Network Security 13: IDS and Apps

i.g.batten@bham.ac.uk

Recap on Firewalls

- Firewalls look at packet headers and block/pass based on protocols, sources and destinations
- Help implement policy on access to applications and keep out random probing, but have limited benefits for well-run networks

Recap on Proxies

- Terminate connections, process through protection mechanism, run new connection to service
- Various approaches for encryption
- Protect against protocol-level attacks
- Require complex new code for each new protocol, or alternatively a pass-through which misses the point
  - Look up SOCKS5

Recap on Malware Protection

- Multiple means of operation:
  - Signature-based, looking for patterns in files
  - Heuristic-based, looking for virus-like behaviour in files
  - Behaviour-based, monitoring the actual behaviour of running programs (cf. Apple sandboxes)

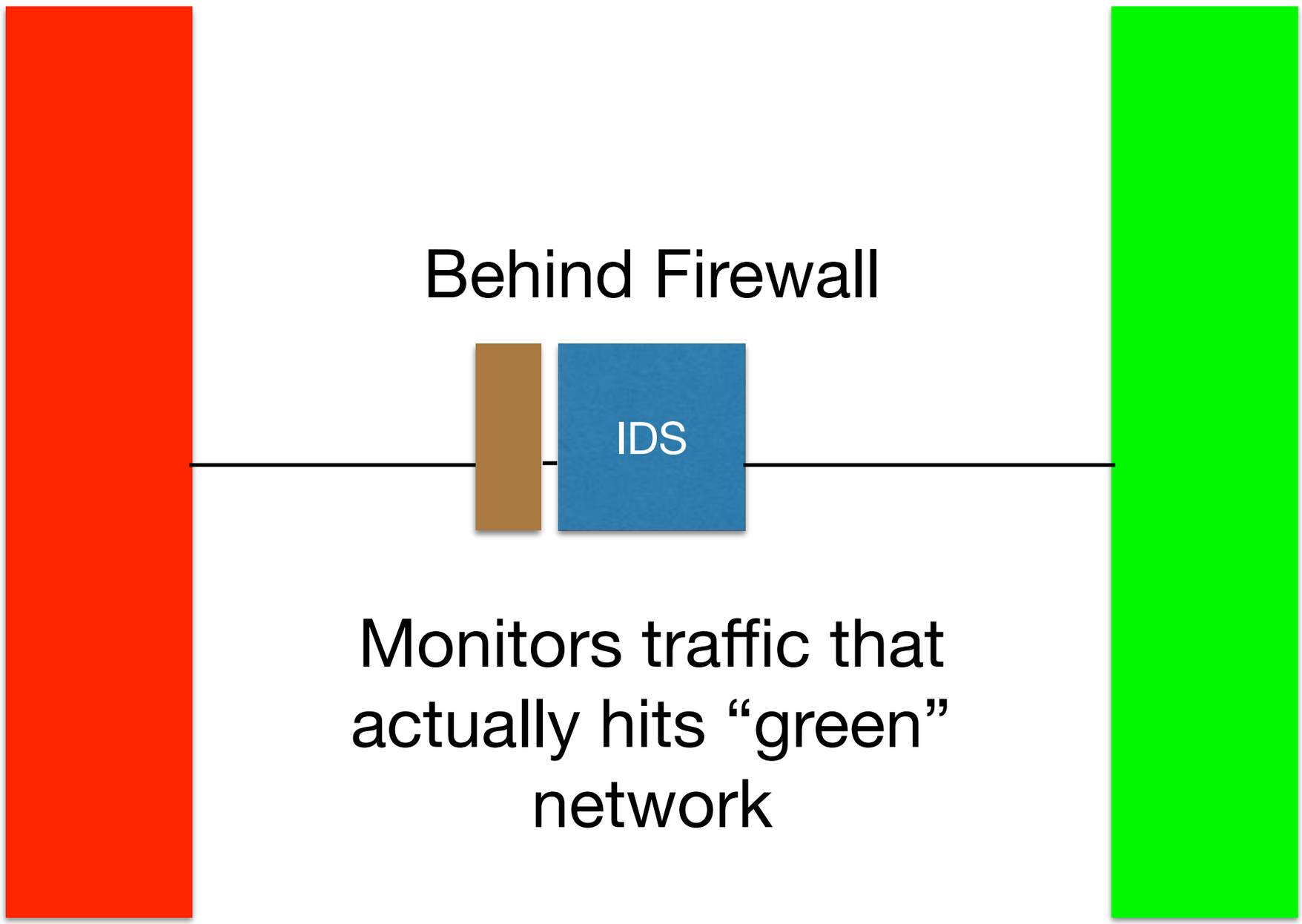
IDS Concept

- Intrusion Detection System
  - Pass all traffic through a system which treats network flows as target for malware analysis
  - Has access to headers, payloads, timing, source and destination…

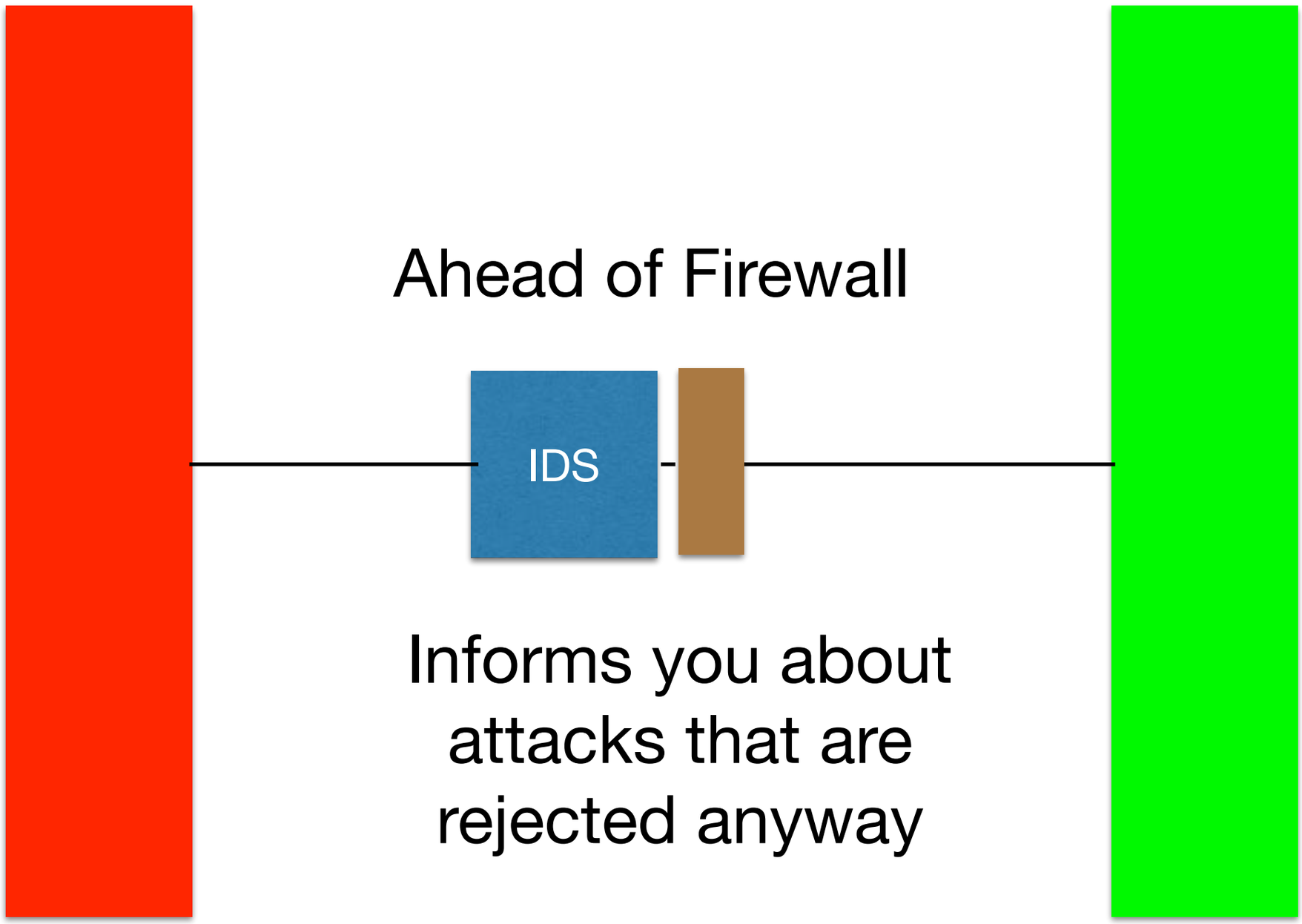
IDS Triggers

- Patterns in packets: matching payloads of packets against signatures of known malware or other attacks
- Behaviour: matching against known patterns of malware behaviour (probe this, probe that, report to the other)
- Heuristic: matching against changes in network performance, or obviously dubious activity (lots of failed connections, say)

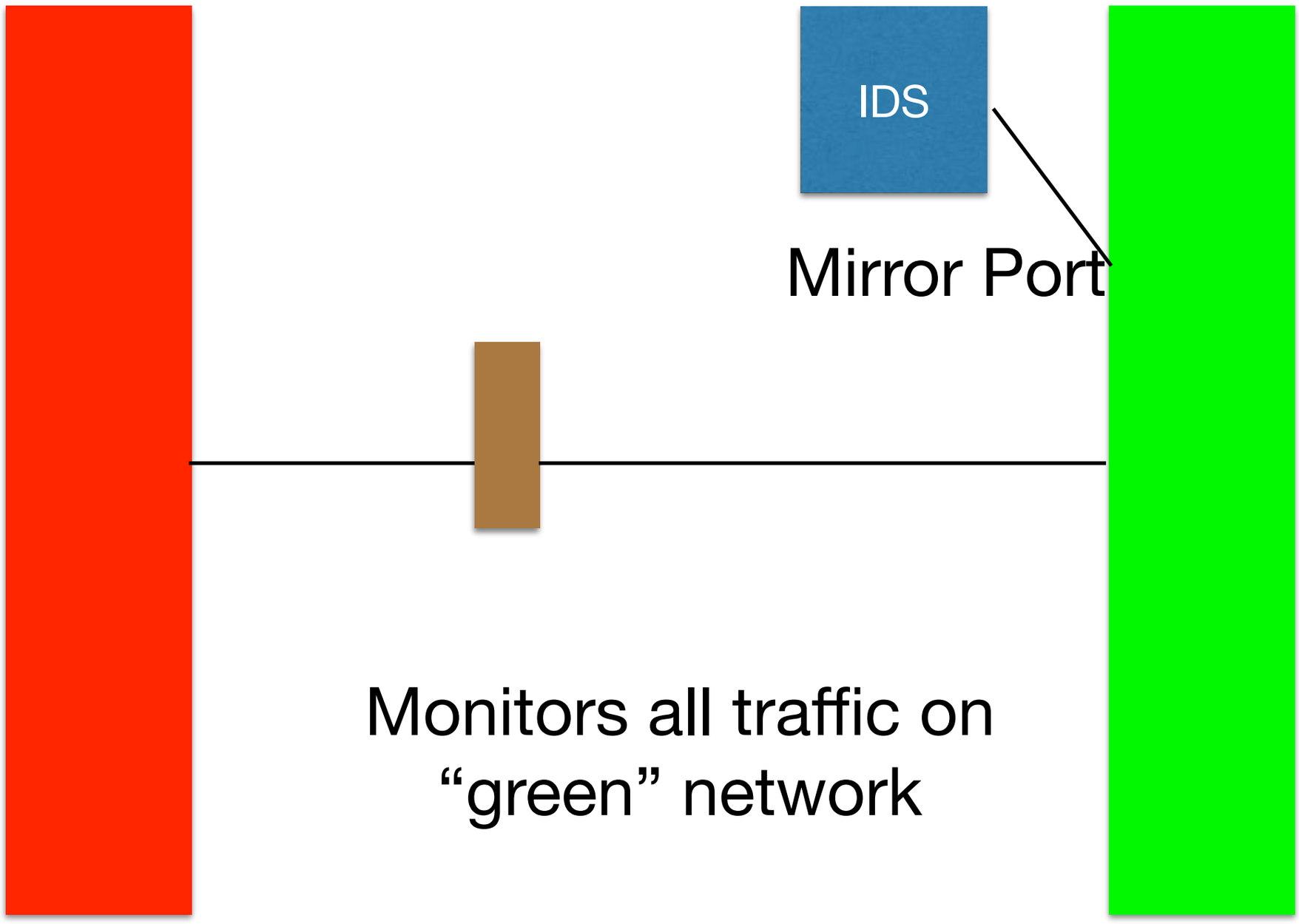
Obvious Deployment



Research Deployment



Ideal Deployment



Usually “Mirror”

- Typical switch has facility for “mirror” port that receives traffic to and from all other interfaces
- Meant for debugging and monitoring
- Obvious issues over performance: some buffering to cope with bursts, but will drop traffic under load
  - Switches rarely have deep buffering, and mirror ports don’t normally have extra buffers

IDS: Drinking from Firehose

- IDS therefore receives all traffic, including unicast between stations on network
- Modern fast processor can handle >1Gbps stream with fairly complex matching, 10Gbps with multicore
- Architectures are often multi-stage, to discard least interesting stuff as early as possible

IDS Problems

- Match packets against range of threats
- Lots of false positives
- Many attacks old / patched / irrelevant to current infrastructure
- Heuristics against zero-day attacks triggered by new software of many types
- Highly unlikely to detect spear-phishing
- At best, another line of defence against rapidly-spreading malware and (if you are very lucky) zero-day buffer-overrun type attacks
- "We think IDS is dead. It's failed to provide enterprise value," [Gartner] says. "In order for it to survive, it has to go faster, at wire speed, and it has to solve the false-alarm problem."

Sample IDS Rule

```
alert tcp any any -> any 21 \
(flow:to_server,established; \
content:"root"; pcre:"/user\s+root/i";)
```

Two-stage rule: match “root” (fast) then confirm with regular expression

IDS Rules

- If your FTP server accepts root logins which really get root, you have massive problems, and spotting the attempts in network streams is no substitute for proper audit
  - FTP servers should run chroot() under all circumstances
- But if your FTP server doesn’t accept root logins, why do you care if someone tries?
  - Might be a sign of reconnaissance by attacker, but are skilled attackers really so obvious?
- Provides way for real attackers to drown you in noise

IDS Rules

- Lots of rules are fragments of known attacks, often buffer overruns and the like
- Trivially easy for attackers to mutate
- If you have time to update rules file, you have time to patch or mitigate the actual problem

IDS->IPS

- Can be in-line (responds faster) or mirror port (easier to deploy)
- Re-programs firewalls or switches to close off attack
- All the same false positive problems, but now automated
  - Less work, or more trouble?

Lots of products

- Snort (free)
- Tipping Point ($$$)
- Many, many others
- I would be very interested to see a mature, site-wide, well-tuned implementation
  - Suspect they’re turned off before they are finished, but welcome counter-examples

Research Topics

- Machine learning on protocols, traffic patterns, etc
- Spot “normal” behaviour, trigger on “abnormal”
  - Probably involves so much hysteresis that response time would be in hours
- Lots of research, lots of adverts, not a lot of big deployments
- Plenty of potential for a PhD

Research Topics

- Legal and ethical issues over scanning all packets
  - Inside an enterprise it's all OK
  - Still some debate about legality of ISPs scanning email for viruses and spam
  - "Deep Packet Inspection" at ISP level relies on fine distinctions about whether algorithms constitute "reading".
- Machine learning based on S/Flow data very interesting

HIDS

- Host Intrusion Detection
  - Tripwire
  - OSSEC

Tripwire

- Take MD5 / SHA1 hashes of critical files
- Compare them with files on disk periodically
- Many technical problems (how do you secure the hashes, the scanner, the kernel…?)
  - VM introspection looks interesting
- But useful against naive attacks

Using Tripwire

- Boot off secure installation (CD, for example)
- Do tricky things with multi-tenant storage, NAS, etc
- Protects against some insider threats
- Also against careless installation procedures that overwrite system files

OSSEC / HIDS

- Host Intrusion Detection
- Originally developed by Trend (tier-2 virus scanner vendor) as open source with their support
- Now open source without their support (pretty much)

OSSEC

- Tripwire, plus log analysis, plus filtering and response
- Can, for example, update firewall after multiple bad login attempts
- Works best with live feed of logging information

OSSEC Rules

```
<rule id="5702" level="5">
    <if_sid>5700</if_sid>
    <match>^reverse mapping</match>
    <regex>failed - POSSIBLE BREAK</regex>
    <description>Reverse lookup error (bad ISP or attack).</description>
  </rule>
```

```
<rule id="5703" level="10" frequency="4" timeframe="360">
  <if_matched_sid>5702</if_matched_sid>
  <description>Possible breakin attempt </description>
  <description>(high number of reverse lookup errors).</description>
</rule>
```

Host Intrusion Works, ish

- Knows the platform it's on, so focuses on attacks that are relevant (ie, not looking for Windows attacks against Unix platforms)
- Sat behind all firewalls and other access control, so only looking at abnormal events
  - Downside: has less information to work with
- Running it on small networks yields low levels of false positives after minor amounts of tuning, so might be worthwhile…
- …except has never detected anything terribly serious either…

HIDS Problems

- Automated log-reading is very tricky, and small changes to software can break it
- Small changes to messages seen as dangerous can cause them to be ignored
- You can end up needing to match everything and raising all unexpected log messages as attacks
- And we’re back to the false positive / noise debate

fail2ban, etc

- HIDS can be used to detect repeated failure of login or other access attempts, and then program firewalls to block them
- Simpler software (tcp_wrappers, fail2ban, built-in features in firewalls and daemons) can do the job just as well
- Not clear that stopping repeated login attempts at network level prevents break-ins
  - Rate limiting on individual users is good, but most such probing tries varying username/password combinations
  - Can provide mechanism for DoS attack (do _not_ lock accounts based on repeated login attempts unless you really understand the risks)
- However, will reduce noise in logs.

Firewall Friendliness

- Applications will need to live behind firewalls, IDSes and so on
- It’s important they behave sensibly and securely

Single connection

- Where possible, applications should work over a single TCP connection (performance, security, firewall states)
  - Starting multiple encrypted connections is particularly expensive
- If they need multiple connections (IMAP is an example), each one should go through the full authentication protocol or the mechanism should be very carefully examined by an expert

Client always initiates, server always listens

- The client application should be the only party to call connect(), the server should be the only party to call llisten() and accept()
- Otherwise doesn’t work via NAT or through a firewall

No port numbers or addresses in payload

- Protocols should never pass address information within their payload
  - Probably implies you're going to break rules on previous page
  - Doesn't get NAT'd, doesn't work well with firewalls

Logging

- Despite all my scepticism about IDS/HIDS/etc, it is vital that as much network activity as possible is logged (yes, I realise “as much…as possible” is a somewhat open statement).
- **Forensic Readiness** is next year’s hot topic. Even if you don’t know what to do with the logs, a forensic team might find them useful.

Logging

- Logging is one of the advantages of running network proxies: you can centralise logs of activities in places which both attackers and local users cannot easily manipulate.