Resumo

Introdução

- **Porquê comprometer a máquina de um user**:
  - roubar passwords
  - randomware
  - usar o CPU (bitcoins)
  - parecer um user (spam, DDOS, clicks)
- **Porquê comprometer servers**:
  - data breaches
  - motivaçes políticas e geo-estratégicas
  - supply-chain attacks
  - web-server attacks
- **Segurança** - A propriedade de um sistema que se comporta como esperado. É relativa. Define-se pela negativo. Depende do contexto.
- **Ator** - Alguém/alguma coisa que intrevém no sitema: users, processos, organizações, empresas, máquinas, etc... Atacantes são atores (externos/internos).
- Quem desenha o sistema não deve estar respnsãvel pela segurança porque está demasiado focado em features.

Segurança

- **Modelo – binário** (prova de segurança):
  - definir formalmente capacidades dos atacantes X
  - definir formalmente objetivos de segurança Y
  - nenhum atacante limitado a X consegue quebrar Y
  - **limitações**: não escala para sistemas complexos; os modelos formais podem estar errados (side-channels)
- **Modelo – Gestão de Risco** (resiliência, mitigação, risco):
  - típico em engenharia de software e segurança no mundo real;
  - minimizar risco em função das ameaças mais prováveis;
  - otimizar o custo das medidas de segurança vs. potenciais perdas;
  - **limitações**: análise de risco pode estar errada; uma ameaça mal classificada pode deitar tudo por terra.
- **Ativo** - recurso que detem valor para um ator do sistema: informação; reputação/image; dinheiro/recurso que vale money; infra-estrutura.
- **CIA** - risco de perda de valor por quebra de:
  - **Confidencialidade** (segredo, privacidade);
  - **Integridade** (não alteração, dados fidedignos, autenticidade de origem);
  - **Disponibilidade** (existência, consistencia).
- **Matriz de análise de risco** - a ação a tomar depende do **potencial impacto** e da **probabilidade**.
- **Vulnerabilidade** - falha que está acessível a um adversário que poderá ter a capacidade de a explorar.
- **Ataque** - ocorre quando alguém tenta explorar um vulnerabilidade (passivos, ativos, denial-of-service). Quando é bem sucedido o sistema diz-se **comprometido**. Motivação/ameaça + vulnerabilidade + método/exploit.
- **Modelo de ameaças**:
  - o que queremos proteger
  - de quem/do que queremos proteger
  - que tipo de ataques temos de precaver
  - que ataques podemos descartar
  - perímetro de segurança
  - superfície de ataque
- **Política de segurança** - um conjunto de processos/mecanismos que devem ser seguidos para garantir segurança num determinado modelo de ameaças: política de passwords, política de emails, etc...
- **Mecanismo de segurança** - método/ferramenta/procedimento que permite implantar uma **política de segurança**: identificação/autenticação, controlo de acessos, criptografia, controlos físicos, e auditorias.
- **Confiabilidade** - Análise de requisitos de segurança + definir modelo de ameaças + definir modelo de confiança + definir solução (politicas de segurança) + validar a solução.

Controlo

```
-- high addr
--- stack (v)
...
```

```
--- heap  (^)
--- uninitialized data(bbs)
--- initialized data
--- text
-- low addr
```

- Frame contém desde frame pointer da anterior até return address da próxima.
- **Stack Smashing** - é o buffer overflow.
- **Overflows na Heap** - apontadores para funções (virtual function tables e global offset tables para dynamic linking); exception handlers; longjmp buffers.
- **Heap spraying** - injetar o codigo malicioso em vários pontos da heap.
- **Use after free** - usar instancia de class destruida.
- **Int overflow** - perda de informação por misrepresentation de int pelo CPU.
- **Strings de formatação** - passam a string para o printf feitos burros.
- Reutilizar bibliotecas: libc tem system e mprotect.
  - **ROP** - Return-oriented programming com gadgets da libc

Defesas

- **defesa em profundidade** - ter varias camadas de proteção.

previnir usurpação de controlo

- **DEP** - Data Execution Prevention - W ^ X - pode ser hw (NX bit) ou SW.
- **ASLR** - Address Space Layout Randomization
- **PIE** - position-independent executable - tornam ROP hard
- **KASLR** - apenas muda de boot para boot; sabendo addr base pode prever-se of outros.
  - **KARL** - o próprio código do kernel é randomized.

deteção em tempo de execução

- Compilador adiciona código de deteção de potencial ataque.
- Transforma execução arbitrária de código em DoS.
- **Stack Canaries**:
  - pode ser randomized a cada execução
  - podem user carateres de terminação para string.h chorar
- **Bypass canários**:
  - escrita para apontador de var local
  - programa recebe apontadores de funções como arg
  - aprender canário usando outra vulnerabilidade
  - brute-force com forks
- **Extra mitigations**:
  - garantir que buffers estão sempre junto ao canário.
  - copiar args para o todo da stack.
  - **Shadow stack** - redundante; tem frame pointer e return addr; verifica-se consistencia no return;
  - ter dangereous types numa stack à parte;

Memory tagging

- suporte hardware para criar tags
- permite ligar pointers a regiao para onde apontam
- comparar tags no acesso (overflows) e free altera tags (use after free)

Control Flow Integrity (CFI)

- Durante compilação verificar pontos de origem para cada destino válido.
- Durante exec verificar consistencia com essa informação.
- Só é necessário proteger saltos/returnos dinamicos (pointers para funcoes).

Segurança de Sistemas

- **Economia nos Mecanismos** - KISS
- **Proteção por omissão** - por default de import um nível de proteção conservador
- **Fail closed** - em caso de falhar ser conservador.
- **Desenho aberto** - arquitetura de segurança e detalhes de funcionamento devem ser publicos. Segredos sao parametros de sistema que podem ser alterados.
- **Defesa em profundidade** - todos os mecanismos de segurança podem falhar. Nao depositar a confiança num só mecanismo.
- **Privilegio Minimo** - cada user/container/programa deve ter apenas os previlegios/permissoes essenciais para desempenhar a sua funcao.
- **Separação de privilegios** - deve haver isolamento entre recursos.
- **Mediacao completa** - para todos os recursos ter uma política de proteção e validar cada acesso com essa política.

Controlo de Acessos

- Ator + ((Recurso + Operação)=permissão).
- Matriz de acessos (sum)
  - Lista de controlo de acessos (ACL) (recurso)
  - Lista de permissões (ator)
- Role Based Access Control: ligar roles a permissoes e atores a roles.
- Attribute-based Access Control: atores e recursos tem atributos; descreve permissoes com base nos atributos; recurso com A acedido por ator com B.

Modelo de confiança

Confiável - sistema faz exatamente (e apenas) aquilo que foi especificado.

- é indutiva;
- o BIOS e Kernel apos instalacao sao confiaveis;
- boot coloca kernel em memoria colocando o pc num estado confiavel;
- nenhum novo processo pode alterar o estado de confiança;
- hibernacao preserva estado de confiança;

Medidas de Mitigação

- Assinaturas digitais;
- Monitorização para detetar burros;
- Processo nao pode aceder ao espaço de mem de outro proc
- CIA e controlo de fluxo do kernel tem de ser de todos os processos que executam em user mode.

Kernel Mapping - quando um proc faz uma system call não é necessário alterar o sistema de mapeamento de páginas:

- a mem relevante para o kernel já está mapeada;
- coexistem no mesmo addr space
- quando se muda de proc de user as tabelas de paginas sao diferentes mas a da kernel mem sao as mesmas.
- **Permissoes de kernel sobre a mem de outros procs** - Impedir o kernel de violar W^X para impedir fugas de info/codigo malicioso no caso de kernel corrompido (can't write).

File system

- **EUID** (determina as permissões), **RUID** (utilizador que lançou o proc), **SUID** (utilizado em transicoes);
- seteuid altera apenas EUID;
- setuid (not root) restaura EUID para RUID ou SUID;
- **perigo**: usar seteuid quando se pretende alteracao permanente (podemos voltar).

Confinamento

reference monitor - onmipresente; media todos os pedidos de acesso a recursos; simples.

SFI + SCI

- **Software Fault Isolation** - isolamento de procs que partilham o mesmo espaço de endereçamento.
  - Limitar zone de memória acessível a uma app;
  - Operações perigorsas precedidas de guardas: load/store e saltos;
- **System Call Interposition** - mediação de todas as system calls. Concentrar acessos num número pequeno de pontos que podem ser monitorizados.
- e.g.: virtualizar o espaço de endereçamento e monitorizar os acessos nos mecanismos de tradução de endereços; separação Kernel vs Userland.

Containers

- **seccomp** - Secure Computing Mode - chama prctl e entra em secure mode; só pode terminar/retornar ou utilizar ficheiros já abertos; uma violação leva o kernel a terminal o proc.
- **seccomp+bpf** - configurar de forma mais fina as system calls. Pode ter vários filtors.

Malware

- **virus** - código que se propaga criando um contexto em que eventualmente será executado;
- **worm** - código que se propaga autonomamente e consegue criar sozinho as condições para se executar e propagar;
- **rootkit** - código desenhado para esconder a sua presença e permitir acesso com privilégios elevados a um atacante;
- **trojan** - código que aparenta ser legitimo mas tem como obj transmitir info para um atacante.

Botnets

- **architecture**:
  - centralizada (com multiplos servidores centrais);
  - peer-to-peer, auto organizada (worker vs proxy), hierarquica.
- **fluxo**:
  - push - server envia comandos;
  - pull - bot pergunta se ha comandos.
- recuperacao/resiliencia no caso de ataque ao controlador (procurar novos servers);
- **deteção**:
  - detetar o malware na maquina;
  - detetar trafego de rede;
  - export honeypots para serem comprometidas e monitorizadas.
- **combate**:
  - limpar maquinas comprometidas e isola-las;
  - isolar/desligar o C2;
  - tomar o controlo do C2 e usa-lo para desativar botnet.

NIDS vs HIDS

- **NIDS**:
  - unico sistema protege N sistemas diversos;
  - simples de gerir e instalar;
  - nao ocupa recursos de sistemas em producao;
  - mais dificil acesso para os atacantes.
- **HIDS**:
  - tem acesso direta a semantica da atividade maliciosa:
    - observa os defeitos do ataque;
    - mais dificil de contornar pk é mais preciso.
  - protege de ameacas que nao vem da rede (USB);
  - é possivel observar dados que estao cifrados na rede;
  - nao afeta o sistema todo (é possível configurar de mais fina).