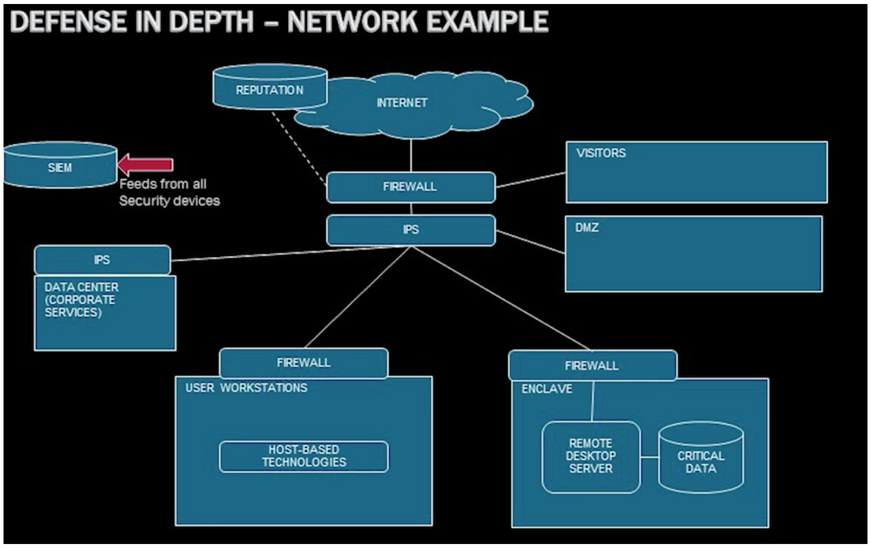
Nicolas Barraclough
11/5/2020
CS373 - Defense Against the Dark Arts

Week 6 Write-Up

This week was all about network security. In order to ensure that a whole network of systems are secure, proper security measures must be implemented.

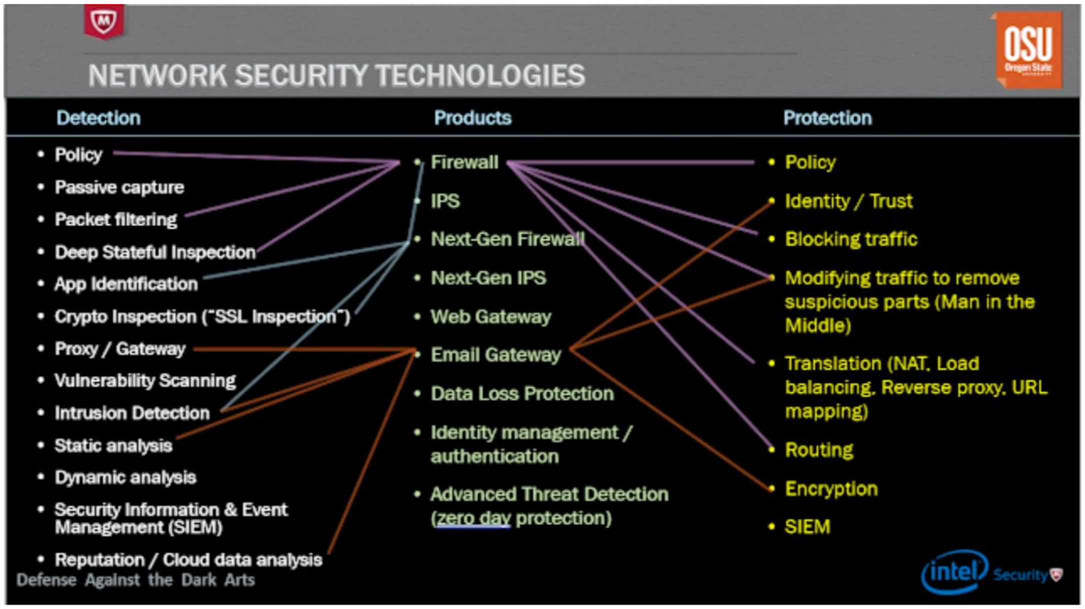
A major factor in keeping a network safe is to implement defense in depth.


DEFENSE IN DEPTH – NETWORK EXAMPLE

Defense in depth is where multiple security measures are implemented at different depths of the network. By ensuring defense in depth, if a portion of the network is compromised, the attacker will still be restricted from the other areas. The lecturer used an analogy of a medieval castle where various areas of the castle each have respective defenses. For example if the attacker(s) take over the barbican of the castle (the DMZ of the network), they will still not have access to anything important. The rest of the network will still be secure.

Firewalls are another way to restrict malicious access to the network. Firewalls are a way for a network to know which actions are permitted, so that any unpermitted actions can be stopped before they are carried out. They are placed between the different zones of the network and define policies for the permitted actions.

A proxy takes in data and regenerates it to send through the network, screening for anything malicious. If a proxy sees something that might be malicious, it can screen just that piece of data, rather than restricting the entire connection. For example, if something within an online advertisement is malicious, a proxy will still allow the entire webpage to be visited, whereas a firewall would block the entire page due to the advertisement.



NETWORK SECURITY TECHNOLOGIES

Detection

- Policy
- Passive capture
- Packet filtering
- Deep Stateful Inspection
- App Identification
- Crypto Inspection ("SSL Inspection")
- Proxy / Gateway
- Vulnerability Scanning
- Intrusion Detection
- Static analysis
- Dynamic analysis
- Security Information & Event Management (SIEM)
- Reputation / Cloud data analysis

Products

- Firewall
- IPS
- Next-Gen Firewall
- Next-Gen IPS
- Web Gateway
- Email Gateway
- Data Loss Protection
- Identity management / authentication
- Advanced Threat Detection (zero day protection)

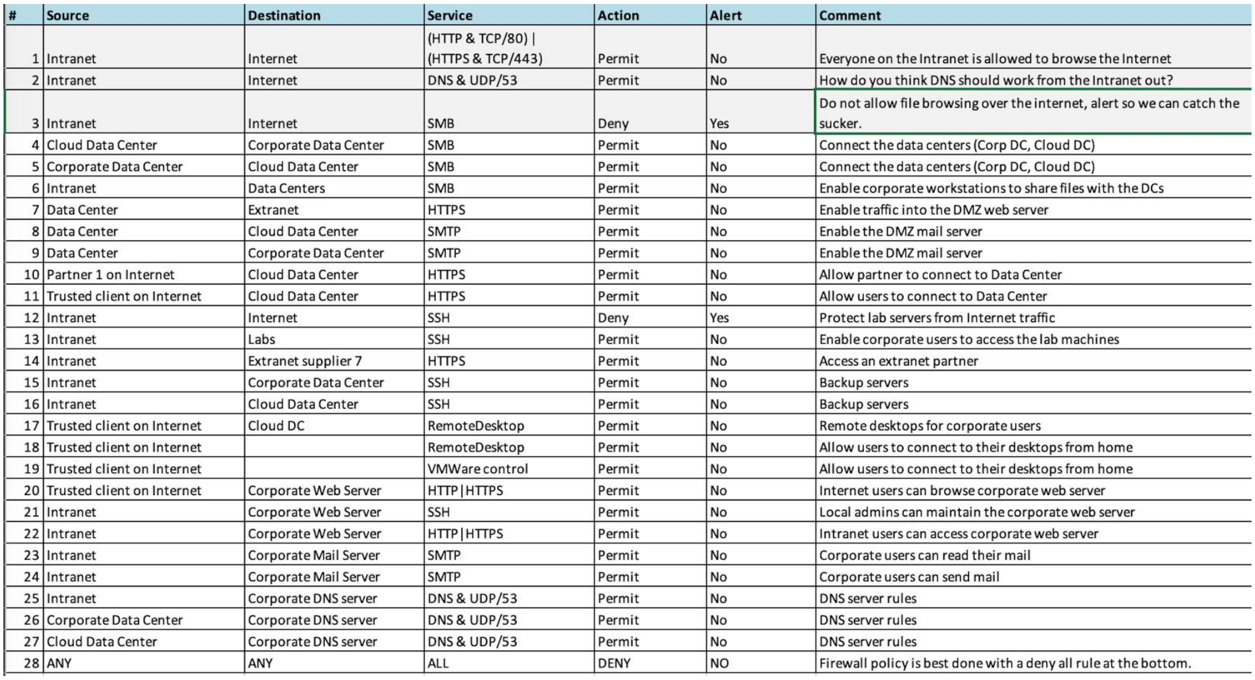
Protection

- Policy
- Identity / Trust
- Blocking traffic
- Modifying traffic to remove suspicious parts (Man in the Middle)
- Translation (NAT, Load balancing, Reverse proxy, URL mapping)
- Routing
- Encryption
- SIEM

Defense Against the Dark Arts

One thing that I had never heard of before this week was quarantining within a network. The lecturer gave the example of an airport's wifi restricting internet access until the user agrees to the terms and conditions. I obviously knew that this existed, but I had no idea how it worked or what it was formally called. I'm interested to learn more about how a network can ensure its own security (after the proper policies and protections are defined by the network security administrators).

Firewall Policy:



#	Source	Destination	Service	Action	Alert	Comment
1	Intranet	Internet	(HTTP & TCP/80) | (HTTPS & TCP/443)	Permit	No	Everyone on the Intranet is allowed to browse the Internet
2	Intranet	Internet	DNS & UDP/53	Permit	No	How do you think DNS should work from the Intranet out?
3	Intranet	Internet	SMB	Deny	Yes	Do not allow file browsing over the internet, alert so we can catch the sucker.
4	Cloud Data Center	Corporate Data Center	SMB	Permit	No	Connect the data centers (Corp DC, Cloud DC)
5	Corporate Data Center	Cloud Data Center	SMB	Permit	No	Connect the data centers (Corp DC, Cloud DC)
6	Intranet	Data Centers	SMB	Permit	No	Enable corporate workstations to share files with the DCs
7	Data Center	Extranet	HTTPS	Permit	No	Enable traffic into the DMZ web server
8	Data Center	Cloud Data Center	SMTP	Permit	No	Enable the DMZ mail server
9	Data Center	Corporate Data Center	SMTP	Permit	No	Enable the DMZ mail server
10	Partner 1 on Internet	Cloud Data Center	HTTPS	Permit	No	Allow partner to connect to Data Center
11	Trusted client on Internet	Cloud Data Center	HTTPS	Permit	No	Allow users to connect to Data Center
12	Intranet	Internet	SSH	Deny	Yes	Protect lab servers from Internet traffic
13	Intranet	Labs	SSH	Permit	No	Enable corporate users to access the lab machines
14	Intranet	Extranet supplier 7	HTTPS	Permit	No	Access an extranet partner
15	Intranet	Corporate Data Center	SSH	Permit	No	Backup servers
16	Intranet	Cloud Data Center	SSH	Permit	No	Backup servers
17	Trusted client on Internet	Cloud DC	RemoteDesktop	Permit	No	Remote desktops for corporate users
18	Trusted client on Internet		RemoteDesktop	Permit	No	Allow users to connect to their desktops from home
19	Trusted client on Internet		VMWare control	Permit	No	Allow users to connect to their desktops from home
20	Trusted client on Internet	Corporate Web Server	HTTP|HTTPS	Permit	No	Internet users can browse corporate web server
21	Intranet	Corporate Web Server	SSH	Permit	No	Local admins can maintain the corporate web server
22	Intranet	Corporate Web Server	HTTP|HTTPS	Permit	No	Intranet users can access corporate web server
23	Intranet	Corporate Mail Server	SMTP	Permit	No	Corporate users can read their mail
24	Intranet	Corporate Mail Server	SMTP	Permit	No	Corporate users can send mail
25	Intranet	Corporate DNS server	DNS & UDP/53	Permit	No	DNS server rules
26	Corporate Data Center	Corporate DNS server	DNS & UDP/53	Permit	No	DNS server rules
27	Cloud Data Center	Corporate DNS server	DNS & UDP/53	Permit	No	DNS server rules
28	ANY	ANY	ALL	DENY	NO	Firewall policy is best done with a deny all rule at the bottom.

Robustness Principle: 1980-1989 from RFC-1122 Jonathan Postel, 1989

Once there was a great man, named Postel. See RFC 2468.

1.2.2 Robustness Principle

At every layer of the protocols, there is a general rule whose application can lead to enormous benefits in robustness and interoperability [ref to rfc760, 1980]:

> "Be liberal in what you accept, and conservative in what you send"

- This seems backwards. I assume it would be safer to be conservative in what is accepted, and liberal in what is sent. Based on what we learned this week, we scrutinize what we receive in order to ensure safety of the network.

Software should be written to deal with every conceivable error, no matter how unlikely; sooner or later a packet will come in with that particular combination of errors and attributes, and unless the software is prepared, chaos can ensue. In general, it is best to assume that the network is filled with malevolent entities that will send in packets designed to have the worst possible effect.

- Due to cost/time constraints, this may not be possible, but ideally, software should be written to deal with every conceivable error.

This assumption will lead to suitable protective design, although the most serious problems in the Internet have been caused by unenvisaged mechanisms triggered by low-probability events; mere human malice would never have taken so devious a course!

- The most serious problems in the internet come from malicious, attacker-designed software, not “low-probability events”. We need to design according to how the system might be attacked, instead of just how it might fail.

Adaptability to change must be designed into all levels of Internet host software. As a simple example, consider a protocol specification that contains an enumeration of values for a particular header field—e.g., a type field, a port number, or an error code; this enumeration must be assumed to be incomplete. Thus, if a protocol specification defines four possible error codes, the software must not break when a fifth code shows up.

An undefined code might be logged (see below), but it must not cause a failure.

- Software will most likely always change in some ways, so designing an adaptable system is highly recommended. The software should have fail-safe mechanisms to handle unexpected errors so that the software doesn’t break due to an exception.

The second part of the principle is almost as important: software on other hosts may contain deficiencies that make it unwise to exploit legal but obscure protocol features. It is unwise to stray far from the obvious and simple, lest untoward effects result elsewhere. A corollary of this is “watch out for misbehaving hosts”; host software should be prepared, not just to survive other misbehaving hosts, but also to cooperate to limit the amount of disruption such hosts can cause to the shared communication facility.

- To be completely honest, I can’t quite understand what this section is talking about, but it sounds correct. I assume it means that hosts should be able to handle deficiencies of other hosts so as to not be disrupted by a defect. All hosts should be able to handle exceptions caused by other hosts.