Application Firewalls

Moving Up the Stack

- Why move up the stack?
- Apart from the limitations of packet filters discussed last time, *firewalls are inherently incapable of protecting against attacks on a higher layer*
- IP packet filters (plus port numbers. . .) can't protect against bogus TCP data
- A TCP-layer firewall can't protect against bugs in SMTP
- SMTP proxies can't protect against problems in the email itself, etc.

Advantages

- Protection can be tuned to the individual application
- More context can be available
- You only pay the performance price for that application, not others

Disadvantages

- Application-layer firewalls don't protect against attacks at *lower* layers!
- They require a separate program per application
- These programs can be quite complex
- They may be very intrusive for user applications, user behavior, etc.

Example: Protecting Email

- Do we protect inbound or outbound email? Some of the code is common; some is quite different
- Do we work at the SMTP level (RFC 2821) or the mail content level (RFC 2822)?
- What about MIME?
- (What about S/MIME- or PGP-protected mail?)
- What are the threats?

Email Threats

Application Firewalls
Moving Up the Stack
Advantages
Disadvantages
Example: Protecting Email
Email Threats
Inbound Email
Different Sublayers
Outbound Email
Combining Firewall Types
Firewalling Email
Enforcement
Outbound Email
The DNS
Application Proxies
Circuit Gateways
Personal and Distributed Firewalls
The Problems with Firewalls

- The usual: defend against protocol implementation bugs
- Virus-scanning
- Anti-spam?
- Javascript? Web bugs in HTML email?
- Violations of organizational email policy?
- Signature-checking?

Inbound Email

- Email is easy to intercept: MX records in the DNS route inbound email to an arbitrary machine
- Possible to use “*” to handle entire domain
- Example: DNS records exist for `att.com` and `*.att.com`
- Net result: all email for that domain is sent to a front end machine

Different Sublayers

- Note that are are multiple layers of protection possible here
- The receiving machine can run a hardened SMTP, providing protection at that layer
- Once the email is received, it can be scanned at the content layer for any threats
- The firewall function can consist of either or both

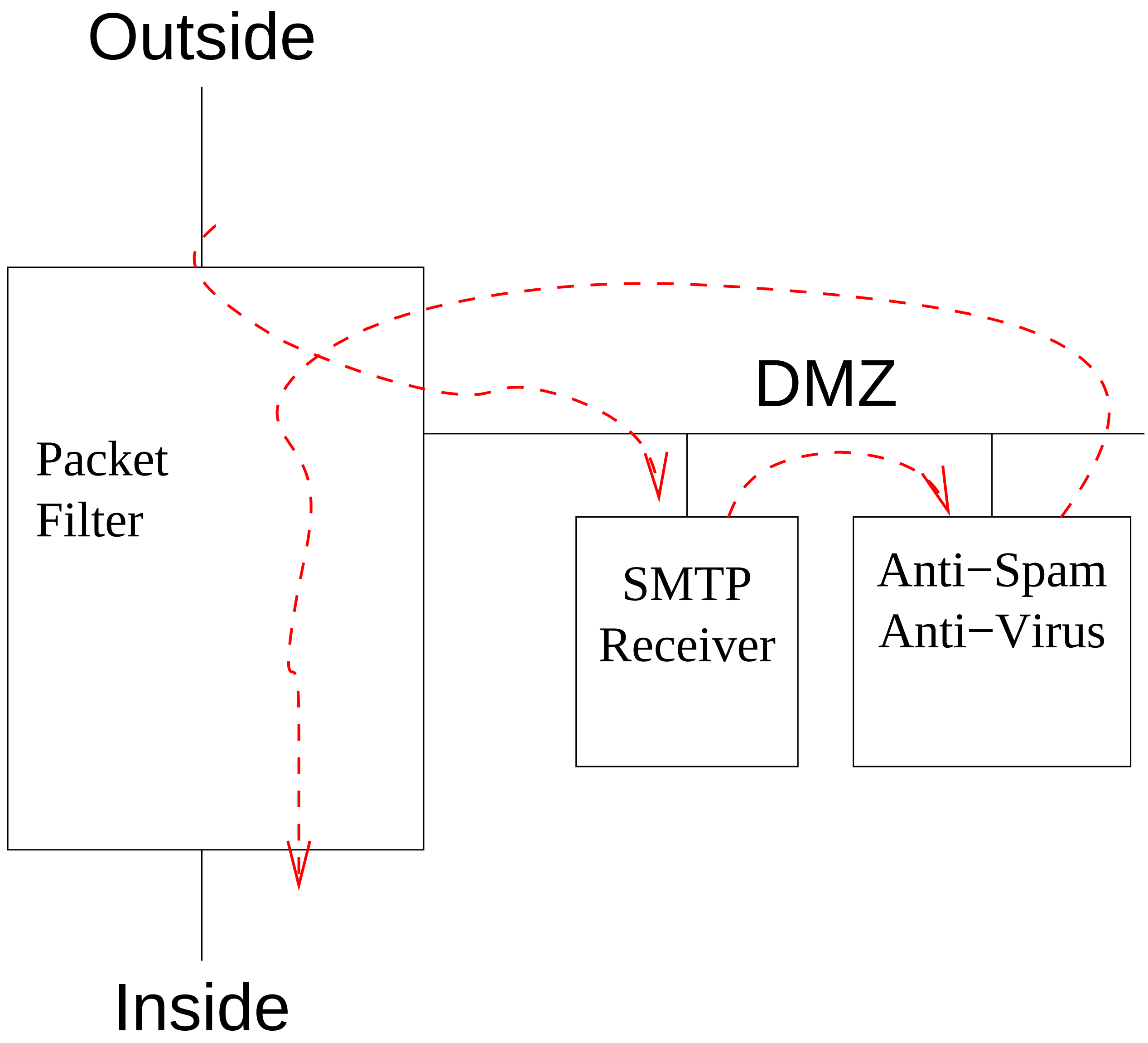
Outbound Email

- No help from the protocol definition here
- But — most MTAs have the ability to forward some or all email to a relay host
- Declare by administrative fiat that this must be done
- (Remember: in a large organization, some groups will run their own MTA.)
- Enforce this with a packet filter. . .

Combining Firewall Types

- Use an application firewall to handle inbound and outbound email
- Use a packet filter to enforce the rules

Firewalling Email



Enforcement

- Email can't flow any other way
- The only SMTP server the outside can talk to is the SMTP receiver
- It forwards the email to the anti-virus/anti-spam filter, via some arbitrary protocol
- That machine speaks SMTP to some inside mail gateway
- Note the other benefit: if the SMTP receiver is compromised, it can't speak directly to the inside

Outbound Email

- Again, we use a packet filter to block direct outbound connections to port 25
- The only machine that can speak to external SMTP receivers is the dedicated outbound email gateway
- That gateway can either live on the inside or on the DMZ

The DNS

DNS Issues

- UDP (discussed previously)
- Internal versus external view
- DNS cache corruption
- Optimizing DNSSEC checks

UDP Issues

- Remember the DNS server location discsussed last time
- In fact, what we did there was use an application-level relay to work around packet filter restrictions
- We're lucky — since the DNS protocol includes provision for recursion, it requires no application changes for this to work

Internal Versus External View

- Should outsiders be able to see the names of all internal machines?
- What about `secretproject.foobar.com`?
- Solution: use two DNS servers, one for internal requests and one for external request
- Put one on each side of the firewall
- Issue: which machine does the NS record for `foobar.com` point to, the inside or the outside server?
- Can be trickier than it seems — must make sure that internal machines don’t see NS records that will make them try to go outside directly

Cache Contamination Attacks

- DNS servers cache results from queries
- Responses can contain “additional information” — data that may be helpful but isn’t part of the answer
- Send bogus DNS records as additional information; confuse a later querier

DNS Filtering

- All internal DNS queries go to a *DNS switch*
- If it's an internal query, forward the query to the internal server or pass back internal NS record
- If it's an external query, forward the query to outside, but:
  - Scrub the result to remove any references to inside machines
  - Scrub the result to remove any references to any NS records; this prevents attempts to go outside directly
- Use a packet filter to block direct DNS communication

Application Proxies

Small Application Gateways

- Some protocols don't need full-fledged handling at the application level
- That said, a packet filter isn't adequate
- Solution: examine some of the traffic via an application-specific proxy; react accordingly

FTP Proxy

- Remember the problem with the PORT command?
- Scan the FTP control channel
- If a PORT command is spotted, tell the firewall to open that port temporarily for an incoming connection
- (Can do similar things with RPC — define filters based on RPC applications, rather than port numbers)

Attacks Via FTP Proxy

- Downloaded Java applets can call back to the originating host
- A malicious applet can open an FTP channel, and send a PORT command listing a vulnerable port on a nominally-protected host
- The firewall will let that connection through
- Solution: make the firewall smarter about what host and port numbers can appear in PORT commands. . .

Web Proxies

- Again, built-in protocol support
- Provide performance advantage: caching
- Can enforce site-specific filtering rules

Circuit Gateways

Circuit Gateways

- Circuit gateways operate at (more or less) the TCP layer
- No application-specific semantics
- Avoid complexities of packet filters
- Allow controlled inband connections, i.e., for FTP
- Handle UDP
- Most common one: SOCKS. Supported by many common applications, such as Firefox and Pidgin.

Application Modifications

- Application must be changed to speak the circuit gateway protocol instead of TCP or UDP
- Easy for open source
- Socket-compatible circuit gateway libraries have been written for SOCKS — use those instead of standard C library to convert application

Adding Authentication

Application Firewalls
The DNS
Application Proxies
Circuit Gateways
Circuit Gateways
Application Modifications
Adding Authentication
Personal and Distributed Firewalls
The Problems with Firewalls

- Because of the circuit (rather than packet) orientation, it's feasible to add authentication
- Purpose: extrusion control

Personal and Distributed Firewalls

Rationale

- Conventional firewalls rely on topological assumptions — these are questionable today
- Instead, install protection on the end system
- Let it protect itself

Personal Firewalls

- Add-on to the main protocol stack
- The "inside" is the host itself; everything else is the "outside"
- Most act like packet filters
- Rules can be set by individual or by administrator

Saying “No”, Saying “Yes”

- It’s easy to reject protocols you don’t like with a personal firewall
- The hard part is saying “yes” safely
- There’s no topology — all that you have is the sender’s IP address
- Spoofing IP addresses isn’t that hard, especially for UDP

Application-Linked Firewalls

- Most personal firewalls act on port numbers
- At least one such firewall is tied to applications — individual programs are or are not allowed to talk, locally or globally
- Pros: don't worry about cryptic port numbers; handle auxiliary ports just fine
- Cons: application names can be just as cryptic; service applications operate on behalf of some other application

Distributed Firewalls

- In some sense similar to personal firewalls, though with central policy control
- Use IPsec to distinguish "inside" from "outside"
- Insiders have inside-issued certificates; outsiders don't
- Only trust other machines with the proper certificate
- No reliance on topology; insider laptops are protected when traveling; outsider laptops aren't a threat when they visit

The Problems with Firewalls

Problems

- Corrupt insiders
- IPsec versus Firewalls
- Connectivity
- Laptops
- Evasion

IPsec versus Firewalls

- Suppose hosts routinely use IPsec to talk to the outside world.
- An inbound, ESP-protected packet arrives at the firewall.
- Should it be allowed in? Does it conform to security policies?
- The destination port number is encrypted. The ACK flag is encrypted. It might even be a tunnel mode packet.
- There is no way to for the firewall to make a decision!

Corrupt Insiders

- Firewalls assume that everyone on the inside is good
- Obviously, that's not true
- Beyond that, active content and subverted machines mean there are bad actors on the inside

Connectivity

- Firewalls rely on topology
- If there are too many conections, some will bypass the firewall
- Sometimes, that's even necessary; it isn't possible to effectively firewall all external partners
- A large company may have hundreds or even thousands of external links, most of which are unknown to the official networking people

Laptops

- Laptops, more or less by definition, travel
- When they're outside the firewall, what protects them?
- At one conference, I spotted at least a dozen other attendee machines that were infected with the Code Red virus
- (Code Red only infected web servers. Why were laptops running web servers?)

Evasion

- Firewalls and firewall administrators got too good
- Some applications weren't able to run
- Vendors started building things that ran over HTTP
- HTTP usually gets through firewalls and even web proxies. . .