Fundamentos de Segurança Informática (FSI)

2021/2022 - LEIC

Manuel Barbosa
mbb@fc.up.pt

Aula 22
Segurança de Redes 2

Camada de Transporte (TCP)

Terminação de Ligações

- No TCP não existe qualquer forma de um nó saber se um pacote veio, efetivamente, do emissor com quem pretende falar
  - basta que o pacote tenha o formato correto (endereços e número de sequência) para ser aceite
- Um ataque simples é a terminação de ligações indesejadas:
  - Enviar RST a um cliente termina a tentativa de ligação
  - Esta técnica é utilizada pela Grande Firewall da China (https://en.wikipedia.org/wiki/Great_Firewall)
- Para lançar este tipo de ataques é importante controlar (parte da) infra-estrutura

Great Firewall

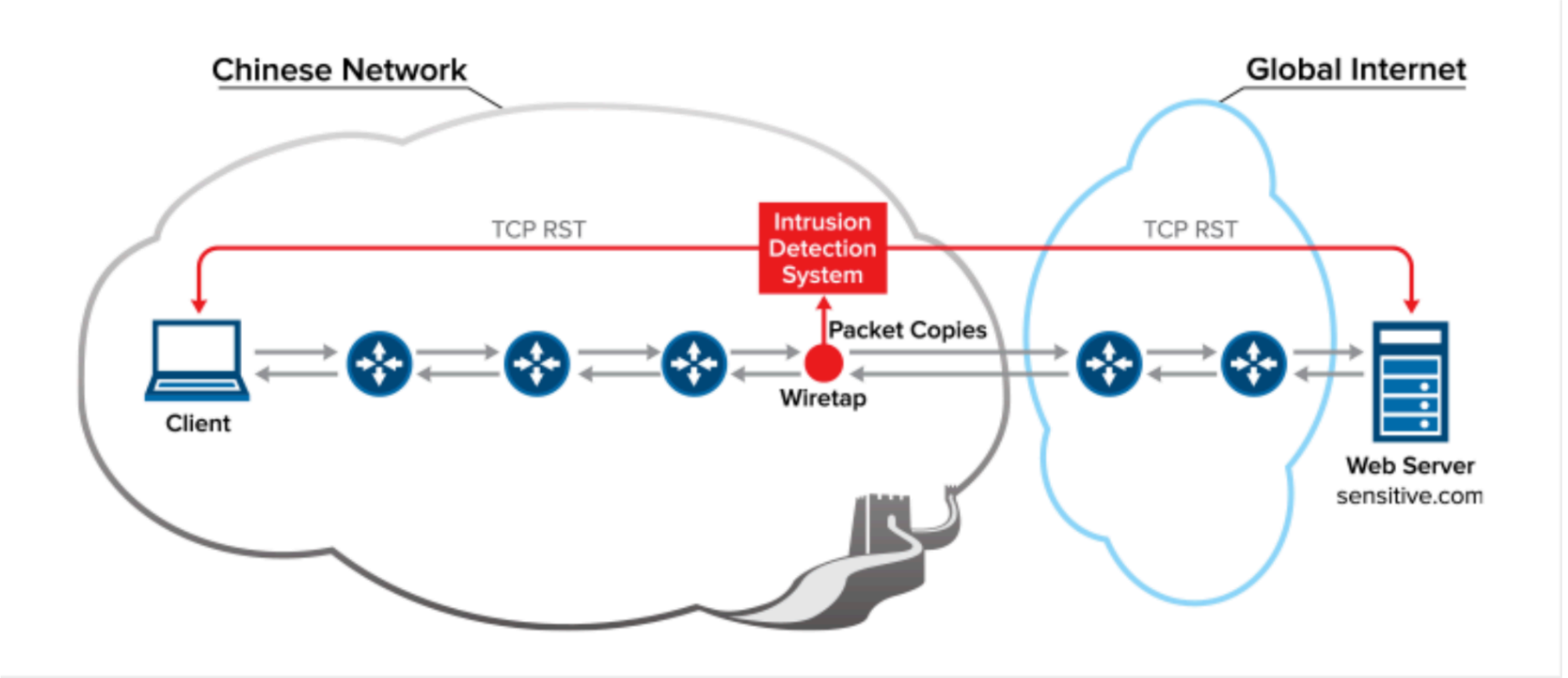


Figure 2: Filtering routers pass copies of passing traffic to out-of-band IDS devices that inspect for blacklisted keywords.
Sensitive content is blocked by injections of forged TCP resets.

https://www.thousandeyes.com/blog/deconstructing-great-firewall-china/

Spoofing às cegas (off path)

- Será que conseguimos estabelecer uma sessão em nome de uma origem que não controlamos?
  - envia-se mensagem inicial SYN
  - não conseguimos ver o número de sequência na resposta
  - adivinhamos o número de sequência (muitas vezes baseado no relógio!)
- Mitigação: usar números de sequência aleatórios

TCP Session hijacking

- Usurpação de uma sessão legítima por parte de um atacante
  - O atacante intromete-se na sessão para entregar mensagens em nome de outra entidade
- Assumindo que existe uma troca inicial que estabelece uma autenticação, o atacante já não necessita de ter credenciais
  - Faz hijacking da sessão já depois de o utilizador legítimo se ter autenticado
  - Diferente de spoofing: aí o atacante inicia a sessão em nome de um utilizador legitimo e precisaria das credenciais

TCP session hijacking

- Fases: Tracking, Des-sincronização, Injecção
- Porque é eficaz
  - Não há contra-medidas eficazes (a não ser criptografia)
  - O TCP/IP é intrinsecamente vulnerável e o ataque é simples de executar
  - Permite fazer bypass a processos de autenticação que não são efectuados/ associados sobre/a um canal seguro:
    - passwords, one-time passwords, challenge-response, kerberos sem cifração, etc.
  - Permite enganar routers/proxies que estão a filtrar sources porque os pacotes aparentam vir de fontes/sessões legítimas

Dificuldade

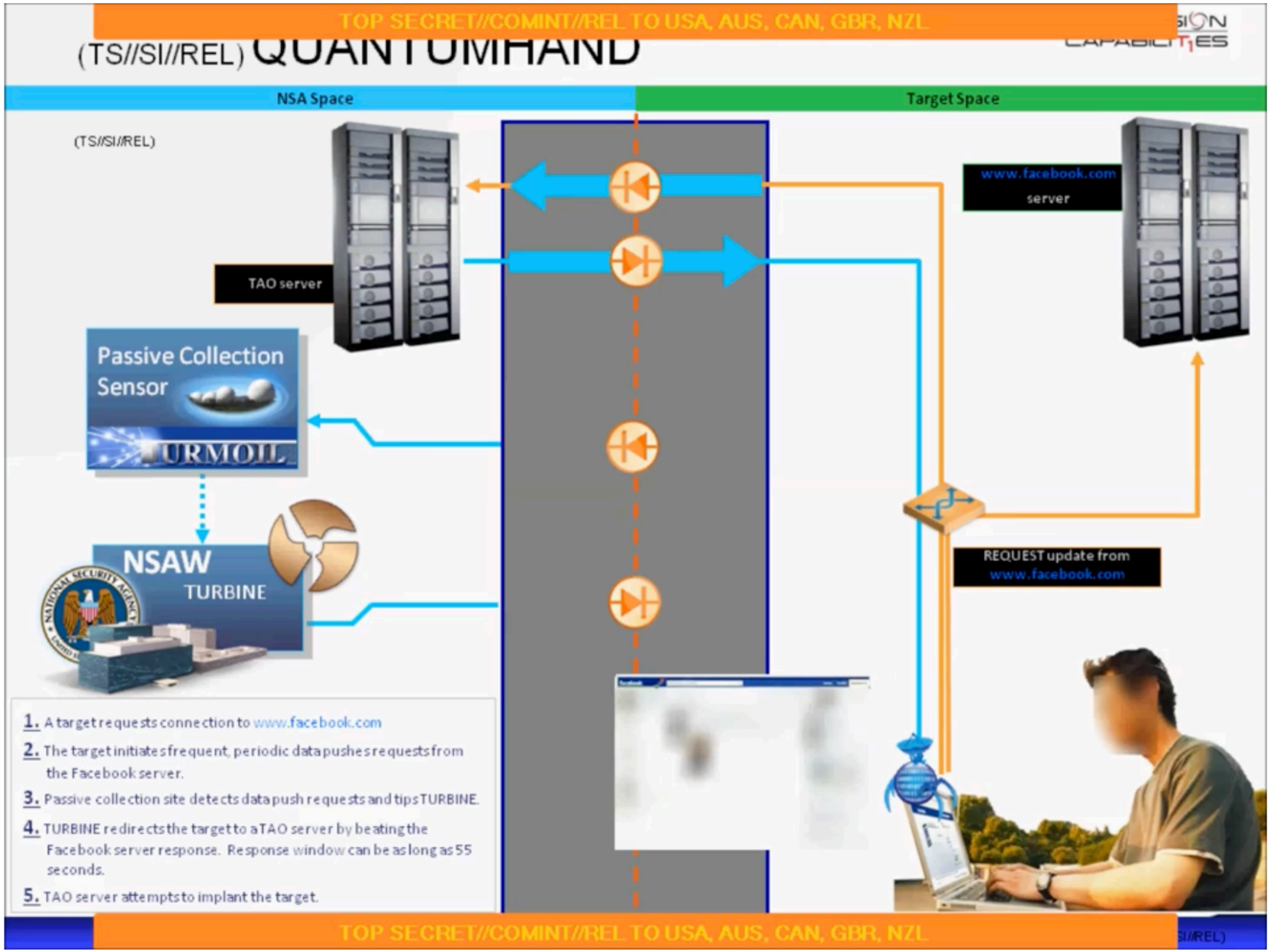
- Mais difícil que spoofing e exige capacidade de observar tráfego:
  - Esperar que alvo estabeleça sessão
  - Remover alvo do sistema, impedindo-o de comunicar
  - Tomar conta da sessão estabelecida
  - Implica ser capaz de saber tanto sobre a sessão como o alvo, incluindo os TCP sequence numbers, e tomar conta da comunicação
- Mas pode ser a única opção
  - No caso de haver credenciais secretas necessárias para autenticação

Mais em detalhe

- Sniffing para obter a informação necessária e identificar o momento de intervir
- Des-sincronizar: e.g. avançar o seq number de servidor com mensagem
  - O alvo e o servidor ficam a tentar re-sincronizar, mas não encerram a sessão
- Agora o atacante
  - sabe os números certos de sequência corretos
  - pode comunicar com o alvo entregando-lhe mensagens e vendo respostas
  - ou mesmo tentar um ataque Man-in-the-Middle

UDP Hijacking

- No UDP não há controlo de tráfego como no TCP
- Um atacante consegue facilmente interferir na transferência, e tentar responder primeiro que os pares
- Utilizando técnicas MiM o ataque torna-se mais simples: não é necessário ser mais rápido na resposta



https://theintercept.com/2014/03/12/nsa-plans-infect-millions-computers-malware/

SECONDDATE

- SECONDDATE is an exploitation technique that takes advantage of web-based protocols and man-in-the-middle (MitM) positioning.
- SECONDDATE influences real-time communications between client and server and can quietly redirect web-browsers to FA servers for individual client exploitation.
- This allows mass exploitation potential for clients passing through network choke points, but is configurable to provide surgical target selection as well.

https://theintercept.com/2014/03/12/nsa-plans-infect-millions-computers-malware/

Conclusão

- Os protocolos de rede não são secure by design: segurança é adicionada à posteriori
- As soluções criptográficas permitem resolver o problema em parte:
  - segurança entre pontos de terminação (endpoints)
    - camada de rede (IPSec) => E.g., ligar dois routers diretamente um ao outro
    - camada de transporte (TLS) => todas as ligações HTTPS, próxima aula
    - camada de aplicação (e.g., Signal, Whatsapp) => próxima aula
- não evitam ataques à infra-estrutura, metadados, DoS, etc.
  - evidência de que estes ataques são explorados na prática em grande escala

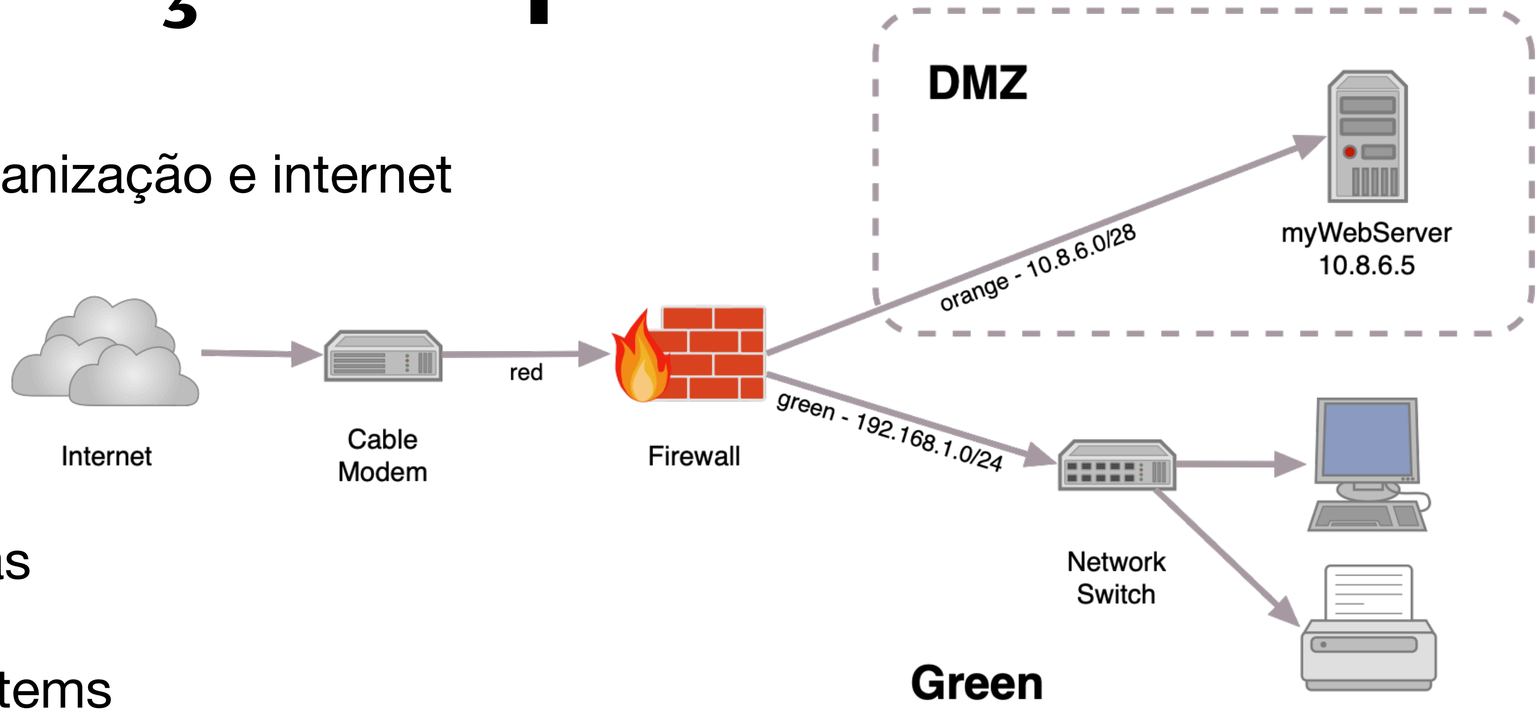
Defesas

Problema

- Como proteger as nossas máquinas ligadas à rede?
  - Maior número de máquinas expostas => maior a superfície de ataque
  - Chega desligar todos os serviços desnecessários?
    - Difícil de fazer
      - número de máquinas, diversidade de sistemas, propósitos, utilizadores, administradores pode ser muito grande
    - Insuficiente
      - definição pouco clara da superfície de ataque
      - não permite defesa em profundidade, mediação completa

Solução típica

- Definir uma fronteira clara entre organização e internet
  - Firewalls
  - Network Address Translation
  - Proxies de aplicações específicas
  - Network Intrusion Detection Systems

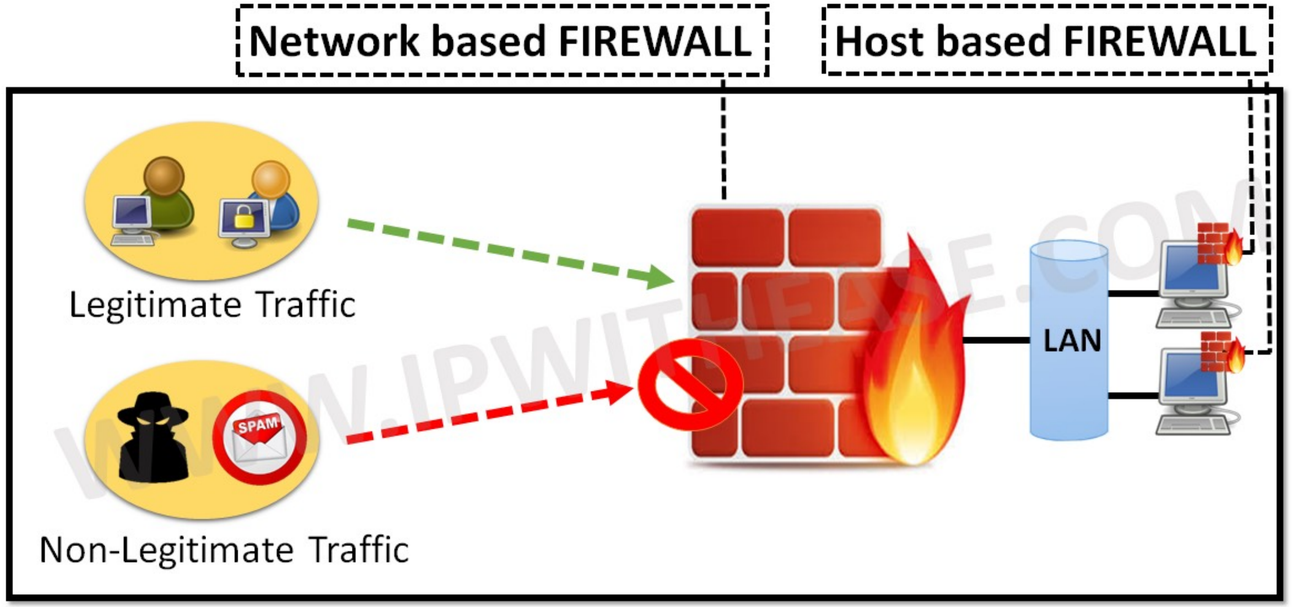


- Sempre: proteger a rede interna de ameaças externas
- Por vezes: proteger a rede interna de máquinas locais infetadas
- Tecnologia: filtrar/limitar tráfego de rede => como/onde se faz essa filtragem?

https://wiki.ipfire.org/configuration/firewall/rules/dmz-setup

Firewalls

- Firewalls locais (host-based)
  - Correm nas máquinas das aplicações/utilizadores
  - Podem incluir filtros baseados no contexto da aplicação/utilizador
- Firewalls na rede
  - Interceptam comunicações de/para muitas máquinas
  - definem fronteira com o exterior: tudo o que está fora é hostil



https://ipwithease.com/network-based-firewall-vs-host-based-firewall/

Firewalls

- Filtragem de pacotes
  - olham tipicamente para os cabeçalhos dos pacotes: endereços IP, protocolo/portas TCP/UDP
  - tentam eliminar comunicações “maliciosas”
  - protegem máquinas internas de acesso externo
  - fornecem às máquinas internas funcionalidades que exigem acesso externo
- Distinguir de proxies
  - trabalham ao nível de uma aplicação específica
  - exemplo: proxies web/HTTP

Políticas de Controlo de Acessos

- Uma firewall implementa uma política de controlo de acessos:
  - Quem fala com quem e quem acede a que serviço
- Distingue tráfego recebido de tráfego enviado
- Política simples (suficiente?):
  - permitir todos os acessos de dentro para fora
  - restringir acessos de fora para dentro:
    - permitir acesso a serviços explicitamente designados para exposição
    - negar todos os outros acessos a serviços designados como internos

Políticas de Controlo de Acessos

- Como tratar tráfego que não referido explicitamente nas políticas?
  - Default allow => permitir todos os acessos exceto problemas específicos
  - Default deny => permitir apenas alguns acessos comuns a todos os sistemas
- Default deny => escolha mais conservadora, proteção por omissão
  - começar por esta escolha e depois adicionar exceções
  - provoca sempre instabilidade inicial

Filtragem de Pacotes

- Cada pacote é verificado relativamente a regras => forward/drop
- Utiliza-se informação das camadas de rede e de transposte
  - source/destination IP, source/destination port, flags (e.g., ACK)
- Exemplo de regras:
  - bloquear propagação de DNS (porta 53) exceto de servidores conhecidos
  - bloquear pedidos de HTTPS (porta 443) exceto aqueles vindo de IPs da organização (e.g., web services vs websites)
  - bloquear endereços internos desconhecidos

Filtragem de Pacotes: Tipos

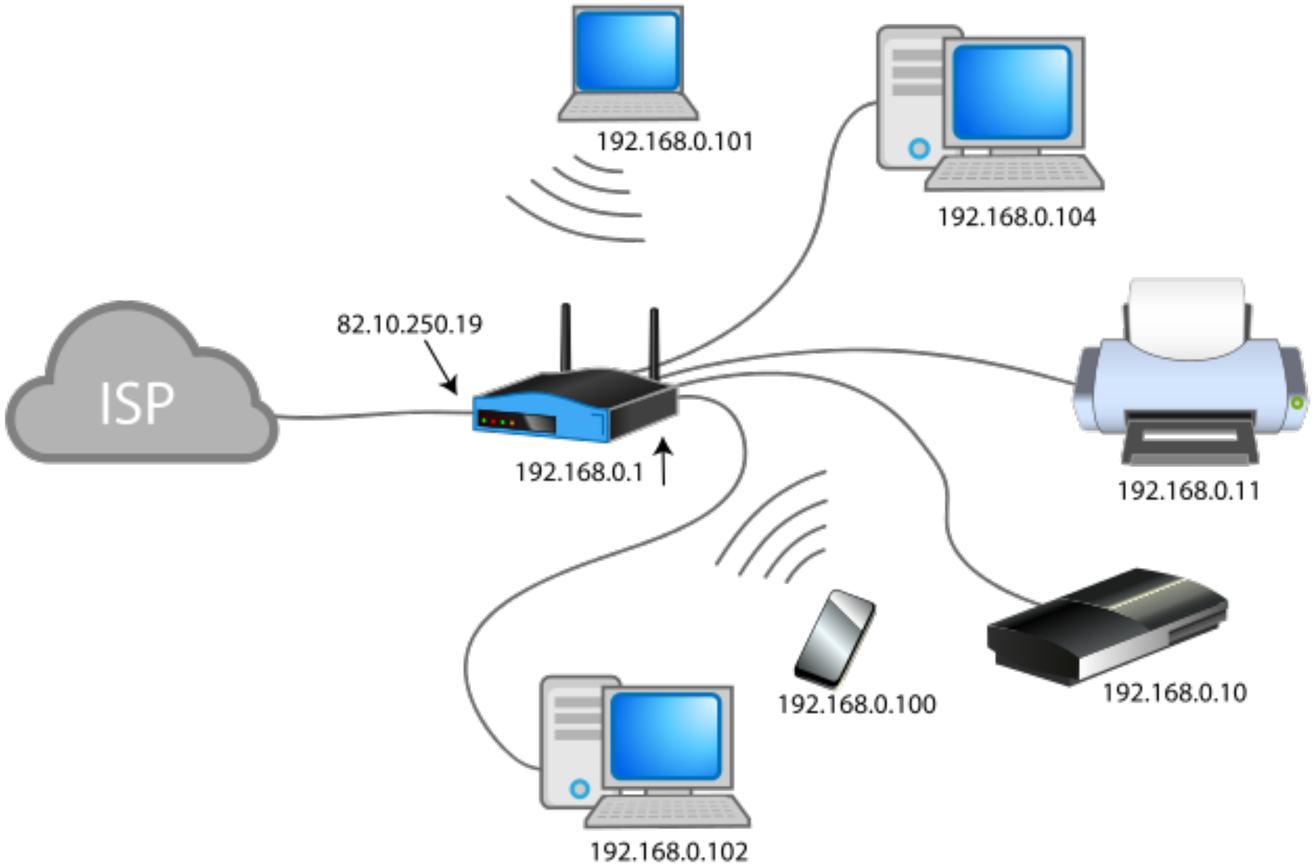
- Filtragem sem estado:
  - forma limitada de filtragem porque não tem em conta se pacotes estão no contexto de uma ligação => tem de ser mais permissiva
- Filtragem com estado:
  - tenta manter registo de ligações ativas para verificar se pacote “faz sentido” nesse contexto
  - por exemplo: se máquina interna estabeleceu ligação TCP, então permite pacotes externos que contêm a resposta
  - difícil de fazer para contextos elaborados:
  - e.g. como eliminar sequências de pacotes que fazem login como root?
    - inúmeras maneiras de segmentar os comandos em pacotes

Contornar Filtragem de Pacotes

- Isto pode ser necessário para utilizador legítimos
  - ou pode ser utilizado por adversários (e.g. internos para “exfiltrar” info)
- Padrão: usar porta geralmente usada por outro serviço
- Túneis:
  - encapsular um protocolo dentro de outro protocolo
  - exemplos: SSH, VPN, etc.

Network Address Translation

- Os endereços IP não podem ser únicos globalmente (não temos suficientes)
- NAT => tradução de endereços “globais” para endereços “locais”
- Quase todas as redes locais (e.g., em casa) usam NAT



https://en.wikibooks.org/wiki/A-level_Computing/AQA/Paper_2/Fundamentals_of_communication_and_networking/Network_Address_Translation_(NAT)

Funcionamento NAT

- Mantém tabela: (endereço IP, porta) internos => porta na rede pública
- Para mensagens enviadas para o exterior:
  - verifica se tabela contém porta pública => senão cria entrada
  - traduz (endereço IP, porta) internos => (endereço router, porta pública)
- Para enviadas recebidas pelo router do exterior
  - verifica se tabela contém porta pública => senão faz drop
  - traduz (endereço router, porta pública) => (endereço IP, porta) internos

Vantagens/desvantagens NAT

- Vantagens
  - Reduz exposição ao exterior:
    - apenas um endereço IP
    - ligações externas descartadas, a não ser que iniciadas internamente (ou configuradas estaticamente)
- Desvantagens
  - Pode perturbar o funcionamento de alguns protocolos
  - Fácil de fazer bypass para adversário ativo (este exemplo chama-se *slipstream*)
    - site do atacante corre JavaScript em cliente na rede interna
    - envia mensagens de dentro para fora que “baralham as tabelas NAT”
    - e.g., passa a ser possível aceder a outras portas na mesma máquina

https://samy.pl/slipstream/

Proxies de aplicação

- Ideia:
  - obrigar tráfego de uma ou mais aplicações a passar por proxy
  - um proxy é um man-in-the-middle “do bem” (?!?)
- Exemplos:
  - SMTP => fazer scan de virus, spam
  - SSH => log de autenticação, inspecionar texto cifrado
  - HTTP/HTTPS => bloquear URLs proibidas

Deteção/Prevenção de Intrusões

- Já vimos:
  - deteção/prevenção de malware na rede vs host-based
    (aka application-layer filtering)
  - vimos que os anti-virus são formas de deteção de malware host-based
  - mencionámos que é possível fazer IDS/IPS ao nível das comunicações
- Veremos agora como fazer IDS/IPS ao nível das comunicações mais em detalhe:
  - ideia: monitorizar tráfego e tentar identificar um ataque
    (e.g., scanning, login como root, etc.)
  - também pode ser feito host-based ou globalmente na rede

Network IDS/IPS

- Ideia:
  - manter tabela das ligações ativas e acompanhar o estado de cada uma
  - procurar padrões (parciais) como /etc/passwd
- Vantagem: não é necessário alterar máquinas individuais
- Desvantagem:
  - exigente em termos de processamento em ligações de alto débito
  - menos preciso (para reduzir falsos negativos => muitos falsos positivos)
    - como lidar com todos os ataques a todos os tipos de sistemas?
    - como lidar com todas as formas de evasão (segmentação de pacotes, codificação de strings, etc.)?
    - como lidar com código cifrado (e.g., HTTPS)? Podemos dar as chaves ao IDS?

Host-Based IDS/IPS

- Defesa em profundidade:
  - fazer deteção mais agressiva em algumas máquinas
- Exemplo: servidores web
  - é possível fazer deteção após remoção de camada criptográfica
  - é possível procurar padrões de ataque específicos para servidores web
- Não se generaliza por causa do custo de adotar esta aproximação em todas as máquinas

Análise de Logs

- Ferramentas de deteção de intrusões “baratas” (e.g., fail2ban)
  - podem ser corridas off-line, e.g., todas as noites
  - analisam a interpretação dos servidores/máquinas
    - tudo o que sejam técnicas de evasão (segmentação, escape de strings, etc.) já foram removidas
  - apenas reativo => deteção após o ataque (por vezes muito tempo)
    - não permite prevenir/bloquear ataque em tempo real
  - O atacante pode modificar os próprios logs
- Podem ser utilizados para actualizar regras de firewalls/IDS (black/ban lists) => perigo de self-block

Pen-Testing

- Ideia:
  - em vez de esperar por um ataque …
  - simular situações de ataque e ver como o sistema reage
- Opções:
  - o mínimo: ferramentas de vulnerability scanning automáticas
  - auditoria técnica feita por especialistas/ethical hackers

Pen-Testing

- Vantagens:
  - proativo: falha não causa perda de valor e permite corrigir sistema
  - optimização: permite melhorar sistema/reduzir falsos positivos
    - ataque tratado noutro local não precisa de ser monitorizado
- Desvantagens:
  - custos podem ser elevados
  - por vezes um ataque, mesmo simulado, pode causar danos (e.g., down time)

Honey-Pots

- Em combinação com os sistemas anteriores usa-se um chamariz:
  - objetivo => acelerar deteção expondo um sistema atraente para atacantes
  - entidades internas sabem que sistema não serve para nada
    - necessário disfarçar isto dos atacantes
  - qualquer tentativa de acesso é ataque ou erro
  - funciona melhor com ataques automatizados