Zero Trust

Firewalls

“All of ARPA’s [the firewall] protection has, by design, left the internal AT&T machines untested—a sort of crunchy shell around a soft, chewy center.”

—*The Design of a Secure Internet Gateway*, Bill Cheswick, 1990

The Purpose of Firewalls

- “Firewalls are *not* a solution to network problems. They are a network response to a host security problem.
- “More precisely, they are a response to the dismal state of software engineering; taken as a whole, the profession does not know how to produce software that is secure, correct, and easy to administer.
- “Consequently, better network protocols will not obviate the need for firewalls. The best cryptography in the world will not guard against buggy code.”

(Bellovin, 1994)

Firewalls

“Firewalls are useless against attacks from the inside. An inside attack can be from a legitimate user who has turned to the dark side, or from someone who has obtained access to an internal machine by other means. Malicious code that executes on an internal machine, perhaps having arrived via an e-mail virus or by exploiting a buffer overflow on the machine, can also be viewed as an inside attacker.”

—*Firewalls and Internet Security, Second Edition*, Cheswick, Bellovin, and Rubin, 2003

Firewalls

“Since the dawn of the commercial Internet, firewalls have been a mainstay of the defense. . . That said, their utility, and in particular the protection they provide, has diminished markedly over the years. The time has come to ask whether the general-purpose firewall—the one protecting an enterprise—is still worth its capital, operational, and productivity cost.

. . .

“There are three properties necessary for a firewall to be effective:

. . .

“3. All nodes on the “inside” must be ‘good’; all nodes on the outside are, if not actually ‘bad,’ untrusted.”

—*Thinking Security*, Steven M. Bellovin, 2016

Firewalls?

- Centralized firewalls as a primary defense are, if not obsolete, at least obsolescent
- Topology changes, communication needs, mobile devices, and threat model changes have all combined: the solution of 1990 and 1994 no longer works
- But—some of the same drivers still exist
- The question is what to do
- One answer: zero-trust architectures

What is a “Zero Trust” Architecture?

- Assume that any element is or will be compromised
- Hosts, routers, switches, links, Active Directory nodes—anything!
- How do you stay secure?
- What must you do?

Functional Requirements

- Strong authentication
- Proper authorization for all requests
- Ubiquitous encryption
- Constant monitoring

More Requirements

- Usability
- Access to all necessary services
- Support for common devices
- Support for all necessary software
- A migration path

Consequences

- No difference between “inside” and “outside” networks
- Easy access to all resources from the open Internet
- No corporate firewall!
- No VPNs!
- All resources are just as accessible from the open Internet as they are from “inside” the company

But How Do We Do This?

Authentication

- All authentication is via some sort of public key protocol
- Why?

Authentication

- All authentication is via some sort of public key protocol
- Why?
- If a password database is compromised, there's the risk of large-scale password-guessing
- If a machine accepting passwords is compromised, passwords can be collected that way
- There are also risks from phishing, keystroke loggers, etc.
- With public key authentication, none of that matters

However. . .

- Where do key pairs come from?
- How are they stored?
- Why are they trusted?

Certificates

- Certificates are self-authenticating
- Much better than locally stored public keys
- That assumes that the CA hasn’t been compromised, but it can be offline or separated by an airgap
- Air gaps aren’t perfect, but they’re very strong

Private Keys

- Don’t store a private key on a user machine—use something like FIDO2 or a smart card
- (The Federal PIV card has a chip that can store keys and certificates)
- Prevent compromise of the private key

Authenticating Devices

- Authenticate devices, too, again via public key cryptography
- Reject—and flag—valid user logins from unknown devices
- Guard against credential theft, including physical theft
- But—don’t (if feasible) trust a device without a user credential, too

Usable Authentication

- Cannot ask users to log in for every network operation
- Must have *single sign-on*
- The sign-on operation effectively enables the device and user private keys
- What if the user's device is compromised?

Single Sign-On?

- What is “single sign-on”?

Single Sign-On?

- What is “single sign-on”?
- Sign on *once* (per day) to an organization
- You (somehow) get credentials good for the rest of the day (or at least for several hours) for all resources within that organization
- On the open web, that function is often performed via “Log in with Google” or “Log in with Facebook”
- In an enterprise, it’s most often Kerberos, generally as part of Windows Active Directory
- What are the risks and benefits of this?

Authorization

- Authorization is crucial—*must* limit what anyone can do
- Authorization is by device and user—a compromised device can't get to places the user is not authorized for
- Note well: authorization is at the *application level*, not the network

Centralized or Decentralized Authorization?

- If you centralize authorization, you have a single point of security failure
- If you decentralize it, administration becomes far more difficult
- How do you handle employees leaving? Known compromises?

Coarse- or Fine-Grained Authorization?

- Coarse authorization is good for things like “may access this database”
- ☞ This is probably best done centrally
- Fine authorization can control access to particular rows or columns
- ☞ Easiest if done locally, to avoid overloading the central authorization servers
- ☞ Note well: proper fine-grained authorization can compensate for a hacked authorization server

Attributes and Certificates

- Put attributes into certificates to simplify administration
- Example: “ACCESS: Personnel_Database”
- Access control decisions can be complex: attributes, name, organization, time of day, and more

The Certificate Process

- Don’t forget the people and the process
- *People* have to issue employee certificates
- *People* have to set up authorization
- Use audits as a check

Encryption

- Encrypt all network traffic
- Tie encryption to bilateral authentication
- Prevent network eavesdropping
- More important, prevent injection of fraudulent traffic
- Note well: encryption is at the application layer, not the network

Intrusion Detection

- Comprehensive intrusion detection is central to the whole scheme
- Otherwise, an attacker can go hop by hop, with failures unnoticed
- Most intrusion detection has to happen on end-hosts—with all traffic encrypted, there's no way to do it in the network
- However—metadata matters, too

☞ Is too much data being exported from a given server?

The Manning Case

- Chelsea Manning downloaded very many classified documents

The Manning Case

- Chelsea Manning downloaded very many classified documents
- ☞ Why didn’t the document server notice the bulk downloads?

The Manning Case

- Chelsea Manning downloaded very many classified documents
- ☞ Why didn't the document server notice the bulk downloads?
- From Manning in a chat: "Weak servers, weak logging, weak physical security, weak counter-intelligence, inattentive signal analysis. . . a perfect storm"

Logging

- Logs from every element must go to a secure log server
- All analysis is done there
- Logs from different components can be correlated
- Suspect machines and/or users can be identified and their credentials revoked

Device Inventory

- Keep a record of every device and its status
- Include frequency of use data—might the device's patches be out of date?
- That feeds into the access control mechanism, too

Did We Succeed?

- Do we really trust nothing?
- What happens if some component is compromised?
- And is the resulting network secure?

Did We Succeed?

- Do we really trust nothing?
- What happens if some component is compromised?
- And is the resulting network secure?
- Answers: No, oops, not really...

Is it Really ϵ Trust?

- We trust the CA—and the people running it—to only issue certificates to the right people
- We trust the authorization database(s)
- If the log server is compromised, we'll never notice failed intrusion attempts

Attack Surfaces

- More seriously—this simplistic approach to zero trust doesn't address the issue of attack surface
- A system that is vulnerable because of buggy code, rather than stolen credentials, is still vulnerable
- The purpose of a firewall is to keep the attackers away from buggy code—and we've gotten rid of the firewall. . .

Other Issues

- Legacy software that doesn’t support encryption
- Third-party software that doesn’t do our sort of authorization
- Legacy devices that don’t support encryption, e.g., many VoIP phones
- Machine-to-machine communication, e.g., for email, with no users involved
- Emergencies and outages

Google’s Approach: BeyondCorp

- Access Proxies
- VLANs
- Tunnels
- Design for reliability
- Design for scale

Access Proxies

- Most internal Google services are web-based and hence handle TLS
- Have users, external or internal, connect to an Access Proxy (AP)
- The AP does authentication via client-side certificates and coarse authorization
- Communication to the actual web server—which is encrypted and authenticated—goes through the AP

☞ Note: what if the AP is compromised?

VLANs: Virtual LANs

- (What's a VLAN? A virtual LAN: Ethernet and WiFi can support multiple independent networks on the same physical infrastructure)
- Put servers on a separate VLAN
- Do the same with legacy devices
- ☞ This isolates them from direct access from the Internet *or* the corporate intranet
- As part of the transition effort, users authenticate to the network; this determines what VLAN they're assigned to
- Ultimately, users are assigned to a non-privileged VLAN

Tunnels

- To support legacy software, “tunnel” drivers are used
- A tunnel driver sets up an encrypted, authenticated connection between two machines, one of which is typically an AP
- As usual, the AP enforces access control

Reliability

- Google (obviously!) pays a lot of attention to reliability
- The APs are a crucial link, so they're replicated
- In fact, they're often load-balancing front ends of the type we've discussed
- There are paths that don't go through the APs, in order to repair systems when something has made the APs non-functional

Scale

- Google is *big*
- Correction: Google is *really* big
- Access control policies for different internal services are controlled by individual groups
- The access control language was written to be as flexible as possible
- As much as possible, e.g., provisioning loaner laptops, is done via a self-service model

Is this Truly Zero Trust?

Is this Truly Zero Trust?

- No, of course not

Is this Truly Zero Trust?

- No, of course not
- The APs enforce a lot of policy and restrictions
- The authorization servers say who can talk to whom
- The VLAN assignment processing could put people on a net with privileged servers
- And then there's the AP repair path

However. . .

- There is no longer a firewall per se
- Usage from more or less anywhere is the same for most employees
- The overall attack surface of the *enterprise* is vastly reduced compared with firewall-based architectures
- It's a low trust architecture, with the most critical component—the APs—the most standardized and hence the easiest to get right

What Did Google Really Do?

- They have done fine-grain separation of their network
- They could have used separate LANs instead of VLANs, post-migration, but VLANs are more convenient
- The APs serve as application-layer firewalls; privileges are granted by user and device identity, not topology
- There is far more intrusion detection

Client Devices

- Client devices are truly mobile
- (But if you read the fine print, there's a reliance on a Chrome extension that you have to disable manually to, e.g., talk to hotel or airport login services)
- This part of the ZTA architecture was completely realized

Servers

- Servers, with their valuable data and large attack surfaces, are still protected from the Internet
- We call the protection Access Proxies instead of firewalls, but the concept is the same
- They have to do fine-grained authentication, but there's a large infrastructure to support that

The Network

- The network is completely untrusted
- Everything is encrypted, even internally
- This is application-level encryption, not IPsec or other network-layer encryption
- That permits user-grain granularity of keying, at the cost of modifying applications
- But HTTP, which already supports TLS, is by far the most important protocol, so no change was needed

Intrusion Detection

- They've centralized logging
- (We've already discussed why that's a good idea in any event)
- They've souped up intrusion detection
- They do look at network metadata, but rely on hosts to do content-level detection and logging

Revisiting the “Chewy Center”

- A compromised inside host has no inherent access to other hosts
- A malicious insider doesn’t have another user’s credentials
- There are nodes that have to be trusted, but there are many fewer of them
- Authentication can be end-to-end
- *Everything* is verified

Questions?

(Red-winged blackbird attacking a great egret, Central Park, July 26, 2020)