NETFLIX

Improving Cloud Security with Attacker Profiling

Bryan D. Payne Engineering Manager, Platform Security

NETFLIX

Who is out to get me?

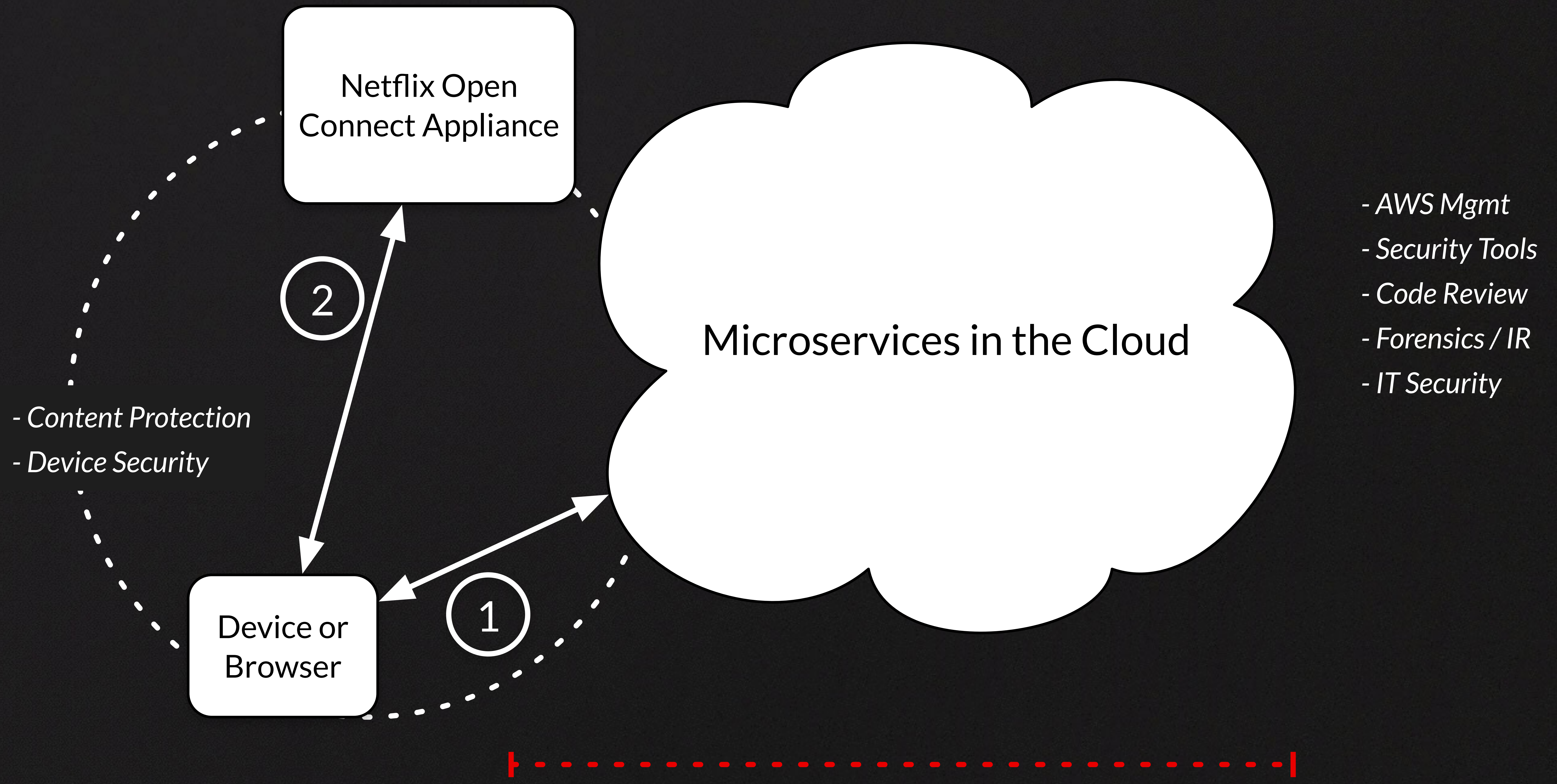
What do they want?

Why are we losing?

NETFLIX

Platform Security at Netflix

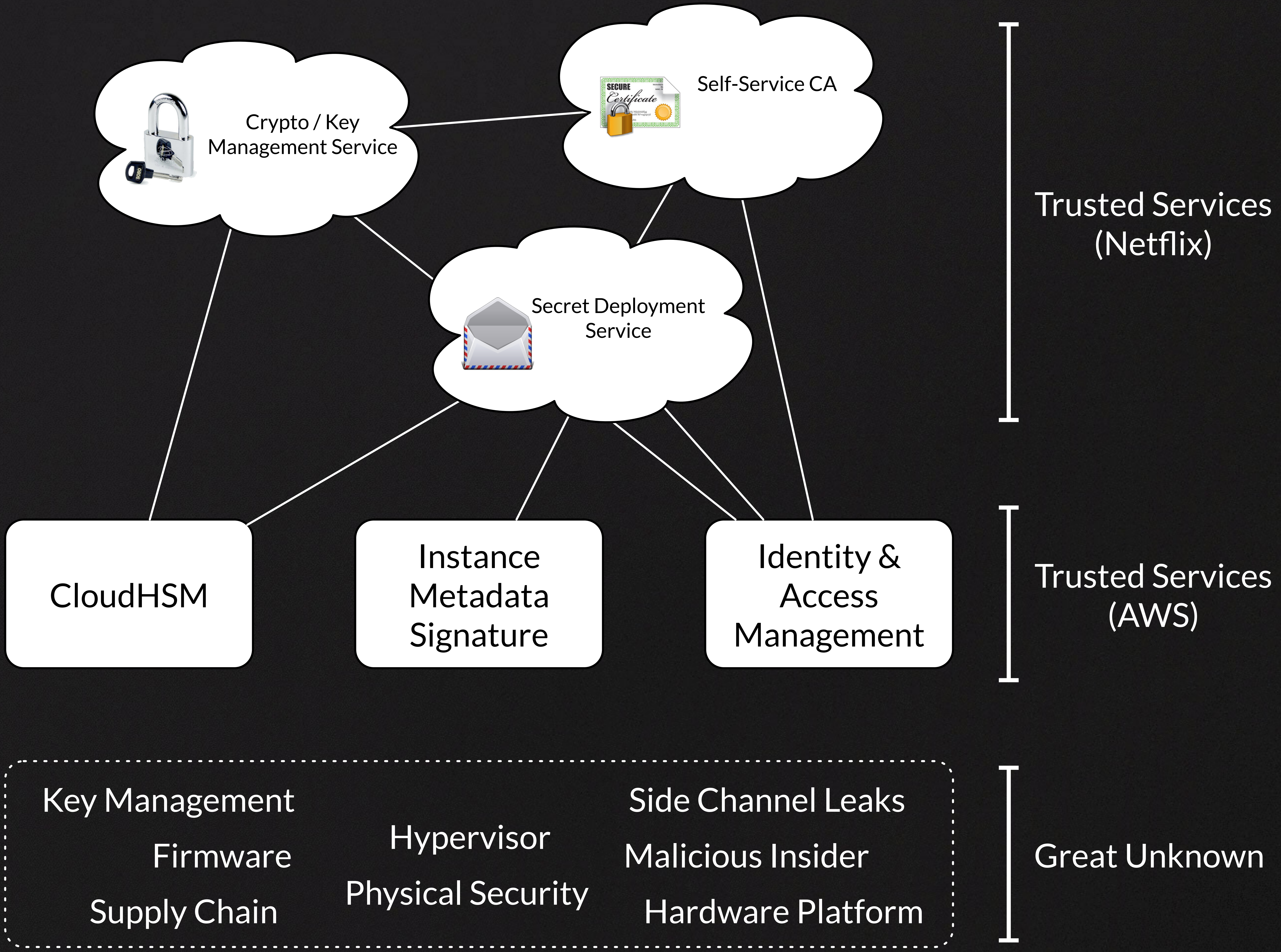
Platform Security Overview



Platform Security

- *Foundational Security Services*
- *Security in Common Platform*
- *Security by Default in base AMI*

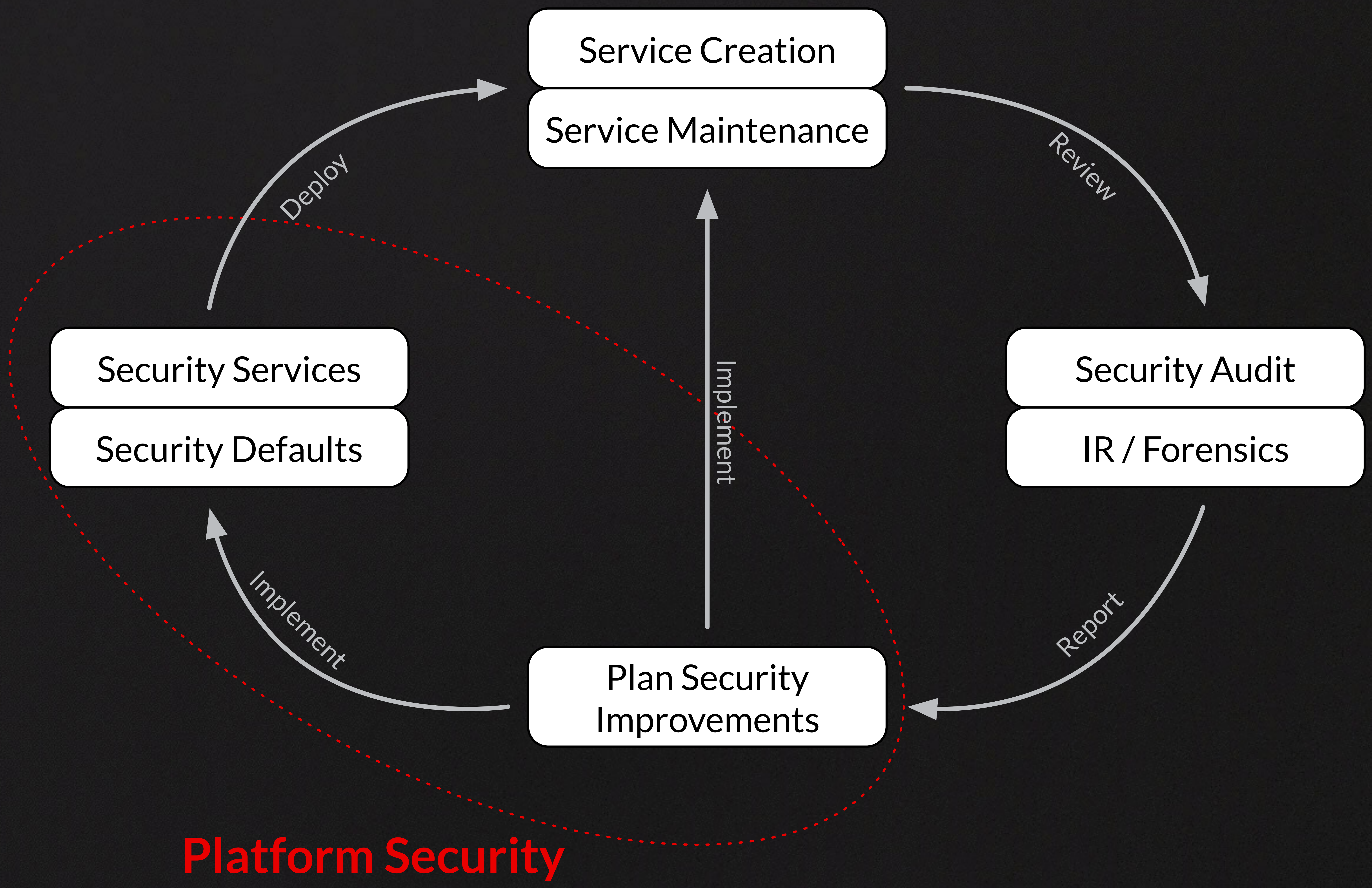
Classic Security via AWS



NETFLIX

Ubiquitous Security

- Partner with other teams
- Make security transparent (or easy)
- Focus on common components
- Also focus on strategic risks

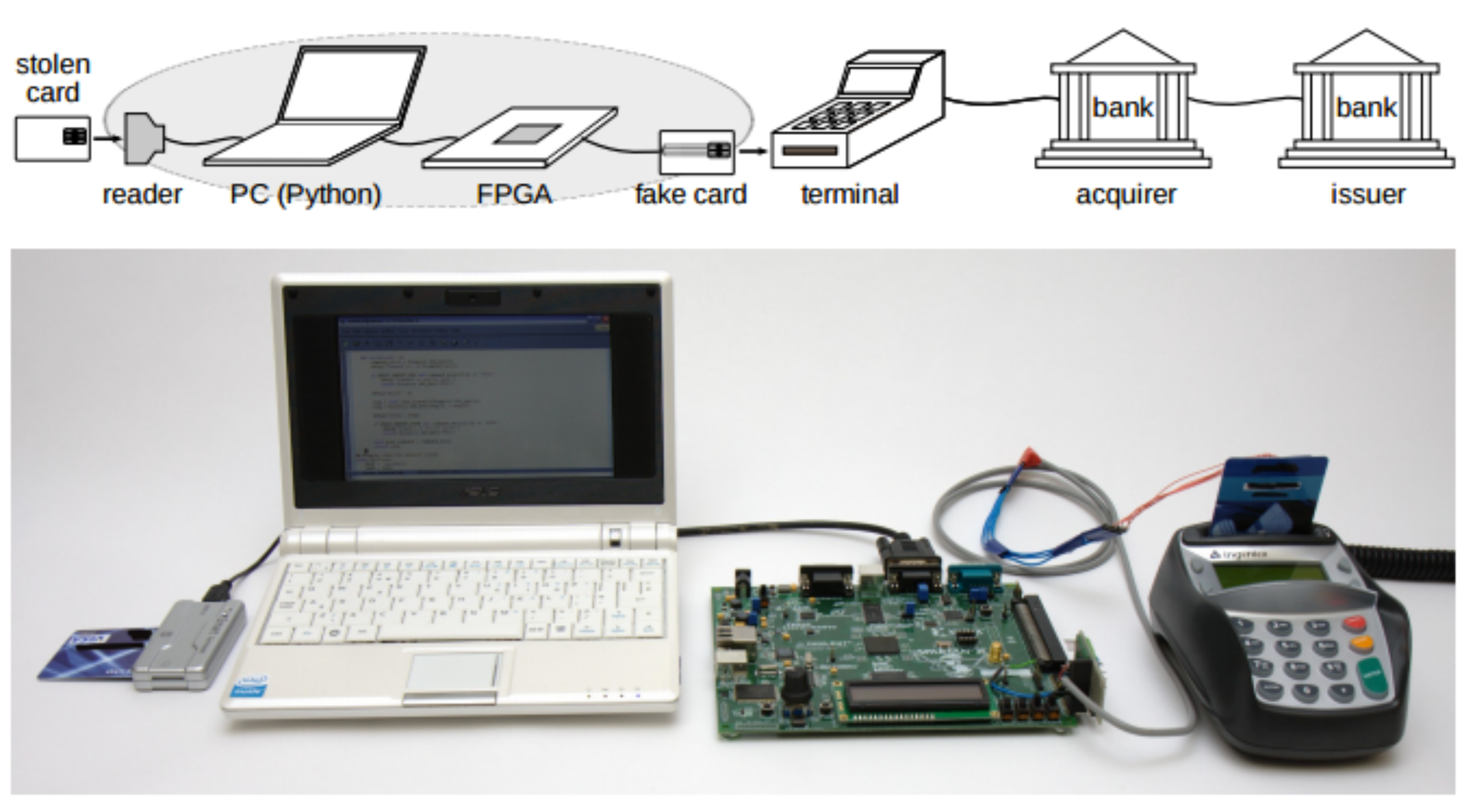


NETFLIX

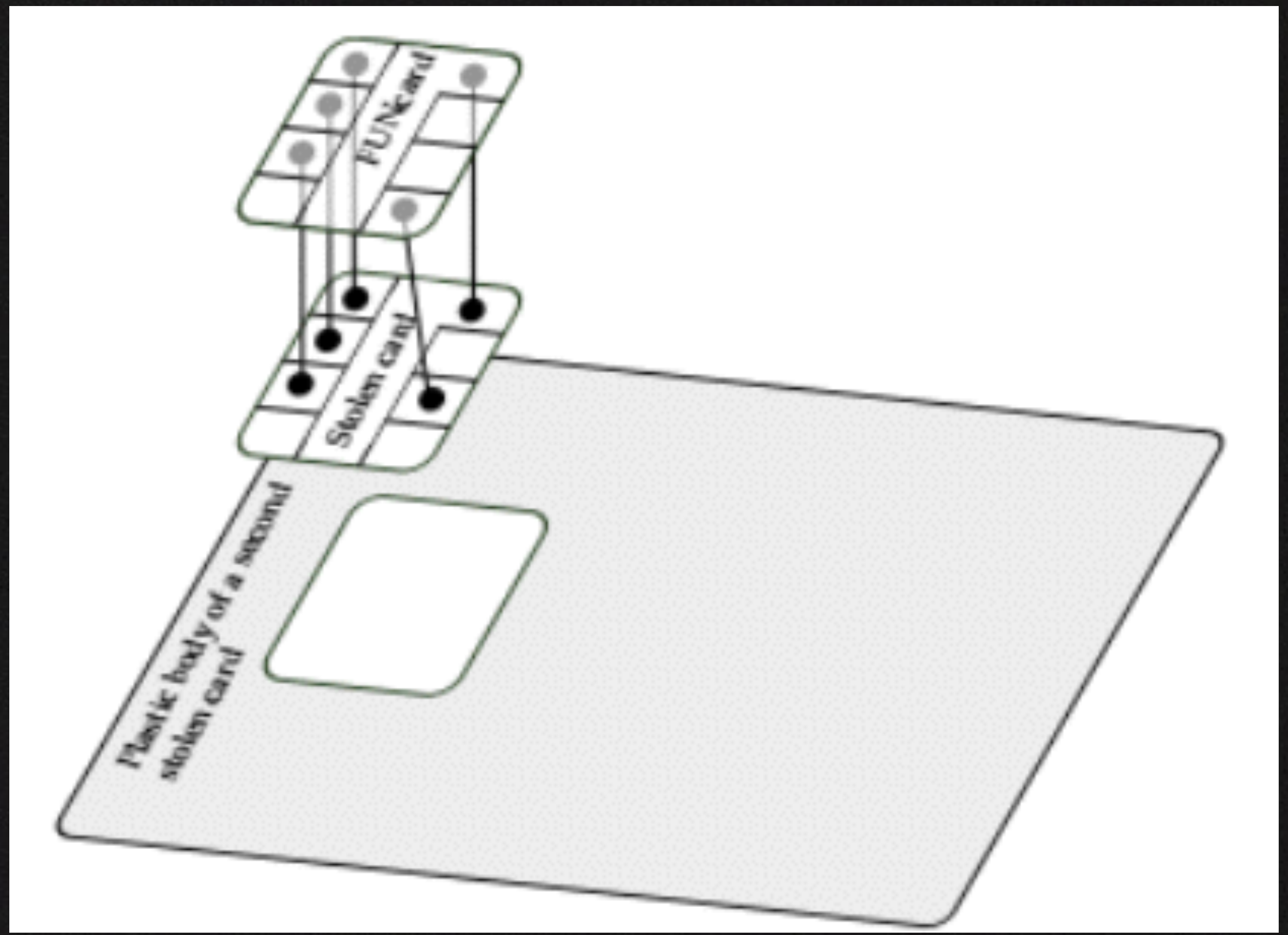
Who is out to get me?


Own your tomorrow
charles SCHWAB BANK
4809 9010 0000 0000
4809
VALID THRU 01/17
DEBIT
VISA
thankyou PREFERRED
citi
4128 0033 0000 0000
4128
Valid from 01/15
Expiration date 02/28/19
Member since 15
VISA SIGNATURE
BankAmericard | Travel Rewards
4313 0738 0000 0000
4313
VALID THRU 05/15
CARDHOLDER SINCE 04
VISA SIGNATURE
WELLS FARGO
4147 1801 0000 0000
4147
CUSTOMER SINCE 2015
GOOD THRU 01/31/18
VISA SIGNATURE

BBC Newsnight, 11 February 2010
https://www.youtube.com/watch?v=1pMuV2o4Lrw



Murdoch et al, Chip and PIN is Broken,
IEEE Symposium on Security and Privacy, 2010

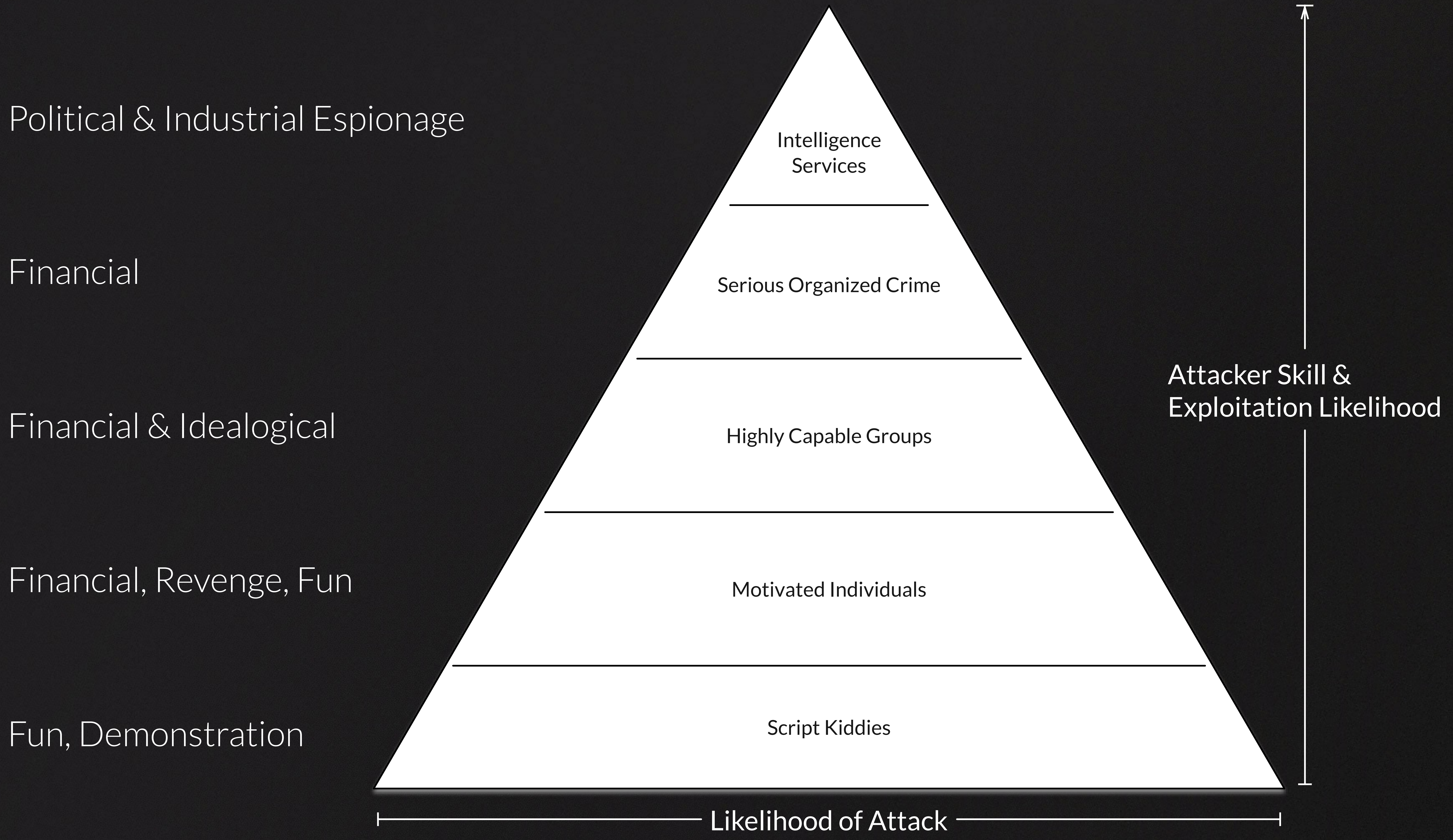


Greenberg, X-Ray Scans Expose and Ingenious
Chip-and-PIN Card Hack, Wired, 19 October 2015

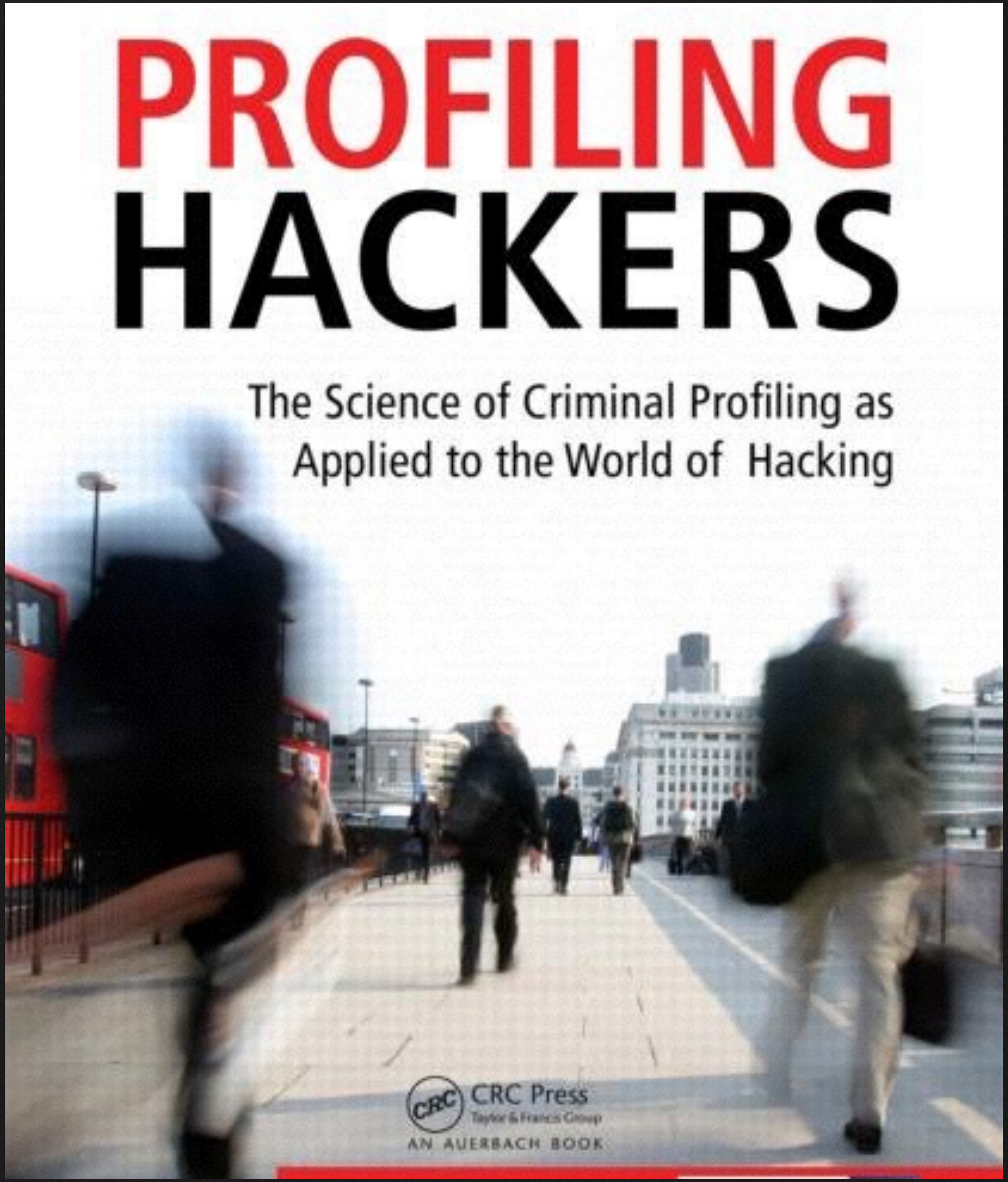
NETFLIX

Attacker Motivations

- financial / business
- political / idealogical
- revenge
- demonstration
- fun


Political & Industrial Espionage
Financial
Financial & Idealogical
Financial, Revenge, Fun
Fun, Demonstration
Intelligence Services
Serious Organized Crime
Highly Capable Groups
Motivated Individuals
Script Kiddies
Attacker Skill & Exploitation Likelihood
Likelihood of Attack

- Little trust in authorities
- Desire control
- Hacker life kept secret
- “Don’t foul your own nest”



NETFLIX

Attacker Characteristics

- creative and brilliant
- curious
- motivated
- shy in real life
- comfortable with computers

"Yes, I am a criminal. My crime is that of curiosity."

The Hacker Manifesto

NETFLIX

Attack Characteristics

- access (nmap, exploit, configuration error, etc)
- file cleaners
- backdoor
- password cracking
- monitor system admin
- proceed with goals (files, network sniffing, etc)

Photo Credit: Google
http://www.google.com/about/datacenters/gallery/

NETFLIX

What do they want?

"Antwerpen Hoveniersstraat" by Thorsten1997 - Own work. Licensed under Public Domain via Commons

19 February 2003
BBC News
http://news.bbc.co.uk/2/hi/europe/2782305.stm

Diamond heist baffles police

Belgian police are trying to unravel events behind a daring robbery in the diamond-cutting capital of the world.

Antwerp: At the centre of the diamond trade for 400 years

Thieves cleared out 123 of the 160 vaults in the maximum security cellars at Antwerp's Diamond Centre at the weekend, but the raid was only discovered the next day.

The precise value of the stolen diamonds is not known, but Belgian media have speculated it could run to millions of dollars.

Diamond traders in the city have been shocked by the audacity of the robbery and fear it could be a blow to their industry.

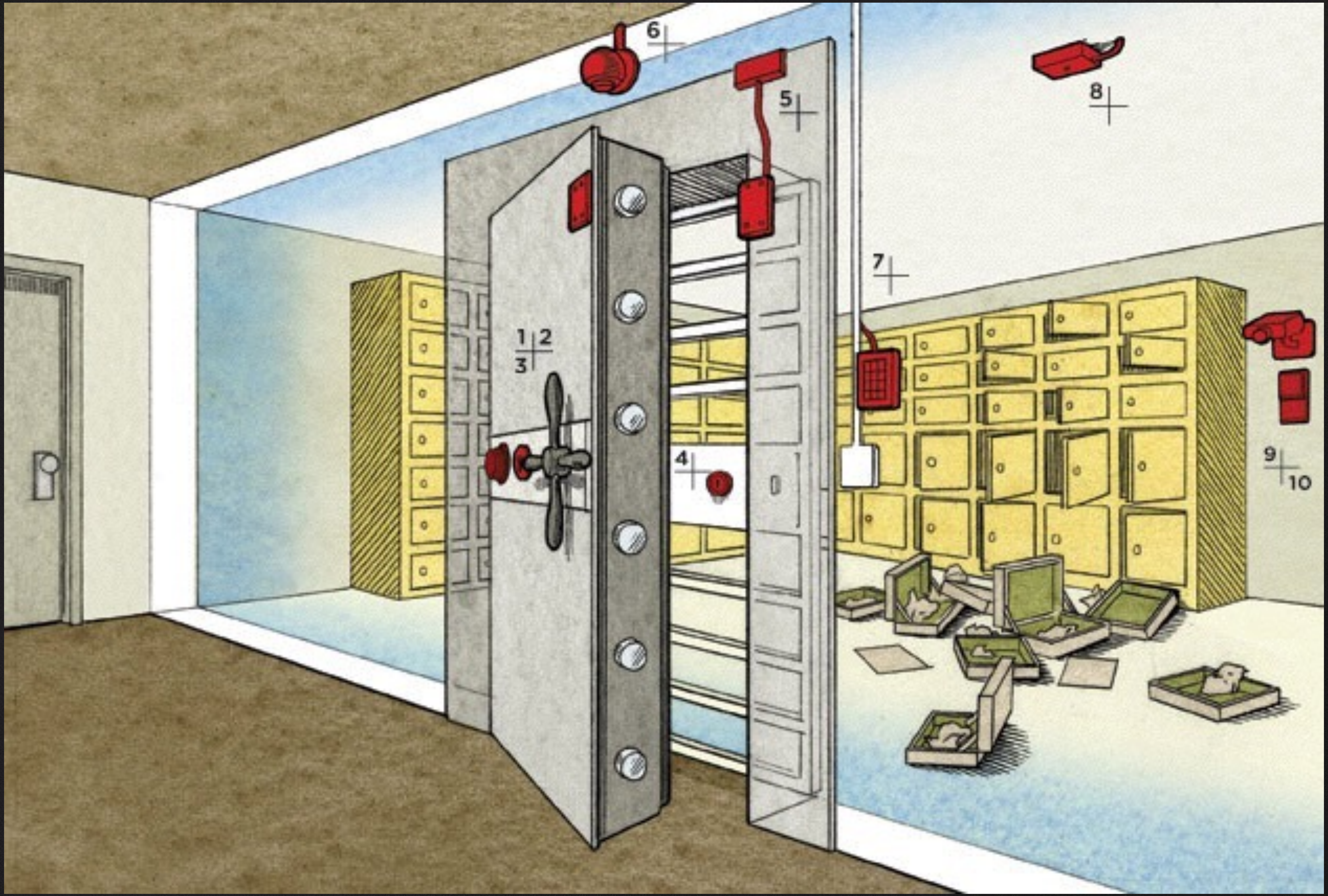
Inside job?

The Diamond Centre building, located in the heart of Antwerp's historic diamond district, is closely guarded.

There are surveillance camers, entry codes, 24-hour security guards, and even cameras in the vaults.

But with no signs of a break-in, police suspect the thieves could have had inside help and have been questioning staff and owners of the safes at the centre.

Antwerp's Diamond High Council, which represents the gemstone traders, has admitted the robbery could have serious implications for an industry proud of its discretion and security.



1. Combination dial
2. Keyed lock
3. Seismic sensor
4. Locked steel grate
5. Magnetic sensor
6. External security camera
7. Keypad to disarm sensors
8. Light sensor
9. Internal security camera
10. Heat / motion sensor

Joshua Davis. *The Untold Story of the World's Biggest Diamond Heist*.
Wired, http://archive.wired.com/politics/law/magazine/17-04/ff_diamonds

UNITED STATES OFFICE OF PERSONNEL MANAGEMENT

Adobe

ASHLEY MADISON.COM
Life is Short. Have an Affair.

THE HOME DEPOT

ebay

JPMorganChase

TARGET

T · · Mobile ·

Experian

Anthem

- USG employee background checks & fingerprints
- Credit cards
- User data
- PPI: SSN, driver's license, phone, address, DoB, etc
- Passwords

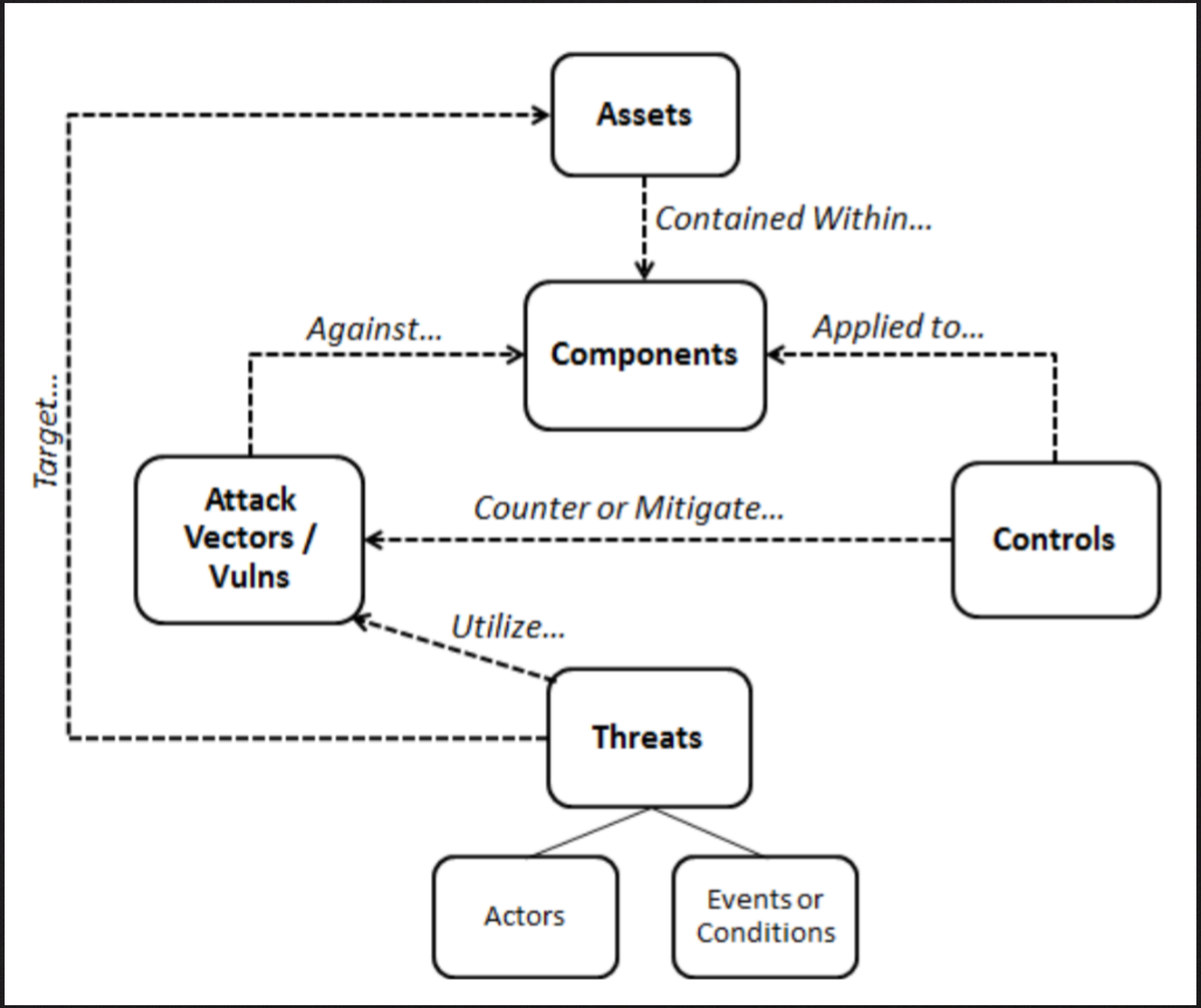
Photo Credit: Jonathunder (CC BY-SA 3.0)
https://en.wikipedia.org/wiki/Bank_vault#/media/File:WinonaSavingsBankVault.JPG

Photo Credit: Tom Varco (CC BY-SA 3.0)
https://en.wikipedia.org/wiki/Safe#/media/File:Safe.jpg

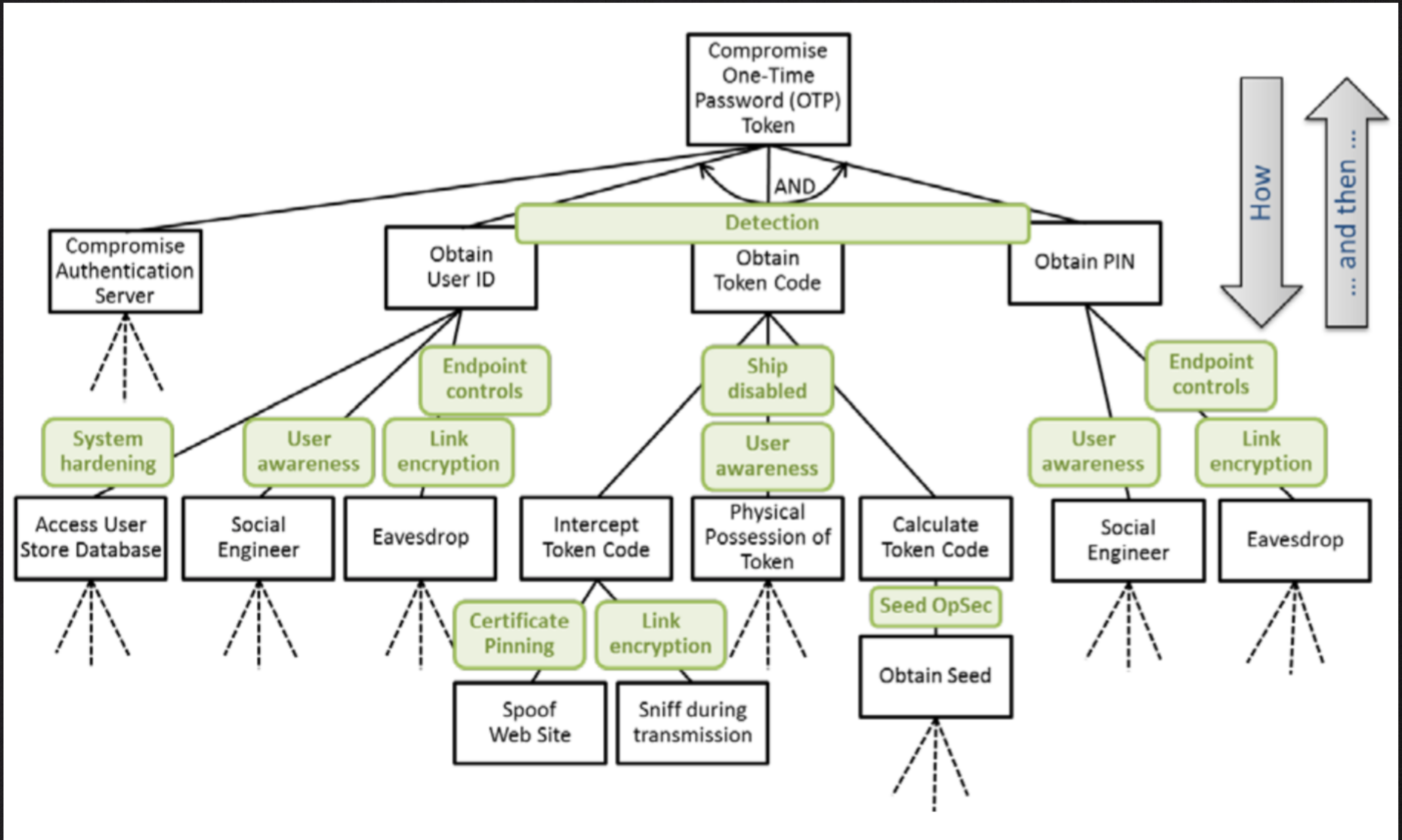
$$\text{risk} \propto \text{threat} \cdot \text{vulnerability} \cdot \text{consequence}$$

risk $\propto$ threat • vulnerability • consequence

asset *attack vectors* *controls*



http://lockheedmartin.com/content/dam/lockheed/data/isgs/documents/Threat-Driven%20Approach%20whitepaper.pdf



http://lockheedmartin.com/content/dam/lockheed/data/isgs/documents/Threat-Driven%20Approach%20whitepaper.pdf

NETFLIX

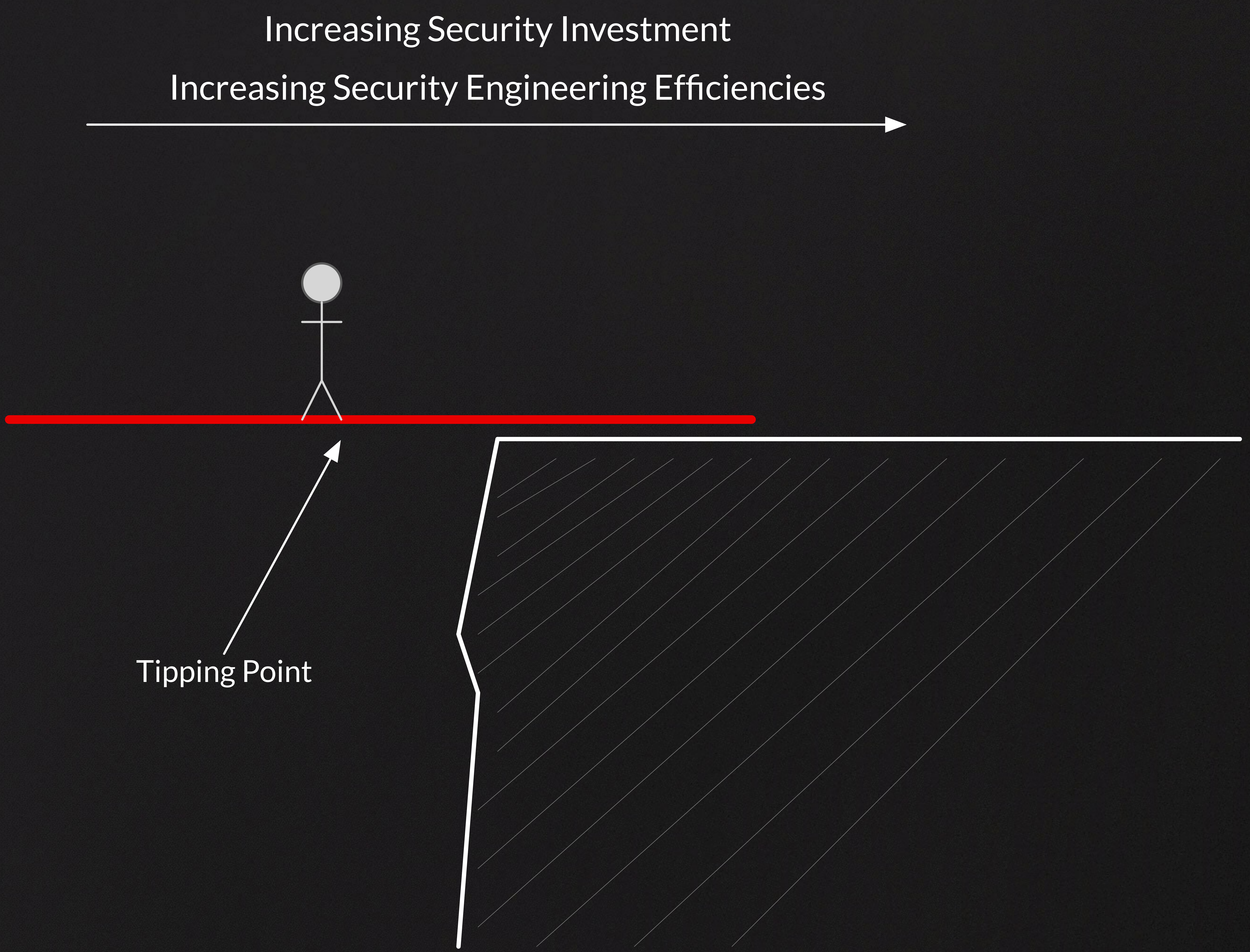
Cloud Attack Graphs

- Cloud account credentials
- Instance account credentials
- Your employees, supply chains, code
- Provider's employees, supply chains, code
- Corporate network
- Build pipeline

NETFLIX

Why are we losing?

… *and how can we improve?*


Increasing Security Investment
Increasing Security Engineering Efficiencies
Tipping Point

Simple Libraries

(e.g., python-cryptography)

```
from cryptography.fernet import Fernet

key = Fernet.generate_key()
f = Fernet(key)
ciphertext = f.encrypt(b”A message.”)
plaintext = f.decrypt(ciphertext)
```

Traditional Libraries

(e.g., openssl)

```
#include <openssl/conf.h>
#include <openssl/evp.h>
#include <openssl/err.h>
#include <string.h>

int main(int arc, char *argv[])
{
  /* Set up the key and iv. Do I need to say to not hard code these in a
   * real application? :-)
   */

  /* A 256 bit key */
  unsigned char *key = "01234567890123456789012345678901";

  /* A 128 bit IV */
  unsigned char *iv = "0123456789012345";

  /* Message to be encrypted */
  unsigned char *plaintext =
    "The quick brown fox jumps over the lazy dog";

  /* Buffer for ciphertext. Ensure the buffer is long enough for the
   * ciphertext which may be longer than the plaintext, dependant on the
   * algorithm and mode
   */
  unsigned char ciphertext[128];

  /* Buffer for the decrypted text */
  unsigned char decryptedtext[128];

  int decryptedtext_len, ciphertext_len;

  /* Initialise the library */
  ERR_load_crypto_strings();
  OpenSSL_add_all_algorithms();
  OPENSSL_config(NULL);

  /* Encrypt the plaintext */
  ciphertext_len = encrypt(plaintext, strlen(plaintext), key, iv,
    ciphertext);

  /* Do something useful with the ciphertext here */
  printf("Ciphertext is:\n");
  BIO_dump_fp(stdout, ciphertext, ciphertext_len);

  /* Decrypt the ciphertext */
  decryptedtext_len = decrypt(ciphertext, ciphertext_len, key, iv,
    decryptedtext);

  /* Add a NULL terminator. We are expecting printable text */
  decryptedtext[decryptedtext_len] = '\0';

  /* Show the decrypted text */
  printf("Decrypted text is:\n");
  printf("%s\n", decryptedtext);

  /* Clean up */
  EVP_cleanup();
  ERR_free_strings();

  return 0;
}

int encrypt(unsigned char *plaintext, int plaintext_len, unsigned char *key,
  unsigned char *iv, unsigned char *ciphertext)
{
  EVP_CIPHER_CTX *ctx;

  int len;

  int ciphertext_len;

  /* Create and initialise the context */
  if(!(ctx = EVP_CIPHER_CTX_new())) handleErrors();

  /* Initialise the encryption operation. IMPORTANT - ensure you use a key
   * and IV size appropriate for your cipher
   * In this example we are using 256 bit AES (i.e. a 256 bit key). The
   * IV size for *most* modes is the same as the block size. For AES this
   * is 128 bits */
  if(1 != EVP_EncryptInit_ex(ctx, EVP_aes_256_cbc(), NULL, key, iv))
    handleErrors();

  /* Provide the message to be encrypted, and obtain the encrypted output.
   * EVP_EncryptUpdate can be called multiple times if necessary
   */
  if(1 != EVP_EncryptUpdate(ctx, ciphertext, &len, plaintext, plaintext_len))
    handleErrors();
  ciphertext_len = len;

  /* Finalise the encryption. Further ciphertext bytes may be written at
   * this stage.
   */
  if(1 != EVP_EncryptFinal_ex(ctx, ciphertext + len, &len)) handleErrors();
  ciphertext_len += len;

  /* Clean up */
  EVP_CIPHER_CTX_free(ctx);

  return ciphertext_len;
}


int decrypt(unsigned char *ciphertext, int ciphertext_len, unsigned char *key,
  unsigned char *iv, unsigned char *plaintext)
{
  EVP_CIPHER_CTX *ctx;

  int len;

  int plaintext_len;

  /* Create and initialise the context */
  if(!(ctx = EVP_CIPHER_CTX_new())) handleErrors();

  /* Initialise the decryption operation. IMPORTANT - ensure you use a key
   * and IV size appropriate for your cipher
   * In this example we are using 256 bit AES (i.e. a 256 bit key). The
   * IV size for *most* modes is the same as the block size. For AES this
   * is 128 bits */
  if(1 != EVP_DecryptInit_ex(ctx, EVP_aes_256_cbc(), NULL, key, iv))
    handleErrors();

  /* Provide the message to be decrypted, and obtain the plaintext output.
   * EVP_DecryptUpdate can be called multiple times if necessary
   */
  if(1 != EVP_DecryptUpdate(ctx, plaintext, &len, ciphertext, ciphertext_len))
    handleErrors();
  plaintext_len = len;

  /* Finalise the decryption. Further plaintext bytes may be written at
   * this stage.
   */
  if(1 != EVP_DecryptFinal_ex(ctx, plaintext + len, &len)) handleErrors();
  plaintext_len += len;

  /* Clean up */
  EVP_CIPHER_CTX_free(ctx);

  return plaintext_len;
}
[edit]
```

NETFLIX

Sidebar: Key Management @Netflix

Simple Framework for Key Handling

	Throughput	Protection	It's Exposed!	It lives…
Low Sensitivity	High	Low	No biggie	In lots of VMs
Medium Sensitivity	Medium	Medium	It'll be a long week.	In very few VMs
High Sensitivity	Low	High	No. Just. No.	In Special Hardware

NETFLIX

Use Case of a Key Implies Handling Requirements

TLS Session Key - Fast, Handled in Dynamic Environment

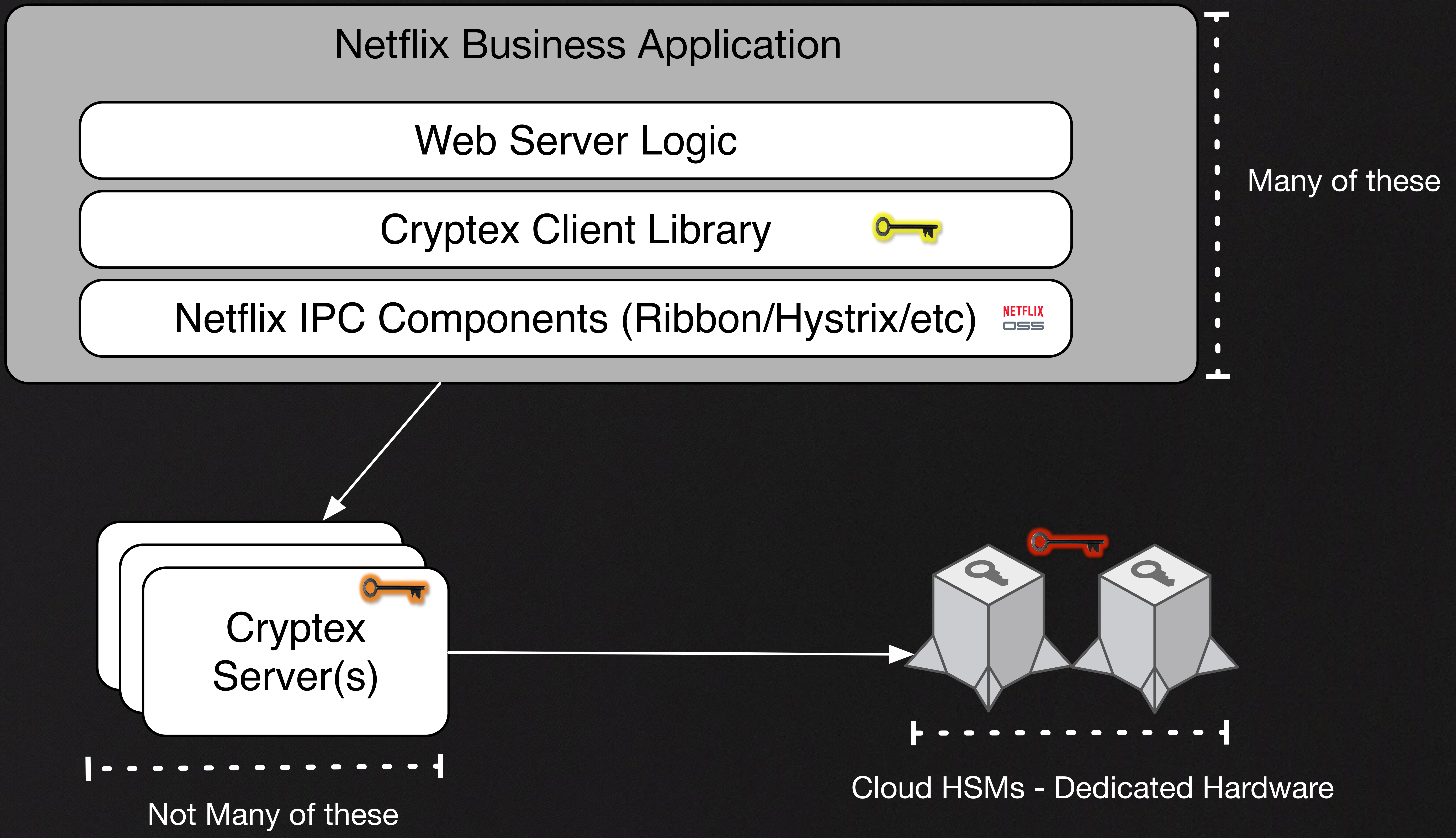
- *But easy to have a reasonable policy if we lose it*

Certificate Authority Private Key - Maybe not used so much

- *Probably way more important that you just don’t lose it*

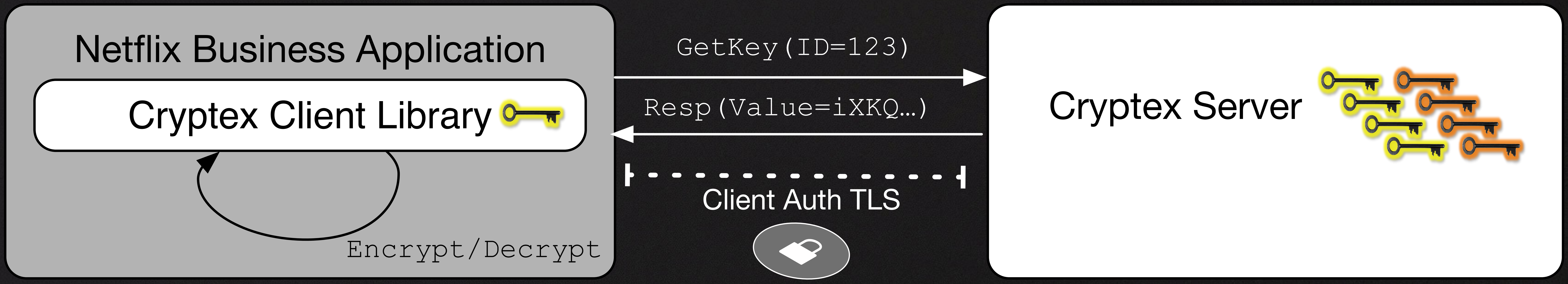
NETFLIX

Cryptex - Our Framework for Key Handling



NETFLIX

"Low" Key Handling

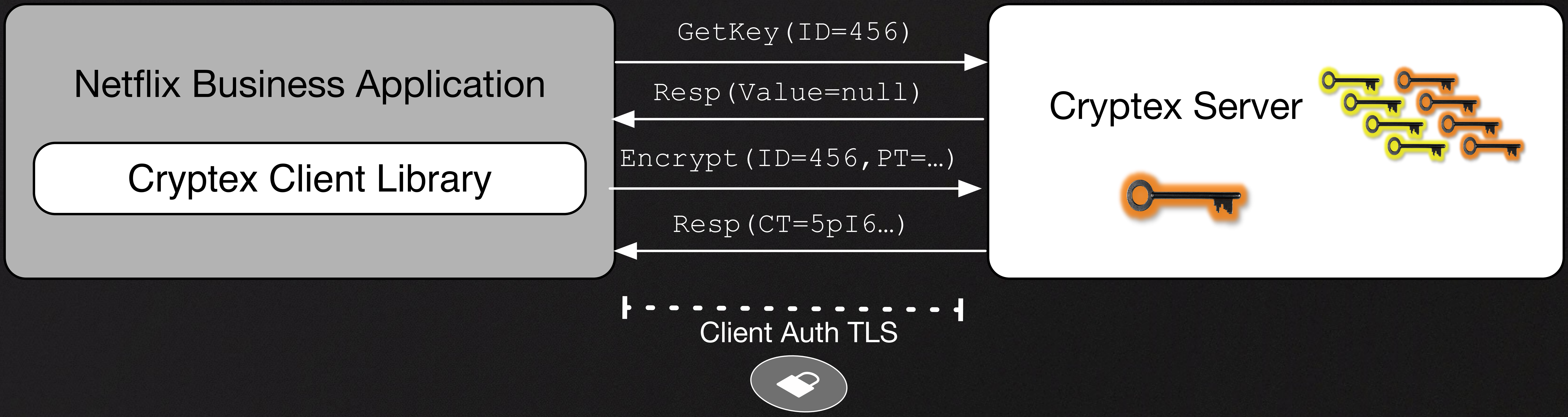


Key Exported Out to Every Client

- *Extremely High Throughput*
- *Client Library Attempts to be Mindful of Key Handling*

NETFLIX

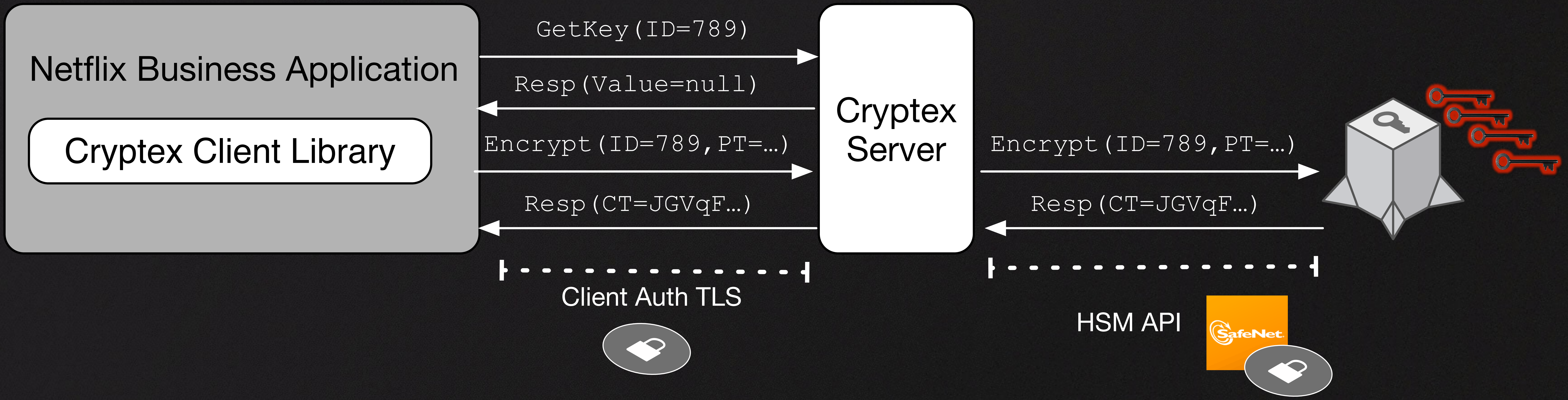
"Medium" Key Handling



Every Operation is a REST Call

- *Luckily we don't have many bulk encrypt use cases for these*
- *Cryptex servers not publicly facing; ostensibly harder to get onto*

“High” Key Handling

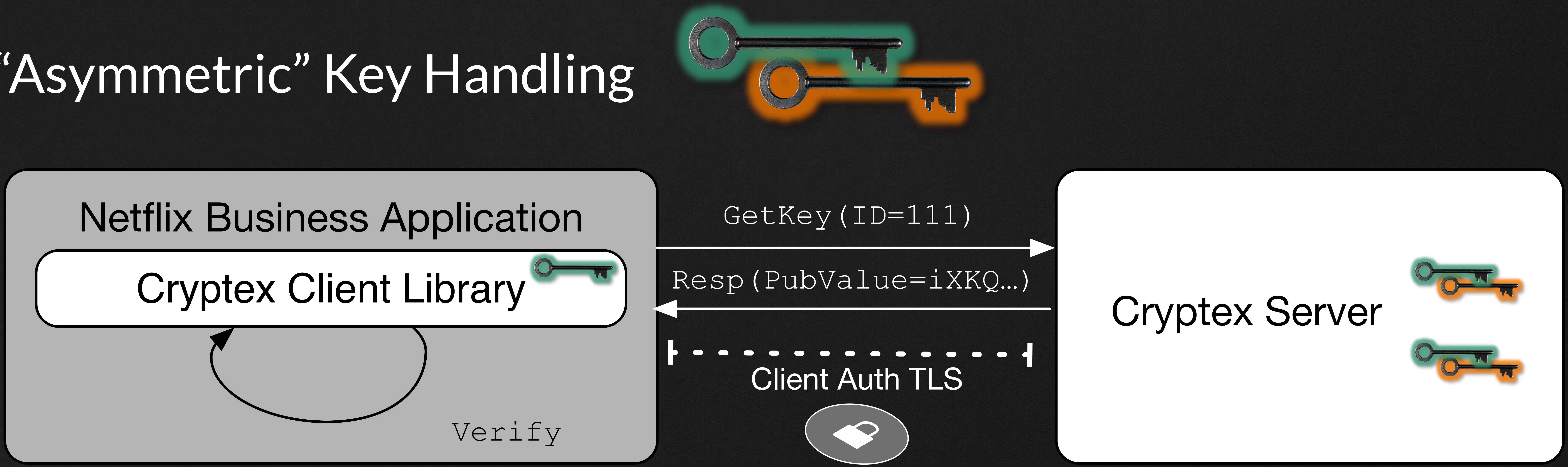


Every Operation is a call to specialized hardware

- *HSM API challenging relative to REST calls (only Cryptex does it)*
- *Very constrained throughput; VM side channel attacks negated*

NETFLIX

"Asymmetric" Key Handling



We support the basics: AES, HMAC-SHA, RSA

- *Optimize RSA verify/encrypt by pushing public key to edge*
- *At scale computational intensity of RSA quite apparent*


IN GOD WE TRUST
LIBERTY
2013
S
VDB

NETFLIX

Managing Security at Scale

what you deploy

deployment pipeline

runtime consistency

NETFLIX

Attackers Are Creative

- 802.11a/b/g/n/ac
- Bluetooth
- Gigabit Ethernet
- Out-of-band SSH access over 4G/GSM cell networks

https://www.pwnieexpress.com/product/pwn-plug-r3penetration-testing-device/

A team participating in a CTF competition at DEFCON 17

Photo Credit: Nate Grigg (CC BY 2.0)
http://www.flickr.com/photos/nateone/3792232737/

NETFLIX

Questions?

bryanp@netflix.com
http://bryanpayne.org

[PS… I’m hiring!]