Segurança de Sistemas e Dados — Guia de Estudo Completo

Índice

Aula 1 — Autenticação & Passwords

Conceito de Autenticação

- **Autenticação de utilizador** é o processo de verificar a identidade de um utilizador.
- Tem **dois passos**: identificação (fornecer identificador) + verificação (ligar a entidade ao identificador).
- É a base do controlo de acesso e da responsabilização.

4 Meios de Autenticação

#	Baseado em...	Exemplo
1	**Algo que sabe**	Password, PIN
2	**Algo que possui**	Chave, token, smartcard
3	**Algo que é** (biometria estática)	Impressão digital, retina
4	**Algo que faz** (biometria dinâmica)	Voz, assinatura

Passwords — Problemas e Ataques

- **Vulnerabilidades**: ataque de dicionário offline, ataque a conta específica, ataque a password popular, adivinhar contra um utilizador, hijacking de workstation, monitorização eletrónica.
- **Contramedidas**: bloquear acesso ao ficheiro de passwords, deteção de intrusão, bloqueio de conta, políticas contra passwords comuns, formação, logout automático, ligações de rede encriptadas.

Armazenamento de Passwords — Hash + Salt

- **Nunca** guardar passwords em texto simples.
- Guardar `y = h(password)` — hash da password.
- **Problema**: Trudy pode fazer forward search (ataque de dicionário pré-computado).
- **Solução: Salt** — escolher salt aleatório `s`, guardar `(s, h(password, s))`.
  - Salt não é secreto, mas força Trudy a recalcular o dicionário para cada utilizador.
- **Implementação UNIX original**: salt de 12 bits, DES, 25 iterações — hoje considerada insegura.
- **Melhorias modernas**: MD5/SHA-1/SHA-2 com salt de 48 bits, 1000 iterações; **Bcrypt** (OpenBSD) com salt de 128 bits.

Rainbow Tables

- Tabelas pré-computadas de hashes para todos os salts possíveis.
- Uma tabela de 1.4 GB quebra 99.9% das passwords alfanuméricas do Windows em < 5 segundos.

- **Defesa**: usar salts grandes torna as rainbow tables inviáveis.

Porquê Passwords?

- **Custo**: são gratuitas.
- **Conveniência**: mais fácil para administradores repor uma password do que emitir um novo token.

Aula 2 — Passwords na Prática & Password Managers

Chaves Criptográficas vs Passwords

- Chave de 64 bits → 2^{64} chaves possíveis, escolhida aleatoriamente.
- Password de 8 caracteres → teoricamente $256^8 = 2^{64}$ possibilidades, mas **os utilizadores não escolhem aleatoriamente**.
- Atacante precisa de tentar muito menos do que 2^{63} passwords (ataque de dicionário).

Escolha de Passwords

- Utilizadores tendem a escolher passwords curtas ou adivinháveis.
- Num estudo com 14.000 passwords encriptadas, quase ¼ foram adivinhadas.

Técnicas para Melhores Passwords

- **Educação do utilizador**
- **Passwords geradas por computador**
- **Verificação reativa de passwords** (testar depois de definidas)
- **Verificação proativa de passwords**: regras + Markov Models + Bloom Filter para rejeitar passwords fracas antes de serem aceites.
- Melhor conselho: usar **passphrase** como base.

Ataques a Passwords

- Alvo: uma conta específica, qualquer conta no sistema, qualquer conta em qualquer sistema, DoS.
- Caminho comum: Exterior → utilizador normal → administrador.

Password Managers

- **Definição**: aplicação cliente + serviço backend que armazena credenciais num cofre encriptado e oferece preenchimento automático.
- **Elementos core**:
  - **Vault encryption**: AES-256 com autenticação de integridade.
  - **Key derivation**: KDF (PBKDF2, scrypt, Argon2) para resistir a adivinhação offline.
  - **Zero-knowledge design**: o fornecedor não consegue desencriptar o cofre.
  - **2FA/MFA**: segundo fator adicional.
- **Capacidades**: geração de passwords fortes e únicas, auto-fill, deteção de passwords reutilizadas/fracas/comprometidas.

Ataques Mitigados pelos Password Managers

Ataque	Mitigação	Limite
Credential stuffing	Passwords únicas por serviço	—
Adivinhação online	Passwords longas e aleatórias	—
Cracking offline de hashes	Passwords aleatórias = mais esforço para crack	—
Phishing	Auto-fill ligado ao domínio registado	Se domínio comprometido, falha
Shoulder surfing	Menos digitação visível	—
Keylogging/malware	Reduz teclas digitadas	Endpoint comprometido → falha

Risco de Concentração

- O cofre é um alvo de alto valor → a segurança da **master password** + **MFA** + **segurança do dispositivo** são críticas.

Paper — Estudo Gallus 2025: Segurança de Password Managers

- 10 gestores analisados: Bitwarden, 1Password, Dashlane, Keeper, LastPass, NordPass, ProtonPass, RoboForm, Google PM, KeePass.
- **Cifra**: AES-256 é padrão; **XChaCha-20** (Bitwarden, KeePass) como alternativa moderna com nonce de 192 bits.
- **Zero-knowledge**: garantido por todos exceto Google PM e KeePass (ambos podem aceder ao cofre).
- **KDF**: Argon2 (vencedor da Password Hashing Competition), PBKDF2 (mais comum), bcrypt.
- **SOC 2**: conformidade de auditoria de segurança — maioritariamente presente, exceto KeePass.
- **Alertas de violação (breach alerts)**: presentes em todos exceto KeePass.
- **Testes de penetração realizados**: força bruta da master password, phishing, exploração de backup, extração de memória.

Paper — A Ciência de Adivinhar Passwords (Bonneau 2012)

- **Problema com Shannon entropy H_1**: matematicamente inapropriada para passwords — mede incerteza antes de qualquer tentativa; não captura o custo real de adivinhação parcial.
- **Métricas de adivinhação parcial**:
  - **λ_β**: número de tentativas para comprometer fração β dos utilizadores.
  - **μ_α**: fração de utilizadores comprometidos com α tentativas.
  - **$\tilde{G}_\alpha$ (α-guesswork)**: número esperado de tentativas para comprometer fração α dos utilizadores.
- **Resultados práticos**: <10 bits efetivos para ataques online com throttling; ~20 bits para ataques offline.
- **Efeito demográfico**: ganho máximo de fator 2 ao conhecer género/idade — surpreendentemente pequeno.

Aula 3 — Biometria & Tokens de Segurança

Biometria — "You are your key" (Schneier)

- Exemplos: impressão digital, assinatura, reconhecimento facial, voz, marcha, iris.
- Autentica com base em características físicas do utilizador.

Biometria Ideal

Propriedade	Significado
Universal	Aplica-se a (quase) todos
Distinguível	Distingue com certeza
Permanente	A característica não muda
Colecionável	Fácil de recolher

Dois Modos Biométricos

- **Identificação**: quem és? → comparação **1:N** (ex: base de dados do FBI).
- **Autenticação**: és quem dizes ser? → comparação **1:1** (ex: rato com leitor de impressão digital).
- Identificação é muito mais difícil. **O foco do curso é autenticação.**

Erros Biométricos

- **FAR (False Accept Rate / Fraud rate)**: Trudy autenticada como Alice (Falso Positivo).
- **FRR (False Reject Rate / Insult rate)**: Alice não autenticada como Alice (Falso Negativo).

- **EER (Equal Error Rate)**: ponto onde FAR = FRR — usado para comparar biométricos.
- Aumentar o threshold → menos fraude, mais insultos (e vice-versa).

EER por Biométrico

Biométrico	EER
Impressão digital	~5%
Geometria da mão	~10^{-3}
Iris scan	~10^{-6} (teórico)

Fases de um Sistema Biométrico

1. **Enrollment**: captura e armazena o template biométrico.
2. **Recognition**: captura e compara com o template armazenado.

Tokens — "Algo que Possui"

- **Memory Card**: armazena dados (sem processamento). Ex: cartão bancário antigo. Necessita de leitor especial.
- **Smartcard**: tem processador, memória, I/O. Pode ter co-processador criptográfico. Executa protocolos de autenticação.
- **Cartão de Cidadão (PT)**: inclui BI, NIF, SNS, cartão de eleitor, título de viagem — tudo num só smartcard.

2-Factor Authentication (2FA)

- Requer **2 de 3** fatores: algo que sabe + algo que possui + algo que é.
- Exemplos: ATM (cartão + PIN), crédito (cartão + assinatura).

YubiKey

- Token OTP de baixo custo; liga via USB; funciona como teclado (sem drivers).
- Gera OTP de 40 caracteres com pressão de botão.
- **Modos**: Yubikey OTP, OATH-HOTP, OATH-TOTP, Static Password, Challenge-Response.
- Suportado por Google, PayPal, LastPass, etc.

Aula 4 — Tokens, 2FA Móvel & FIDO

Problemas com SMS como 2FA

- Custo (especialmente no estrangeiro), demora (>60s), inconveniência.
- **Não é seguro**: dispositivos StingRay (IMSI catchers) e vulnerabilidades SS7 permitem intercetar SMS.

Google Authenticator (TOTP/HOTP)

- App que gera OTPs baseados em **tempo** (TOTP) ou **contador** (HOTP).
- Configurado via **QR code** (partilha de segredo).
- Segurança baseada em: sincronização de relógio + segredo partilhado inicial.
- Não necessita de ligação após configuração.

QR-Login — Vantagens

- Password não é digitada → keyloggers não capturam nada.
- Password pode ser longa e aleatória (não precisa de ser memorizada).
- Protege contra phishing (telemóvel só envia para sites na sua base de dados).
- Seguro em computadores não confiáveis.

FIDO Alliance

- Formada em 2012 com o objetivo de substituir passwords por métodos mais fortes.
- Membros: Google, PayPal, Microsoft, Bank of America, MasterCard, VISA, Samsung, etc.
- **Ideia central**: usar autenticação local do dispositivo para autenticação online.

FIDO UAF (Universal Authentication Framework)

- Framework passwordless baseado em **criptografia assimétrica**.
- **Componentes**:
  - **FIDO Client**: interage com autenticadores e com o agente do utilizador.
  - **FIDO Server (Relying Party)**: valida attestation, gere associações de autenticadores a contas.
  - **Authenticator**: entidade segura que cria material de chave; pode ser biométrico, PIN, ou hardware bearer.
  - **ASM (Authenticator Specific Module)**: abstração que permite múltiplos autenticadores.

Fluxos FIDO UAF

1. **Registration**: descoberta de autenticadores → validação de attestation → associação à conta.
2. **Authentication**: desafio criptográfico → resposta do autenticador → validação.
3. **Transaction Confirmation**: WYSIWYWS (What You See is What You Sign).
4. **Deregistration**: remoção das chaves do autenticador.

Privacidade no FIDO UAF

- Sem identificador global visível entre relying parties.
- Par de chaves único por dispositivo, por utilizador e por relying party → **unlinkability**.
- Dados biométricos **não saem** do dispositivo.
- Consentimento explícito do utilizador para cada relying party.
- Certificados de attestation em lotes de ≥ 100.000 → não identificáveis individualmente.

U2F (Universal 2nd Factor)

- Segundo fator universal baseado em **criptografia de chave pública**.
- **Vantagens sobre OTP/SMS**:
  - **Proteção anti-phishing**: usa `appid` ligado ao domínio TLS.
  - **Proteção anti-MITM**: incompatibilidade de `appid/keyhandle`.
  - **Sem segredos partilhados** (ao contrário do OATH).
  - **Anónimo**: novas chaves geradas por site.
  - **Universal**: USB, NFC, BLE; ilimitados sites por dispositivo.
  - **Open standard**.

FIDO2

- Evolução passwordless do U2F.
- Inclui **WebAuthn** (W3C, 2021) — API JavaScript para autenticação no browser.
- **Três modos**: Single Factor (só chave), Second Factor (chave + password), Multi-Factor (chave + PIN).

Paper — Phishing: Uma Análise em 2021

- Termo “phishing” cunhado em **1996** no contexto de ataques à AOL.
- **4 fases do ataque**: planeamento → preparação → ataque → aquisição de valor.
- **Variantes**: spear-phishing (alvo específico/personalizado), vishing (por voz), smishing (por SMS).
- **90%** das organizações reportaram ataques de phishing direcionados em 2019.

Paper — Autenticação à Escala: Lições do Google

- **Autorização centrada no dispositivo**: os dispositivos registados do utilizador são a unidade de confiança.
- **2SV (Two-Step Verification)**: SMS/TOTP/app offline como segundo fator; cookie de 30 dias para "lembrar" o dispositivo.
- 2SV foi abusada por hijackers — mesmo assim reduziu drasticamente sequestros de conta.
- **ASP (Application-Specific Passwords)**: passwords separadas para apps legacy que não suportam 2SV (ex: IMAP, SMTP). Revogáveis individualmente.
- **Protótipo USB smartcard**: chave pública física, sem segredos partilhados → evolução direta que resultou no standard U2F/FIDO.

Paper — Is FIDO2 the Kingslayer? Adoção pelo Utilizador

- **94 participantes** — primeiro estudo qualitativo de adoção de FIDO2 como único fator (1FA).
- Utilizadores **aceitam FIDO2 como 1FA** mas com preocupações claras.
- Preocupações principais: **perda do dispositivo**, falta de modelos mentais, ausência de recuperação de conta.
- FIDO2 não satisfaz: **Nothing-to-Carry** (exige dispositivo extra) e **Easy-Recovery-from-Loss** (sem mecanismo nativo).
- FIDO2 = duas especificações: **WebAuthn** (API browser/W3C) + **CTAP2** (protocolo device↔authenticator).

Aula 5 — Controlo de Acesso, UNIX, RBAC, MLS & Canais Cobertos

Controlo de Acesso

- "Prevenção do uso não autorizado de um recurso."
- **Autenticação**: És quem dizes ser? (identidade)
- **Autorização**: Tens permissão para fazer isso? (controlo de acesso)

Elementos do Controlo de Acesso

- **Sujeito**: entidade que acede a objetos (processo representando utilizador/aplicação).
- **Objeto**: recurso controlado (ficheiro, diretório, registo, programa).
- **Direito de acesso**: forma de acesso (leitura, escrita, execução, eliminação, criação, pesquisa).

Matriz de Controlo de Acesso (Lampson)

- Linhas = sujeitos; colunas = objetos; células = direitos.
- Pode ser muito grande (1000s utilizadores × 1000s recursos).
- **ACLs (Access Control Lists)**: matriz por coluna — fácil de mudar direitos de um recurso.
- **Capabilities (C-lists)**: matriz por linha — fácil de delegar, evita o "confused deputy".

Confused Deputy

- O compilador (deputy) age em nome de Alice, mas usa **os seus próprios direitos**, não os de Alice.
- Com ACLs: difícil de evitar.
- Com Capabilities: mais fácil — mantém associação entre autoridade e propósito.

Permissões UNIX

- Baseadas em **inodes** — estrutura com atributos e permissões.
- Formato: `rwxrwxrwx` (owner | group | others).
- **r** (read), **w** (write), **x** (execute).
- Para **diretórios**: r = listar, w = criar/apagar ficheiros, x = entrar/traversar.
- **SetUID/SetGID**: usa temporariamente os direitos do owner/grupo do ficheiro.

- **Sticky bit**: em diretórios, apenas o dono pode renomear/mover/apagar.
- **Superuser**: isento das restrições normais de controlo de acesso.

RBAC (Role-Based Access Control)

- Utilizadores são atribuídos a **roles** (funções/papéis).
- Roles têm permissões sobre objetos.
- Simplifica gestão: em vez de gerir permissões por utilizador, gere por role.
- Ex. bancário: HR atribui roles → roles têm acesso a aplicações.

MLS — Multi-Level Security

- Necessário quando sujeitos/objetos em diferentes níveis de classificação usam o mesmo sistema.
- **Classificações** (para objetos): TOP SECRET > SECRET > CONFIDENTIAL > UNCLASSIFIED (DoD).
- **Clearances** (para sujeitos): habilitação de segurança.

Modelo Bell-LaPadula (BLP) — Confidencialidade

- **Simple Security Condition**: S pode **ler** O se L(O) ≤ L(S) — **"no read up"**.
- **★-Property (Star Property)**: S pode **escrever** O se L(S) ≤ L(O) — **"no write down"**.
- Objetivo: prevenir leitura não autorizada (confidencialidade).
- **Crítica de McLean**: "System Z" — admin pode reclassificar e "write down".
- **Resposta**: propriedade de **tranquilidade** (forte: labels nunca mudam; fraca: só mudam sem violar política — permite "least privilege" / high water mark).
- **Conclusão**: BLP é simples, talvez demasiado; inspirou outros modelos.

Canal Coberto (Covert Channel)

- **Definição**: caminho de comunicação não intencional pelos designers do sistema.
- **Exemplo**: Alice (TOP SECRET) cria/apaga um ficheiro; Bob (CONFIDENTIAL) lista os ficheiros periodicamente → Alice vaza info TOP SECRET para Bob a 1 bit/min.
- **Condições para existir**: (1) recurso partilhado, (2) emissor pode variar propriedade que recetor observa, (3) comunicação pode ser sincronizada.
- São **ubíquos** — praticamente impossível de eliminar.
- **Objetivo realista (DoD)**: reduzir capacidade do canal coberto para ≤ 1 bit/segundo.
- **Exemplo real**: ocultar dados no campo `Sequence Number` do cabeçalho TCP (ferramenta `covert_TCP`).

Paper — Crítica de McLean ao Bell-LaPadula

- McLean demonstrou que o **Basic Security Theorem (BST)** é provável para o **†-property** — a inversão direta da ★-property, que permite *escrever para baixo livremente*.
- O †-property é claramente **inseguro** (permite fuga de informação) — prova que o BST captura demasiado pouco.
- O modelo formal do BLP não exclui sistemas trivialmente inseguros; o teorema é satisfeito por políticas seguras e por políticas perigosas.

Paper — Break-the-Glass no Hospital S. João (Porto)

- **BTG (Break-the-Glass)**: política de acesso de emergência — médico obtém acesso total a registos clínicos fora das suas permissões habituais quando clinicamente necessário.
- Filosofia central: **máxima liberdade + máxima responsabilidade**.
- Modelo: híbrido **RBAC + IBAC** (Identity-Based Access Control).
- **Fluxo de acesso BTG**: pedido negado → prompt BTG → utilizador introduz razão clínica → conta fica "frozen" → hierarquia notificada → acesso concedido + log de auditoria completo imutável.
- Princípio chave: a auditoria posterior substitui o controlo preventivo — em emergência não se pode atrasar o acesso.

Aula 6 — CAPTCHAs & Segurança de Software

CAPTCHA

- **CAPTCHA** = Completely Automated Public Turing test to tell Computers and Humans Apart.
- Também conhecido como **HIP** (Human Interactive Proof).
- Paradoxo: o computador cria e avalia testes que ele próprio não consegue passar.
- Objetivo: **controlo de acesso** — garantir que apenas humanos acedem.
- Usos: evitar bots em serviços de email, votações, indexação automática.
- **Regras**: fácil para humanos, difícil para máquinas (mesmo com acesso ao software).
- **Tipos**: visual (OCR distorcido), áudio (palavras distorcidas).
- **Desafio atual**: LLMs e IA estão a tornar os CAPTCHAs cada vez mais fáceis de resolver.

Turing Test

- Proposto por Alan Turing em 1950.
- Humano faz perguntas a um humano e a um computador sem ver nenhum dos dois.
- Se não conseguir distinguir → computador passa no teste.
- CAPTCHA é como um **teste de Turing inverso**.

Software Security — Introdução

- Virtualmente toda a segurança é implementada em software → software vulnerável = segurança comprometida.
- **"Complexity is the enemy of security"** — Paul Kocher.
- Estimativa conservadora: **5 bugs por 1000 LOC**.
- Num computador típico: ~150.000 bugs; rede de 30.000 nós: ~4.5 mil milhões de bugs.

Erros, Falhas e Faltas

- **Erro** (error): engano de programação (humano).
- **Falta/Defeito** (fault): estado interno incorreto resultante de um erro — **não observável externamente**.
- **Falha** (failure): comportamento externo incorreto resultante de uma falta — **observável externamente**.
- Cadeia: **error** → **fault** → **failure**.

Categorias de Problemas de Software

- **Program flaws** (não intencionais): buffer overflow, incomplete mediation, race conditions.
- **Malicious software** (intencional): vírus, worms, outros malware.

Aula 7 — Malware: Propagação, Evolução & Contramedidas

Tipos de Malware

Tipo	Característica
Vírus	Propagação passiva (precisa de hospedeiro)
Worm	Propagação ativa (autónomo)
Trojan horse	Funcionalidade inesperada/oculta
Trapdoor/Backdoor	Acesso não autorizado
Rabbit	Esgota rapidamente recursos do sistema

Onde Vivem os Vírus?

- Boot sector, memória residente, aplicações/macros, rotinas de biblioteca, compiladores/debuggers/antivírus, **firmware** (UEFI).

Linha do Tempo do Malware

Ano	Malware	Nota
1971	Creeper	Primeiro worm
1982	Elk Cloner	Primeiro vírus de PC
1986	Brain	Primeiro vírus de boot sector
1988	Morris Worm	Primeiro worm grande da Internet
2001	Code Red	250.000 sistemas em 15 horas
2003/4	SQL Slammer	250.000 em 10 minutos
2010	Stuxnet	Primeiro cyberweapon (ICS/SCADA)
2017	WannaCry / NotPetya	Ransomware massivo
2020	Sunburst	Supply-chain backdoor

Morris Worm (1988) — Caso de Estudo

- Tentou determinar onde espalhar, espalhar a infeção e ficar não detetado.
- Usou: adivinhação de passwords, buffer overflow no `fingerd`, trapdoor no `sendmail`.
- **Flaws no fingerd e sendmail eram conhecidos mas não corrigidos.**
- Consequência: chocou a internet, originou o **CERT**.

Code Red (2001)

- Explorou buffer overflow no **Microsoft IIS**.
- Monitorizou tráfego na porta 80 para encontrar servidores vulneráveis.
- Dias 1-19: propagação; Dias 20-27: ataque DDoS ao whitehouse.gov.
- Variantes incluíram backdoor para acesso remoto.

SQL Slammer (2003/4)

- 250.000 sistemas em **10 minutos** (Code Red demorou 15 horas para o mesmo).
- Pico: infeções duplicavam a cada **8.5 segundos**.
- Cabia num **único pacote UDP de 376 bytes**.
- Firewalls deixavam passar por assumirem que pacotes pequenos eram inofensivos.

Malware Polimórfico vs Metamórfico

Tipo	Mecanismo	Detetável?
Polimórfico	Encripta o corpo com chave diferente em cada propagação; inclui código de decifra	Sim (pelo código de decifra — tem assinatura)
Metamórfico	Muta o código inteiro antes de infetar novo sistema	Problema ainda por resolver

Paper — Obfuscation: The Hidden Malware

- **Corrida de armamentos em 4 fases**: infeção → packing → deteção estática → deteção dinâmica.
- **Packers**: comprimem e/ou encriptam o executável; **~48,92%** dos ficheiros de malware são packed.
- **Motor polimórfico**: payload encriptado com chave diferente em cada cópia + stub de decifra; stub tem assinatura reconhecível — ainda detetável.
- **Motor metamórfico — 5 passos**:
  1. Desassemblar o código em representação intermédia.
  2. Encolher (*shrink*) — remover instruções redundantes.
  3. Permutar — reordenar blocos e instruções.
  4. Expandir — substituir por equivalentes funcionais (*junk code*).
  5. Reassemblar — código completamente reescrito, sem padrão reconhecível.
- **Win32/Apparition**: primeiro vírus metamórfico documentado.

- **Análise de entropia** como contra-medida: ferramentas **PHAD**, **PE-Probe**, **MRC** detetam executáveis packed pela sua entropia anormalmente alta.
- **Ferramenta Reminder**: desmascara e analisa executáveis comprimidos/encriptados.

Aula 8 — Worms, Deteção de Malware & Automação Ofensiva com IA

Worms — Características

- Programa replicante que se propaga pela rede (email, remote exec, remote login).
- Fases: **dormência** → **propagação** → **ativação** → **execução**.
- Pode disfarçar-se como processo de sistema.
- Tecnologias modernas: multiplatforma, multi-exploit, ultra-rápido, polimórfico, metamórfico, zero-day exploits.

Modelo de Propagação de Worms

- Curva em S: fase de arranque lento → fase de propagação rápida → fase de abrandamento.

Bots e Botnets

- Bot: programa que toma controlo de outro computador para lançar ataques difíceis de rastrear.
- **Botnet**: conjunto de bots coordenados.
- Mecanismo de controlo remoto: IRC, HTTP, DHT (Distributed Hash Tables).
- Contramedidas: IDSes, honeypots, digital immune systems.

Rootkits

- Conjunto de programas instalados para acesso administrativo malicioso e furtivo.
- Ocultam a sua existência (processos, ficheiros, entradas de registry).
- **User-mode** ou **kernel-mode**.
- Modificam a **system call table** para redirecionar chamadas do sistema.

Métodos de Deteção de Malware

Método	Vantagens	Desvantagens
Signature Detection	Eficaz para malware conhecido; mínimo esforço do admin	Ficheiro de assinaturas enorme; não deteta desconhecidos; não deteta polimórfico/metamórfico
Change Detection	Virtualmente sem falsos negativos; deteta malware desconhecido	Muitos falsos positivos; ficheiros mudam frequentemente; pesado para admins
Anomaly Detection	Pode detetar malware desconhecido	Não provado na prática; atacante pode simular comportamento normal; difícil definir “normal”

Honeypots

- Sistemas isco preenchidos com informação fabricada, monitorizados.
- Desviam e mantêm o atacante, recolhendo informação sobre as suas atividades.
- Úteis para **deteção e aviso precoce de ataques zero-day**.

Outros Ataques de Software

- **Salami attack**: programador “fatia” pequenas quantias (ex: frações de cêntimos em juros bancários) — difíceis de detetar individualmente.
- **Time bomb**: código malicioso ativado numa data/condição específica.
- **Linearization attack**: explorar lógica de verificação serial.

Paper — Levantamento de Botnets: Estrutura e Deteção

- **Topologias de C2 (Command-and-Control)**: | Topologia | Sobrevivência | Latência | Garantia entrega | Exemplo | |—|—|—|—|—| | **Centralizada** (IRC/HTTP) | Baixa (single point of failure) | Baixa | Alta | SDBot | | **P2P** estruturado | Média | Média | Média | Storm | | **Não estruturada** | Alta | Alta | Não garantida | — |
- **SDBot**: botnet histórica baseada em IRC — fácil de controlar mas fácil de desativar.
- **Storm**: botnet P2P muito resistente a takedowns; usou DHT.
- **BotHunter**: ferramenta de deteção por correlação de comportamentos cooperativos de rede.
- Métodos de deteção: assinaturas, comportamentos cooperativos, anomalia de tráfego, medição.

Aula 9 — Confiança em Software, Engenharia Inversa & Proteção

Pode-se Confiar em Software? (Ken Thompson, 1984)

- Cenário: compilador C com vírus que ao compilar o `login` cria uma backdoor, e ao recompilar o próprio compilador, incorpora-se a si mesmo.
- **Recomeçar do zero não resolve** — o compilador comprometido compila o novo compilador.
- Moral: **a confiança deve estar nas ferramentas que usas para construir software**.

Software Reverse Engineering (SRE)

- Também chamado RCE (Reverse Code Engineering) ou "reversing".
- **Usos legítimos**: análise de malware, compreensão de código legado.
- **Usos maliciosos**: remover restrições de software (pirataria), encontrar vulnerabilidades, fazer batota em jogos.

Ferramentas de SRE

Ferramenta	Função
Disassembler (IDA Pro)	Converte executável em assembly — análise estática
Debugger (SoftICE, OllyDbg)	Execução passo a passo — análise dinâmica
Hex Editor (UltraEdit, HIEW)	Ver/modificar bytes do executável (patching)

- Disassembler dá visão estática; debugger dá visão dinâmica. **Ambos são necessários** para SRE sério.
- **Formatos executáveis**: PE (Portable Executable) no Windows; ELF no UNIX/Linux.

Técnicas Anti-SRE

Técnica	Mecanismo
Anti-disassembly	Código encriptado, false disassembly, código auto-modificável
Anti-debugging	Monitorizar uso de registos de debug, breakpoints inseridos, threads interativas
Tamper-resistance	Código faz hash de si próprio; falha se adulterado ("guards")
Code Obfuscation	Torna o código difícil de entender (spaghetti code, predicados opacos)

Obfuscação de Código

- Objetivo: dificultar a compreensão do código pelo atacante.
- **Predicado opaco**: condição cujo resultado é sempre verdadeiro/falso mas é difícil de determinar estaticamente.
- Investigação recente mostrou que obfuscação **provavelmente não pode fornecer segurança "forte"**.
- Útil na prática para aumentar o trabalho do atacante (especialmente na autenticação: atrasar a descoberta do "bit" de sucesso/falha).

BOBE — Break Once, Break Everywhere

- Software distribuído em cópias idênticas (clonagem) → um ataque bem-sucedido quebra todas as cópias.
- **Software Metamórfico**: cada cópia distribuída é internamente diferente mas funcionalmente idêntica.
- Melhora a resistência a BOBE: N cópias metamórficas → N vezes mais trabalho para quebrar todas.
- Analogia: **diversidade genética** — uma doença que mata todas as plantas geneticamente idênticas só mata algumas num campo diversificado.

Paper — Heartbleed (CVE-2014-0160)

- **Bug**: buffer *over-read* (não overflow) no protocolo TLS Heartbeat (RFC 6520) — leitura além dos limites, não escrita.
- **Mecanismo**: cliente declara comprimento do payload (até 65.535 bytes) sem enviar dados reais; servidor ecoa esse comprimento de memória RAM não relacionada.
- **Verificação em falta**: `payload_length ≤ length − 1 − 2 − 16` nunca verificada no OpenSSL.
- **Impacto**: atacante lê até **64 KB de RAM por pedido** — pode incluir chaves privadas TLS, cookies, passwords.
- **Linha do tempo**: bug aceite em 31 dez 2012; divulgado em 7 abr 2014 — **~16 meses** em produção.
- **Causa raiz**: C/C++ não têm verificação automática de limites de arrays/buffers.
- **"Tragédia dos comuns"**: OpenSSL (~450.000 LOC) é infraestrutura crítica de toda a Internet mas com poucos mantenedores remunerados.

Paper — The Cathedral and the Bazaar (Eric S. Raymond)

- **Catedral**: desenvolvimento centralizado, planeado, formal — versões infrequentes.
- **Bazar**: desenvolvimento aberto, colaborativo, iterativo — versões frequentes ("release early, release often").
- **Lei de Linus**: *"given enough eyeballs, all bugs are shallow"* — com revisores suficientes, todos os bugs são fáceis de encontrar.
- **Projeto fetchmail**: caso de estudo de sucesso do modelo bazar.
- 19 lições, incluindo: "planeia atirar fora uma versão", "trata os utilizadores como co-desenvolvedores".

Paper — Open vs. Closed Source em Segurança (Ross Anderson)

- **Modelo de crescimento de fiabilidade**: $E \approx K/t$ — erros descobertos decrescem com o tempo de exposição t.
- **Argumento de simetria**: num mundo ideal, open e closed source têm fiabilidade equivalente ao longo do tempo.
- **Quebras de simetria** (que favorecem o atacante):
    1. Atacante foca-se em alvos específicos; defensor protege tudo.
    2. Comportamento do vendedor ("Wild West") — sem incentivo real para corrigir.
    3. Custos de transação elevados para utilizadores aplicarem patches.
    4. Utilizadores transientes sem relação de longo prazo com o produto.
- **TCPA (Trusted Computing Platform Alliance)**: criticado como instrumento de vantagem comercial, não de segurança genuína do utilizador.

Aula 10 — Digital Rights Management (DRM)

O que é DRM?

- Conjunto de mecanismos técnicos que mantêm media encriptado durante a distribuição, fornecem chaves de decifra apenas a utilizadores/dispositivos autorizados e aplicam regras de uso.
- **Problema fundamental**: "remote control" — distribuir conteúdo digital mantendo controlo sobre o seu uso após entrega.

- **"Persistent protection"**: como aplicar restrições (sem cópia, número limitado de leituras, limites de tempo, sem reencaminhamento)?

Streaming Adaptativo

- **DASH** (Dynamic Adaptive Streaming over HTTP): conteúdo dividido em segmentos; player adapta qualidade às condições de rede; manifesto **MPD** (Media Presentation Description).
- **HLS** (HTTP Live Streaming): similar ao DASH; playlists em formato **M3U8**; dominante no ecossistema Apple.
- Segmentos premium são **encriptados**; player obtém licença (chaves + política) antes de decifrar.

Entidades Core no DRM

Entidade	Função
CDN (Content Delivery Network)	Distribui segmentos encriptados à escala global
License Server	Valida titularidade; emite chaves de decifra e política de uso
CDM (Content Decryption Module)	Corre no dispositivo cliente; solicita licenças; decifra conteúdo

CDM — Content Decryption Module

- Componente confiável integrado no browser ou SO.
- **Funções**: gera pedidos de licença, recebe e armazena chaves, decifra áudio/vídeo, aplica regras DRM.
- **Proteções**: chaves em memória protegida (hardware quando possível), sandboxing, obfuscação de código, anti-debugging, verificação de integridade.

Analog Hole

- Quando o conteúdo é renderizado, pode ser capturado em forma analógica.
- **DRM não pode prevenir este ataque** — objetivo é aumentar o custo de captura de qualidade.

Software-based DRM — Limitações

- **Impossível** ter DRM forte apenas em software.
- Razão: não é possível esconder verdadeiramente um segredo em software; não é possível prevenir SRE; utilizador com privilégios de admin pode quebrar qualquer proteção.
- **Killer attack**: SRE do software DRM para extrair chaves.
- **DRM é "hide and seek"** com uma chave oculta/obscurecida em software.

DRM — Estado Atual

- Na melhor das hipóteses: **security by obscurity** — considerado pejorativo em segurança.
- Viola o **Princípio de Kerckhoffs** (designs secretos).
- Frase relevante: *"Whoever thinks his problem can be solved using cryptography, doesn't understand his problem."*

HDCP

- **High-bandwidth Digital Content Protection** — proteção de conteúdo em ligações HDMI/DisplayPort.
- Objetivo: proteger o caminho de saída do conteúdo decifrado.

Paper — A First Look at DRM: Widevine, FairPlay, PlayReady

- **Widevine Levels (Google)**: | Nível | Segurança | Criptografia | Processamento media | |—|—|—|—| | **L1** | Máxima | TEE (hardware) | TEE (hardware) | | **L2** | Intermédia | TEE (hardware) | Software | | **L3** | Mínima | Software puro | Software puro |
- **OEMCrypto**: módulo hardware Widevine que efetua operações criptográficas no TEE do dispositivo.
- **FairPlay (Apple)**: protocolo **SPC/CKC** (*Service Provider Certificate / Content Key Context*) — dois documentos trocados durante a aquisição de licença.

- **PlayReady (Microsoft)**: usa **HDS** (*Header Decryption Scheme*).
- **CENC (Common Encryption)**: standard que permite encriptar conteúdo uma vez e desencriptá-lo com múltiplos DRM (Widevine, PlayReady, FairPlay) — simplifica Multi-DRM.

Aula 11 — DRM Web & Desenvolvimento Seguro de Software

DRM Key Systems

- Um **DRM key system** é o ecossistema DRM completo: CDM + protocolo de licença + linguagem de política.
- **Principais key systems**:
  - **Google Widevine**: cross-platform, amplamente usado em browsers.
  - **Microsoft PlayReady**: ecossistema Windows.
  - **Apple FairPlay**: ecossistema Apple (Safari/iOS).
- **Multi-DRM**: o mesmo conteúdo é frequentemente empacotado em múltiplas variantes DRM para cobrir todas as plataformas.

EME — Encrypted Media Extensions

- Standard **W3C** que permite a uma página web interagir com sistemas DRM.
- **Padroniza**: API JavaScript (key system selection, session creation, request/response orchestration).
- **NÃO padroniza**: formatos de mensagem DRM (são "blobs" opacos), esquemas criptográficos, provisionamento de dispositivos, semântica de política de licença.
- **Mensagens opacas**: a licença request/response é produzida/consumida pelo CDM — o browser não interpreta o conteúdo.

Fluxo EME (License Acquisition)

1. Site pede key system e configuração.
2. Browser instancia o CDM.
3. Site cria sessão e chama `generateRequest(initData)`.
4. CDM produz pedido de licença opaco.
5. Site envia pedido ao license server.
6. License server retorna resposta de licença opaca.
7. Site chama `update(response)` → CDM instala chaves e política.
8. CDM decifra media para playback.

Tipos de Sessões EME

- **Temporary**: chaves destruídas quando sessão fecha.
- **Persistent-license**: chaves e metadados podem ser armazenados e recarregados (offline viewing).

Quem Faz o Quê em EME

Componente	Responsabilidade
Web app (JS)	Chama a API EME
Browser (user agent)	Implementa EME; reencaminha mensagens opacas
CDM	Gera/consome mensagens; guarda chaves; decifra media
License server	Valida titularidade; emite chaves e política

Ataques ao Pipeline DRM

Classe	Exemplos
Client-side	Screen capture (analog hole), bypass de HDCP, SRE/tampering do CDM
Service-side	Credential stuffing, session/token replay, scraping automatizado de catálogo

Defesas DRM

- Proteção hardware-backed para chaves quando disponível.
- Tokens de curta duração ligados ao dispositivo/sessão.
- Rate-limiting e deteção de anomalias nos endpoints de licença.
- **Forensic watermarking**: marcas rastreáveis para identificar fonte de fugas.
- Monitorização e takedown de canais de pirataria.

Penetrate and Patch — O Modelo Errado

- Abordagem habitual: desenvolver rápido → lançar sem testes adequados → corrigir quando surgem falhas.
- **Porque acontece**: vantagem de ser primeiro no mercado (network economics); sem incentivo económico serio (fornecedores de software não são legalmente responsáveis por falhas).
- **Por que é uma falácia**: patches muitas vezes introduzem novas falhas; software é alvo móvel (novas versões, funcionalidades, usos).

Open Source vs Closed Source — Segurança

	Open Source	Closed Source
Argumento	"Mais olhos" → menos falhas (variante do Princípio de Kerckhoffs)	Falhas menos visíveis → "security by obscurity"
Realidade	wu-ftp: 8000 LOC, amplamente usado, falhas graves só descobertas após 10 anos; HeartBleed; segue modelo "penetrate and patch"	Exploits não requerem código fonte; análise de código fechado é trabalhosa mas possível
Conclusão	Nenhuma vantagem óbvia de segurança para nenhum dos lados	O que importa são as **práticas de desenvolvimento**

- **Microsoft Fallacy**: (1) Microsoft faz mau software; (2) software da Microsoft é closed source; (3) logo todo o software closed source é mau. → **Falácia lógica.**
- Microsoft é atacada frequentemente porque é um **alvo grande** (mais "bang for the buck").

Paper — Your DRM Can Watch You Too (Privacidade do EME)

- **Widevine Client ID**: identificador único e estável derivado do hardware do dispositivo — não é um cookie.
- Os browsers partilham-no com servidores de licença **sem consentimento explícito** do utilizador.
- **Sobrevive à limpeza de cookies**, histórico e outras medidas de privacidade.
- Pode ser usado para **cross-site tracking** se múltiplos sites usarem o mesmo servidor de licenças.
- **Ferramenta EME Track**: documenta o Client ID e o fluxo completo de aquisição de licença (**21 passos**).
- Ironia: o mecanismo DRM criado para proteger conteúdo pode ser subvertido para rastrear utilizadores.
