Bezpečnost v IT

6. přednáška

Radek Janoštík

Univerzita Palackého v Olomouci

3. 5. 2024

Outline

- Aktuální (kyber)bezpečnostní situace
- RADIUS
- OAUTH
- Šifrování souborů, adresářů, disků, systému
- Filtrace (+ firewall)
- Proxy, Brány

Aktuální (kyber)bezpečnostní situace

- Sledujete zprávy z dění v kyberprostoru? Co se stalo?
- 24 let stará chyba v glibc při převodu mezi znakovými sadami
  - Postiženo PHP při převodu na znakovou sadu `ISO-2022-CN-EXT` možný buffer overflow
  - $\Rightarrow$ možnost napadení aplikace $\Rightarrow$ poté serveru
- Vydáno Chrome 124, které zavádí mechanismy *postkvantové kryptografie*
  - Nový ochranný mechanismus při navazování TLS 1.3 a QUIC spojení
  - Některé implementace serverů/firewallů mají problém s dodatečnými parametry
  - Chybně implementují standard
- Google vydal zprávu o boji proti škodlivým aplikacím na Play
  - V roce 2023 zablokováno přes 2.2 milionu aplikací
  - V roce 2022 zablokováno „jen“ 1.4 milionu
- V Británii začal platit nový zákon upravující výchozí hesla zařízení
  - Nesmí být pevné výchozí heslo pro zařízení
  - Musí být náhodné, bez vazby na HW identifikátor (MAC adresu)
  - Silné proti bruteforce
  - Pomůcka pro výrobce

Remote Authentication Dial-In User Service (RADIUS)

- = síťový protokol pro autentizace, autorizaci a správu účtů uživatelů na síti
  - AAA protokol – *Authentication, authorization, accounting*
  - „Kdo se kam, s jakými oprávněními a nastavením může připojit“
- Počátky $\approx$ 1992, RFC-2865, RFC-2866 (accounting)
  - Postupně doplňován o funkcionalitu, zabezpečení (asi 40 RFC)
  - Široce rozšířen, implementován (téměř) všemi výrobci zařízení
- Mnoho implementací serveru (komerční i open-source)
  - Od primitivních – uživatelé v textovém souboru, bez správy účtu
  - Až po komplexní – uživatelé např v LDAP či AD
- Využívá UDP porty 1812 a 1813
- Možnost *Roamingu* – spolupracující organizace umožní autentizaci uživatelů pomocí „jejich“ RADIUS-serveru
  - Přístup uživatelů spolupracujících organizací do jejich sítí (možné jiné nastavení)
  - Např. „náš“ eduroam

RADIUS – Autentizace

- Modelová situace – přístup uživatele(počítače) do Wi-Fi sítě (např. zde na univerzitě)
- V roli RADIUS-klienta není uživatelův počítač, ale AP, ke kterému se chce připojit
  - Počítači ještě není umožněno komunikovat v síti
  - AP od klienta vyžádá jméno a heslo
  - A vyřídí za něj komunikaci s RADIUS-serverem
- AP vytvoří *access request* obsahující ID AP a autentizační data uživatele
  - Odešle jej šifrovaně na RADIUS-server
  - Je možné použít PAP, CHAP, EAP
- Server v databázi ověří správnost údajů a dá AP vědět, co vše může uživatel používat (TCP/IP)
- AP umožní přístup počítači do sítě s daným nastavením
- Obrázek

RADIUS – struktura paketu

- Paket má pro všechny zprávy stejnou strukturu
  - Rozšiřován pomocí *Attribute-value párů*
- 1B – Kód zprávy: `1-Access-Request, 2-Access-Accept, 3-Access-Reject, 4-Accounting-Request, 5-Accounting-Response, 11-Access-Challenge, 12-Status-Server, 13-Status-Client, 255-Reserved`
- 1B – ID zprávy – párování požadavku-odpovědi. Server duplikát (dle IP klienta)
- 2B – délka zprávy
- 16B – Autentikátor
  - Požadavku – XOR sdíleného tajemství klienta a serveru a MD5 hesla uživatele
  - Odpovědi – MD5 z kódu zprávy, ID zprávy, délky, autentikátoru požadavku, z atributu a sdíleného tajemství
- Volitelná délka – atributy(1B typ, 1B délka, hodnota). 64 předdefinovaných atributů

Open Authorization (OAuth) 2.0

- Autorizační framework pro (převážně) webové služby
- RFC-6749 – nahrazuje OAUTH 1.0
- Umožňuje poskytnout webovým službám (některá) data o uživatelích, aniž by služby znaly heslo
  - Delegování (částečného) přístupu k uživatelským datům
- Dnes v Internetu široce používáno (velké firmy jako Google, Facebook, GitHub, . . .)
  - Na webových stránkách možno vidět jako: „Přihlásit se pomocí . . .“ – není čistý OAuth
  - Přístupy k API službám
- Široká podpora ze strany webových frameworků
  - ASP.NET
  - Frameworky pro čtení dat z API

OAuth – delegace oprávnění (obrázek)

- *Klient(webová aplikace)* žádá *uživatele* o autorizaci(udělení přístupu) k jeho datům
- Uživatel oprávnění udělí
- Webová aplikace o uděleném oprávnění informuje *autorizační server*
- *Autorizační server* vytvoří *Access token*, kterým se bude webová aplikace prokazovat
- Webová aplikace přistupuje k *Resource serveru* s Access tokenem
- Resource server ověří platnost tokenu a zpřístupní uživatelova data, ke kterým dal povolení
- Výše zmíněné se reálně děje sérií přesměrování uživatelova prohlížeče

OAuth – autentizace

- OAuth je primárně autorizační protokol
- *Access token* nic neříká o tom, kdo jej používá
  - Pouze uděluje oprávnění ke konkrétním zdrojům
  - Paralela – přístupová karta v hotelu
- Mohou nad ním být postaveny autentizační protokoly
- Doporučuji pročíst: `https://oauth.net/articles/authentication/`
- Podrobnější studium OAuth ponechávám na vaše samostudium

Šifrování dat – motivace

- S fyzickým přístupem k nezašifrovanému disku máme přístup prakticky kamkoliv
  - Můžeme číst/měnit data
  - Měnit hesla
  - Dělat vše pod rootem
- Pro rychlý servisní zásah se to hodí, pro počítače s důvěrnými/citlivými/drahými daty to je problém
- V praxi použití hlavně šifry `AES-192, AES-256` (rychlost)
- Možné však použít jakoukoliv šifru
- Možné offline bruteforce útoky (ale relativně pomalé)

Šifrování jednotlivých souborů

- Nic nám nebrání si šifrování naprogramovat sami
  - ⇒ možná chybovost, nízká uživatelská přívětivost
- Použití standardizovaných nástrojů

```
openssl aes-256-cbc -in soubor.txt -out sifrovany
openssl aes-256-cbc -d -in sifrovany -out desifrovany.txt
```

  - Při každé úpravě musím ručně dešifrovat, upravit, zašifrovat
- Lepší je použít standardizovaný formát
  - Šifrované PDF, LibreOffice, . . .
  - ⇒ uživatelská přívětivost

Šifrování adresářů

- Možnost „zabalit“ do archivu a ten šifrovat (např. 7zip)
  - Horší práce, rozbalení změna, zabalení
  - Bezpečné smazání mezikroku
- Windows umožňuje zapnout šifrování pro adresář
  - `Properties→Advanced→Encrypt contents to secure data`
  - Využita RSA
  - Možnost export certifikátu (a privátního klíče)
  - `start→Manage file encryption certificates`
- Linux – připojení šifrované složky pomocí `cryfs`
  - Transparentní práce
  - Výchozí AES-256, podpora více šifer

    ```
    cryfs --show-ciphers
    ```
  - Jak je na tom MacOS?

Šifrování oddílů

- $\Rightarrow$ Po spuštění se OS zeptá na heslo, poté je oddíl korektně připojen
- Transparentní práce
- V Linuxu zažitá praxe: oddělení `/home` od systému
  - Data zašifrována, systém nikoliv (výkon)
  - $\Rightarrow$ pořád možný chroot, nepřečtení dat, snadnější obnova
- Pozor citlivá data mohou být k přečtení v dočasných souborech nebo ve swap
- Linux – LUKS, cryptsetup
- Windows – Start$\rightarrow$ Settings $\rightarrow$ Privacy & security $\rightarrow$ Device encryption
- MacOS – ???

Šifrování celého systému

- Bez správného klíče k oddílům nezačne systém ani bootovat
- Bez spolupráce s UEFI je nutné mít nešifrovaný bootsector
  - Načte se „minimální kód nutný k dešifrování“
  - Poté již může vše řešit systém sám
- Možné zpomalení celého OS (nikoliv jen ukládání dat)
- Horší práce při „recovery“

Windows – Bitlocker

- Komplexní řešení pro šifrování celého systému a dat
- Spolupráce s kryptografickým čipem TPM (Trusted platform module) + UEFI Secure Boot
  - Generování privátních klíčů
  - Hashe HW konfigurace počítače ⇒ automatické odemknutí (+ detekce přepojení „jinam“)
  - Možno fungovat i bez TPM (heslo nebo HW klíč)
- Po zašifrování nezapomenout uložit „klíče pro obnovu“
  - Možnost zálohovat do Active Directory Domain Services
- Pro aplikace je práce transparentní „neví o tom“
- Úbytek výkonu do 10%

Bezpečné mazání souborů

- Co se reálně děje, pokud smažeme nějaký soubor na disku?
- Kvůli rychlosti se pouze odmažou metadata/smaže *inode*
  - „Jedničky a nuly dat“ zůstávají na místě
  - Různými *recovery* nástroje lze některá data získat
  - Často už jen fragmenty
- Je potřeba data reálně přepsat
  - Nástroje, které stejné místo na disku přepíší náhodnými data
  - `shred`, `sfill`, `srm`
  - Spolupráce s filesystémem
  - Komplikace na SSD (proč?)
- Surové přepsání celého oddílu

```
dd if=/dev/urandom of=/dev/sda1
```

  - Některé firmy(PČR) tvrdí, že i přes přepsaní se dá z HDD něco „vydolovat“ (otázka motivace)
  - Vícenásobný přepis

Filtrace

- = kontrola dat, které procházejí aktivním prvkem; polopropustný filtr
  - Možné některou komunikaci nepovolit
  - Nemění data
- Kritéria pro filtraci
  - Režijní informace protokolů – adresy, porty, příznaky (SYN/ACK)
  - Data aplikačního protokolu (musíme znát a „rozumět“)
- Filtrační politiky
  - Co není zakázáno, je povoleno – „blacklisting“
  - Co není povoleno, je zakázáno – „whitelisting“
- Provádí *managovatelný switch*(linková vrstva), *router*(IP a TCP)
  - Případně specializovaný HW přes který „teče veškerý provoz“
  - SW(firewall) na klientských stanicích

Filtrace protokolu IP

- Povolení/zakázání komunikace mezi počítači; údaje ze záhlaví paketu
  - IP adresa odesílatele/příjemce
  - Protokol vyšší vrstvy
  - Příznaky (explicitní směrování)
- Filtrace protokolu ICMP – nežádoucí zprávy (redirect, změna směrování, echo?)
- Možné blokovat i rozsahy adres (jak je na tom geolokace?)
- Bez kombinace s TCP/UDP filtrem poměrně bezzubé
- *Reflexivní filtr* – sleduje spojení vyššího protokolu
  - Umožní zevnitř navázat relaci
  - Dovnitř propustí pouze data s ní související
- *Adress-spoofing attack* – útočník nastaví adresu z vnitřní sítě
  - Nedorazí k němu však odpověď
  - Možné takto doručit falešnou DNS odpověď a tím přesměrovat oběť „jinam“

Filtrace na TCP/UDP

- Povolení/zakázání komunikace určitých aplikací
  - Kombinace s IP filtrem – „tyto počítače můžou komunikovat jen těmito aplikacemi“
  - TCP/UDP záhlaví: porty, příznaky SYN/ACK
- Kvůli fragmentaci nemusí všechny pakety obsahovat TCP záhlaví
  - Filtr zastaví pouze první paket
  - Ostatní jsou doručeny k cíli, pokusí se stavit paket
  - Neuspěje a informuje protějšek ICMP zprávou
  - Čekaní na celý TCP segment a zamítnutí všech(náročnější, pomalejší, jedna cesta)
  - Filtrace ICMP
- Filtrace příchozích spojení (SYN)
  - Po navázání spojení dočasné povolení obousměrného provozu (bez SYN, ale s ACK, RST)
  - Kontrola čísel paketů, stejných IP adres, portů
- Nutné rozumět (a nahlížet do) aplikačním protokolům
  - Např. FTP vytváří datový kanál, aktivní režim (spojení z druhého směru)
  - UDP + DNS – pouštět jen odpovědi po dotazu

Firewall

- = Aplikace či HW, který provádí filtraci provozu + logování
- Dnes často specializovaný HW se podpůrným SW
  - Pro velký provoz potřeba větší výkon
  - Kvalitní síťové karty (větší buffery), redundance
  - Porozumění protokolům vyšších vrstev
  - Někdy podporují i inspekci SSL provozu (MITM)
- Často stovky až tisíce filtračních pravidel

Firewall – ukázky pravidel

- Při práci s větší sítí je dobré zadávat pravidlo pro rozsahy IP adres
  - ⇒ Rozumně rozdělit rozsahy dle logického rozdělení sítě
  - Pojmenovat si rozsahy ⇒ větší přehlednost pravidel

```
ip firewall address-list
add address=158.194.92.201-158.194.92.218 list=Ucebna-5002
add address=158.194.92.180-158.194.92.200 list=Ucebna-5004+vsechn
add address=158.194.92.219-158.194.92.236 list=Ucebna-5003
add address=158.194.92.237-158.194.92.249 list=Ucebna-1029
add address=158.194.92.180-158.194.92.249 list=Vsechny-PC-ucebny
add address=158.194.80.210-158.194.80.212 list=tiskarny
...
add address=158.194.80.0/24 list=vlan-80/92
add address=158.194.92.0/24 list=vlan-80/92
add address=158.194.0.0/16 list=Sit-UP
...
```

Firewall – ukázky pravidel

- U každého pravidla nezapomínejte psát komentáře
  - Za chvíli zapomenete, proč tam to pravidlo máte
  - Zvýšení zastupitelnosti, snížení následků nízkého *bus factoru*
- Proč jsou v našem FW následující pravidla:

```
add action=accept chain=forward  dst-port=445 \
    protocol=tcp src-address=158.194.0.0/16
add action=accept chain=forward  dst-port=445 \
    protocol=udp src-address=158.194.0.0/16
...
add action=drop chain=forward dst-port=445 protocol=tcp
add action=drop chain=forward dst-port=445 protocol=udp
```

- Původní komentáře:

```
comment="TCP-ACCESS SMB z UP"
comment="TCP-DROP SMB"
```

Firewall – ukázky pravidel

- Co dělají následující pravidla:

```
add action=accept chain=forward  dst-address-list=tiskarny \
    src-address=158.194.80.0/24
add action=accept chain=forward dst-address-list=tiskarny \
    src-address=158.194.92.0/24
add action=accept chain=forward dst-address-list=tiskarny \
    src-address-list=SMTP
...
add action=drop chain=forward dst-address-list=tiskarny
add action=drop chain=forward dst-address-list=diskova-pole \
    src-address-list=tiskarny
add action=drop chain=forward  src-address-list=tiskarny \
   dst-address-list=nodes-studentsky-cluster
add action=drop chain=forward src-address-list=tiskarny \
   dst-address-list=infrastrukturni-hypervizory
```

Firewall – komentáře

- Komentáře:

```
comment="ACCESS komunikace k tiskarnam z vlan80"
comment="ACCESS komunikace k tiskarnam z vlan92"
comment="ACCESS SMTP UP -> VLAN 80"
comment="DROP komunikace k tiskarnam"
comment="DROP z tiskaren -> diskova pole (kvuli ramsomware)"
comment="DROP z tiskaren -> studentsky cluster (kvuli ramsomware)
comment="DROP z tiskaren -> infrastrukturni hypervizory (kvuli ra
```

Skenování portů

- = Posílání specifických paketů protokolem TCP či UDP pro zjištění běžících služeb
- Snaha o (většinou) navázání spojení $\rightarrow$ rozumná odpověď $\Rightarrow$ na portu *poslouchá* nějaká služba
- Regulérní taktika při diagnostice sítě
  - Běží daná služba? Má otevřený port?
  - Neblokuje ji nějaký firewall?
  - Co nám odpovídá?
- Avšak také taktika útočníků k mapování sítě
  - Objevení potenciálně zranitelných aplikací
  - Odhad běžících operačních systém
  - Odhad počtu stanic
- Hromadné, masivní skenování může být považováno za útok
  - Zvlášť když po sobě neuzavíráme spojení (čerpání zdrojů)
- Velké množství sw scannerů, pro základy postačuje `nmap`
  - `shodan.io`

Skenování portů – typy

- **TCP connect** – vykoná kompletní třífázový handshake
  - Trvá poměrně dlouho
  - Pravděpodobně bude na cílovém systému zalogován
  - Možné provádět (non-root) běžným uživatelem
- **TCP SYN** – odeslán pouze SYN paket
  - Po obdržení `SYN,ACK` můžeme odvodit, že na portu služba běží
  - Pokud přijde `RST,ACK` je nejspíš port uzavřen
  - ACK už neposíláme $\Rightarrow$ možný DOS cíle (neukončeno spojení)
- **TCP FIN** – na port odešleme pouze `FIN`
  - Dle RFC 793 by měl systém pro všechny uzavřené porty odpovědět `RST`
  - Nemusí však RFC striktně dodržovat
- **TCP Null** – všechny příznaky nulové, měl by odpovědět stejně jako výše
- **UDP** – odešle paket na cílový port (není spojení)
  - Je-li odpověď `ICMP port unreachable` je port uzavřen
  - Jinak můžeme usuzovat, že je otevřen

Nmap – ukázka

- Doporučuji pročíst manuál

Proxy

- = aplikace poskytující službu počítačům (většinou) v interní síti
  - Serverová část – přijímá požadavky od klientů
  - Klientská část – chová se jako klient a „posílá požadavky dál“
  - Prostředník mezi komunikujícími uzly
- Většinou na rozhraní dvou sítí
- Zná aplikační protokol, který zpracovává
- Hlavní účely
  - Filtrace – před „přeložením paketů“ aplikace sady pravidel
    - Kdo se kam může připojit
    - Vidí aplikační data – HTTP metody, JavaScript, ...
    - Blacklist DNS, URL
  - Cache – ukládání provedený požadavků a znovupoužití (s dnešními SPA nepoužitelné)
- Některé protokoly mají *chování jako proxy* (DNS, SMTP), Obrázek

Klasická proxy

- Klient (=aplikace) se přímo připojuje na proxy (např. `proxy.inf.upol.cz`)
- Po připojení řekne proxy: „Chci se připojit na `server.nekde.v.internetu`“
- Proxy toto připojení zprostředkuje
- Obrázek
- Jakým způsobem bude proxy a klient komunikovat?
  - Např. rozšiřující příkazy v protokolech (FTP, telnet, SMTP)
  - Nestandardní domluva – úprava aplikací klienta i proxy $\Rightarrow$ nákladné
- Klient potřebuje umět adresovat pouze proxy, překlad `server.nekde.v.internetu` již provádí proxy

Generická proxy

- Upravování klientských aplikací někdy není možné (proprietární software)
- Idea: Do proxy „zadrátujeme“ adresu cílového serveru a port
- Pro každý server, kam je potřeba se připojit vytvoříme instanci na jiném portu
- Obrázek
- Není potřeba dělat úpravy klientských aplikací
- Omezené množství koncových serverů
- Jednoduchá implementace, stejný program pouze s jiným nastavením cílové IP a portů
- Vhodné pro aplikační protokoly bez rozšířených příkazů (SSH, IMAP, POP, . . .)

Transparentní proxy

- RFC-1919 – pojednává o transparentní proxy, pěkné čtení
- Klient musí umět adresovat server v internetu, vytvoří pakety pro něj a normálně odešle
- Paket přesměrován na transparentní proxy
- Proxy vykoná dotaz(+ filtrace, cache) za klienta, příjme odpověď a vrátí ji klientovi
- Není potřeba upravovat (ani nastavovat) klientské aplikace
- Obrázek

Brána (gateway)

- = uzel podobný proxy, avšak mění aplikační protokol mezi serverovou a klientskou části
- Nejčastěji – brána HTTP $\Leftrightarrow$ FTP
  - SW brány generuje webovou stránku zobrazující obsah FTP
  - Poté i překlad přenášených dat
- Rozlišovat druhy bran
  - Aplikační – překládají aplikační protokoly (výše)
  - Síťové/protokolové – např IoT gateway (LoRaWAN, Zigbee gateway, . . .)
- Obrázek

Doporučená četba

- Ciampa M. Security+ Guide to Network Security Fundamentals, Course Technology, Cengage Learning. 2012. ISBN 9781111640170
  - RADIUS
- Hassel J. Radius. O'Reilly Media. 2002. ISBN 9780596003227
- Richer J., Sanso A. OAuth 2 in Action, Manning 2017. ISBN 9781617293276
- `https://learn.microsoft.com/en-us/windows/security/operating-system-security/data-protection/bitlocker/`
  - Bitlocker oficiální dokumentace