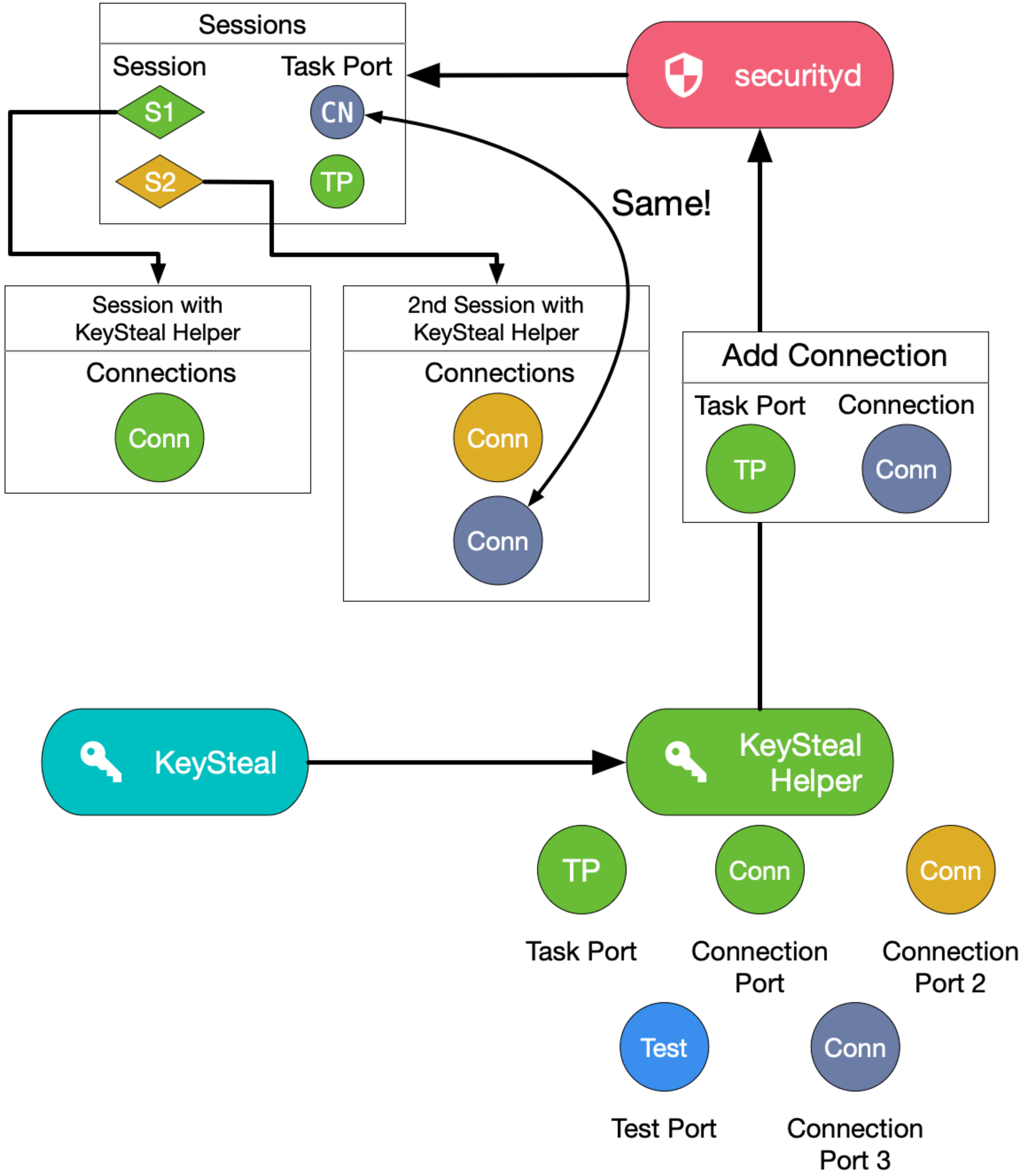
Test

Test Port

Conn

Connection Port 3

RECLAIMING



Sessions

Session | Task Port

S1 | CN

S2 | TP

Same!

securityd

Session with KeySteal Helper

Connections

Conn

2nd Session with KeySteal Helper

Connections

Conn

Conn

Add Connection

Task Port | Connection

TP | Conn

KeySteal

KeySteal Helper

TP

Task Port

Conn

Connection Port

Conn

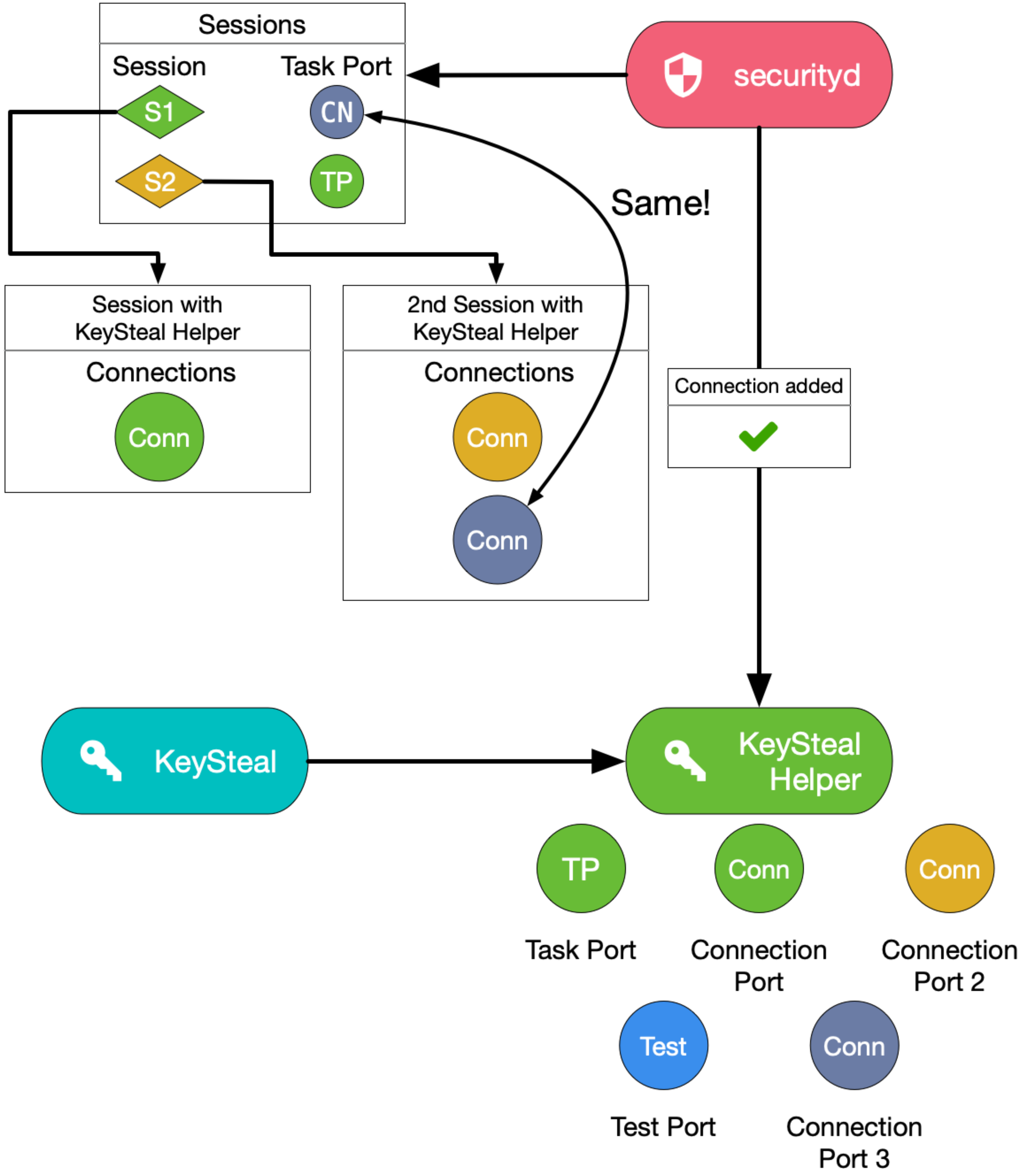
Connection Port 2

Test

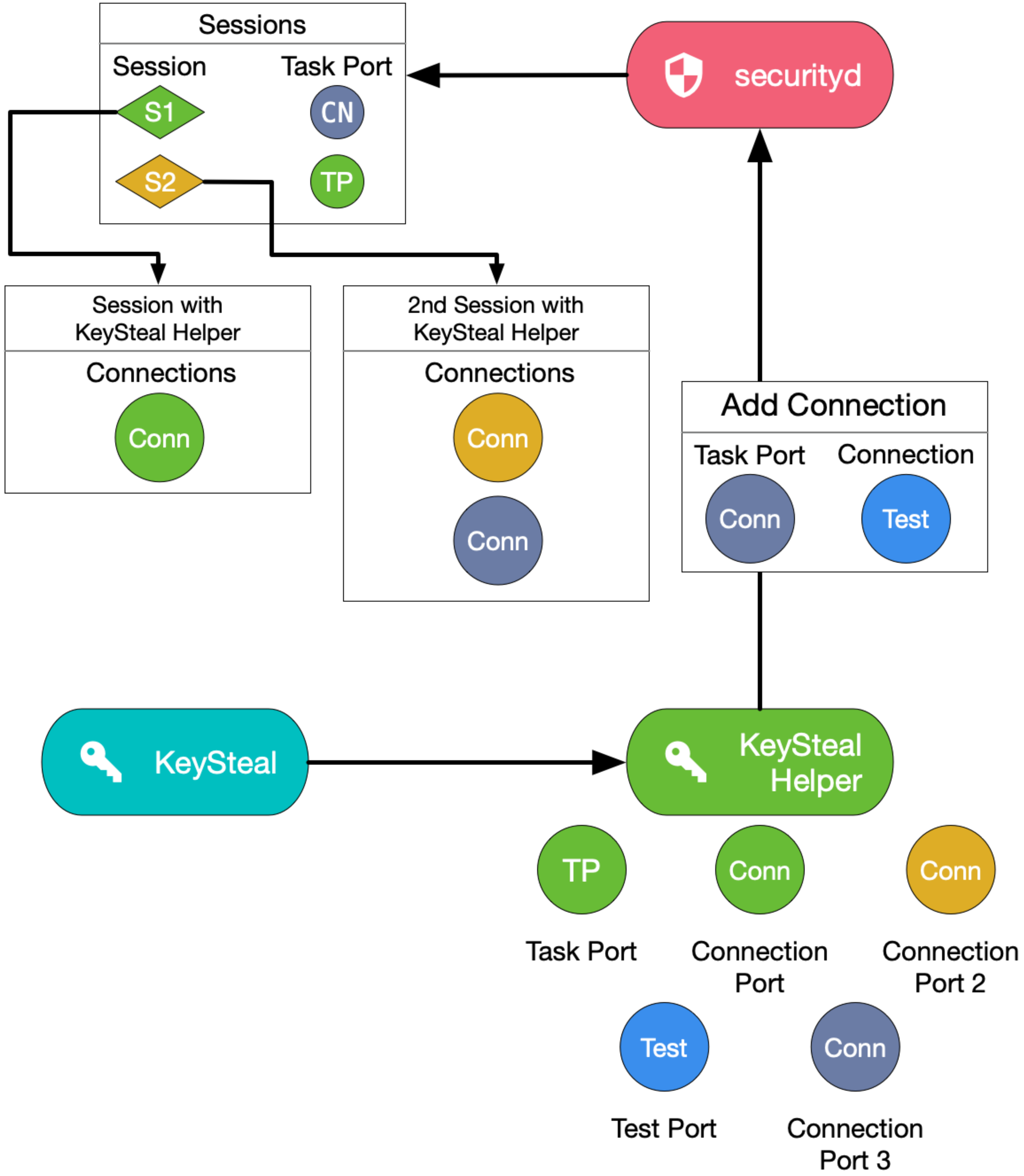
Test Port

Conn

Connection Port 3


Sessions
Session
Task Port
S1
CN
S2
TP
Same!
securityd
Session with KeySteal Helper
Connections
Conn
2nd Session with KeySteal Helper
Connections
Conn
Conn
Connection added
✓
KeySteal
KeySteal Helper
TP
Task Port
Conn
Connection Port
Conn
Connection Port 2
Test
Test Port
Conn
Connection Port 3

RECLAIMING



Sessions

Session | Task Port

S1 | CN

S2 | TP

securityd

Session with KeySteal Helper

Connections

Conn

2nd Session with KeySteal Helper

Connections

Conn

Conn

Add Connection

Task Port | Connection

Conn | Test

KeySteal

KeySteal Helper

TP

Task Port

Conn

Connection Port

Conn

Connection Port 2

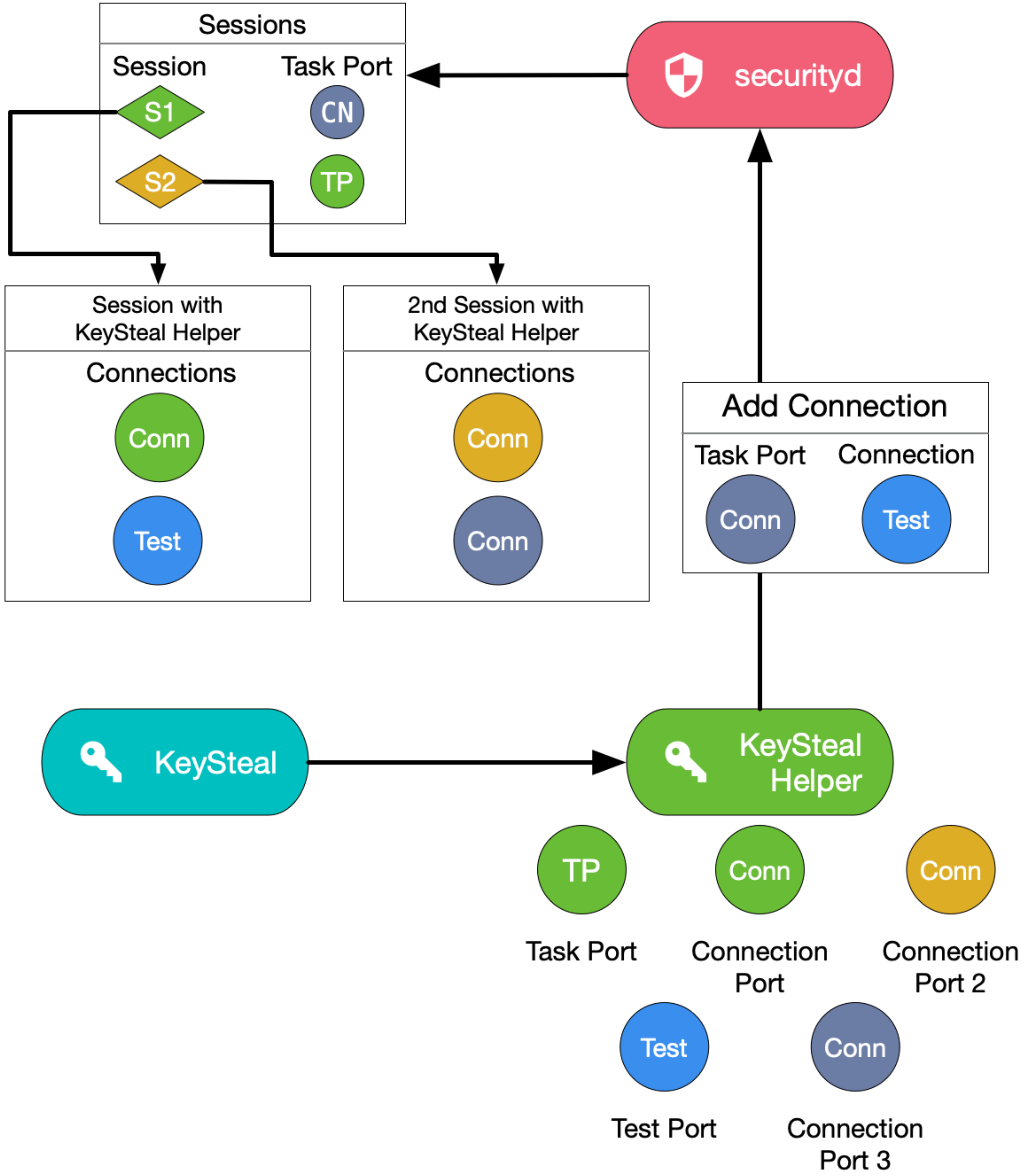
Test

Test Port

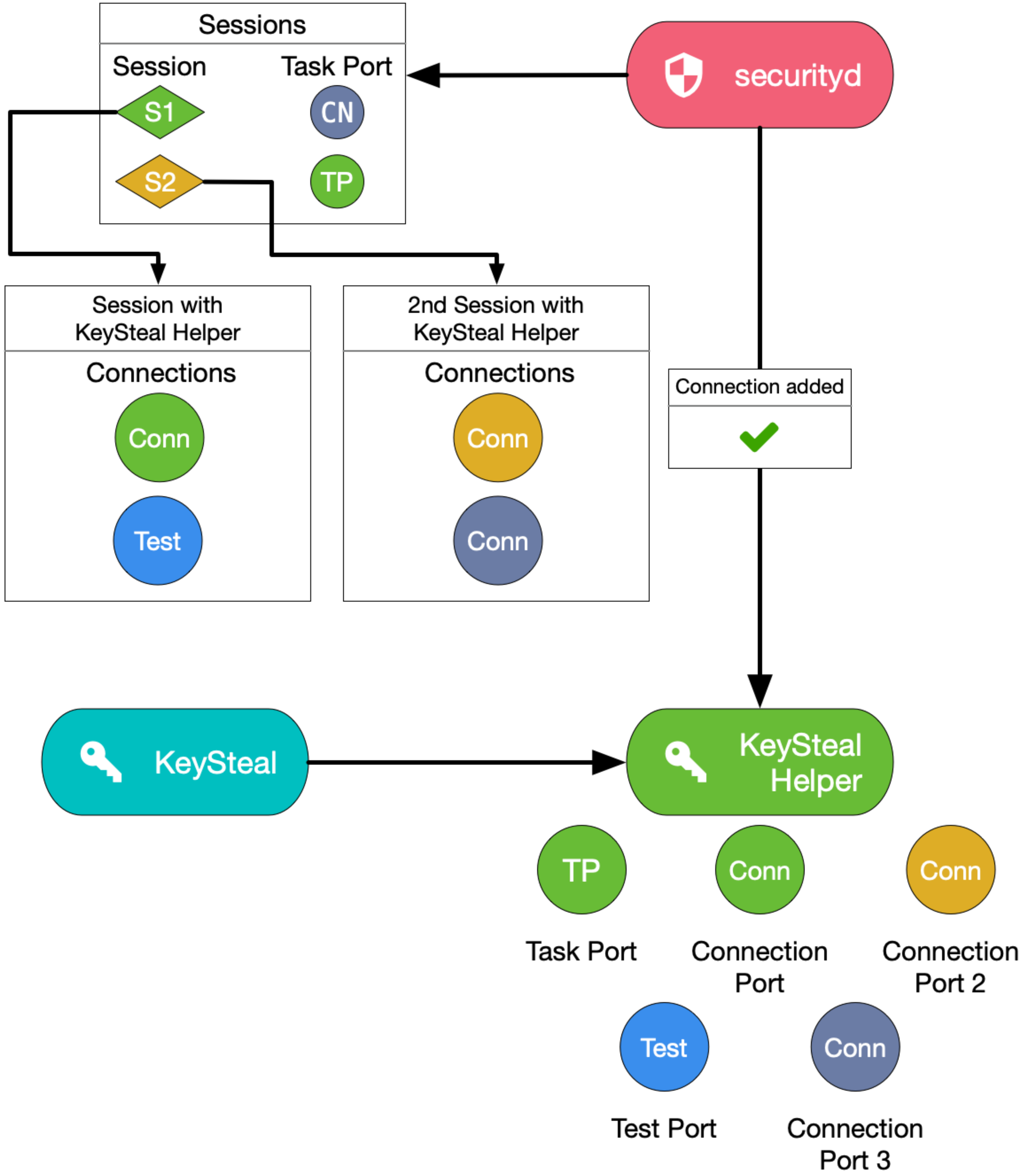
Conn

Connection Port 3

RECLAIMING


Sessions
Session
Task Port
S1
CN
S2
TP
securityd
Session with KeySteal Helper
Connections
Conn
Test
2nd Session with KeySteal Helper
Connections
Conn
Conn
Add Connection
Task Port
Connection
Conn
Test
KeySteal
KeySteal Helper
TP
Task Port
Conn
Connection Port
Conn
Connection Port 2
Test
Test Port
Conn
Connection Port 3

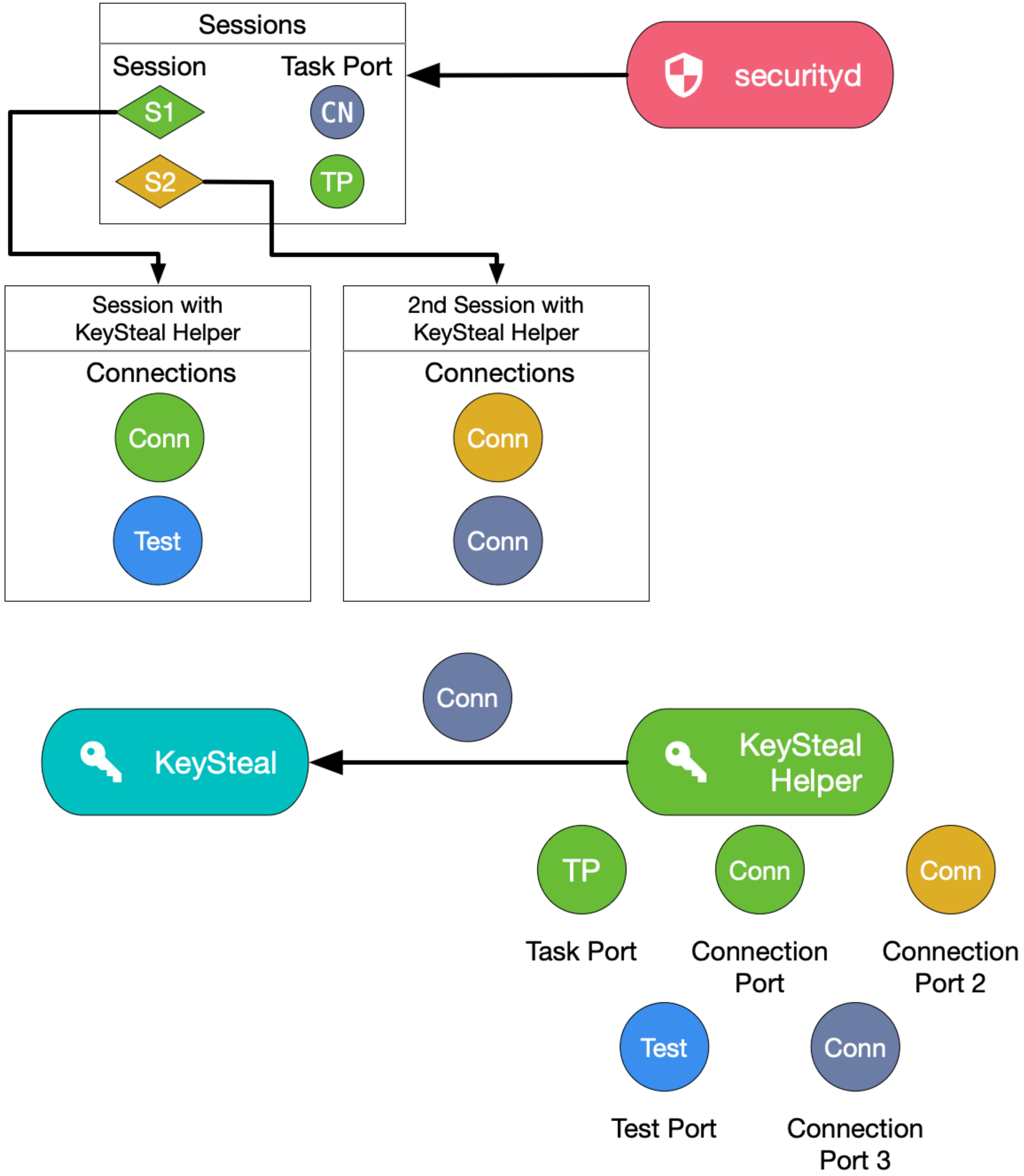
RECLAIMING

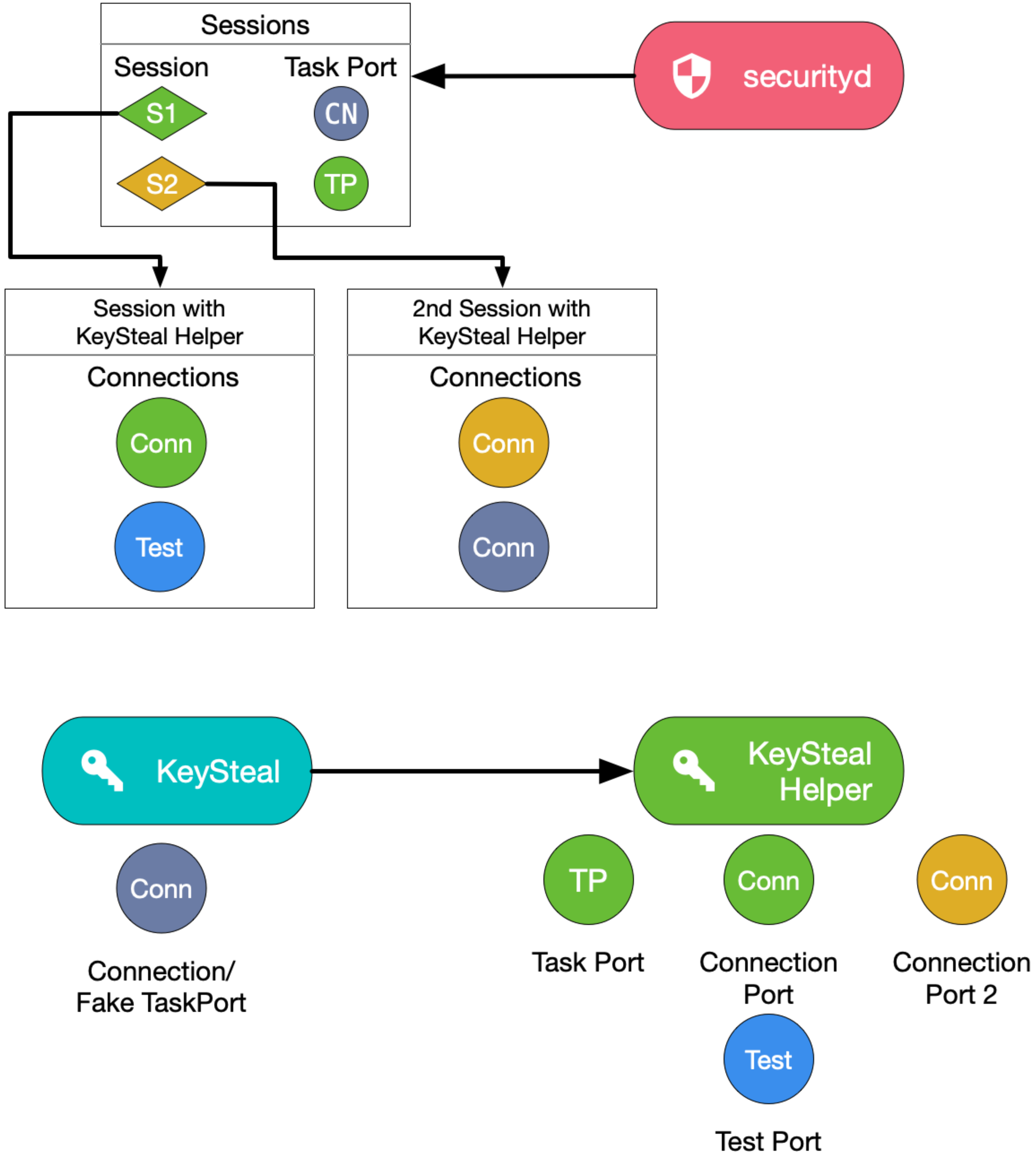

Sessions
Session
Task Port
S1
CN
S2
TP
securityd
Session with KeySteal Helper
Connections
Conn
Test
2nd Session with KeySteal Helper
Connections
Conn
Conn
Connection added
✓
KeySteal
KeySteal Helper
TP
Task Port
Conn
Connection Port
Conn
Connection Port 2
Test
Test Port
Conn
Connection Port 3

RECLAIMING

ATTACK PLAN

1. Create three processes: A, B and C ✓
2. B should create a session with securityd ✓
3. Send task port of B to C ✓
4. Let C free B's task port in securityd ✓
5. B should now reclaim it's session by sending securityd many ports, hoping one of them will get the same number as B's task port had ✓
6. Send this fake task port to A (receive right!) ←


Sessions
Session
Task Port
S1
CN
S2
TP
securityd
Session with KeySteal Helper
Connections
Conn
Test
2nd Session with KeySteal Helper
Connections
Conn
Conn
Conn
KeySteal
KeySteal Helper
TP
Task Port
Conn
Connection Port
Conn
Connection Port 2
Test
Test Port
Conn
Connection Port 3



Sessions

Session

Task Port

S1

CN

S2

TP

securityd

Session with KeySteal Helper

Connections

Conn

Test

2nd Session with KeySteal Helper

Connections

Conn

Conn

KeySteal

KeySteal Helper

Conn

Connection/ Fake TaskPort

TP

Task Port

Conn

Connection Port

Test

Test Port

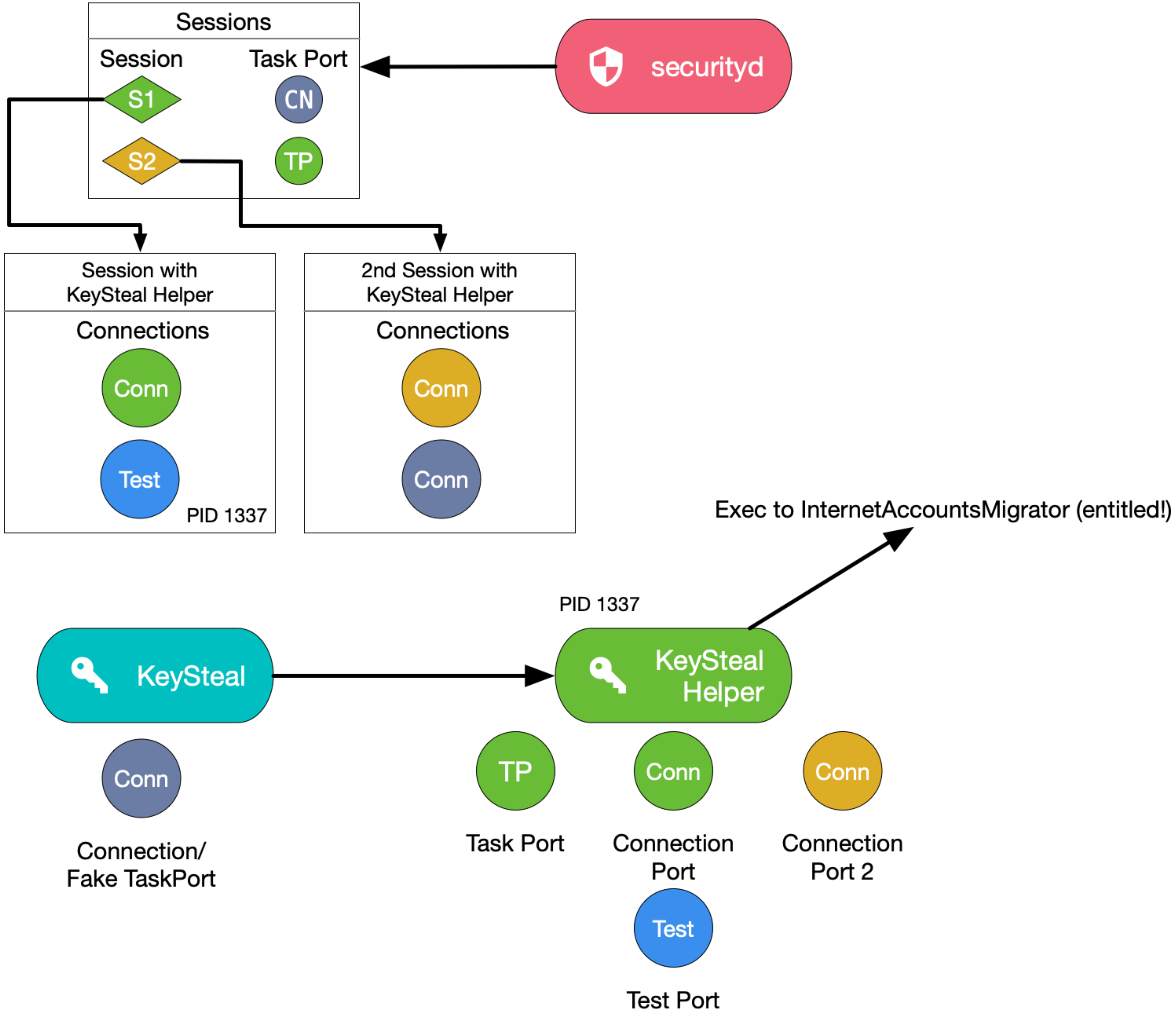
Conn

Connection Port 2

RECLAIMED

ATTACK PLAN

1. Create three processes: A, B and C ✓
2. B should create a session with securityd ✓
3. Send task port of B to C ✓
4. Let C free B's task port in securityd ✓
5. B should now reclaim it's session by sending securityd many ports, hoping one of them will get the same number as B's task port had ✓
6. Send this fake task port to A (receive right!) ✓
7. B should exec internetAccountsMigrator ←
   7.1. Reclaimed session won't be deleted as A now owns the fake task port which therefore won't be deleted



Sessions

Session

Task Port

S1

CN

S2

TP

securityd

Session with KeySteal Helper

Connections

Conn

Test

PID 1337

2nd Session with KeySteal Helper

Connections

Conn

Conn

Exec to InternetAccountsMigrator (entitled!)

PID 1337

KeySteal

KeySteal Helper

Conn

Connection/ Fake TaskPort

TP

Task Port

Conn

Connection Port

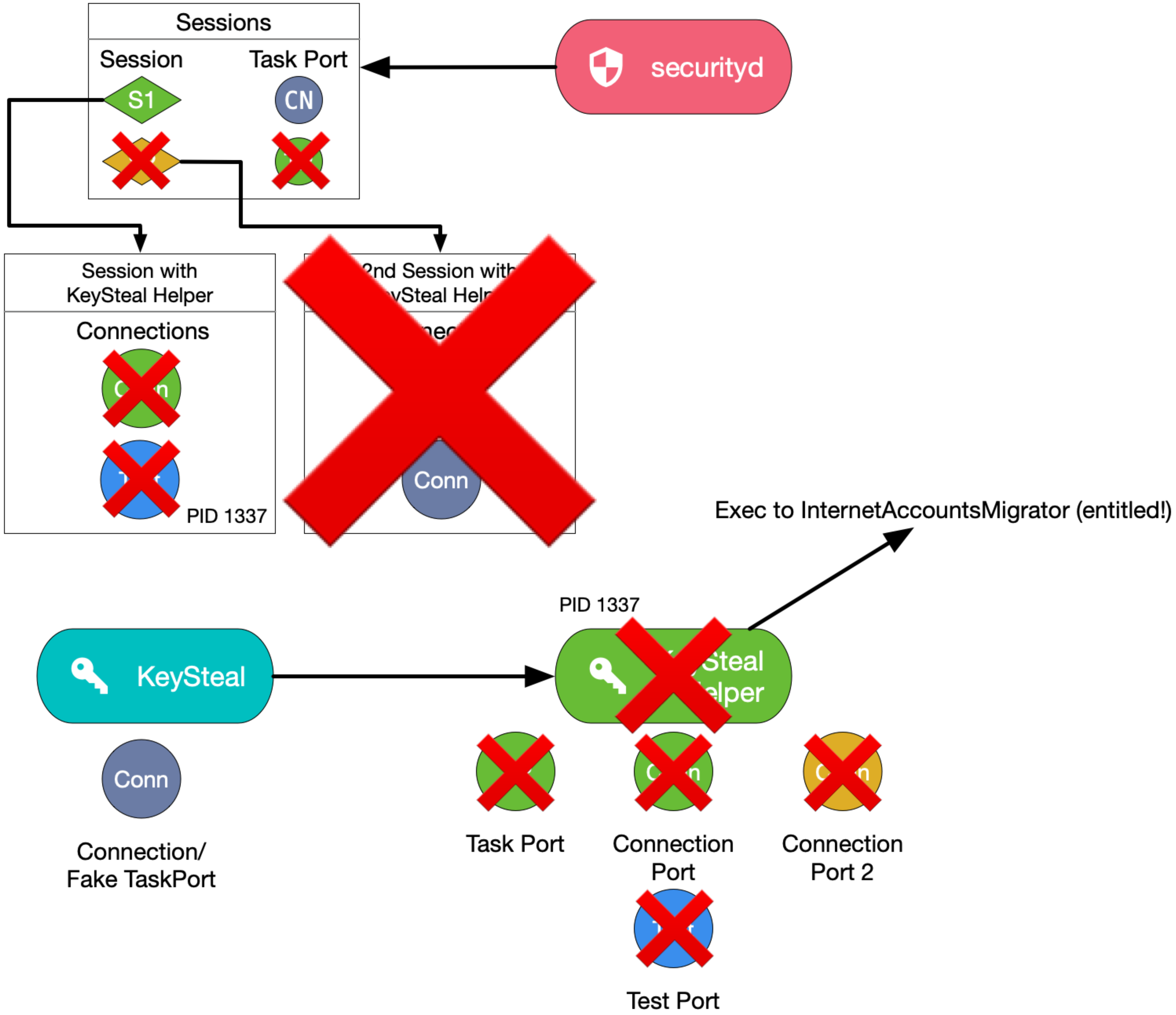
Test

Test Port

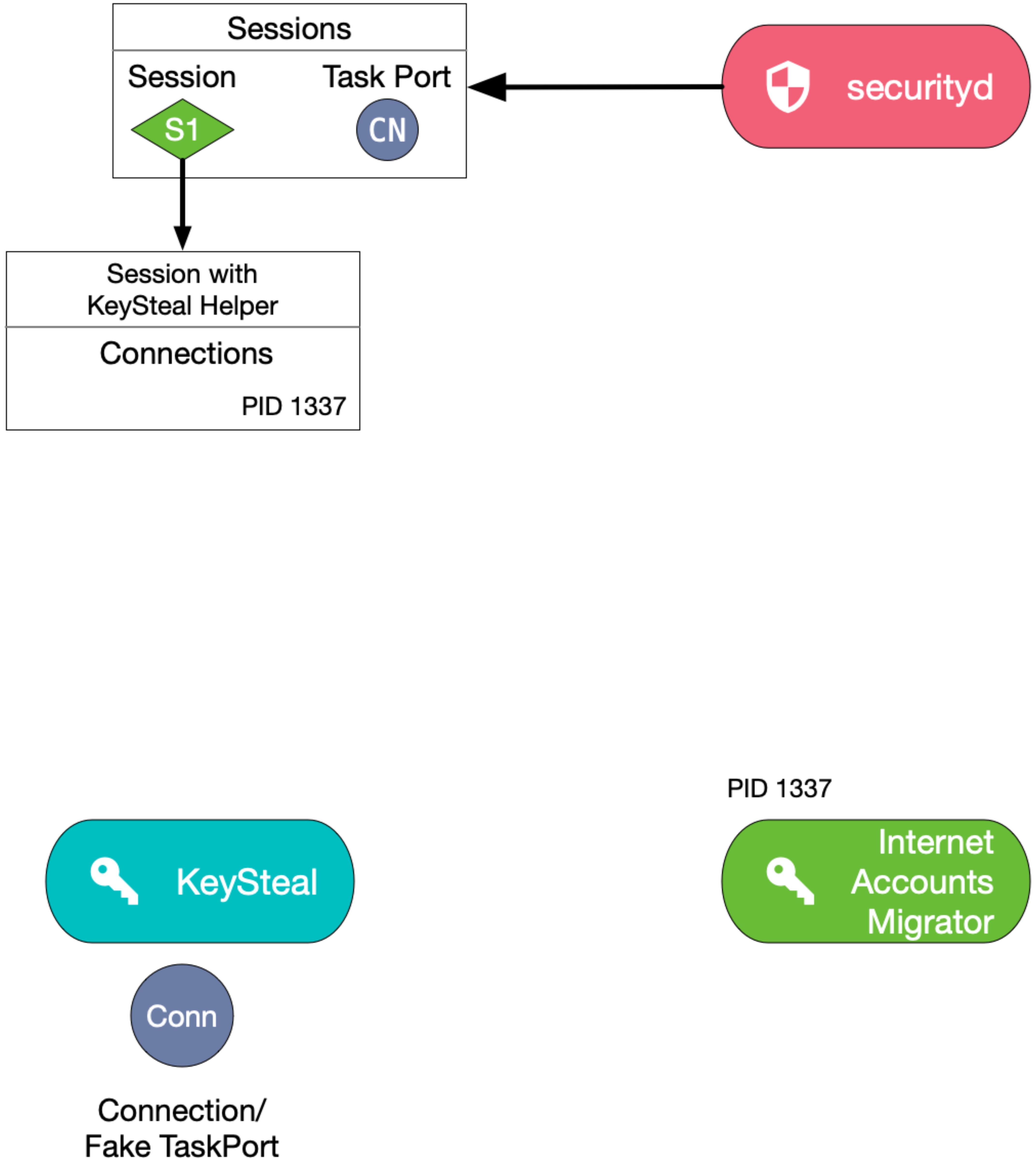
Conn

Connection Port 2

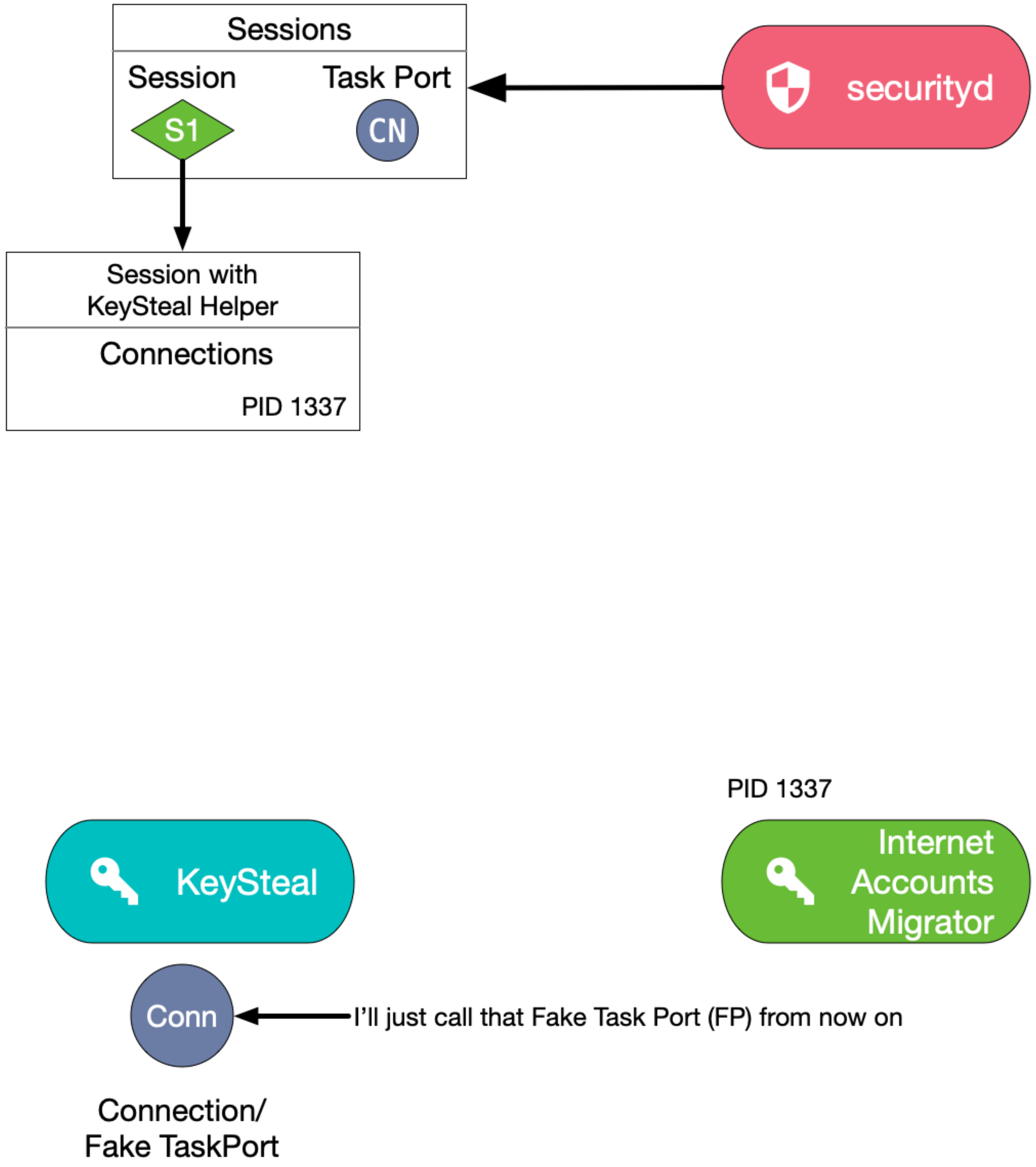
EXEC


Sessions
Session
Task Port
S1
CN
securityd
Session with KeySteal Helper
Connections
PID 1337
2nd Session with
ySteal Help
Conn
Exec to InternetAccountsMigrator (entitled!)
PID 1337
KeySteal
Steal
elper
Conn
Connection/
Fake TaskPort
Task Port
Connection Port
Connection Port 2
Test Port

EXEC


Sessions
Session
Task Port
S1
CN
securityd
Session with KeySteal Helper
Connections
PID 1337
PID 1337
KeySteal
Internet Accounts Migrator
Conn
Connection/ Fake TaskPort

AFTER EXEC

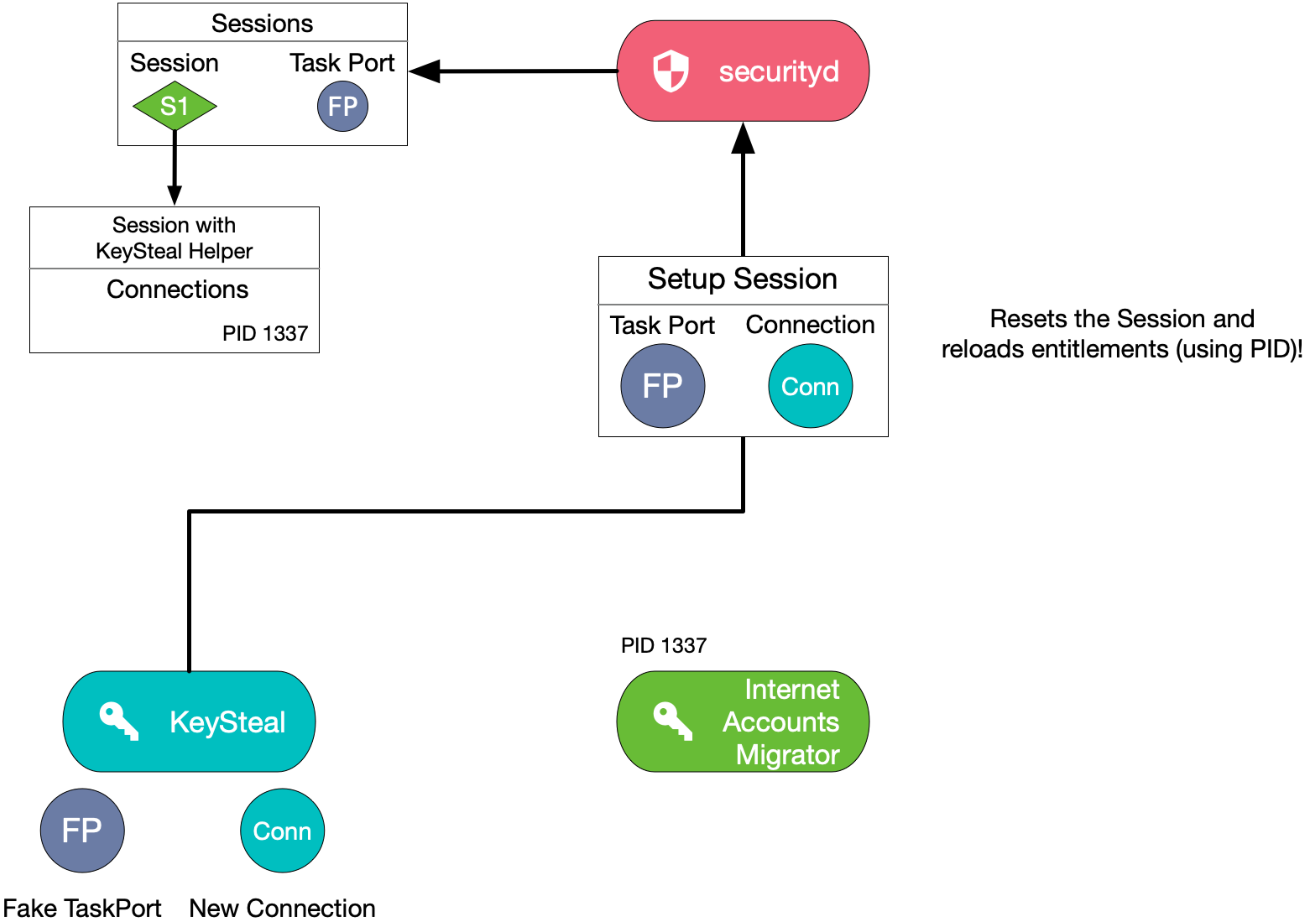

Sessions
Session
Task Port
S1
CN
securityd
Session with KeySteal Helper
Connections
PID 1337
PID 1337
KeySteal
Internet Accounts Migrator
Conn
Connection/ Fake TaskPort

I'll just call that Fake Task Port (FP) from now on

AFTER EXEC

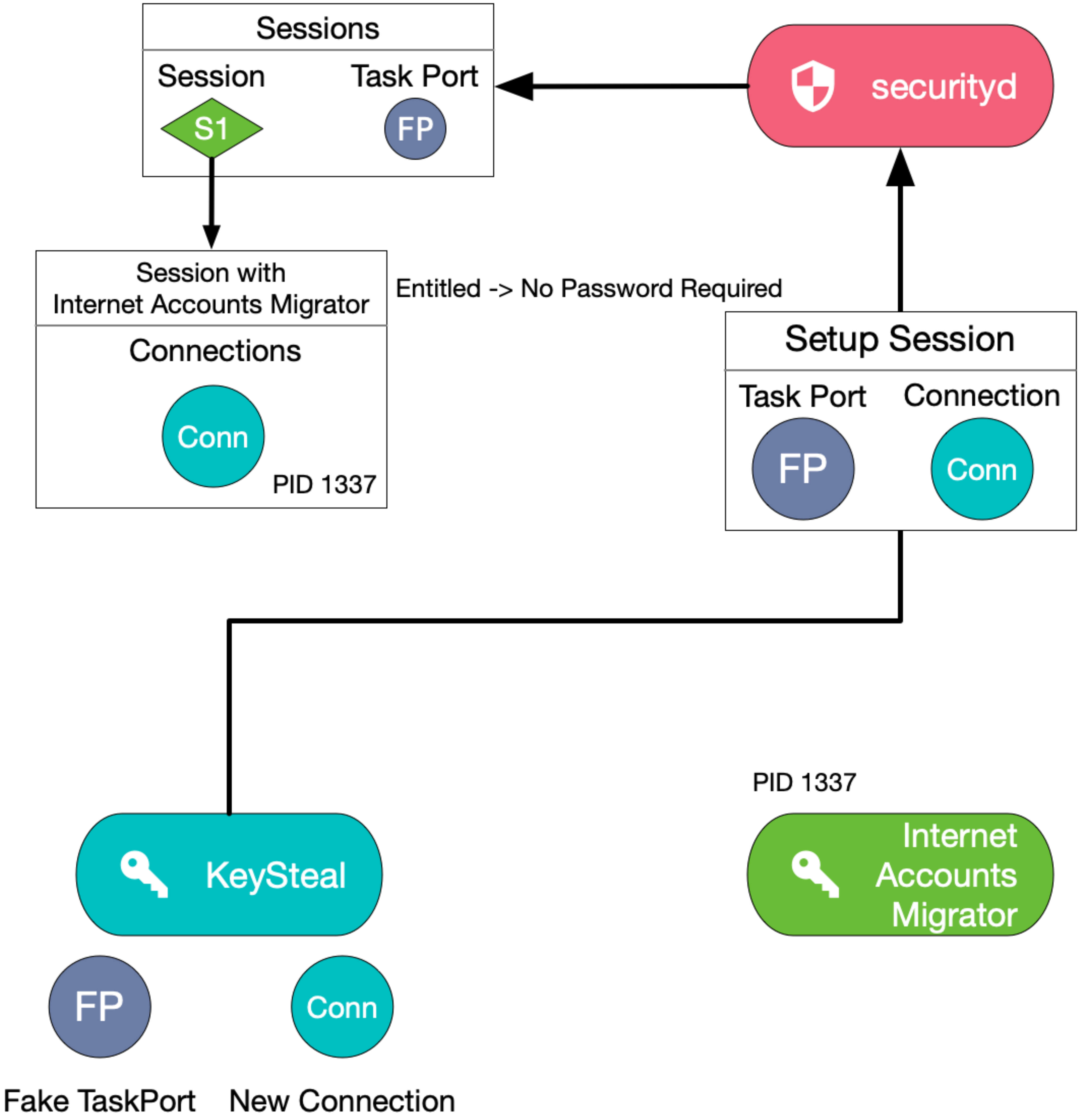
ATTACK PLAN

1. Create three processes: A, B and C ✓
2. B should create a session with securityd ✓
3. Send task port of B to C ✓
4. Let C free B's task port in securityd ✓
5. B should now reclaim it's session by sending securityd many ports, hoping one of them will get the same number as B's task port had ✓
6. Send this fake task port to A (receive right!) ✓
7. B should exec internetAccountsMigrator ✓
8. A can now reset B's session using the fake task port ←
    8.1. Causes the entitlements of internetAccounts migrator to be loaded

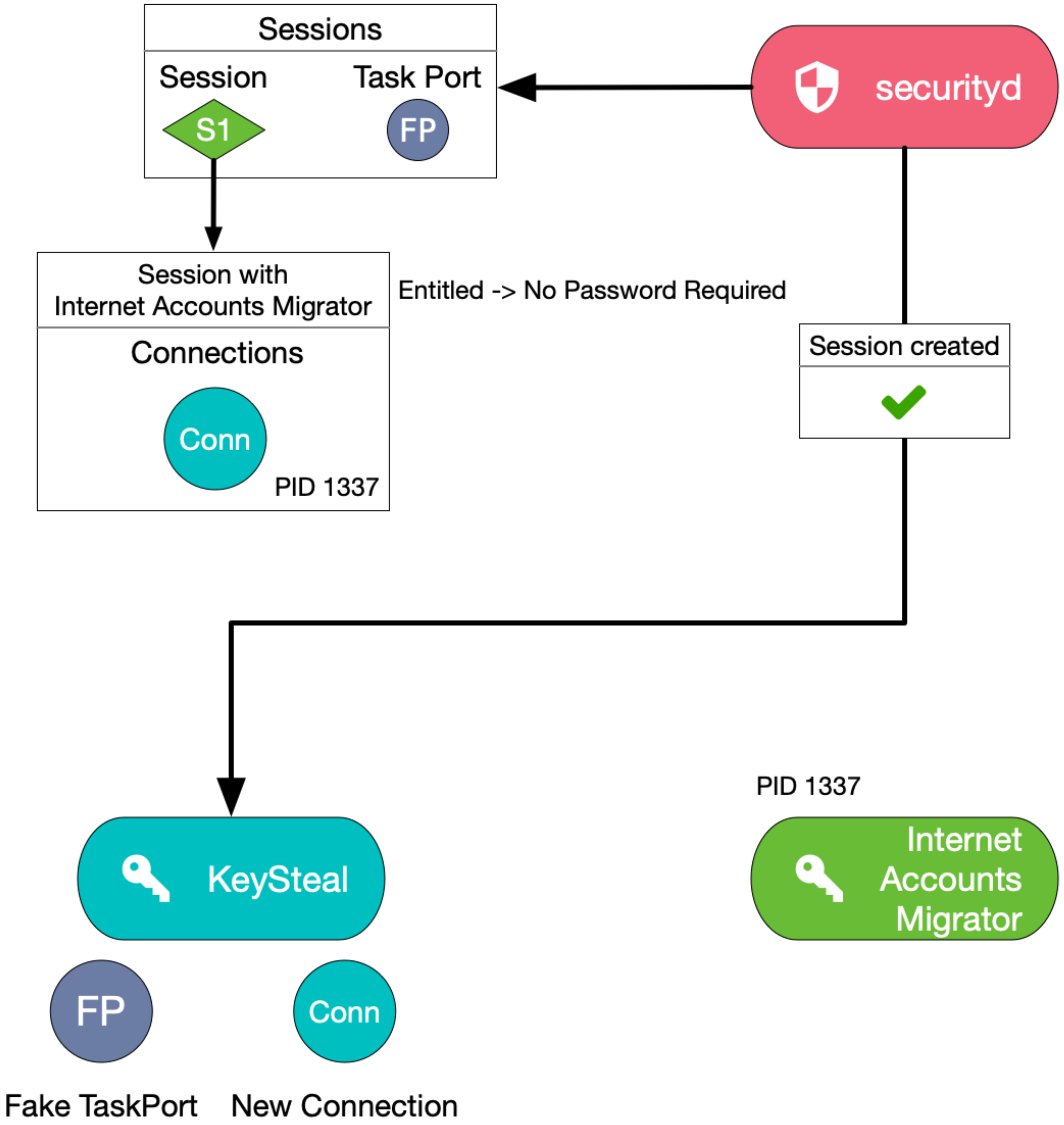


Resets the Session and
reloads entitlements (using PID)!

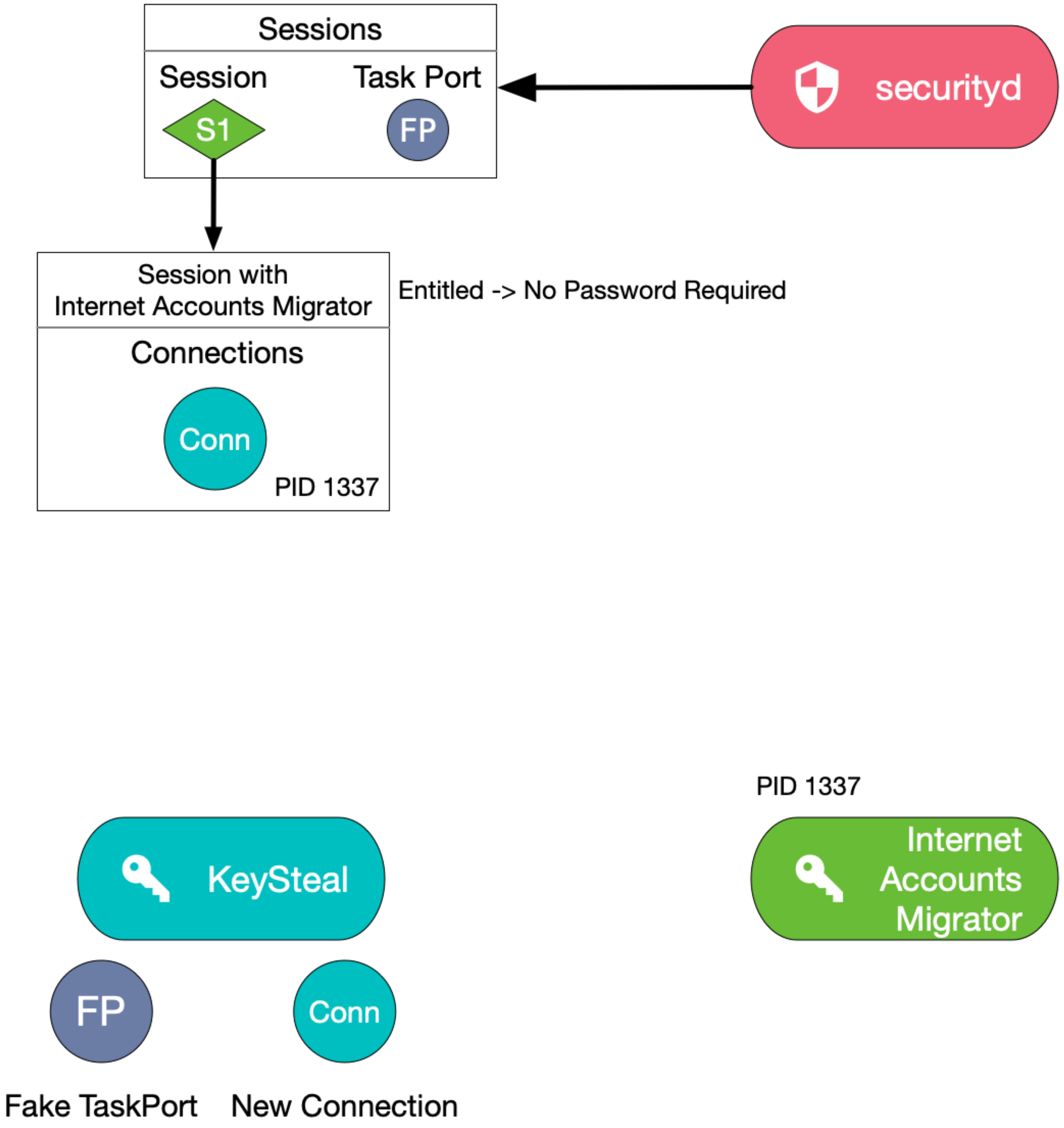
RESET SESSION



RESET SESSION



RESET SESSION



Sessions

Session

Task Port

S1

FP

securityd

Session with
Internet Accounts Migrator

Entitled -> No Password Required

Connections

Conn

PID 1337

PID 1337

KeySteal

Internet Accounts Migrator

FP

Conn

Fake TaskPort

New Connection

DONE

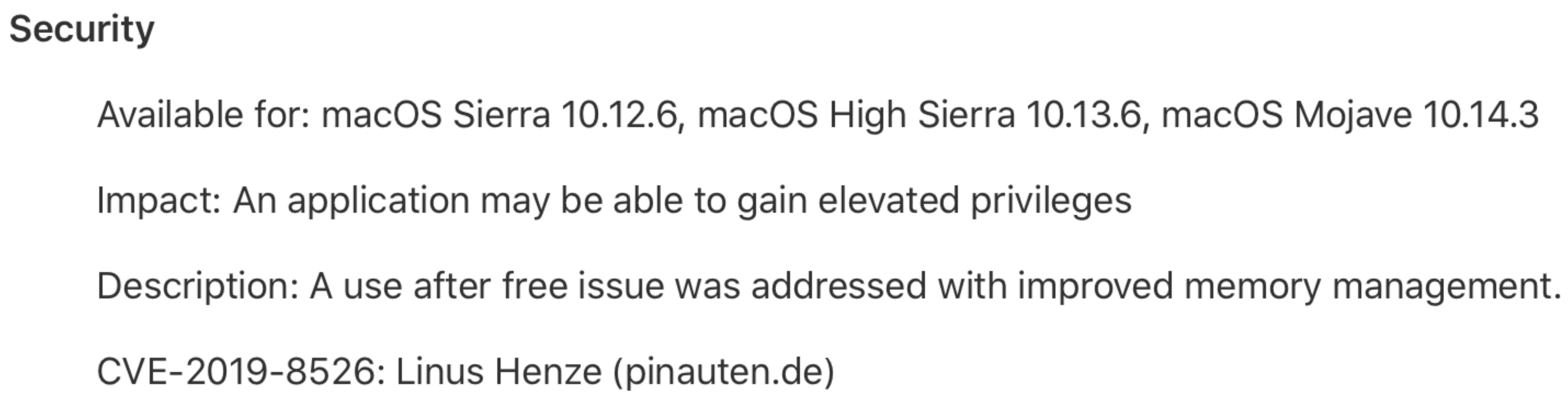
ATTACK PLAN

1. Create three processes: A, B and C ✓
2. B should create a session with securityd ✓
3. Send task port of B to C ✓
4. Let C free B's task port in securityd ✓
5. B should now reclaim it's session by sending securityd many ports, hoping one of them will get the same number as B's task port had ✓
6. Send this fake task port to A (receive right!) ✓
7. B should exec internetAccountsMigrator ✓
8. A can now reset B's session using the fake task port ✓
    8.1. Causes the entitlements of internetAccounts migrator to be loaded
9. Use fake task port to access keychain!!! ←

HOW DID APPLE FIX THE BUG?

KEYCHAIN IS SAFE AGAIN

WHAT APPLE SAYS



Security

Available for: macOS Sierra 10.12.6, macOS High Sierra 10.13.6, macOS Mojave 10.14.3

Impact: An application may be able to gain elevated privileges

Description: A use after free issue was addressed with improved memory management.

CVE-2019-8526: Linus Henze (pinauten.de)

support.apple.com/en-us/HT209600

APPLE'S PATCH

```
//
// Reset Code Signing Hosting state.
// This turns hosting off and clears all children.
//
void CodeSigningHost::reset()
{
  StLock<Mutex> _(mLock);
  switch (mHostingState) {
  case noHosting:
    break; // nothing to do
  case dynamicHosting:
    mHostingPort.deallocate();
    mHostingPort = MACH_PORT_NULL;
    secnotice("SecServer", "%d host unregister", mHostingPort.port());
    break;
  case proxyHosting:
    Server::active().remove(*this);// unhook service handler
    mHostingPort.destroy(); // destroy receive right
    mHostingState = noHosting;
    mHostingPort = MACH_PORT_NULL;
    mGuests.erase(mGuests.begin(), mGuests.end());
    secnotice("SecServer", "%d host unregister", mHostingPort.port());
    break;
  }
}
```

Now calling deallocate instead of destroy

```
//
// Screen a process setup request for an existing process.
// This means the client has requested intialization even though we remember having
// talked to it in the past. This could either be an exec(2), or the client could just
// have forgotten all about its securityd client state. Or it could be an attack...
//
void Process::reset(TaskPort taskPort, const ClientSetupInfo *info, const CommonCriteria::AuditToken &audit)
{
  StLock<Mutex> _(*this);
  if (taskPort != mTaskPort) {
    secnotice("SecServer", "Process %p(%d) reset mismatch (tp %d-%d)",
      this, pid(), taskPort.port(), mTaskPort.port());
    //@@@ CssmError::throwMe(CSSM_ERRCODE_VERIFICATION_FAILURE);   // liar
  }
  setup(info);
  CFCopyRef<SecCodeRef> oldCode = processCode();

  // Note: The following will reload the code signature of the process
  // including all entitlements
  // Now using the generation number as well
  ClientIdentification::setup(this->pid(), this->generationNumber());
  if (CFEqual(oldCode, processCode())) {
        secnotice("SecServer", "%p Client reset amnesia", this);
  } else {
        secnotice("SecServer", "%p Client reset full", this);
    CodeSigningHost::reset();
  }
}
```

Using generation number now

KEYSTEAL ON ACTION

KEYSTEAL DEMO

Thank you!

Linus Henze

@LinusHenze

www.pinauten.de

github.com/LinusHenze/Keysteal

CONTACT