ZER0ing Trust

Do Zero **Trust** Approaches Deliver Real Security?

David "dwizzzle" Weston
Offensive/Device Security, Microsoft

Photo Source

WHAT IS ZER0 TRUST?

Defining zero trust

Security hipsters love the ZTN

…Why?

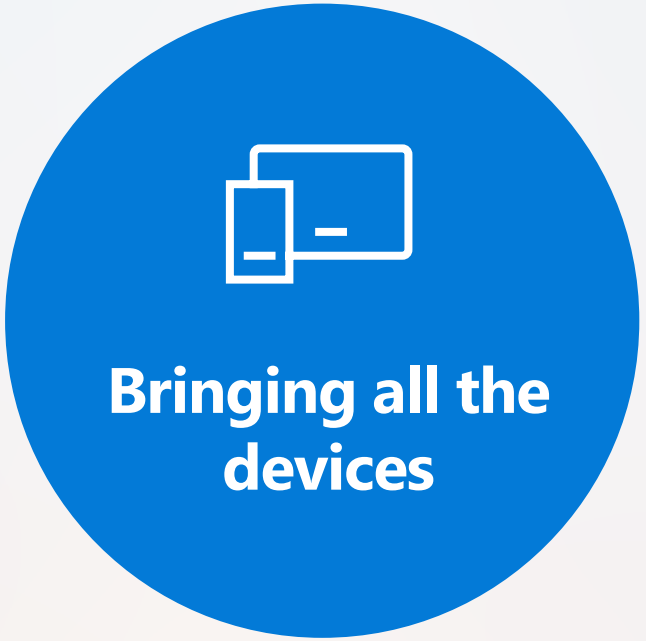
ZTN is so 🔥 rn

Photo Source


REDTEAM/
ATTACKERS
Worst
device on
the network
That user
with no
MFA

source

Why the interest in Zero Trust?

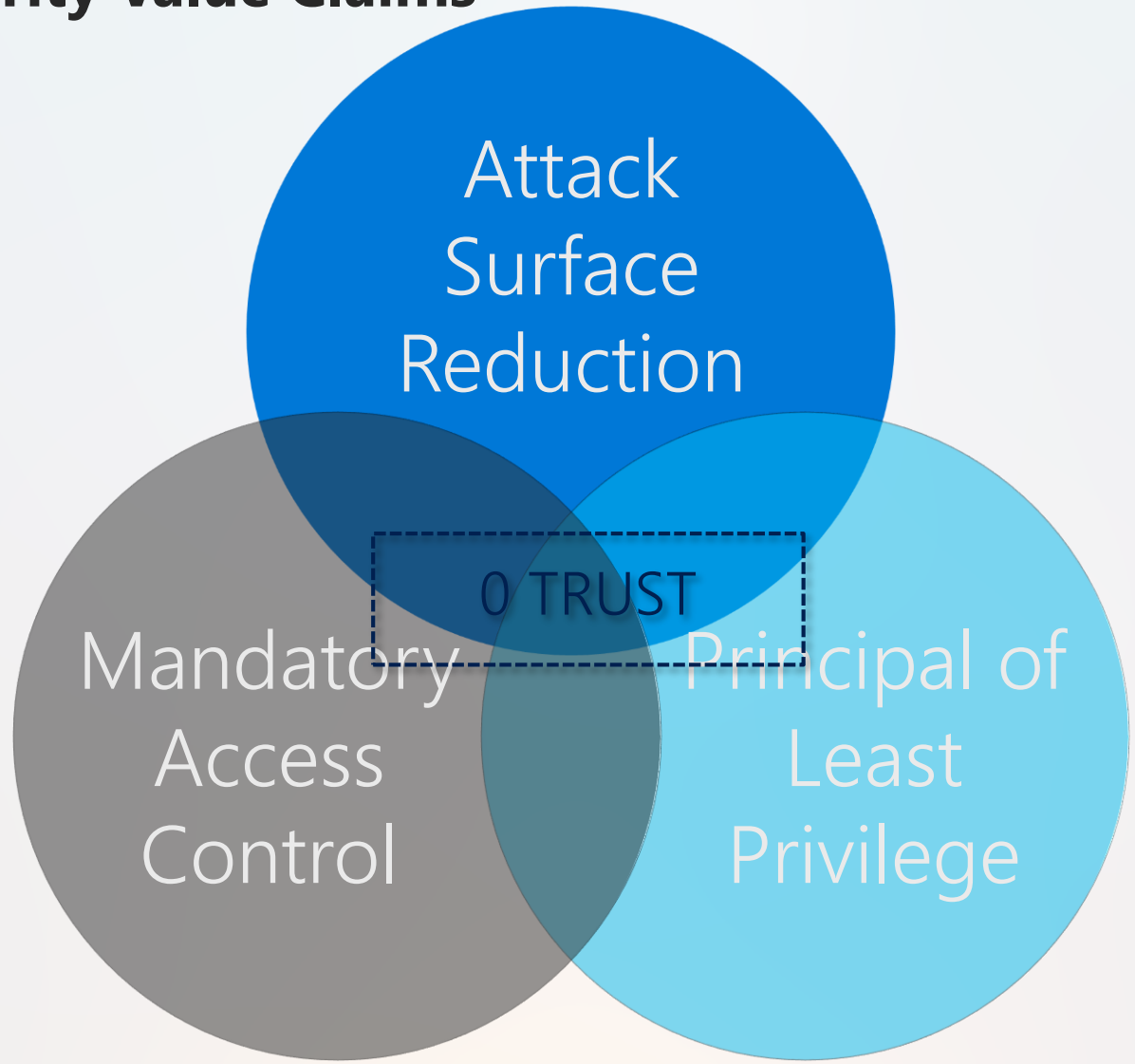


Defining Zero Trust

ZTN security model defined by John Kindervag of Forrester (2010)

Location is nothing	Network MAC	Trust is Device + Identity
Network Perimeter is no longer a security boundary	All traffic flows are **rejected by default**	Access is determined by what is known about the device and the user
Network location provides no inherent privilege or access	Traffic is routed **if and only if it meets security policy** (aka "trusted")	Access policies are dynamic and adjusted based on the trust of the user + device
Every network is *untrusted*	"Segmentation Gateway" is the enforcement engine	
	All traffic is encrypted	

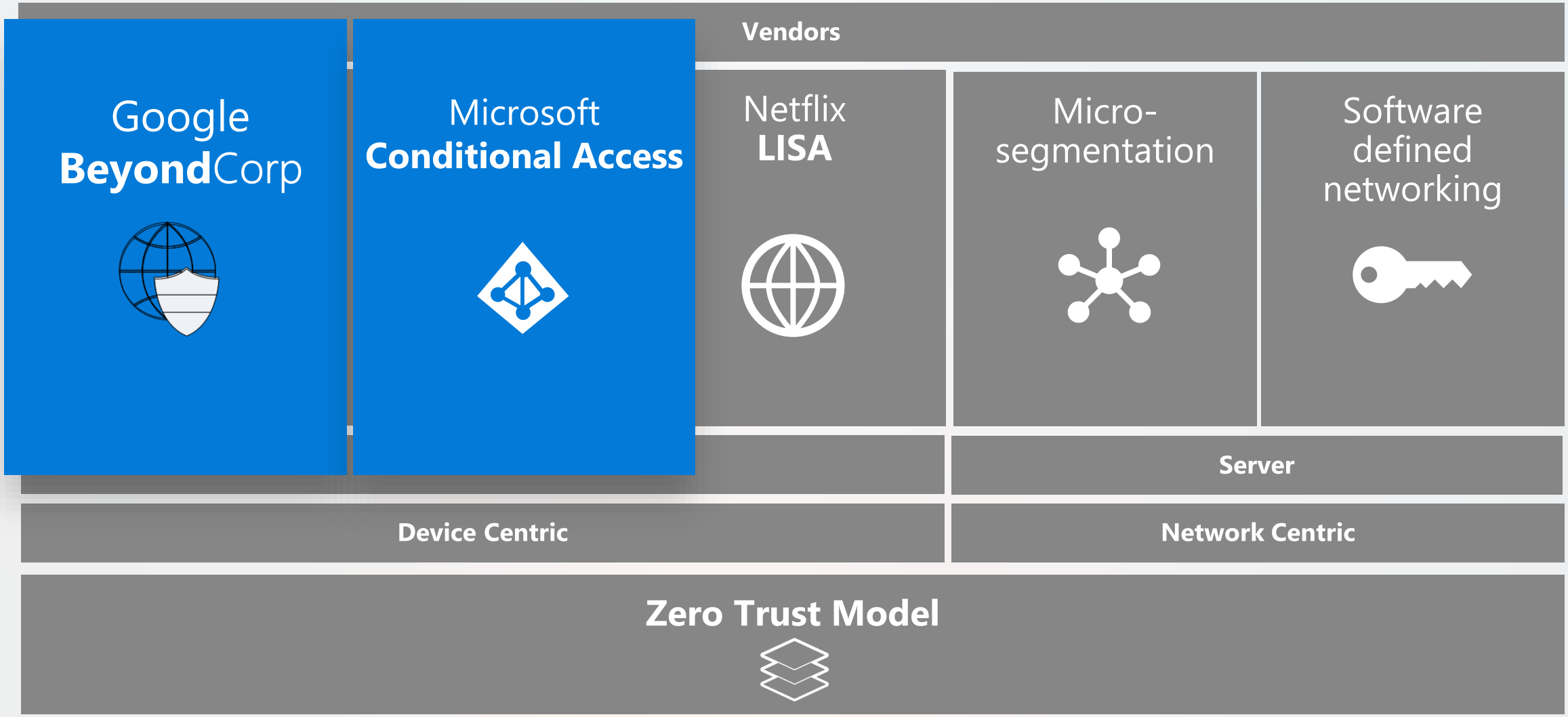
Zero Trust Security Value Claims

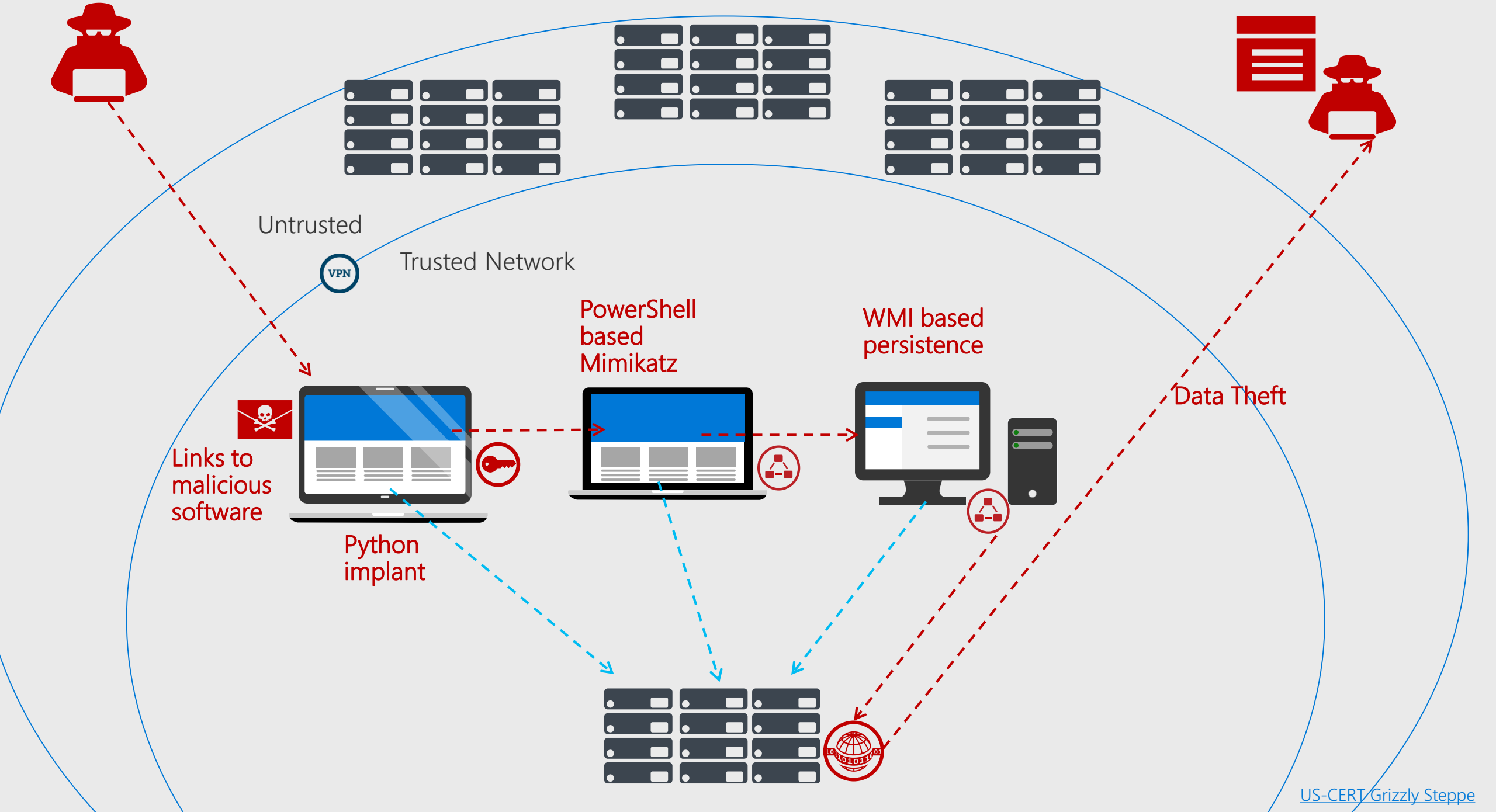


ZER0 Trust is Squishy.

Photo Source

Security Pillar





US-CERT Grizzly Steppe

Google: BeyondCorp

Generalized ZTN security model proposed by Google in response to “Aurora” incident

User Identity Management

Securely Identifying the user

Single Sign-on System

User and Group Database

Examples:

AAD

G-Suite

Device Trust and Identity

Determining trust in a device

Device Identity

Device Inventory Database

Access control engine

Combines knowledge and about the device and user identity to make an access decision

Policy enforcement engine for access

Combines trust information from device data base and SSO/user directory

Provides dynamic access based on changing trust signals

Access Proxy

Enforce access control policies

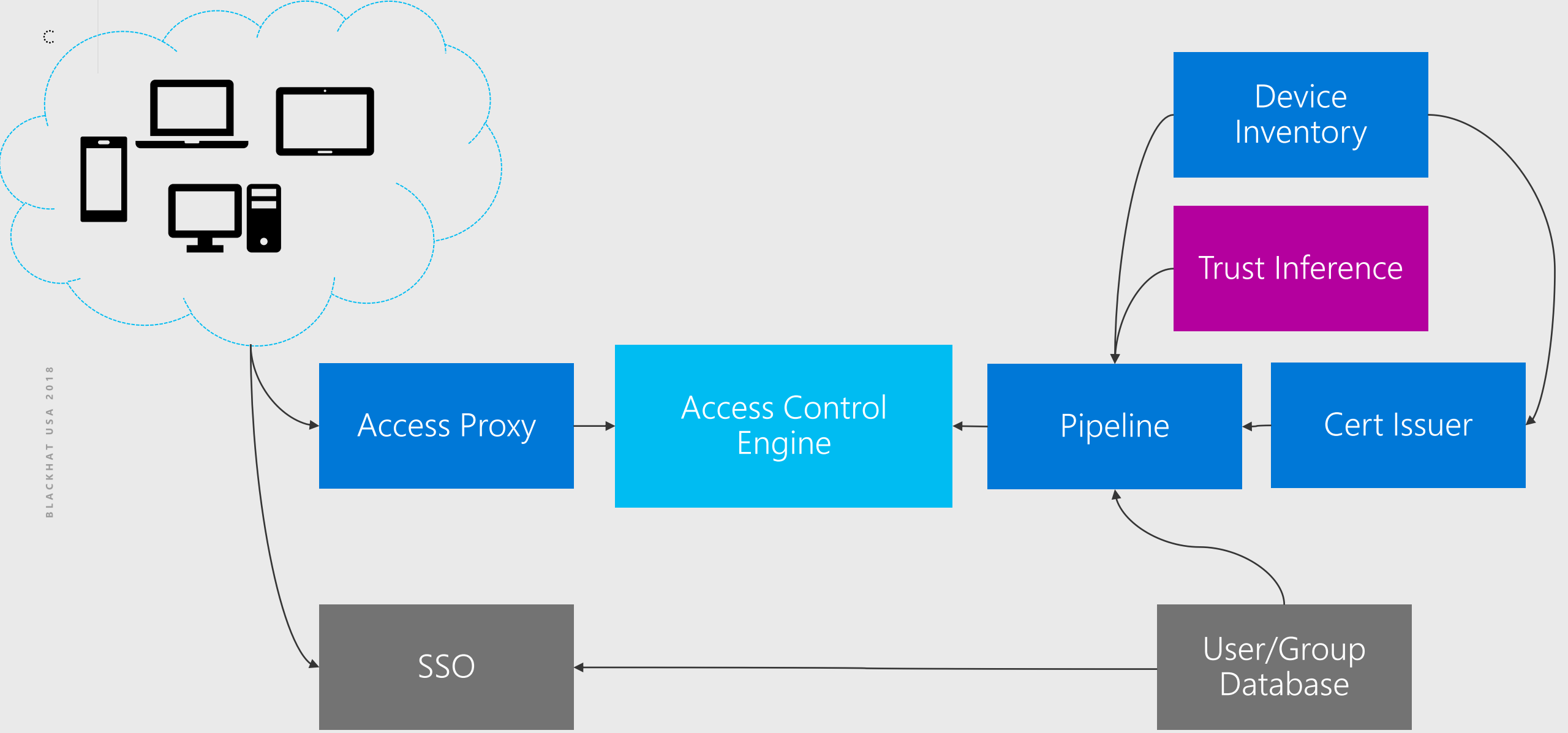
Gates network access to protected resources

Enforcement layer for access control policies

Drops or blocks unauthorized traffic

Primarily HTTP/S

Can require tunneling for other protocols



ZTN Threat Model

Lots of talk about ZTN, little discussion of a formal threat model

To evaluate common ZTN implementations and products need a standardized threat model

MITRE Attack Framework provides a simple publicly documented method for standardize evaluation

Goals

Pre-Exploitation: How well do common ZTN approaches prevent initial exploitation of network device?

Post-Exploitation: Assuming an exploited client, how well do it prevent lateral movement?

Evaluation Approach:

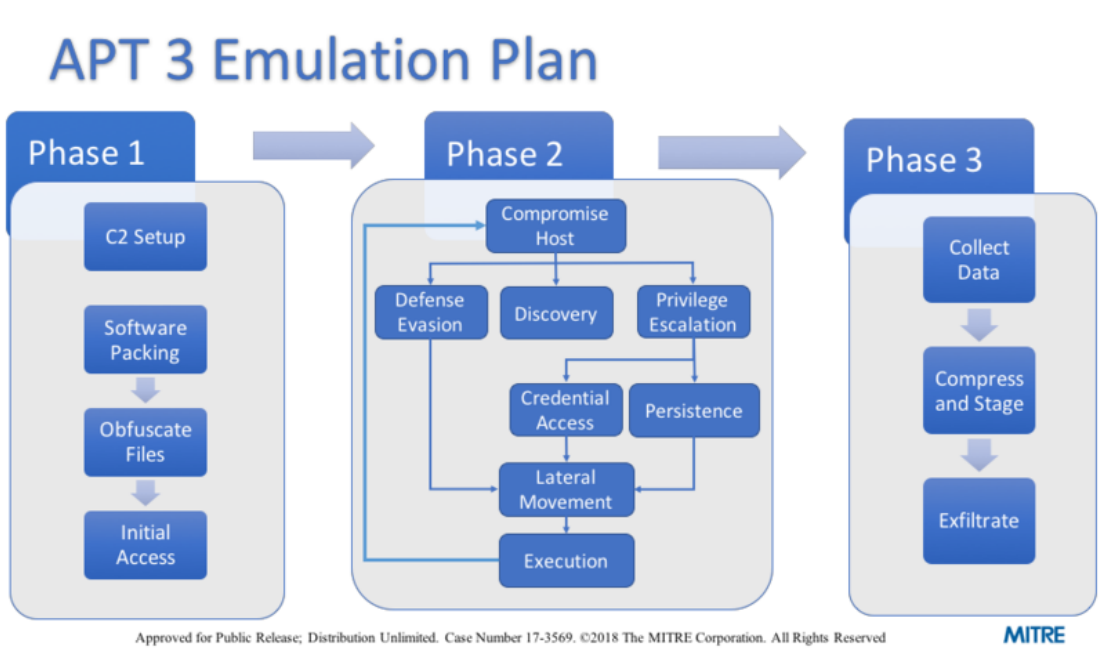
MITRE APT 3 Adversary Emulation

Mobile Emulation

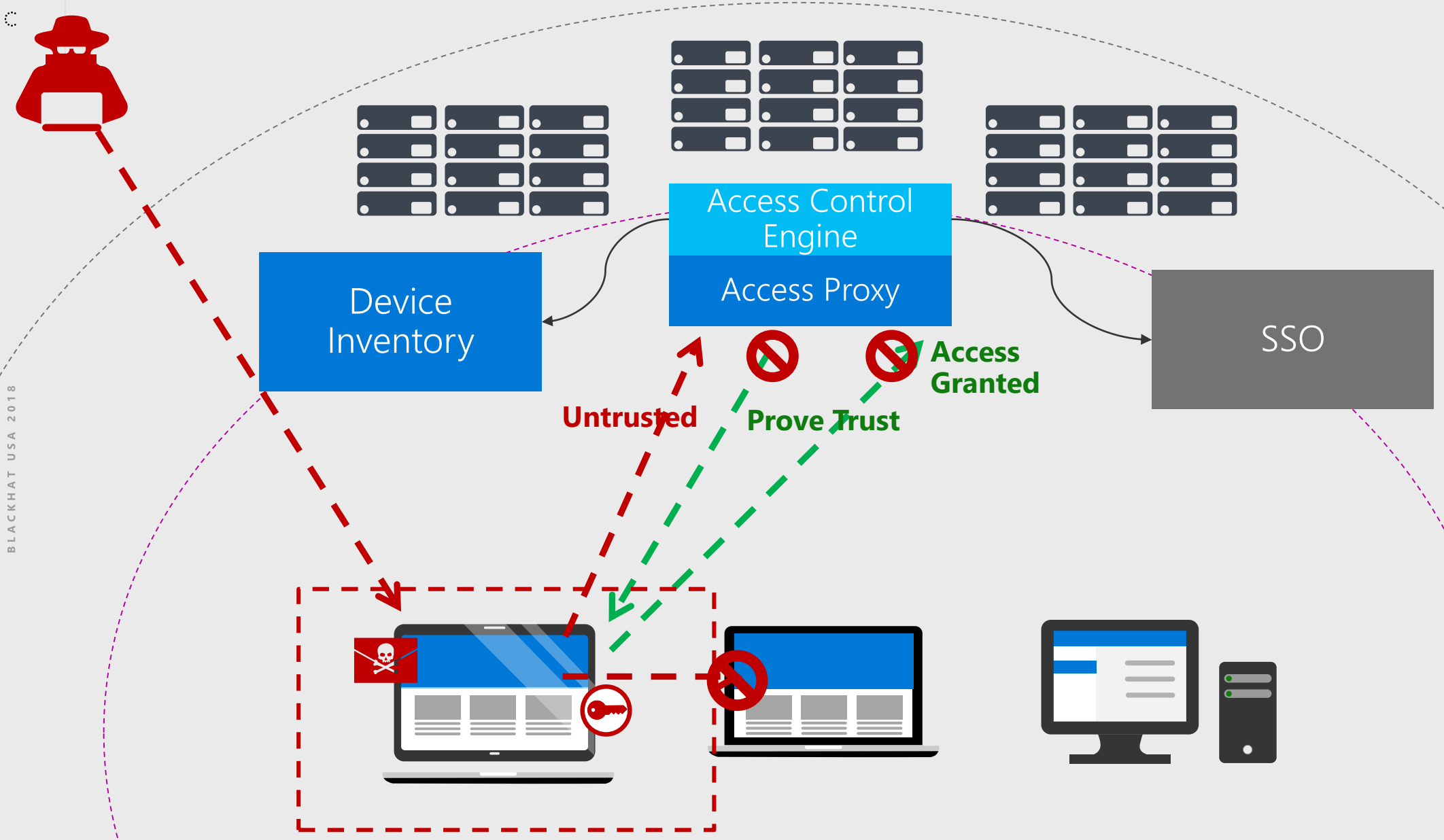
TRIDENT (iOS)

X-AGENT (Android)

Hacking Team (iOS, Android)



Source



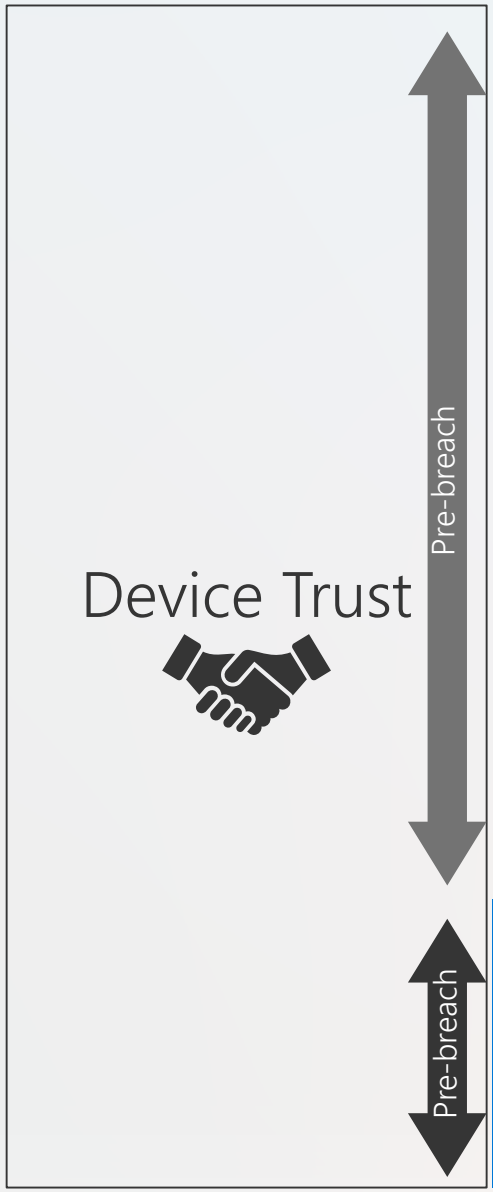
Cool Story?

Source

Establishing Trust in a Device

DEVICE TRUST

Defining Device Trust



Identity

Goal: Strongly identify device ownership

Security Value: Non-enterprise devices have reduced access

Patch State

Goal: Prevent devices which could be compromised from accessing resources

Security Value: Reduce likelihood of data or identity compromise

Security Profile

Goal: Provide access proportional to risk of compromise

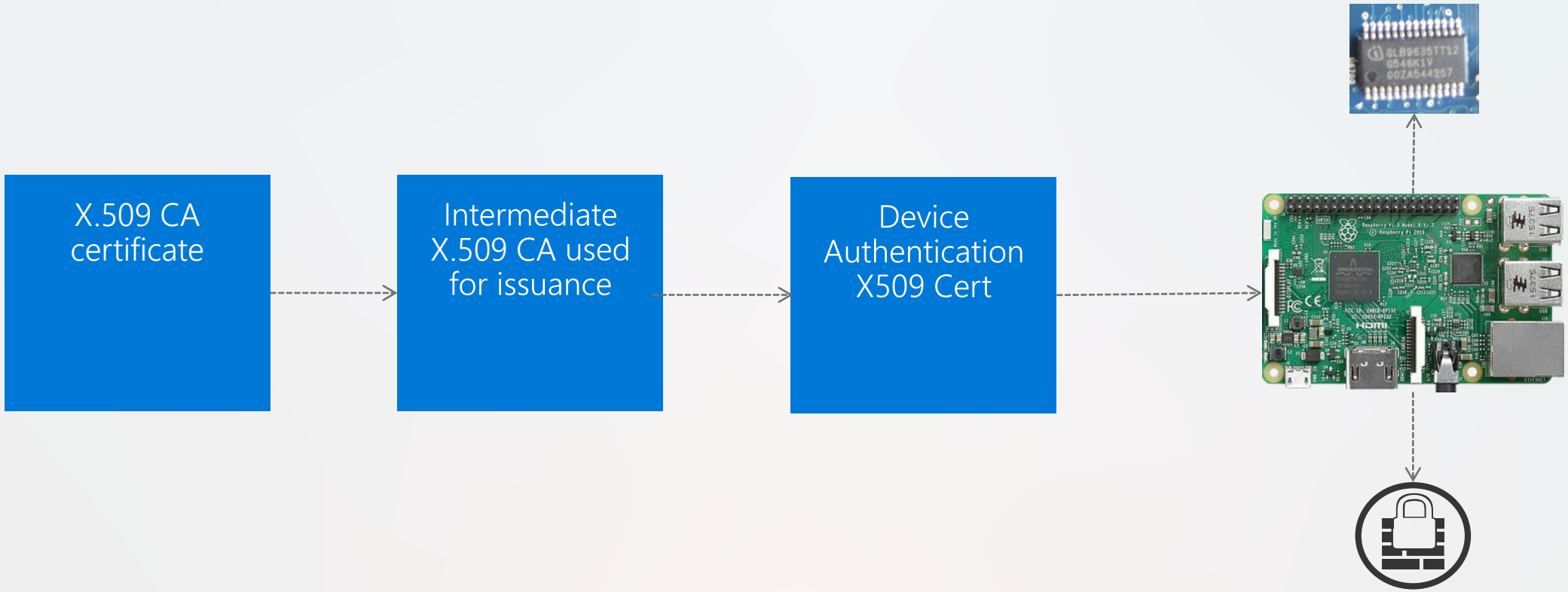
Security Value: Device with reduced security have reduced access

Runtime Integrity

Goal: Remove access if a security invariant is violated

Security Value: Compromised device lose access before lateral movement can occur

Device Authentication



Exporting the unexportable

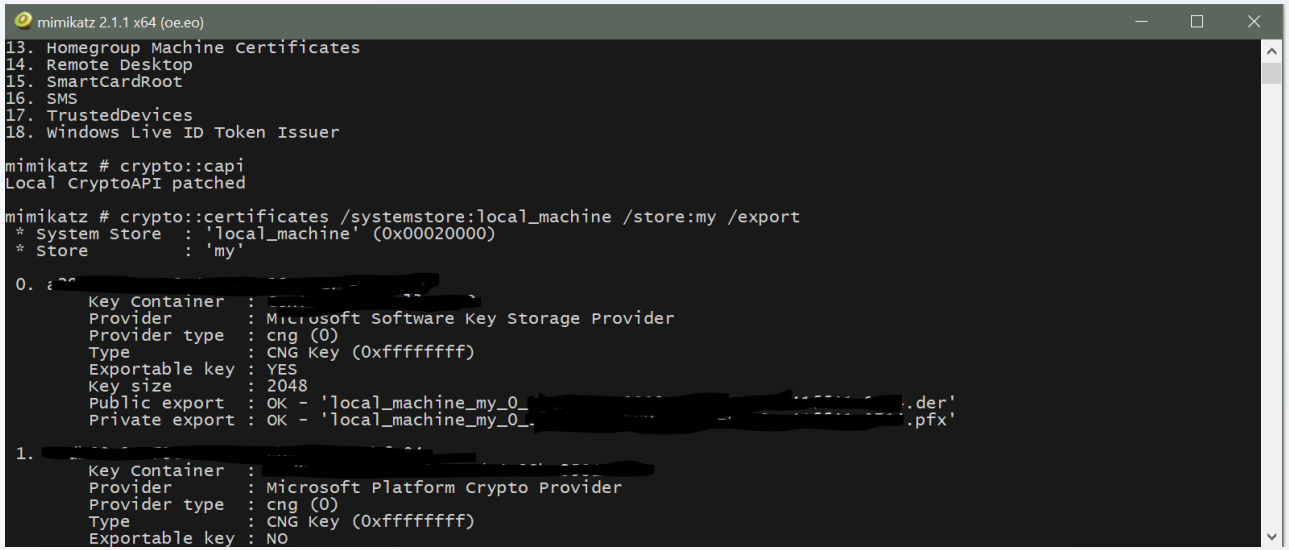
Zero trust trades trust in the network for trust in the device

Most zero-trust vendors reviewed store private key for x509 cert in software

Private key protected by password or on marked “non-exportable”

Even Private keys stored in TPM are often non-attestable

Stealing device cert enables untrusted device access



Storage type	Malicious app attacker	Root attacker	Intercepting root attacker
Bouncy Castle with stored password	✓	X	X
Bouncy Castle with user-provided password	✓	✓ *	X
AndroidKeyStore using the TEE on Qualcomm devices	✓	✓	✓
AndroidKeyStore using the TEE on TI devices	✓	✓	✓
AndroidKeyStore using software-fallback without a PIN to unlock the device	✓	X	X
AndroidKeyStore using software-fallback with a PIN to unlock the device	✓	✓	✓

Analysis of Secure Key Storage Solutions on Android

Secure Method for Device Authentication

PC: TPM factory EK can be used for non-exportable device ID

Android: Keymaster 2 provides hardware backed key store with key attestation

iOS/macOS: Create Device keys within Secure Enclave

1) Fuse EK Seed
2) Generate EK Key Pairs (EK_PRIV, EK_PUB, AIK key pairs)
3) Send EK_PUB to signing service
4) Sign the EK_PUB, issue an EK_CERT
5) Store the EK_CERT on the device

6) Ship the device

7) User purchases and turns it on

7) Device sends the EK_CERT and the EK_PUB to AIK provision service
8) AIK provision service issues a challenge
   - Verifies the EK_CERT
   - Issues a Challenge
   - Generates a random value
   - Encrypts with EK_PUB
   - Sends the challenge
9) Device decrypts the challenge with EK_PRIV
10) Device forwards plain text challenge data and hash of AIK_PUB to provision service
11) AIK validates and provisions device

Assessing Patch State

Most zero trust device agent's use incomplete methods of analysis

Most coverage limited to browser components or granular OS versions

More robust implementations link to device or configuration management (MDM, Windows Defender ATP Secure Score)

Examples:

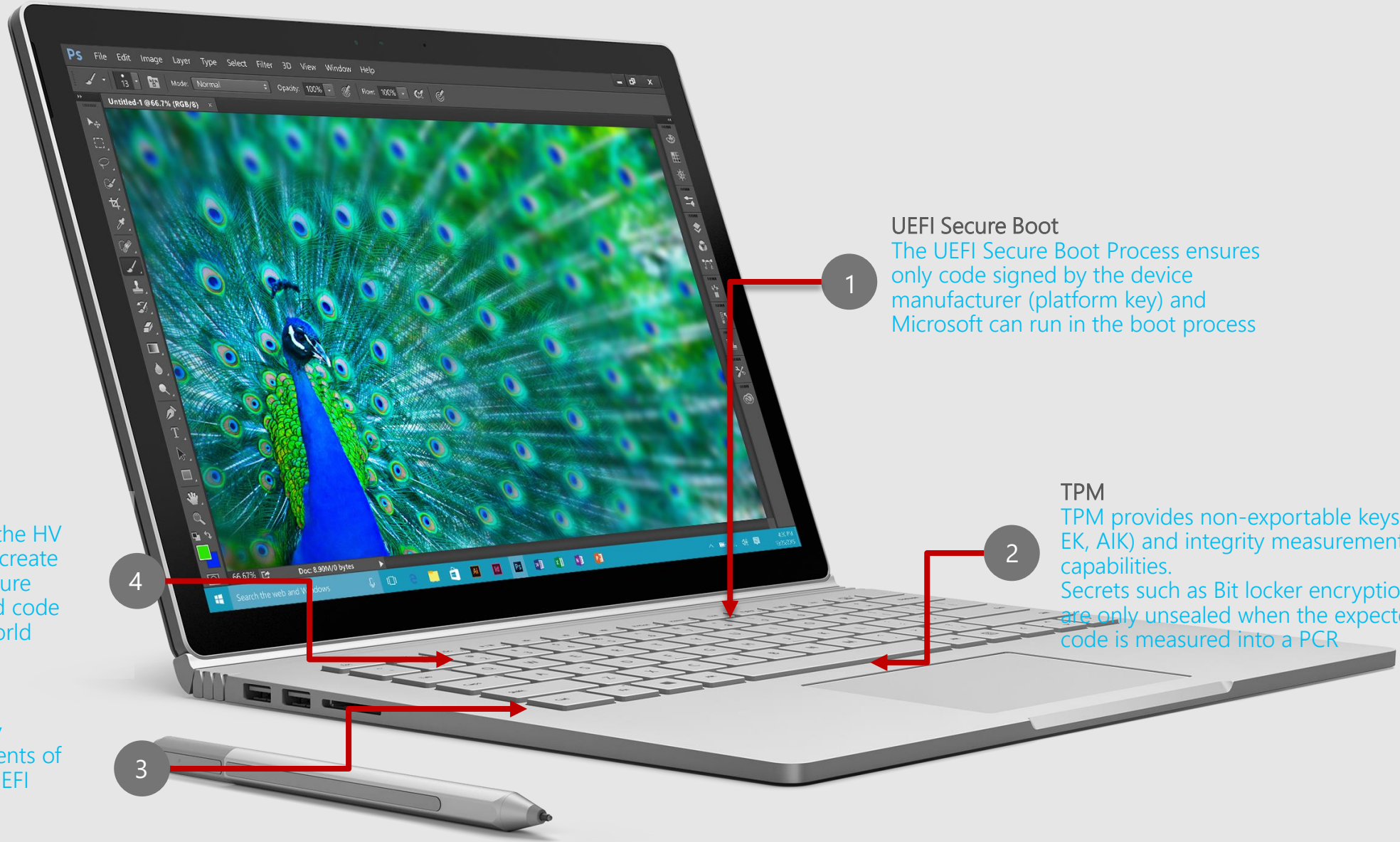
JavaScript from browser to assess patch state

Sandboxed App to access OS state (iOS, Android)

Calls into Windows Security Center API (NAC)

```
typedef enum _WSC_SECURITY_PROVIDER {
WSC_SECURITY_PROVIDER_FIREWALL ,
WSC_SECURITY_PROVIDER_AUTOUPDATE_SETTINGS ,
WSC_SECURITY_PROVIDER_ANTIVIRUS ,
WSC_SECURITY_PROVIDER_ANTISPYWARE ,
WSC_SECURITY_PROVIDER_INTERNET_SETTINGS ,
WSC_SECURITY_PROVIDER_USER_ACCOUNT_CONTROL ,
WSC_SECURITY_PROVIDER_SERVICE ,
WSC_SECURITY_PROVIDER_NONE ,
WSC_SECURITY_PROVIDER_ALL
} WSC_SECURITY_PROVIDER, *PWSC_SECURITY_PROVIDER;
```

Device Trust Properties



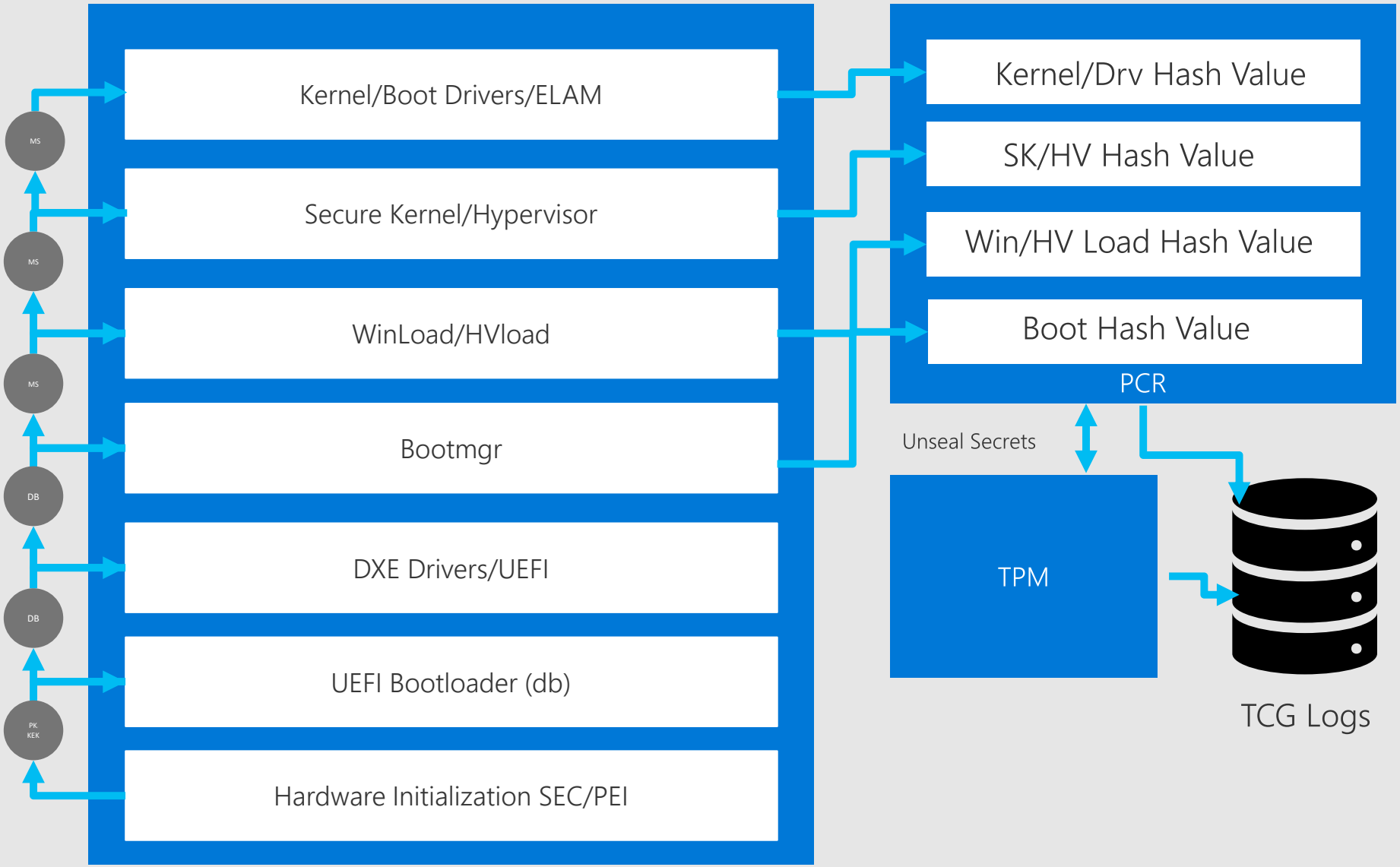
UEFI Secure Boot
The UEFI Secure Boot Process ensures only code signed by the device manufacturer (platform key) and Microsoft can run in the boot process

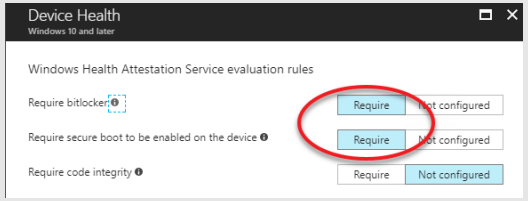
TPM
TPM provides non-exportable keys (e.g. EK, AIK) and integrity measurement capabilities.
Secrets such as Bit locker encryption key are only unsealed when the expected code is measured into a PCR

Intel VT-x and AMD-v
Virtualization extensions allow the HV launched by the bootloader to create the page table segmented "Secure kernel" which is used to isolated code and secrets from the normal world (VTL0)

Intel TXT and AMD SKINIT
DRTM augments UEFI secure by decoupling integrity measurements of OS and HV components from UEFI firmware

Windows Boot Attestation





Device Health Attestation

See here for additional details

Challenges with Attesting to Device Health

Secure boot and static root of trust (SRTM)

Strong dependency on the security UEFI firmware

Widespread issues with OEM firmware can be used to undermine TCB

High-integrity Attestation is at boot time

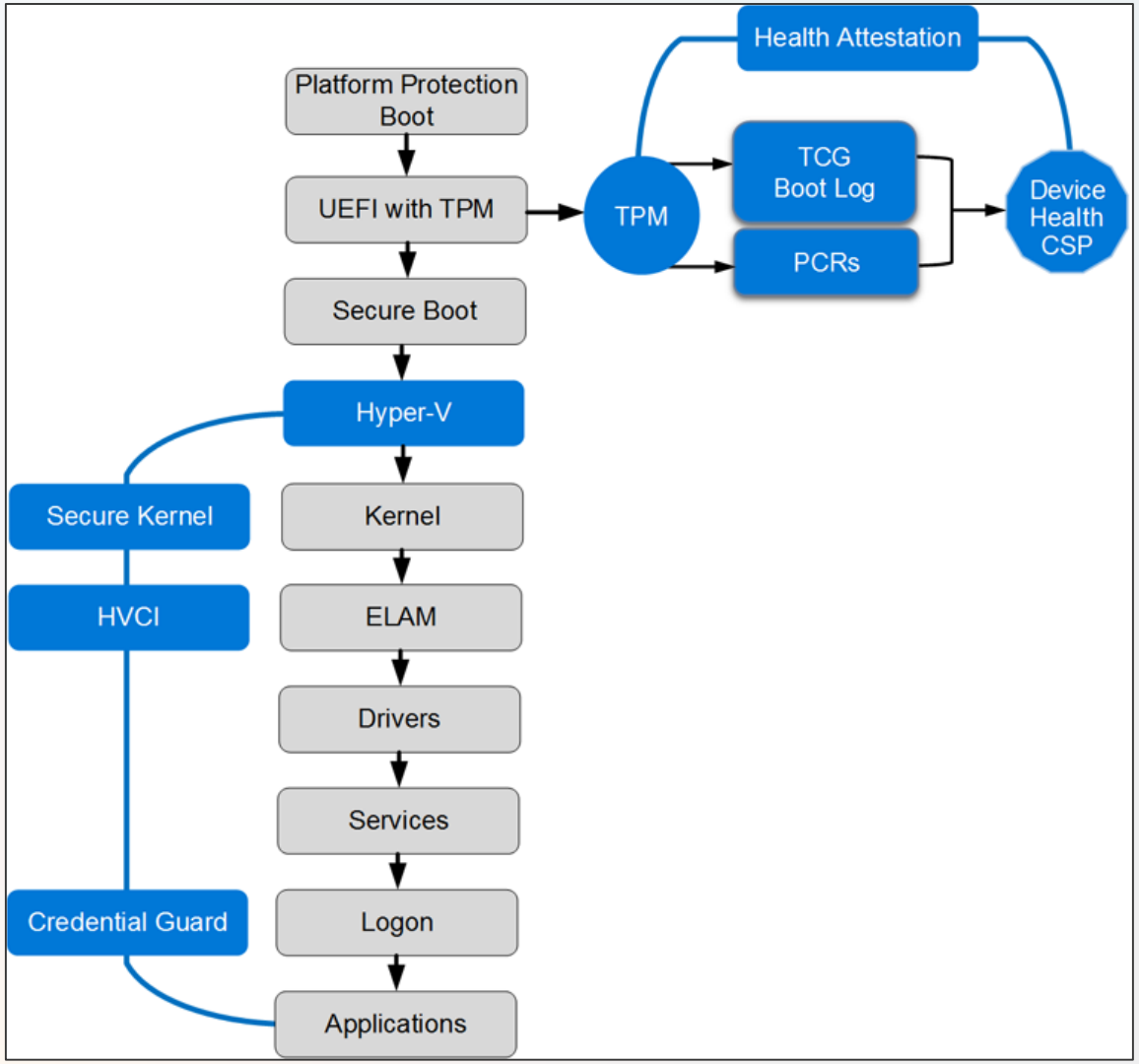
DHA only measures and attest to the integrity of system bring up

Simple runtime attacks do not impact health state

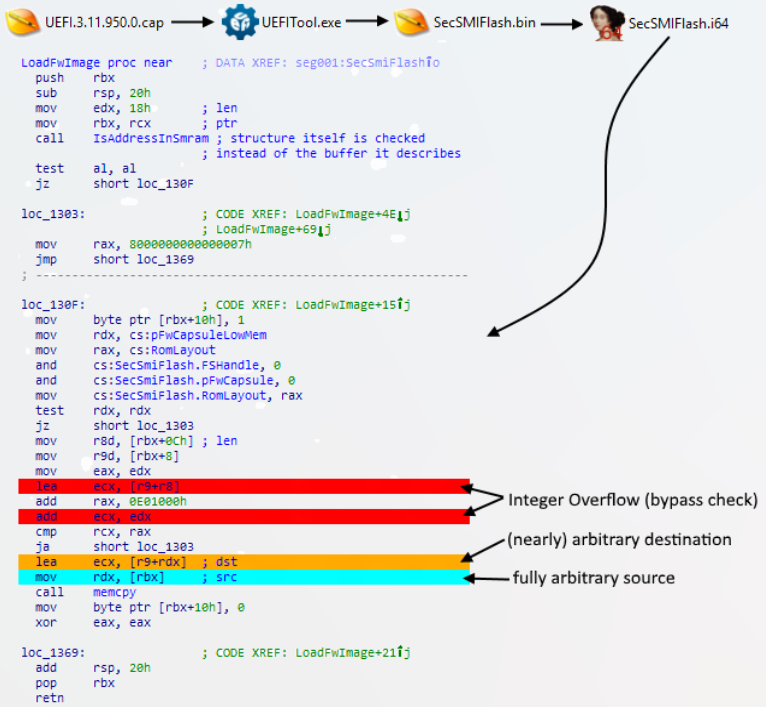
Limited runtime solutions

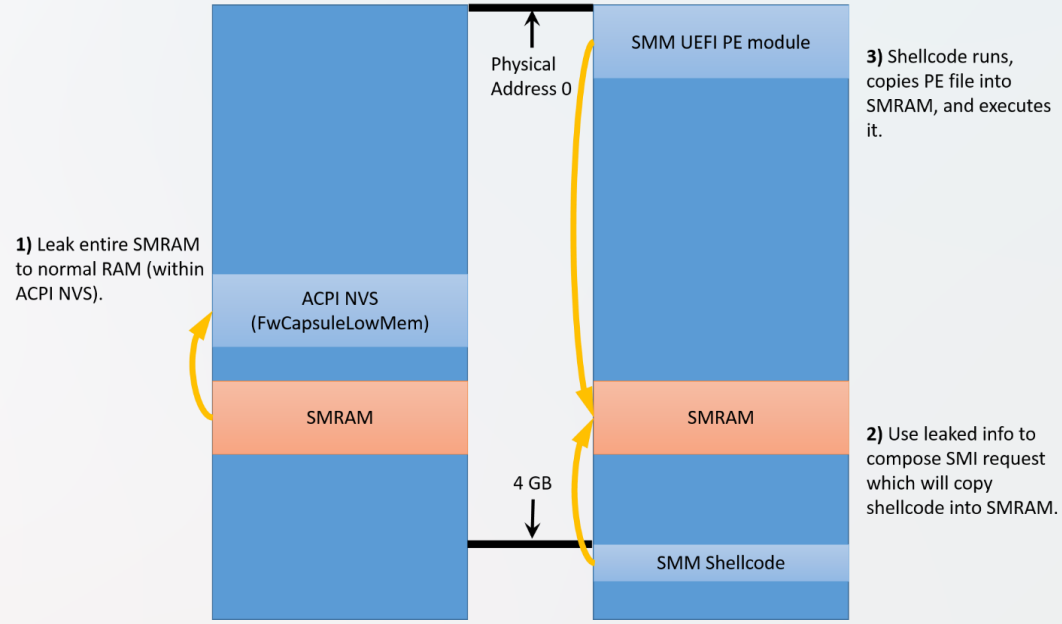
EDR and AV can identify runtime attacks limited integrity (software)

PatchGuard/HyperGuard great but not easily extensible



SMM Attacks





External researchers and OSR REDTEAM highlighted SMM risks for DRTM and VBS

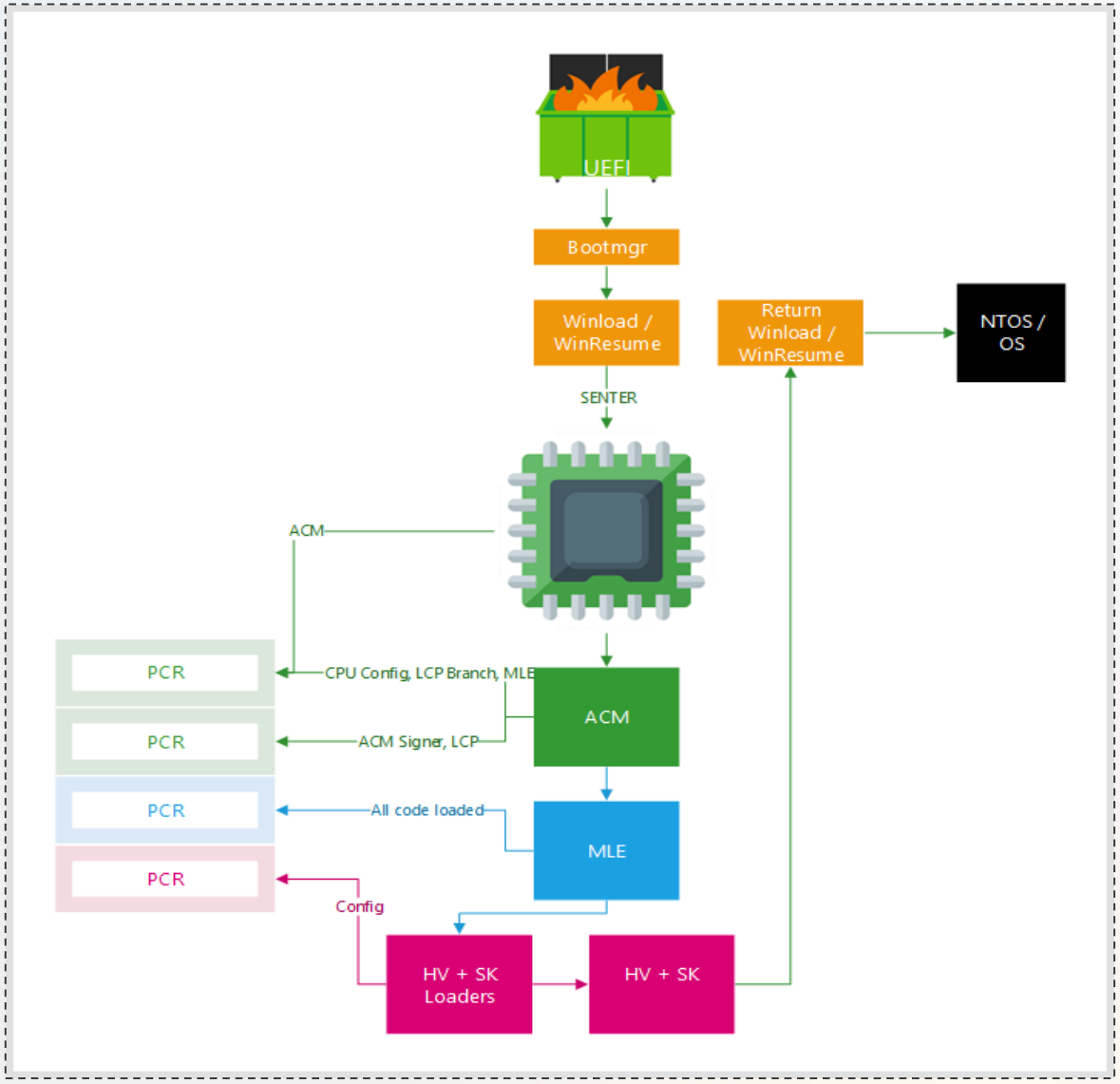
Arbitrary code execution in SMRAM can be used to defeat Hypervisor

Malicious code running in SMM is difficult to detect

Threatens hardware rooted ability to attest to device

SMM vulnerabilities used in OSR REDTEAM reported to Lenovo

DRTM with TXT Boot flow in Windows



Core isolation

Security features available on your device that use virtualization-based security.

This setting is managed by your administrator.

Memory integrity

Prevents attacks from inserting malicious code into high-security processes.

On

Learn more

Firmware protection

Windows Defender System Guard is protecting your device from compromised firmware.

Learn more

Alex Tereshkin
@AlexTereshkin

Follow

Replying to @dwizzzleMSFT @XenoKovah

Like forcing OEMs to implement STM? That would be awesome actually :)

9:41 AM - 14 Jun 2018

Addressing SMM attacks against DRTM

Intel Runtime BIOS resilience feature provides the following security properties for SMM:

SMM entry point locked down

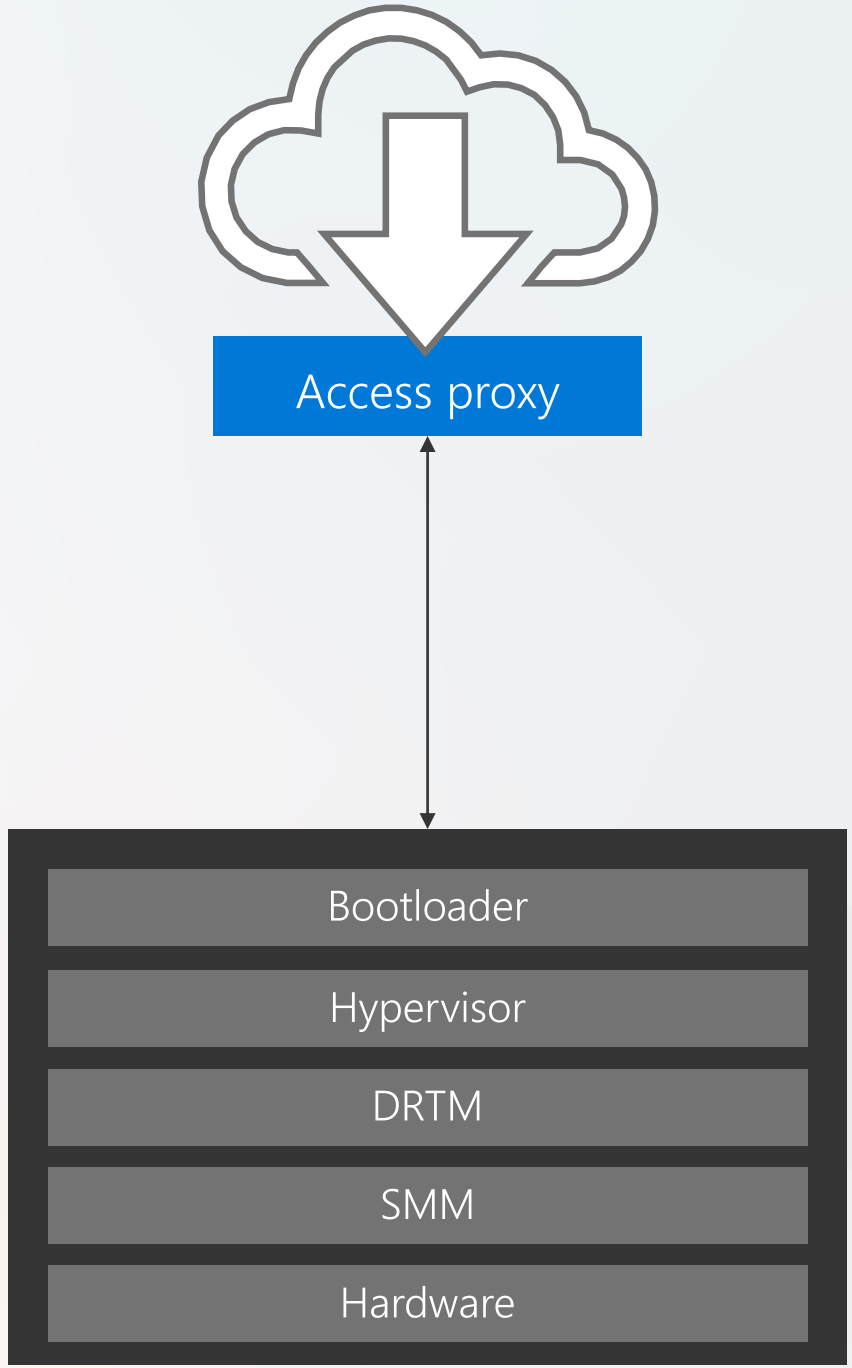
All code within SMM locked down

Memory map and page properties locked down

OS memory not directly accessible from SMM at all

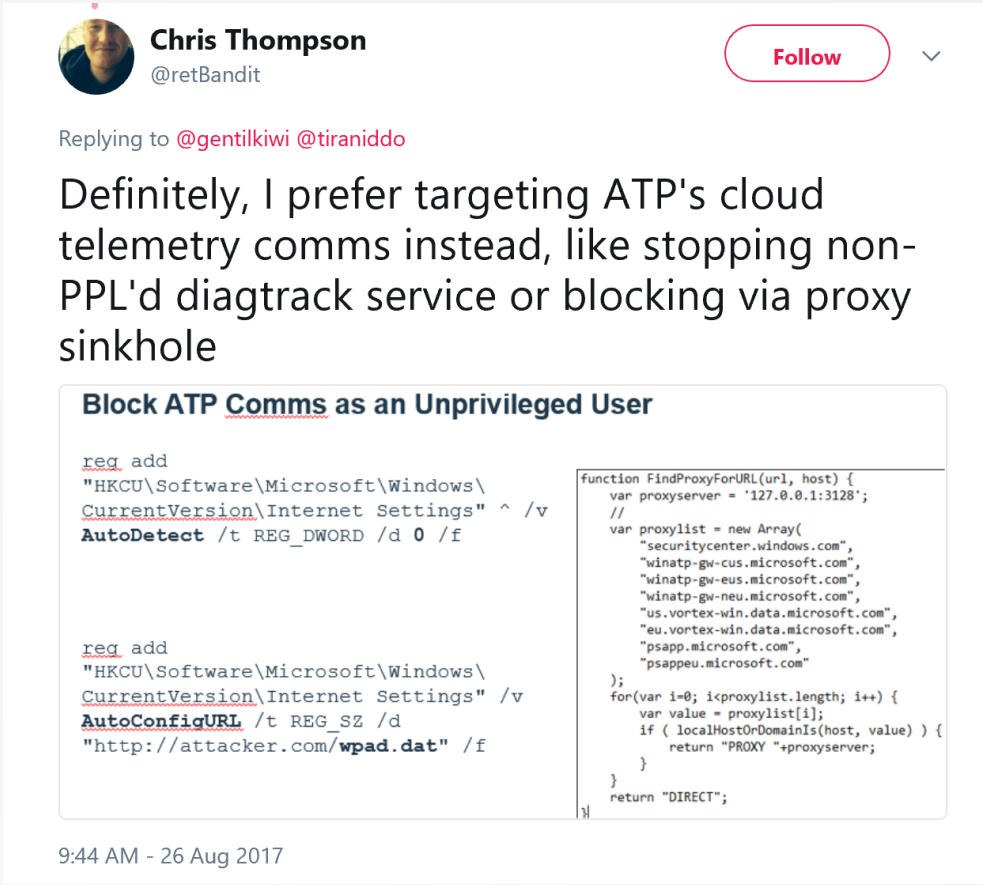
Firmware vendors can also provide the capability to attest to it presence

Attesting the security of DRTM is a key factor in hardware rooted zero trust



Software Device Trust Properties

EDR's provide one of the strongest device trust signals but are susceptible to tampering



Windows Runtime Attestations

Platform Tamper Detection for Windows

Spanning device boot to ongoing runtime process tampering

Designed for remote assessment of device health

Platform approach to benefit a variety of 3rd parties and scenarios

Hardware rooted device trust

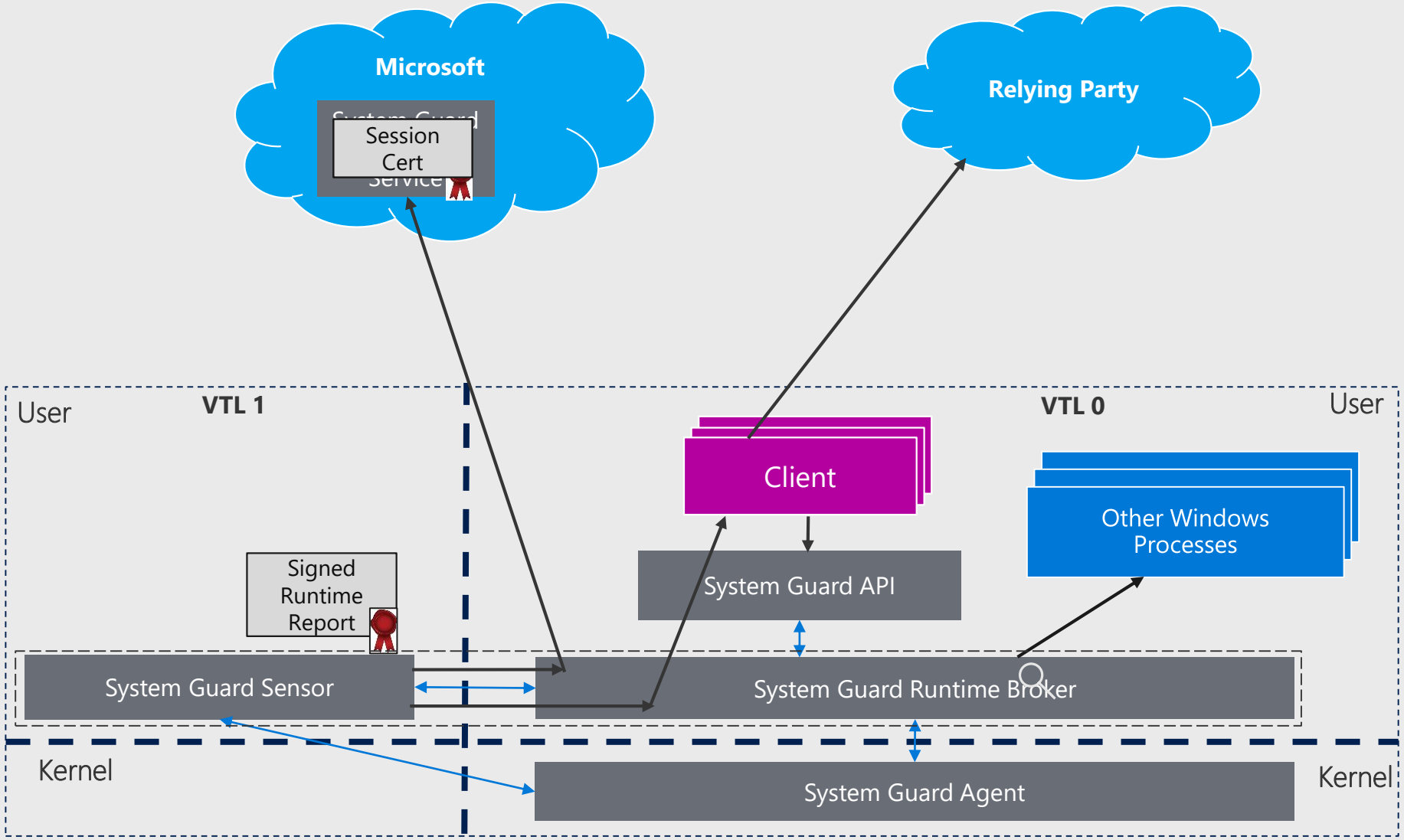
Leverage the VBS security boundary to raise the bar on anti-tampering

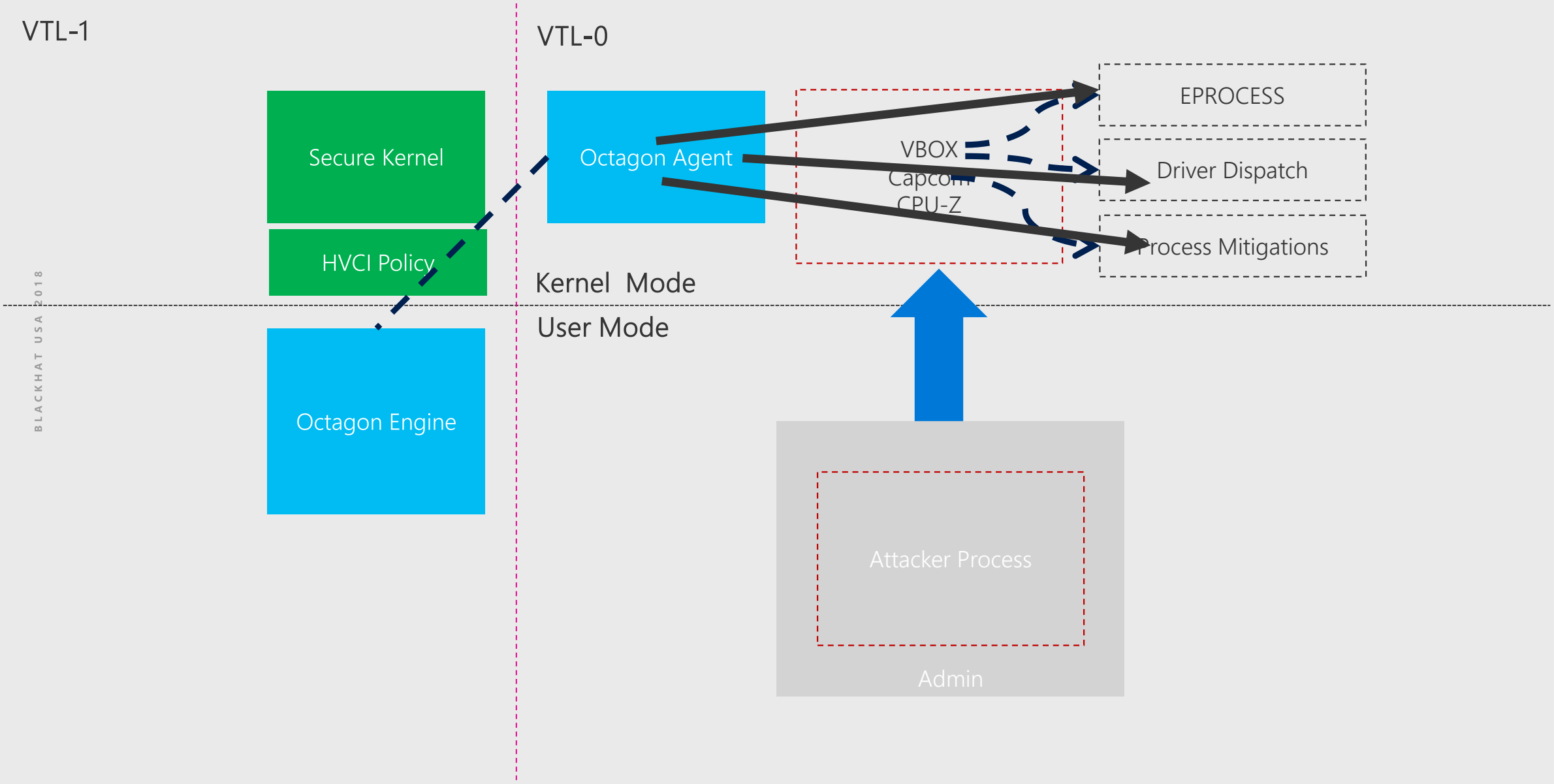
Challenging to build tamper detection schemes on top of Windows

Extensible platform component that can be used via forthcoming public API

Photo Source 1, 2, 3

System Guard Runtime Attestation





System Guard Runtime Attestation API

1. Create a session with a backend supplied nonce → **CreateSessionAsync(**nonce, relyingPartyId**)**

2. Grab the session report and certificate → **GetSessionCertificateAsync()** **GetSessionReportAsJWTAsync()**

3. Generate a runtime report with a backend supplied nonce → **GetRuntimeReportAsJWTAsync(nonce)**

4. Pass outputs of 2 & 3 to your backend (certificate and session+runtime reports)

5. Validate and read the runtime and session reports

GOTO 3. As/When desired generate fresh runtime reports → **GetRuntimeReportAsJWTAsync(nonce)**

GOTO 1. When needed renew the session → **CreateSessionAsync(nonce, relyingPartyId)**

DEMO

1

Secure Enclave Processor
Security circuit designed to perform secure services for the rest of the SOC
Prevents main processor from gaining direct access to sensitive data

2

KTRR/AMCC
Hardware memory controller prevents tampering of kernel data and code

3

Mandatory Access Control Framework
All applications are isolated (kernel + user), signed, and restricted from generating dynamic code

4

Hardware file and partition encryption
Keys for file, partition, cloud, and payments are all isolated user a hardware SoC

5

IBoot guarantees physical access protection
No loss of data from loss of physical access
Remote icloud locking renders it useless

Attestation on iOS

Device trust on iOS attempts to identify Jailbroken or compromised devices

Challenging because most iOS threats assume jailbreak

Zero trust "Agent" apps on the device run with standard app privs

Kernel privilege escalation makes bypassing local attestation bypass possible

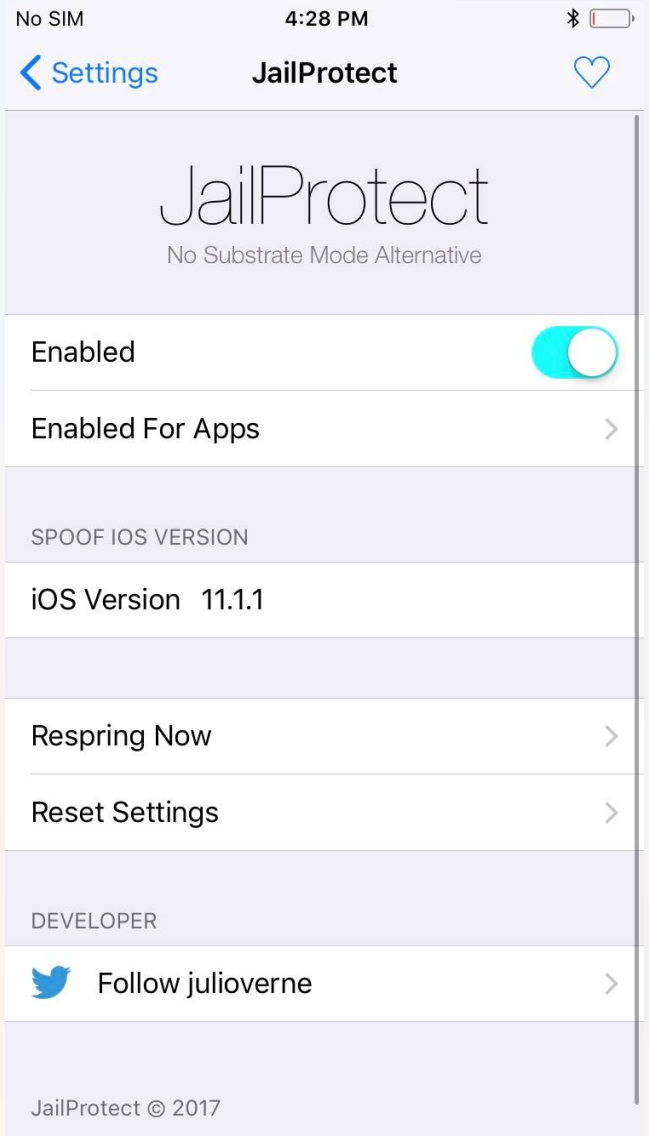
```
} void __cdecl RootCheck__(#493 *self, id a2)
{
…

v2 = "/Applications/Cydia.app";
if ( !mac_syscall(SYS_stat64, "/Applications/Cydia.app", (struct stat64
*)&v12) )
goto LABEL_32;
v2 = "/Library/MobileSubstrate/MobileSubstrate.dylib";
if ( !mac_syscall(SYS_stat64,
"/Library/MobileSubstrate/MobileSubstrate.dylib", (struct stat64 *)&v12) )
goto LABEL_32;
v2 = "/bin/bash";
if ( !mac_syscall(SYS_stat64, "/bin/bash", (struct stat64 *)&v12) )
goto LABEL_32;
v2 = "/bin/sh";
if ( !mac_syscall(SYS_stat64, "/bin/sh", (struct stat64 *)&v12) )
goto LABEL_32;
v2 = "/usr/sbin/sshd";
if ( !mac_syscall(SYS_stat64, "/usr/sbin/sshd", (struct stat64 *)&v12) )
goto LABEL_32;
v2 = "/usr/bin/sshd";
if ( !mac_syscall(SYS_stat64, "/usr/bin/sshd", (struct stat64 *)&v12) )
goto LABEL_32;
v2 = "/etc/apt";
if ( !mac_syscall(SYS_stat64, "/etc/apt", (struct stat64 *)&v12) )
goto LABEL_32;
v2 = "/evasi0n7";
if ( !mac_syscall(SYS_stat64, "/evasi0n7", (struct stat64 *)&v12) )
goto LABEL_32;
v2 = "/panguaxe";
if ( mac_syscall(SYS_stat64, "/panguaxe", (struct stat64 *)&v12)
&& (v2 = "/pguntether", mac_syscall(SYS_stat64, "/pguntether", (struct
stat64 *)&v12))
```

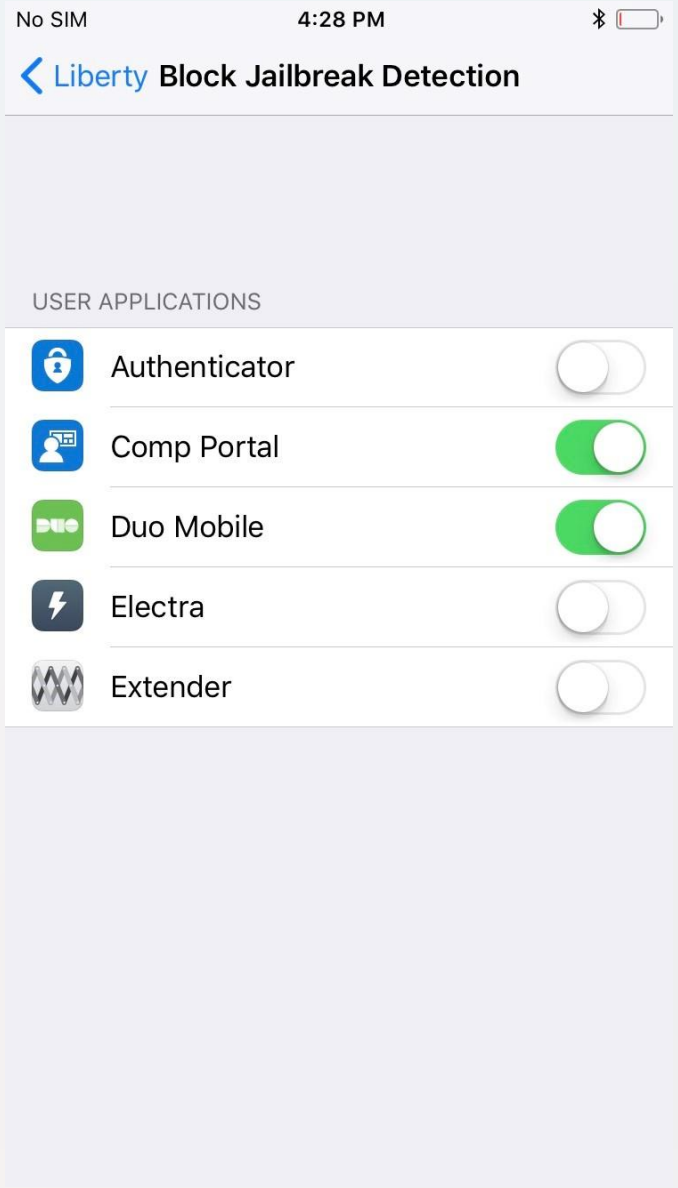
Device Trust Attestation on iOS

Public jailbreak modules in Cydia such as "Liberty" and "JailProtect" bypass most zero trust agent root detection out of the box

Inject hooking code into the app process and spoof software based security properties

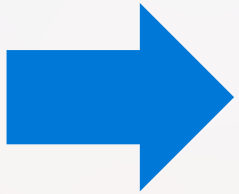
Software approaches which collect local data and perform analysis in the cloud can be more difficult to bypass



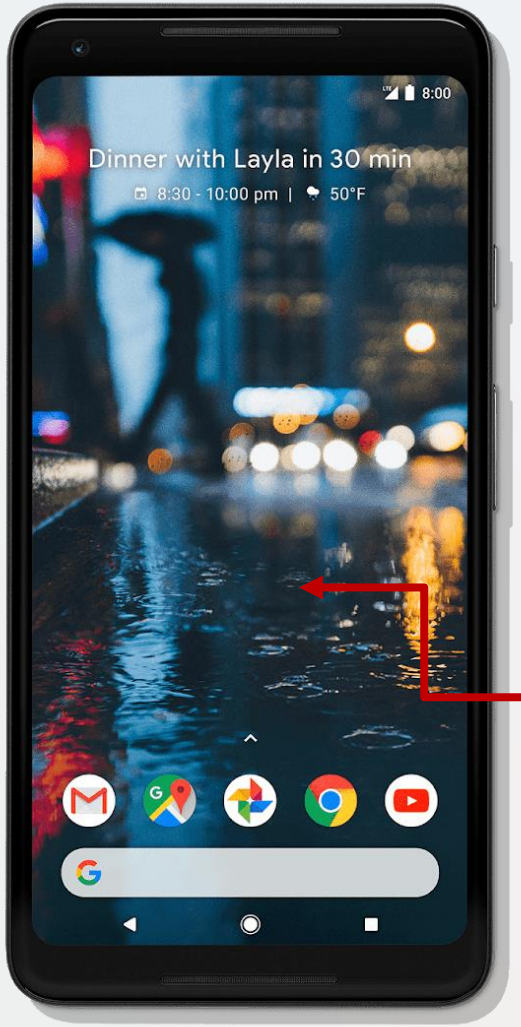


Bypassing Local Jailbreak Checks on iOS (LibertyLite)

```
LABEL_76:
                v93 = "access";
                v94 = replaced_access;
                v95 = (__int64 *)&qword_D100;
                v96 = "CFBundleGetAllBundles";
                v97 = replaced_CFBundleGetAllBundles;
                v98 = &qword_D108;
                v99 = "dlopen";
                v100 = replaced_dlopen;
                v101 = &qword_D110;
                v102 = "dlsym";
                v103 = replaced_dlsym;
                v104 = &qword_D118;
                v105 = "_dyld_image_count";
                v106 = replaced_dyld_image_count;
                v107 = &qword_D120;
                v108 = "fopen";
                v109 = replaced_fopen;
                v110 = &qword_D128;
                v111 = "fork";
                v112 = replaced_fork;
                v113 = &unk_D130;
                v114 = "getenv";
                v115 = replaced_getenv;
                v116 = &qword_D138;
                v117 = "lstat";
                v118 = replaced_lstat;
                v119 = &qword_D140;
                v120 = "open";
                v121 = replaced_open;
                v122 = &qword_D148;
                v123 = "opendir";
                v124 = replaced_opendir;
                v125 = &qword_D150;
                v126 = "stat";
                v127 = replaced_stat;
                v128 = &qword_D158;
                v129 = "statfs";
                v130 = replaced_statfs;
                v131 = &qword_D160;
                v132 = "symlink";
                v133 = replaced_symlink;
                v134 = &qword_D168;
                v135 = "sysctl";
                v136 = replaced_sysctl;
                v137 = &qword_D170;
                v138 = "sysctlbyname";
                v139 = replaced_sysctlbyname;
                v140 = &qword_D178;
                v141 = "system";
                v142 = replaced_system;
                v143 = &unk_D180;
                v144 = "vfork";
                v145 = replaced_vfork;
                v146 = &unk_D188;
                sub_9978(&v93, 18LL);
                if ( (unsigned int)objc_msgSend(
```



```
__data:000000000000C808 _blocked        DCQ aApplications       ; DATA XREF: InitFunc_0+21C↑o
__data:000000000000C808                                         ; replaced_access(char const*,int)+1C↑o ...
__data:000000000000C808                                         ; "/Applications"
__data:000000000000C810                 DCQ aApplications_0     ; "/Applications/"
__data:000000000000C818                 DCQ aApplicationsCy     ; "/Applications/Cydia.app"
__data:000000000000C820                 DCQ aApplicationsCy_0   ; "/Applications/Cydia.app/"
__data:000000000000C828                 DCQ aApplicationsCy_1   ; "/Applications/Cydia.app/Cydia"
__data:000000000000C830                 DCQ aApplicationsCy_2   ; "/Applications/Cydia.app/Info.plist"
__data:000000000000C838                 DCQ aApplicationsCy_3   ; "/Applications/Cydia.app/../Cydia.app"
__data:000000000000C840                 DCQ aApplicationsCy_4   ; "/Applications/Cydia.app/../Cydia.app/"
__data:000000000000C848                 DCQ aApplicationsCy_5   ; "/Applications/Cydia.app/../Cydia.app/In"...
__data:000000000000C850                 DCQ aApplicationsFa     ; "/Applications/FakeCarrier.app"
__data:000000000000C858                 DCQ aApplicationsIc     ; "/Applications/Icy.app"
__data:000000000000C860                 DCQ aApplicationsIn     ; "/Applications/Iny.app"
__data:000000000000C868                 DCQ aApplicationsIf     ; "/Applications/iFile.app"
__data:000000000000C870                 DCQ aApplicationsAc     ; "/Applications/Activator.app"
__data:000000000000C878                 DCQ aApplicationsIn_0   ; "/Applications/IntelliScreen.app"
__data:000000000000C880                 DCQ aApplicationsMx     ; "/Applications/MxTube.app"
__data:000000000000C888                 DCQ aApplicationsRo     ; "/Applications/RockApp.app"
__data:000000000000C890                 DCQ aApplicationsSb     ; "/Applications/SBSettings.app"
__data:000000000000C898                 DCQ aApplicationsWi     ; "/Applications/WinterBoard.app"
__data:000000000000C8A0                 DCQ aApplicationsBl     ; "/Applications/blackra1n.app"
__data:000000000000C8A8                 DCQ aLibraryActivat     ; "/Library/Activator"
__data:000000000000C8B0                 DCQ aLibraryFlipswi     ; "/Library/Flipswitch"
__data:000000000000C8B8                 DCQ aLibraryFramewo     ; "/Library/Frameworks/CydiaSubstrate.fram"...
__data:000000000000C8C0                 DCQ aLibraryMobiles     ; "/Library/MobileSubstrate"
__data:000000000000C8C8                 DCQ aLibraryMobiles_0   ; "/Library/MobileSubstrate/DynamicLibrari"...
__data:000000000000C8D0                 DCQ aLibraryMobiles_1   ; "/Library/MobileSubstrate/DynamicLibrari"...
__data:000000000000C8D8                 DCQ aLibraryMobiles_2   ; "/Library/MobileSubstrate/DynamicLibrari"...
__data:000000000000C8E0                 DCQ aLibraryMobiles_3   ; "/Library/MobileSubstrate/MobileSubstrat"...
__data:000000000000C8E8                 DCQ aLibraryMobiles_4   ; "/Library/MobileSubstrateMobileSubstrate"...
__data:000000000000C8F0                 DCQ aLibraryRington     ; "/Library/Ringtones"
__data:000000000000C8F8                 DCQ aLibrarySwitchs     ; "/Library/Switchs"
__data:000000000000C900                 DCQ aLibraryWallpap     ; "/Library/Wallpaper"
__data:000000000000C908                 DCQ aSystemLibraryL     ; "/System/Library/LaunchDaemons/com.ikey."...
__data:000000000000C910                 DCQ aSystemLibraryL_0   ; "/System/Library/LaunchDaemons/com.sauri"...
__data:000000000000C918                 DCQ aBinBash            ; "/bin/bash"
__data:000000000000C920                 DCQ aBinSh              ; "/bin/sh"
__data:000000000000C928                 DCQ aBin                ; "/bin"
__data:000000000000C930                 DCQ aBinSu              ; "/bin/su"
__data:000000000000C938                 DCQ aEtcApt             ; "/etc/apt"
__data:000000000000C940                 DCQ aEtcApt_0           ; "/etc/apt/"
__data:000000000000C948                 DCQ aEtcClutchConf      ; "/etc/clutch.conf"
__data:000000000000C950                 DCQ aEtcClutchCrack     ; "/etc/clutch_cracked.plist"
__data:000000000000C958                 DCQ aEtcSshSshdConf     ; "/etc/ssh/sshd_config"
__data:000000000000C960                 DCQ aPrivate            ; "/private/"
__data:000000000000C968                 DCQ aPrivate_0          ; "/private"
__data:000000000000C970                 DCQ aPrivateVstbWri     ; "/private/vstb_writable_check"
__data:000000000000C978                 DCQ aPrivateEtcFsta     ; "/private/etc/fstab"
__data:000000000000C980                 DCQ aPrivateMiitomo     ; "/private/Miitomo"
__data:000000000000C988                 DCQ aPrivateVarLibA     ; "/private/var/lib/apt"
__data:000000000000C990                 DCQ aPrivateVarLibA_0   ; "/private/var/lib/apt/"
__data:000000000000C998                 DCQ aPrivateVarLibC     ; "/private/var/lib/cydia"
__data:000000000000C9A0                 DCQ aPrivateVarLibC_0   ; "/private/var/lib/cydia/"
__data:000000000000C9A8                 DCQ aPrivateVarTmpC     ; "/private/var/tmp/cydia.log"
__data:000000000000C9B0                 DCQ aPrivateVarMobi     ; "/private/var/mobile/Library/SBSettings/"...
__data:000000000000C9B8                 DCQ aPrivateVarMobi_0   ; "/private/var/mobileLibrary/SBSettingsTh"...
__data:000000000000C9C0                 DCQ aPrivateVarStas     ; "/private/var/stash"
__data:000000000000C9C8                 DCQ aPrivateVarStas_0   ; "/private/var/stash/"
__data:000000000000C9D0                 DCQ aPrivateVarTmpC     ; "/private/var/tmp/cydia.log"
__data:000000000000C9D8                 DCQ aPrivateVarTmpC_0   ; "/private/var/tmp/Cydia.log"
```



1

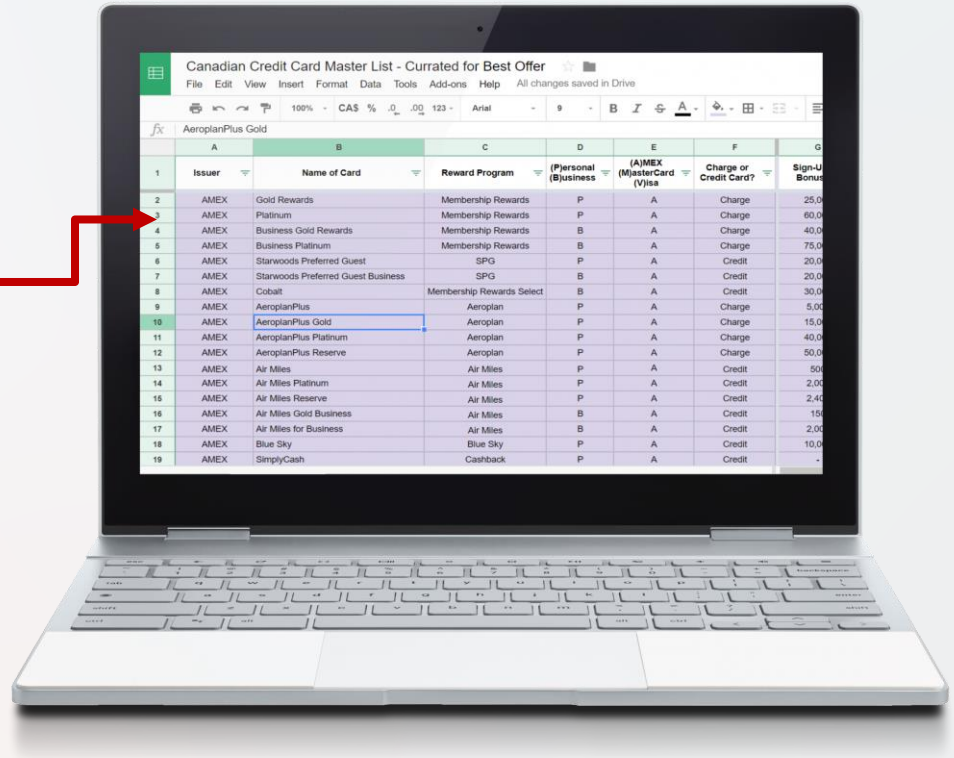
Security Module

Tamper resistant TEE for storing credentials, boot, and encryption keys

Mandatory Isolation

All apps are isolated including Android and arbitrary apps isolated in containers

2



3

OS and boot integrity

Verified Access and SafetyNet provide attestation

Core boot is used to provide tiny TCB in firmware

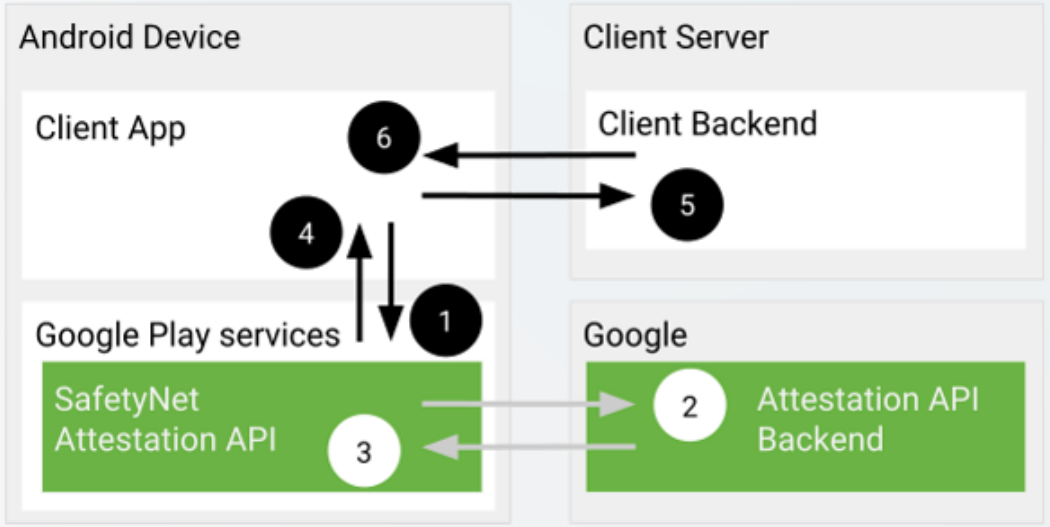
Attesting to Security on Android

To deal with a diverse and fragmented ecosystem Google created SafetyNet a public API for attesting to device security

SafetyNet runs numerous checks including: SELinux, Certificates, Bootloader, DNS, Root detection, etc.

SafetyNet runs in user-mode and use obfuscation and dynamic reflection to make reverse engineering difficult

Excellent presentation from [BH17 covering SafetyNet internals]



Device Status	Value of "ctsProfileMatch"	Value of "basicIntegrity"
Certified, genuine device that passes CTS	true	true
Certified device with unlocked bootloader	false	true
Genuine but uncertified device, such as when the manufacturer doesn't apply for certification	false	true
Device with custom ROM (not rooted)	false	true
Emulator	false	false
No device (protocol emulator script)	false	false
Signs of system integrity compromise, such as rooting	false	false
Signs of other active attacks, such as API hooking	false	false

Device Trust on Android

Similar to iOS, most zero trust agents implemented as low privilege app

Apps reviewed used a combination of SafetyNet plus custom root checks

Commonly available anti-jailbreak tools bypass most zero trust apps

Similar to iOS, most threats involved system capabilities or root which makes bypassing root checks possible

```
private static boolean isDeviceRooted()
{
Object localObject = new ArrayList();
((List)localObject).add("/system/bin");
((List)localObject).add("/system/xbin");
((List)localObject).addAll(Arrays.asList(System.getenv("PATH").split(":")));
localObject = ((List)localObject).iterator();
while (((Iterator)localObject).hasNext())
{
String str = (String)((Iterator)localObject).next();
if (new File(str + "/su").exists()) {
return true;
}
}
return false;
}
```

```
public class RootCloakInstalledRootTest implements IRootTest {
private static final String[] ROOT_CLOAK_PACKAGE_SIGNATURES = new
String[]{"devadvance", "rootcloak"};
protected static final String XPOSED_BRIGE_CLASS =
"de.robv.android.xposed.XposedBridge";
protected static final String XPOSED_METHOD_REPLACEMENT_CLASS =
"de.robv.android.xposed.XC_MethodReplacement";
\private int checkClassForName(String str) {
int i = 0;
try {
str = Class.forName(str, false, getClass().getClassLoader());
} catch (String str2) {
if (StringUtils.indexOfAny(ExceptionUtils.getStackTrace(str2),
ROOT_CLOAK_PACKAGE_SIGNATURES) != -1) {
```

Hiding from Device Trust on Android (Magisk)

```
#!/system/bin/sh
#################################################
#################################################
#
# Magisk Boot Image Patcher
# by topjohnwu
#
# Usage: sh boot_patch.sh <bootimage>
#
# The following additional flags can be set in
environment variables:
# KEEPVERITY, KEEPFORCEENCRYPT, HIGHCOMP
#
# This script should be placed in a directory
with the following files:
#
# File name Type Description
#
# boot_patch.sh script A script to patch boot.
Expect path to boot image as parameter.
# (this file) The script will use b
files in its same directory
# to complete the patching process
# util_functions.sh script A script which
hosts all functions requires for this script
# to work properly
# magiskinit binary The binary to replace
/init, which has the magisk binary embedded
# magiskboot binary A tool to unpack boot
image, decompress ramdisk, extract ramdisk,
# and patch the ramdisk for Magisk support
```

```
static void hide_daemon(int pid) {
    LOGD("hide_daemon: start unmount for pid=[%d]\n", pid);

    char *line, buffer[PATH_MAX];
    struct vector mount_list;

    manage_selinux();
    clean_magisk_props();

    if (switch_mnt_ns(pid))
        goto exit;

    snprintf(buffer, sizeof(buffer), "/proc/%d/mounts", pid);
    vec_init(&mount_list);
    file_to_vector(buffer, &mount_list);

    // Unmount dummy skeletons and /sbin links
    vec_for_each(&mount_list, line) {
        if (strstr(line, "tmpfs /system/") || strstr(line, "tmpfs
/vendor/") || strstr(line, "tmpfs /sbin")) {
            sscanf(line, "%*s %4096s", buffer);
            lazy_unmount(buffer);
        }
        free(line);
    }
```

```
static char *prop_key[] =
    { "ro.boot.vbmeta.device_state",
"ro.boot.verifiedbootstate",
"ro.boot.flash.locked", "ro.boot.veritymode",
     "ro.boot.warranty_bit", "ro.warranty_bit",
"ro.debuggable", "ro.secure",
     "ro.build.type", "ro.build.tags",
"ro.build.selinux", NULL };

static char *prop_value[] =
    { "locked", "green", "1", "enforcing",
     "0", "0", "0", "1",
     "user", "release-keys", "0", NULL };

void hide_sensitive_props() {
    LOGI("hide_utils: Hiding sensitive props\n");

    // Hide all sensitive props
    char *value;
    for (int i = 0; prop_key[i]; ++i) {
        value = getprop(prop_key[i]);
        if (value) {
            if (strcmp(value, prop_value[i]) != 0)
                setprop2(prop_key[i],
prop_value[i], 0);
            free(value);
        }
    }
}
```

Improving Zero Trust on Android

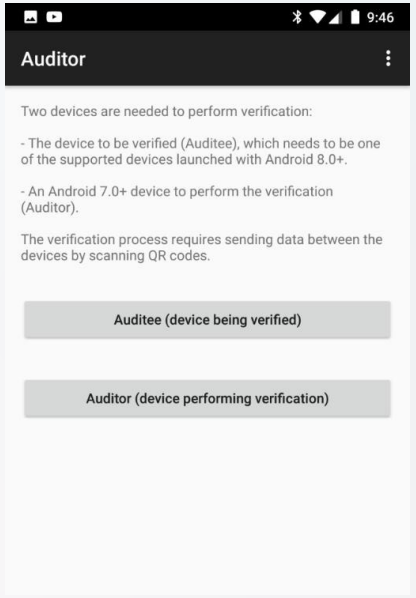
Magisk and most root hiders required unlock-> patching both the bootimg and kernel (e.g. Samsung KNOX RKP)

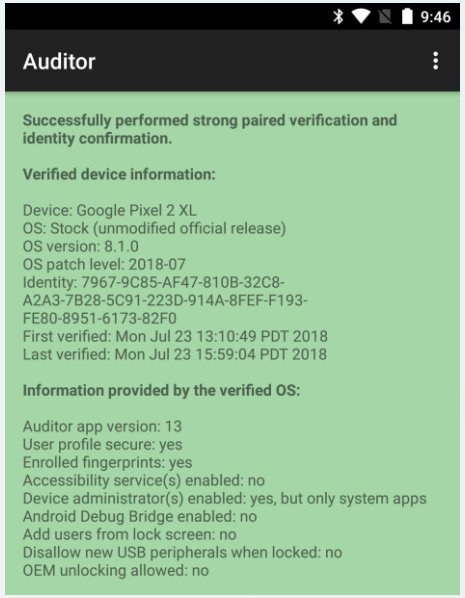
Validating security properties via TEE provides a much stronger guarantee

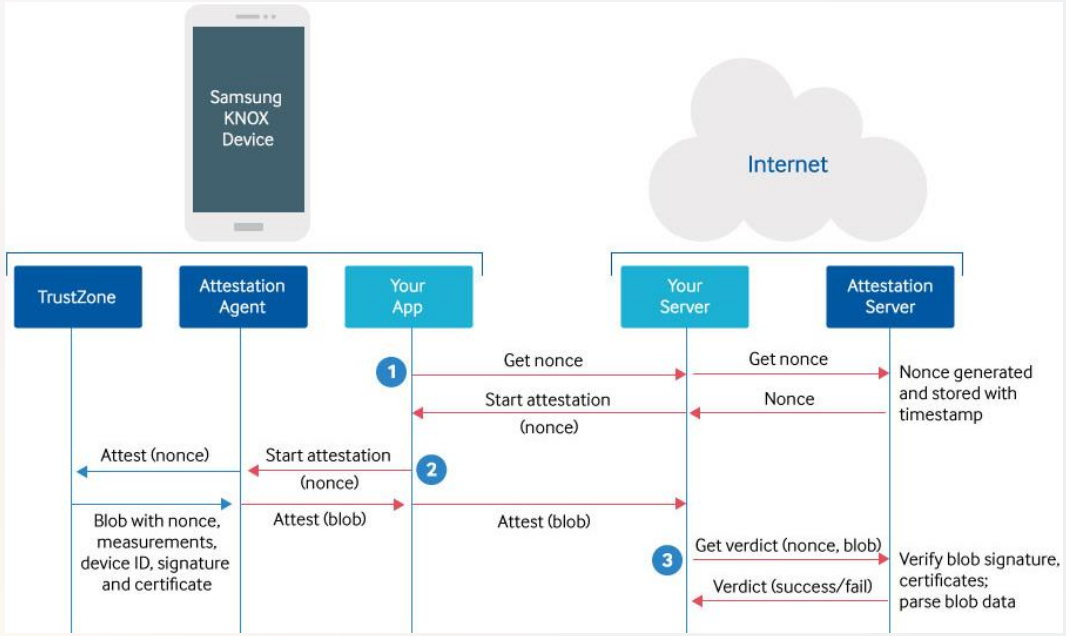
Copperhead OS provides the auditor tool which validates core security properties from TEE

Can be used from either server or client

Samsung also provides an attestation service for KNOX phones







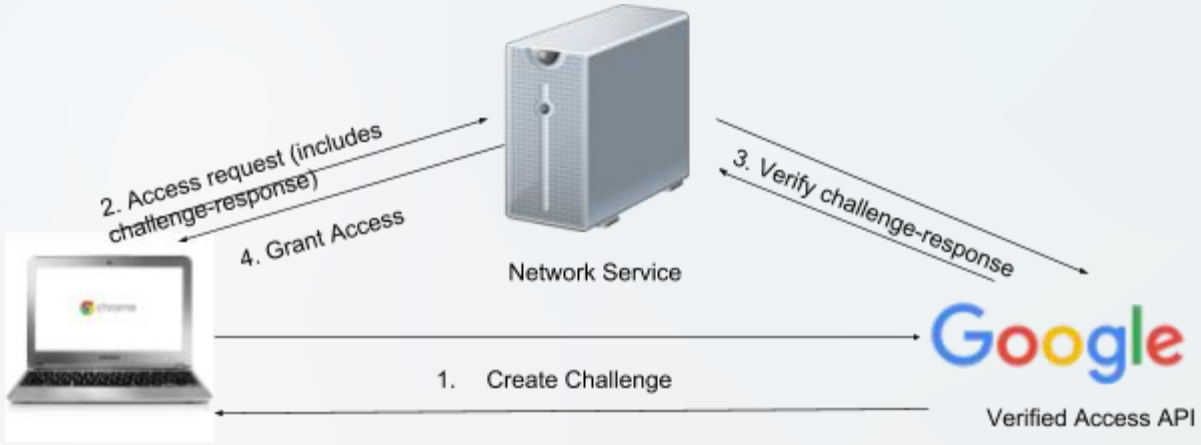
Chrome OS Verified Boot

Verified Boot is a hardware rooted attestation mechanism for Chromebooks

Provides a hardware rooted cryptographic mechanism for attesting to a ChromeOS Devices

Ensures the device is secure including lock status, patch status, and firmware integrity

Used by Duo for “Duo Beyond” zero trust on Chromebooks




```
// Generate challenge response
var encodedChallenge; // obtained by create challenge API call
try {
if (isDeviceVerification) { // isDeviceVerification set by external logic
chrome.enterprise.platformKeys.challengeMachineKey(
decodestr2ab(encodedChallenge), ChallengeCallback);
} else {
chrome.enterprise.platformKeys.challengeUserKey(
decodestr2ab(encodedChallenge), true, ChallengeCallback);
}
} catch (error) {
console.log('ERROR: ' + error);
}
```

Gzob QQ who r u???

Device Trust Takeaways

Device trust is an import part of the Zero Trust Security Model

Most platforms lack a hardware rooted high integrity mechanism for authenticating and evaluating a device

Most zero trust evaluators today are susceptible to tampering and can be defeated

EDR and Vulnerability Management tools are not deeply integrated into most products

Vendors and platforms need to work together to improve device trust capabilities

"These agents go to great lengths in order to attest and prove validity of the measurements they report, but even cursory thought quickly reaches the conclusion that these efforts are generally futile. ***Software-backed measurements lack the protection provided by hardware measurements, and an attacker with sufficient privilege can subvert systems like this****"*

-- Zero Trust Networking – Barth, Gilman

Software != Hardware trust

Establishing Trust in the User

USER TRUST

Identity is the new perimeter

ZTN is the combined ability to leverage device + user trust for every resource access

Bread and butter of attack trade craft:

Attack initial identity and exploit transitive trust

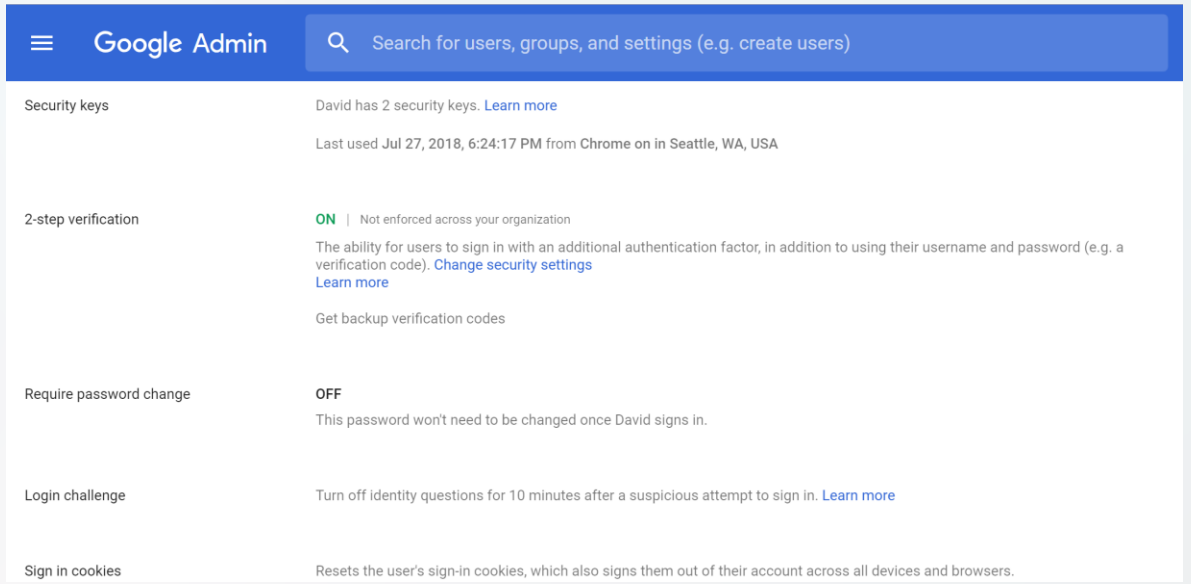
Attacks on identity can occur on or off device (PtH or Phishing)

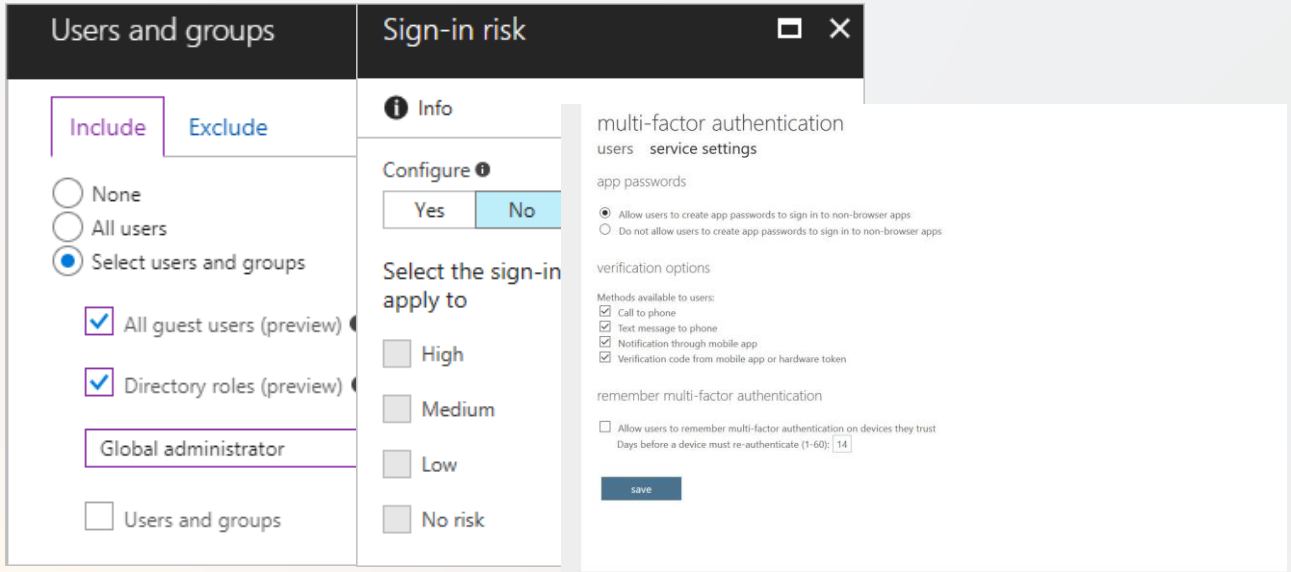
Multiple identity mitigations:

Access Anomalies, Credential Storage, MFA, and fine grained access

Single-Sign on identity providers such as AAD and G-Suite provide a single source of identity

SSO can also be the enforcement points with federation or proxy integrations





Phishing	• Password less authentication (bio, PIN) • MFA (U2F, SEP, etc.)
Lateral Movement	• Hardware credential binding (SEP, Credential Guard) • Anomaly detection • Access Proxy enforcement
Insider	• Least Privilege Access • Access Proxy Enforcement • Anomaly Detection

Zero Trust Identity Challenges

Existing network designs

Identity proxies works greats for cloud native or cloud federated resources

Most attack surface is not cloud

Addressing gap requires backhauling to cloud via n-premise proxy

Introduces scale and latency issues

Non-HTTP protocols

Local Admin accounts, SMB, PSExec, Powershell RMI not handled by HTTP proxy

Proxy Requires tunneling over SSH/HTTP/S

Most common **lateral movement vectors will not be covered** without full rearchitecture

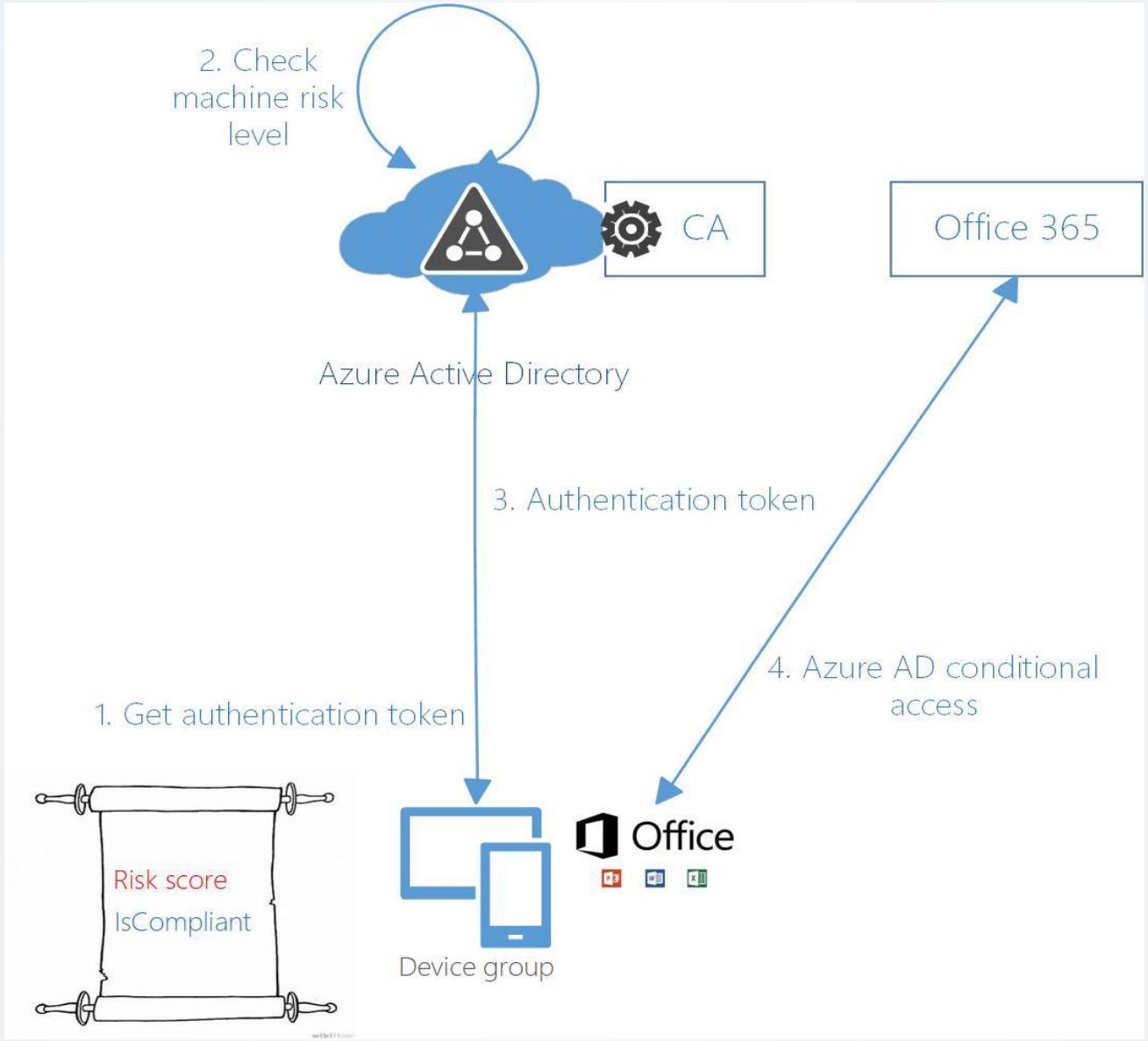
Convenience vs Security

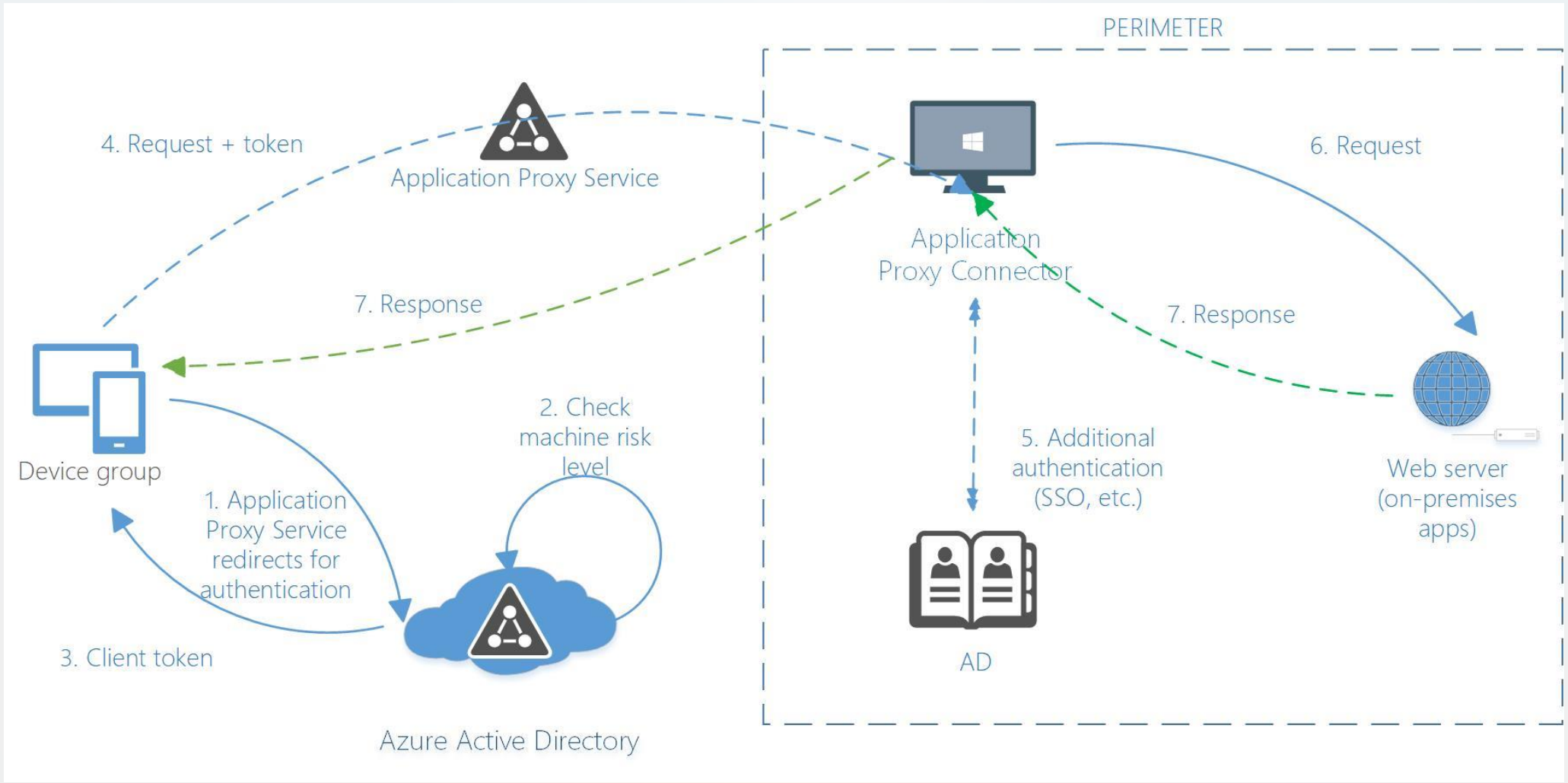
TTL on most tokens will be days or weeks

As a result most authentication is not MFA backed

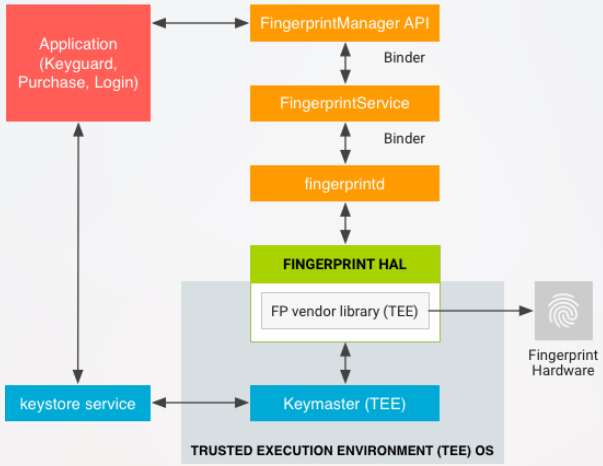
Attackers can do a LOT of damage within the TTL

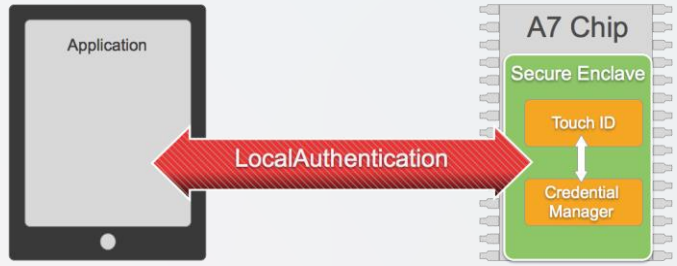
Getting MFA challenges for SMB file copies is not popular


2. Check machine risk level
CA
Office 365
Azure Active Directory
3. Authentication token
4. Azure AD conditional access
1. Get authentication token
Risk score
IsCompliant
Device group
Office


PERIMETER
4. Request + token
Application Proxy Service
6. Request
Application
Proxy Connector
7. Response
7. Response
Device group
2. Check
machine risk
level
1. Application
Proxy Service
redirects for
authentication
5. Additional
authentication
(SSO, etc.)
Web server
(on-premises
apps)
3. Client token
AD
Azure Active Directory

Mitigating Credential Theft



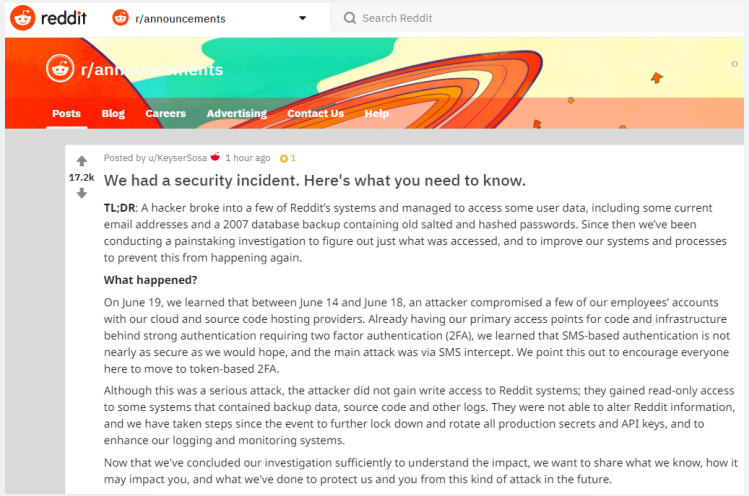


Combining isolated credential storage with SSO identity mitigates many threats

Identity standards such as FIDO also combine local attestation which helps zero trust

U2F and other standard surpass SMS and OTP based 2FA which can still be phished

Not all MFA is created the same



SIM Card Swap

An adversary could convince the mobile network operator (e.g. through social networking or forged identification) to issue a new SIM card and associate it with an existing phone number and account[1]. The adversary could then obtain SMS messages or hijack phone calls intended for someone else[2]. One use case is intercepting authentication messages or phone calls to obtain illicit access to online banking or other online accounts[3].

Exploit SS7 to Redirect Phone Calls/SMS

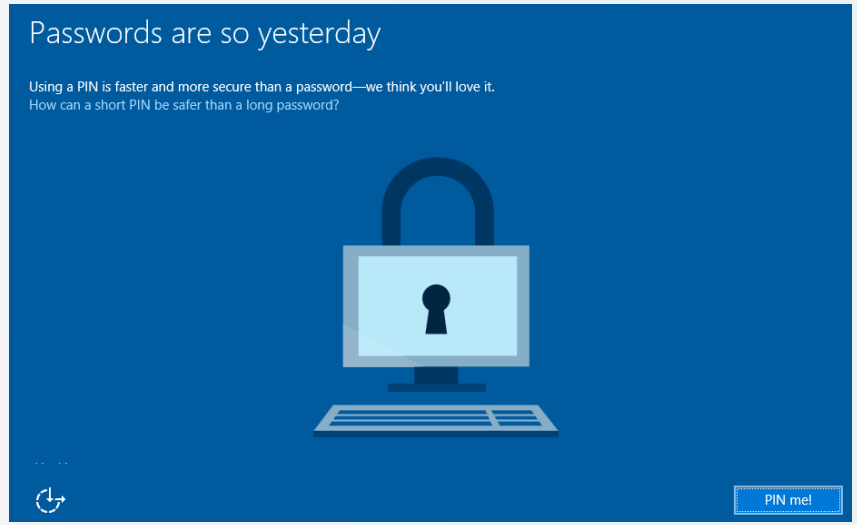
An adversary could exploit signaling system vulnerabilities to redirect calls or text messages to a phone number under the attacker's control. The adversary could then act as a man-in-the-middle to intercept or manipulate the communication. These issues are discussed in [1], [2], [3], [4], as well as in a report from the Communications, Security, Reliability, and Interoperability Council (CSRIC) [5].

Improving Local Identity Protection

Windows Hello and NGC

Offers biometric authentication and hardware backed key storage

PIN vulnerable to input attacks from malicious admin

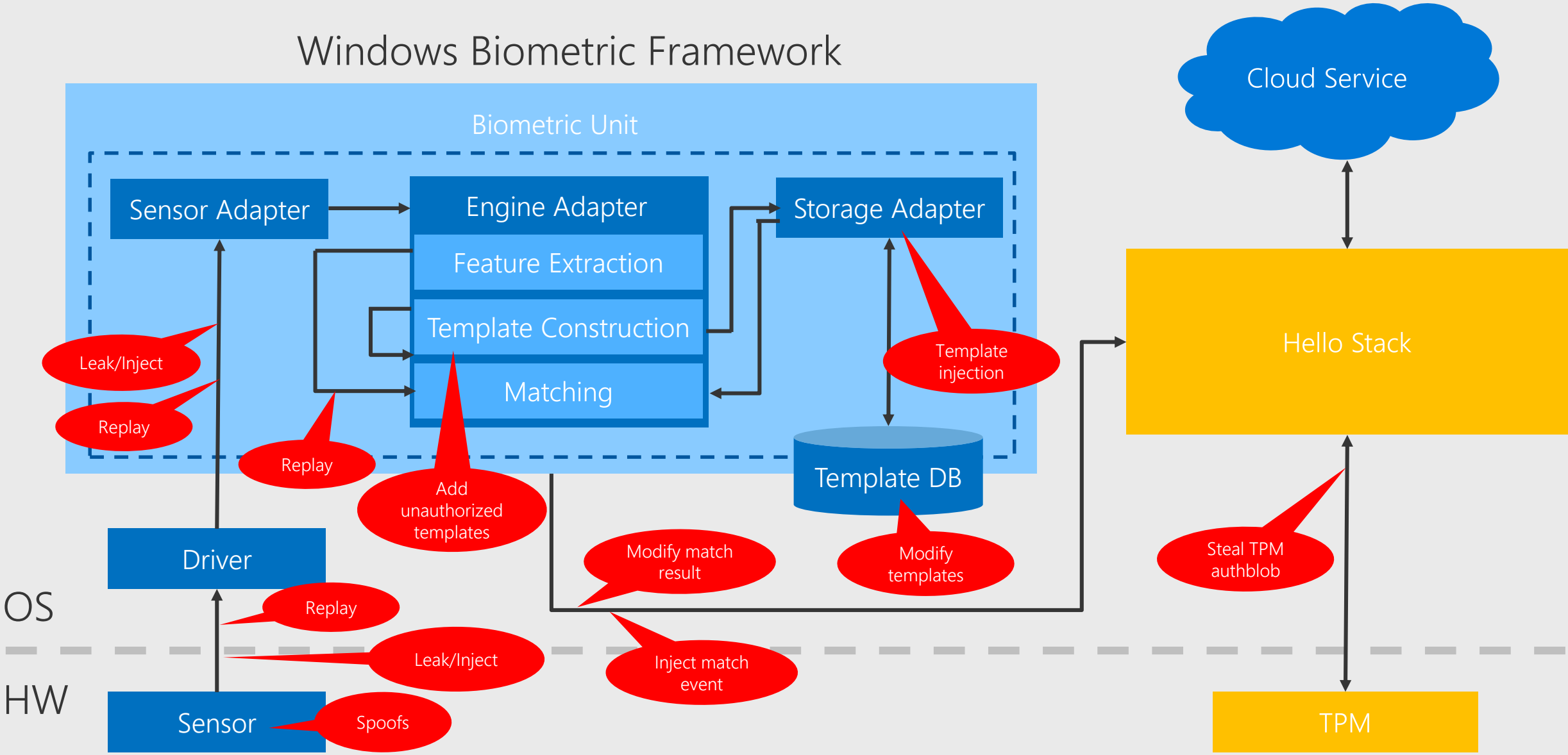


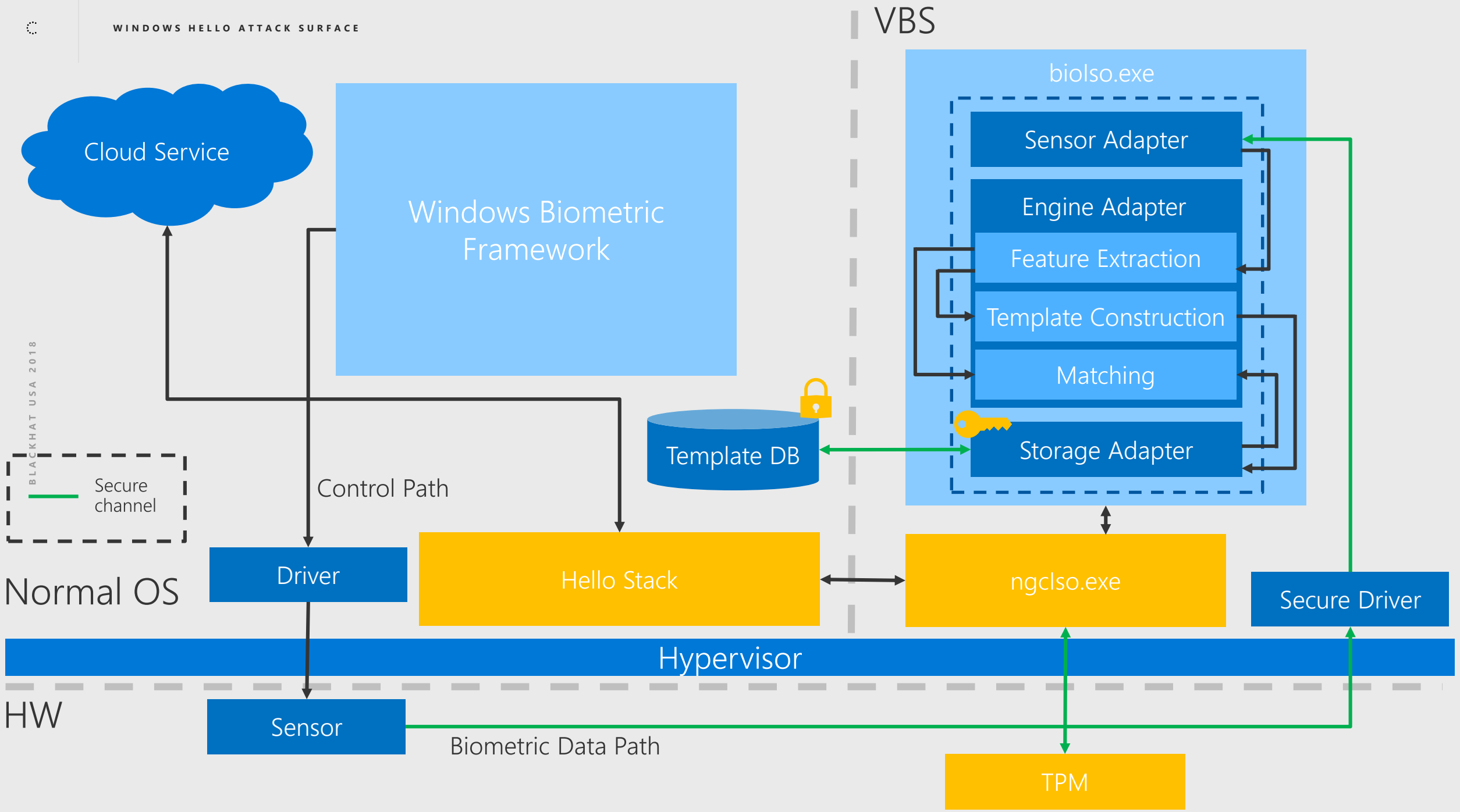
Improving Identity Security

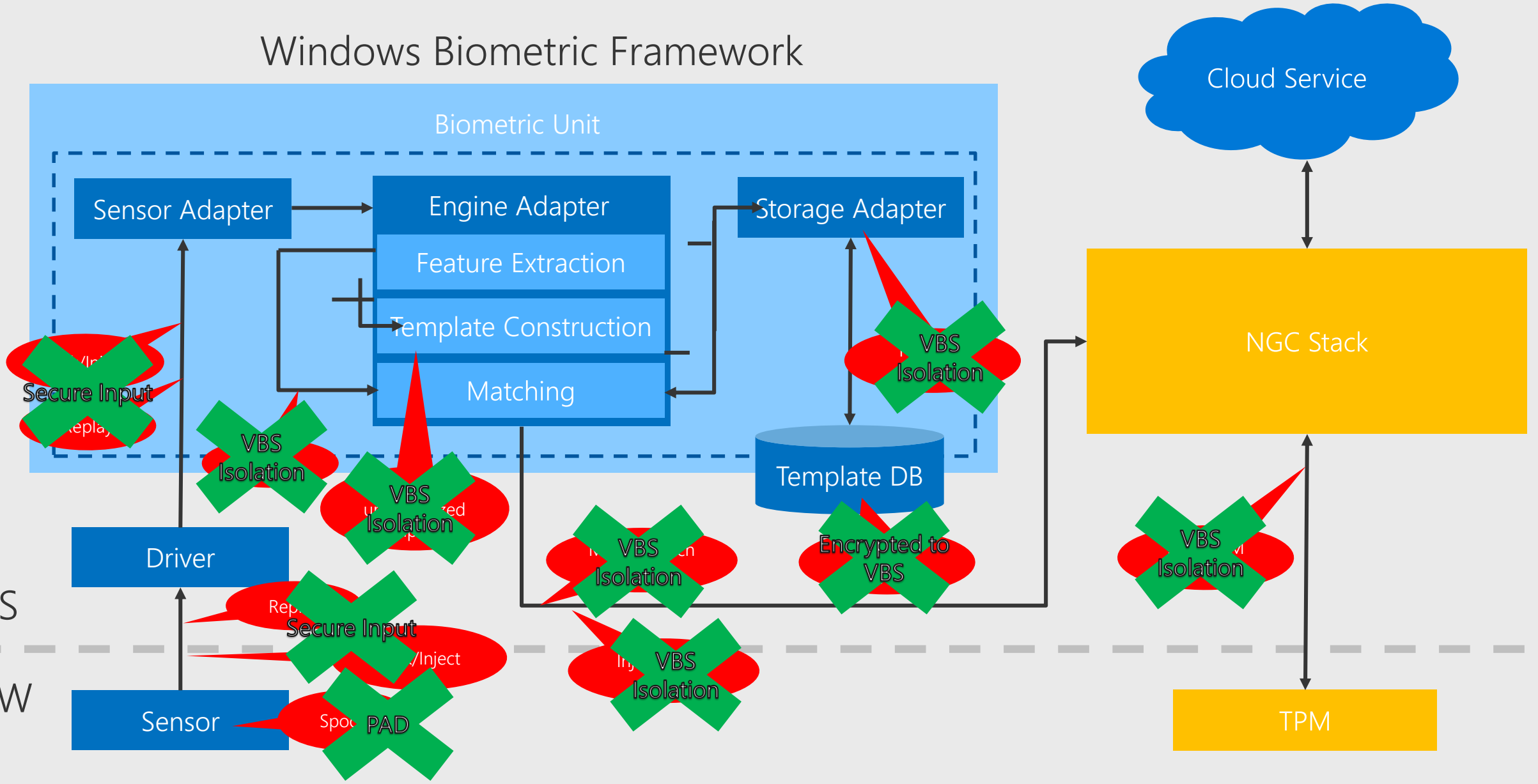
Future version of Windows include biometric hardening enabled through virtualization

Biometric hardening of the data path using virtualization

Hardening of credential release




VBS
biolso.exe
Sensor Adapter
Engine Adapter
Feature Extraction
Template Construction
Matching
Storage Adapter
Cloud Service
Windows Biometric Framework
Template DB
Secure channel
Control Path
Normal OS
Driver
Hello Stack
ngclso.exe
Secure Driver
Hypervisor
HW
Sensor
Biometric Data Path
TPM
BLACKHAT USA 2018


Windows Biometric Framework
Cloud Service
Biometric Unit
Sensor Adapter
Engine Adapter
Feature Extraction
Template Construction
Matching
Storage Adapter
NGC Stack
Template DB
Driver
Sensor
OS
HW
TPM
Secure Input
VBS Isolation
Encrypted to VBS
PAD

Beyond Passwords

A web without passwords

Staying secure on the web is more important than ever. We trust web sites to process credit card numbers, save addresses and personal information, and even to handle sensitive records like medical information. All this data is protected by an ancient security model—the password. But passwords are difficult to remember, and are fundamentally insecure—often re-used, and vulnerable to phishing and cracking.

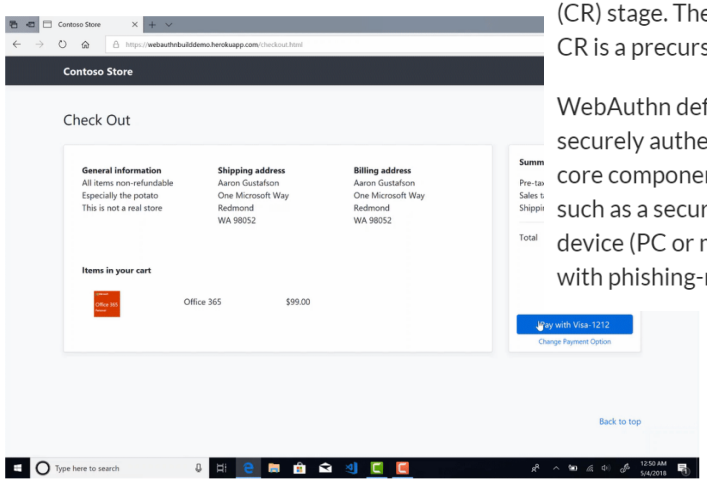
For these reasons, Microsoft has been leading the charge towards a world without passwords, with innovations like Windows Hello biometrics and pioneering work with the FIDO Alliance to create an open standard for passwordless authentication – Web Authentication.

We started this journey in 2016, when we shipped the industry's first preview implementat
Authentication API in Microsoft Edge. Since then, we have been updating our implementa
other vendors and the FIDO alliance to develop the standard. In March, the FIDO Alliance
Authentication APIs have reached Candidate Recommendation (CR) status in the W3C, a m
maturity and interoperability of the specification.

Authenticators in Microsoft Edge

Beginning with build 17723, Microsoft Edge supports the CR version of Web Authenticatio
provides the most complete support for Web Authentication to date, with support for a wi
authenticators than other browsers.

Windows Hello allows users to authenticate without a password on any Windows 10 devi
face and fingerprint recognition—or a PIN number to sign in to web sites. With Windows
users can log in to sites that support Web Authentication in seconds, with just a glance.



FIDO Alliance and W3C Achieve Major Standards Milestone in Global Effort Towards Simpler, Stronger Authentication on the Web

April 10, 2018

With support from Google Chrome, Microsoft Edge and Mozilla Firefox, FIDO2 Project opens new era of ubiquitous, phishing-resistant, strong authentication to protect web users worldwide

MOUNTAIN VIEW, Calif., and https://www.w3.org/ — April 10, 2018 – The FIDO Alliance and the World Wide Web Consortium (W3C) have achieved a major standards milestone in the global effort to bring simpler yet stronger web authentication to users around the world. The W3C has advanced Web Authentication (WebAuthn), a collaborative effort based on Web API specifications submitted by FIDO to the W3C, to the Candidate Recommendation (CR) stage. The CR is the product of the Web Authentication Working Group, which is comprised of representatives from over 30 member organizations. CR is a precursor to final approval of a web standard, and the W3C has invited online services and web app developers to implement WebAuthn.

WebAuthn defines a standard web API that can be incorporated into browsers and related web platform infrastructure which gives users new methods to securely authenticate on the web, in the browser and across sites and devices. WebAuthn has been developed in coordination with FIDO Alliance and is a core component of the FIDO2 Project along with FIDO's Client to Authenticator Protocol (CTAP) specification. CTAP enables an external authenticator, such as a security key or a mobile phone, to communicate strong authentication credentials locally over USB, Bluetooth or NFC to the user's internet access device (PC or mobile phone). The FIDO2 specifications collectively enable users to authenticate easily to online services with desktop or mobile devices with phishing-resistant security.

User Trust Takeaways

Enforcing strong identity and authentication is a strong weapon against lateral movement

Identity protection has a strong dependency on the scope of enforcement

Many of the most common network pathways are difficult to cover with existing ZTN products

To get the best protection, the network needs to undergo a complete reach architecture

Not all MFA is created the same

Using hardware segmentation, secure elements, and biometrics built into the OS can help to balance the user impact of ZTN

MFA is not enough

Where did we land

Conclusion

Does it work?

Can you trust Zero Trust?

Zero Trust is more of a philosophy than a well-defined security model or architecture

Many ways to implement it, core principles matter

The model is a huge step in the right direction and an improvement over perimeter based “tootsie roll” networks

Most vendors are overpromising security value, lots of low hanging fruit and room for improvement

Hardware rooted device identity, user identity and device trust can help

Move to the cloud *should eventually* remove legacy protocol issues

Identity really is the new perimeter

Simultaneously re-architecting network for Zero Trust while moving to the cloud will enable you take advantage of future improvements

Zero trust is promising, but has a transformative cost and is still maturing