KATEDRA
INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMI/BEPS - Bezpečnost počítačových systémů

Poznámky z výuky *(2025 – 2026)*
Verze z *20. 04. 2026*

Vojtěch Netrh
vojtanetrh@gmail.com

Obsah

1 Zkouškové otázky

Zkouška proběhne ústní formou. Vylosujete si dvě otázky:

- Hlavní otázka - komplexnější téma, vysvětlíte více do hloubky.
- Vedlejší otázka - menší téma, u kterého vyžaduji pouze přehledovou znalost, pochopení principů.

Po vylosování otázky budete mít cca 20 minut na písemnou přípravu, poté budete mít prostor prokázat své znalosti. Otázky označené „Počítačové sítě:“ nebyly probírány v tomto předmětu. Jsou ale natolik důležité pro síťovou bezpečnost, je potřeba je znát a budu je zkoušet.

1.1 Hlavní otázky

1. Počítačové sítě: Architektura TCP/IP (obecně), fyzická a linková vrstva (konkrétně)
2. Počítačové sítě: IP protokol
3. Počítačové sítě: Protokoly TCP a UDP
4. Autentizace uživatele – metody, jednorázová hesla, rekurentní algoritmus
5. Symetrická a asymetrická kryptografie
6. RSA
7. Kryptografické hashovací funkce, digitální podpis
8. Public key infrastructure
9. Email a jeho bezpečnost
10. Protokoly SSL/TLS
11. Zabezpečení webových aplikací
12. Bezpečnost síťových technologií
13. Bezpečnost bezdrátových technologií
14. VPN
15. IPsec
16. Filtrace, proxy, brány
17. RADIUS
18. Škodlivý software a obrana
19. Chyby HW, buffer overflow

1.2 Vedlejší otázky

1. Legislativa – průnik do počítačových systému, osobní údaj
2. Blokové šifry – DES a AES, jejich aplikace
3. Bezpečnost RSA
4. Časová razítka
5. Nevyžádaná pošta, útoky na uživatele, phishing
6. Diffie-Hellmanova výměna klíče
7. SQL injection
8. Prolomení hashovacích funkcí
9. WireGuard
10. Prolomení WEP a WPA
11. Tor
12. OAuth 2.0
13. Eliptické křivky
14. Blockchain

2 Počítačové sítě

2.1 TCP/IP protokol

Běžně se používá dnes v internetu. Architektura má 4 vrstvy (ty využívají služby mezi sebou navzájem). Je nutné definovat **protokoly**, což jsou přesná pravidla podle kterých probíhá komunikace. Původně kladeny nulové nároky na bezpečnost, ty se dodaly až postupně v budoucnu (musela být zajištěna ale i zpětná kompatibilita) např. jako mezivrstvy.

> **Příklad 1**
>
> Rozkouskovaný požadavek počítače, který chce navštívit webovou stránku na nějaké adrese.

2.2 Protokol TCP

2.3 Protokol UPD

2.4 Autentizace uživatele

Proces při kterém se ověřuje uživatel/počítač že je to opravdu on. **Autorizace** je udělení konkrétních práv po tom co proběhne autentizace.

Základní způsoby:

- uživatel zná něco (heslo, hash) unikátního
- uživatel má něco unikátního - soukromý klíč, přístup k jinému zařízení
- biometrika uživatele

V reálném světě se obvykle věří nějaké důvěryhodné autoritě a za padělaní hrozí postih (opora v zákoně, ten v technickém světě tak neplatí).

2.4.1 Autentizace heslem

Rozhodující je dnes délka hesla (12-16 znaků je dnes bezpečné). Záleží i na typu útoku (*brute force* vs *rainbow tables*).

Moderní jsou správci hesel (i doporučené).

Dříve se hesla někdy přenášeli v plaintext podobě, dnes už snad vymizelo.

Řešení pro určité případy mohou být i jednorázová hesla. Dříve spíše z pevně daného seznamu (nepraktické – omezená velikost, atd.). Dnes se používá rekurentní algoritmus nebo se předá jiným kanálem (obvykle jako druhý faktor ověření).

> **Rekurentní algoritmus** **Definice 2**
>
> Máme **jednosměrnou funkci** f. Obvykle se používají **hashovací funkce**. Zápis $f^n(x)$ značí n-krát použití funkce f s počátečním řetězcem x (řetězení funkce).
>
> 1. Inicializace – uživatel zvolí tajný řetězec (*seed*) a tajné číslo n
> 2. Spočítá f^n(seed) a tuto dvojici odešle
> 3. [TBA]
>
> Standard **S/KEY** a **OTP**. Tam se to používalo (konkrétně využívalo funkci MD4, která už není bezpečná).

Sociální inženýrství **Poznámka 3**

Cílíme na nejvíce zranitelný článek systému, což je člověk. Využívá se psychologie a manipulativních technik. Takže nejde o krádež údajů jako takovou, ale uživatel je sám nedobrovolně prozradí. Často se na uživatele vyvíjí časový nátlak (neznalý člověk snadněji podlehne). Nejznámější člověk z oboru byl Kevin Mitnick.

3 Kryptografie

[Část která je převzatá od docenta Bartla a je k nalezení v materiálu KMI/KRY.]

Jedná se o tyto tři okruhy:

1. Symetrická a asymetrická kryptografie
2. RSA
3. Kryptografické hashovací funkce, digitální podpis

3.1 Symetrická kryptografie

3.2 Asymetrická kryptografie

3.3 RSA

3.3.1 Praktické aspekty a bezpečnost RSA

Navíc: postranní kanál, poslouchání mikrofonem a měření spotřeby.

3.4 Kryptografické hashovací funkce

3.5 Certifikáty

3.5.1 Digitální podpis

3.5.2 Public key infrastructure

4 Bezpečnost e-mailu

5 Protokol SSL/TLS

Běží na transportní vrstvě (ta je defaultně nezabezpečená). Pro nutnost zabezpečit data na této úrovni vznikly tyto 2 protokoly. Jedná se o dva navzájem nekompatibilní protokoly. Protokol TLS byl historicky první a dnes se již nepoužívá, je nahrazen SSL.

5.1 Jak to funguje?

Jedná se o mezivrstvu mezi aplikační a transportní vrstvou. Architektonicky se jedná o typ *klient-server* (obousměrné). Serverová strana je musí prokázat (autentizovat) certifikátem, pro klienta je to dobrovolné. Šifrování se provádí symetrickou šifrou (kvůli rychlosti). Data z aplikační vrstvy je nutné rozdělit na fragmenty, protože jsou moc velká.

5.2 Struktura SSL

Jeví se jako jediný protokol, ale je složen ze 4 dílčích:

- *Record layer protocol RLP* (manipuluje s daty),
- *Handshake protocol HP* (ustanovuje komunikaci mezi stranami pro RLP),
- *Change cipher specification protocol CCSP* (jediný účel – nastavit budoucí svitu pro HP),
- *Alert protocol AP* (servisní protokol pro předchozí tři)

[Znázornění komunikace na obrázku]

5.2.1 Record Layer Protocol

Rozděluje data na části o max délce 2^{14}. Případně může provádět kompresi. Počítá kontrolní součet a šifruje (na straně příjemce „opačné“ operace). Úvodní data HP jsou nešifrovaná.

V hlavičce RLP se vyskytuje - typ dat (1B), verze (2B), a délka dat (2B).

5.2.2 Change cipher specification protocol

Definuje jedinou zprávu CCS, která dává najevo, že odesílatel přešel na nový způsob šifrování. Může být i uprostřed jediného TCP segmentu.

5.2.3 Handshake protocol

Možnost obnovení relace (rychlejší než navazování nové).

5.2.4 Alert protocol

Slouží ke správě chyb během komunikace. Jsou dva typy – upozornění a fatální chyba (po ní se komunikace končí). Jedná se o 2B zprávy. Domluveno desítky chybových kódů.

5.3 Zranitelnost SSL

5.4 Protokoly využívající SSL/TLS

6 Zabezpečení webových aplikací

Potenciálně výrazně zranitelné, protože jsou dostupné relativně všem a odkudkoliv.

6.1 HTTP basic auth

6.2 Nasazení SSL

6.3 Bezpečnost při přihlašování/registraci

6.3.1 Session hijacking

6.3.2 Cross Site Request Forgery

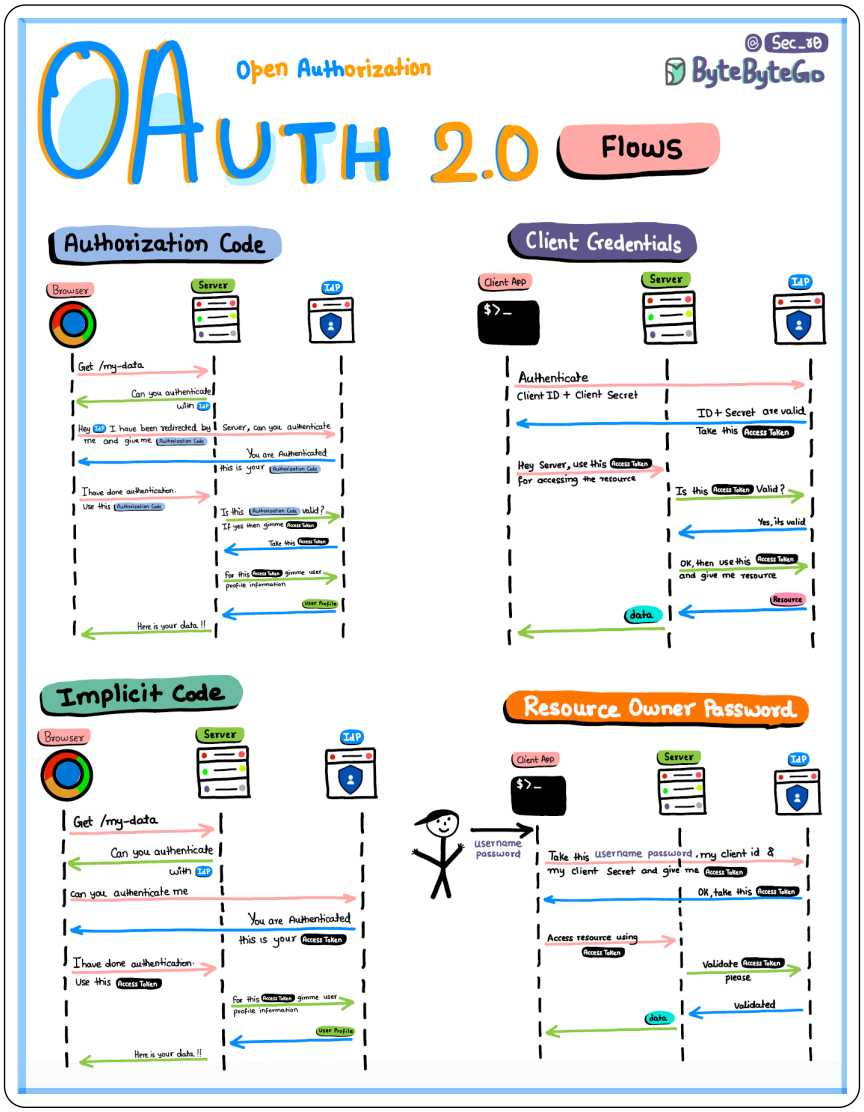
6.4 SQL injection

6.5 Open Authorization (OAuth) 2.0

Jde o autorizační framework používající se převážně ve webových službách. Umožní webové službě poskytnou (některá) data o uživateli aniž by služba znala heslo[1]. Dnes se používá velmi často a má také širokou podporu ve webových frameworcích.

Středobodem je autentizační token. Ten si můžeme představit jako kartu k pokoji v hotelu. Nic neříká o uživateli, pouze mu dává přístup.

[1] Takové to *sign-in using Google, Github, …*



Obrázek 1: OAtuh fungování

7 Škodlivý software (malware)

Ověření fungování SW a že dělá to co chceme. Nikdy to není 100%, ale primitivní metody a pravidla, jejichž dodržování nám dává určitou záruku. Případně ověření podpisu (hashe) aplikace.

Kill switch

Pouze úmyslá „zlá“ funkcionalita. Čili SW dělá něco co uživatel nechce a na uživateli parazituje. Obecně může dělat vše na co má daný software oprávnění (což je často fakt hodně věcí). Např. síŤová komunikace, vytěžování zdrojů, zobrazování reklam, odposlech (kamera, mikrofon, klávesy).

Máme několik typů škodlivých softwarů (malware):

1. Vir – program modifikující ostatní programy (většinou šíření sebe sama)
2. Trojský kůň – funkční program, který přidává nechtěnou funkcionalitu navíc
3. Backdoor – skrytá funkcionalita přidána někým od distributora (obvykle obejití autentizace)
4. Červ – vir šířící své kopie přes síť

5. Spyware – šmírovací software
6. Adware – zobrazování reklam
7. Ransomware – způsobení škody a požadování výkupného (největší otázkou je zda opravdu dodrží slovo)
8. Kontrola počítače – např. botnet
9. Jokeware
10. Kombinace více typů

7.1 Kvalita viru

Několik možných parametrů podle kterých určovat kvalitu.

Špatná detekovatelnost – jak uživatelem, tak i případnými antiviry

Obtížné zneškodnění – změna pozice na disku, tzv. morfovací techniky

Rychlé šíření – jak lokálně, tak po síti

(Ne)náročnost vytvoření a **platformní nezávislost** – cílení na jedinou platformu vs cílím na vše (to má obvykle vliv i na dobu vývoje, trochu se změnilo s AI)

Užitečnost pro tvůrce – primárně asi peněžní obohacení

7.2 Šíření

Dříve s fyzickými vyměnitelnými médii (už pravěk). Později `.exe` soubory šířené obvykle přes e-maily a skrývání přípon (relativně rychle fixnuto). Dnes se nejběžněji šíří přes chyby v PDF souborech (obvykle to souvisí i s prohlížečem PDF). Také běžně přes sociální inženýrství, že někdo někoho donutí spustit specifický program. Síťové chyby v aplikacích (nejčastěji webové aplikace).

Důležité je, že virus se musí spustit. Když jen sedí na disku je vlastně neškodný. Obvykle viry připojují svůj kód k jinému programu, ideálně ho ještě nějakým způsobem zamaskují. Vhodné reagovat i na skončení dané aplikace, třeba zašifrováním a opětovným uložením na disk.

7.3 Oblíbené cíle virů

Nejlepší možností je *boot sektor*. Když vir zvládne získat kontrolu nad boot sektorem, tak může převzít kompletní kontrolu nad celým počítačem a stává se podstatě nedetekovatelným (znázornění na obrázku jak stojí vrstvy nad sebou). Dalším místem jsou tzv. *resident routines*[2], kde se obvykle jedná o nízké funkcionality jako obsluha kláves nebo ošetření chyb. Další možnosti jsou: napadení knihoven (často používané), samotný OS, makra v Excelu.

7.4 Detekce a odstranění malware

Obecně to nejde, ale snažíme se poučit z minulosti a hledáme **vzorce chování**. Přístup který se vlastně vytváří antiviry. Ustanoví se **vzory (signatury) virů** podle kterých se pracuje. Databáze se musí udržovat co nejvíce aktuální.

Po detekci musíme všechny instance zastavit (obvykle v *nouzovém režimu* OS). Kontrola se ideálně provádí ještě před bootem OS. Po odstranění je dobré najít i cestu, kterou se do systému vir dostal a tu zabezpečit.

[2]Funkce OS, které jsou neustále načtené v paměti kvůli rychlosti

Klasifikace virů je složitá a zabývají se tím celé týmy, které je zkoumají a rozdělují do různých kategorií (ty jsou relativně ustálené).

7.4.1 Taktiky proti detekci

V podstatě hra na kočku a myš, obě strany se snaží neustále zlepšovat. Stejně jako všude jinde jsou zlí tvůrci o krok napřed. Viry se snaží mutovat sami sebe, včetně vkládání „zbytečných operací“. Velké zapojení AI u obou stran.

7.4.2 Intrution detection system (IDS)

Obvykle monitorují síťové chování i konkrétního stroje na základě toho rozhodují. Často se jedná i o celý HW stroj. Kontrolují: konkrétní porty, vytíženost stanic. Používá se strojové učení, které porovnává běžné chování s aktuálním. Mohou provádět i zásahy (vypnutí portu, zabití aplikace).

7.4.3 Honeypot

Falešný uzel, který se snaží nalákat škodlivý software a z jeho působení pak sleduje co dělá a zavádí opatření (i pro celou síť). Řada již hotových řešení, které stačí do sítě implementovat.

7.5 Nedůvěryhodná zařízení

Často se zapomíná na fyzické zabezpečení systému. To nám může zhatit i výborné šifrování a hesla. V dnešní době se dá zaheslovat BIOS či šifrovat celý disk. Dnes i malá zařízení (velikost flash disku) jsou schopná mít obrovskou funkcionalitu, které umožní napadnout a ovládnout systém, v horším případě i celou síť. Příkladem pro tyto zařízení může být *USB killer* či *Rubber ducky*, ale i řada dalších[3].

8 Chyby SW i HW

Obecně v dnešní době je SW extrémně komplexní, čímž roste velikost kódu, počet používaných knihoven a kontrola nad takovým množstvím věcí je náročnější. Při používání knihoven a protokolů bychom neměli konat bezmyšlenkovitě a měli bychom alespoň zhruba vědět co to má dělat. Naštěstí i podpůrné nástroje pro programátory jsou lepší a samy pomáhají odhalit řadu chyb.

Bezpečností chyba **Definice 4**

Chyba v SW, které umožní uživateli dělat to, na co nemá oprávnění.

Máme několik různých variant:

1. Arbitrary code execution (ACE) – spuštění libovolného kódu na počítači
2. Remote code execution – vzdálené spuštění libovolného kódu
3. Eskalace práv – navýšení práv uživatele na `root`
4. Neoprávněný přístup do paměti – aplikace může zapisovat/číst na místě paměti jiných procesů[4]
5. Útěk z kontejneru – aplikace opustí kontejner (bezpečné prostředí) určený pro běh

[3] Je možnost mít i hardwarový packet sniffer na ethernetový kabel.
[4] Mohou se tam nacházet privátní klíče, hesla atd.

Exploit **Poznámka 5**

Kód využívající chyby v aplikaci pro svůj prospěch.

8.1 Správa chyb

Chyby by se neměli tajit, ale zveřejnit a donutit správce k updatu či tvůrce SW k opravě.

Vznikla databáze a sytém označování *Common Vulnerabilities and Exposures (CVE)*, který každé chybě přidělí číslo (např. CVE-2014-0160) a zveřejní bezpečné detaily.

Proces obvykle začíná pouze zveřejněním, že je nějaká chyba, aby se správci mohli připravit na update. Bližší informace (exploity, testy, detaily) se zveřejní až později, aby nemohly být ihned zneužity.

Velké firmy mají tzv. *BugBounty* programy, kde nabízí finanční odměny (poměrně štedré) pokud někdo z veřejnosti chybu objeví a společnosti nahlásí.

8.2 Typy chyb

V této kapitole jsou popsány základní typy chyb společně s problémem který způsobují a případnou obranou proti nim.

8.2.1 Buffer overflow

Pokud v nízkoúrovňovém jazyce nekontrolujeme velikost bufferu, do kterého se něco zapisuje, tak může dojít přepsání paměti jiné aplikace. V lepším případě aplikace jen spadne. V horším je možné spustit libovolný kód nebo skočit jinam do kódu.

Obranou jsou *stack canaries* – přidání náhodného čísla před návratovou hodnotu pro kontrolu nebo použití *NX bitu*, který rozdělí paměť na data a kód.

8.2.2 Format string attack

Formátování řetězce a nekontrolování počtu argumentů způsobí, že číst ze zásobníku nebo na něj dokonce zapisovat.

Obranou je kontrola počtu argumentů.

8.2.3 Integer overflow attacks

V celočíselné aritmetice mají čísla omezenou délku na počet bitů. Při sčítání velkých čísel může hodnota přetéct maximální možnou. To umožní psát do místa na paměti, kam by uživatel neměl.

8.2.4 Command injection attacks

Pokud se v programu přímo volá `shell` může dojít k vložení škodlivého příkazu, který `shell` vykoná. Podobný přístup jako SQL injection attack.

8.2.5 Time of check / Time of use attacks

Případ kdy se časově oddělí kontrola oprávnění a provádění operace. Například při zápisu do specifického souboru.

Obranou je otevření souboru a poté ihned zkontrolovat oprávnění (funkce `fstat()`).

8.2.6 XML attacks

Zpracování XML může být složitější neboť se nejedná o triviální záležitost a v parserech mohou být chyby. Nejtypičtější se využívají `XML entity`, které načtou něco co nikdy nekončí nebo soubory, které by neměly být viděny.

8.3 Chyby v HW

Stejně jako u SW, i zde roste komplexnost a s tím i možnost chyby. Instrukční sady jsou různě emulované a je dokonce možné mít OS přímo na procesoru (viz Intel a Minix).

Problém může nastat pokud jsou chyby pevně svázány s designem HW a nedají se opravit aktualizací mikrokódu.

> **Poznámka 6**
>
> Bezpečností chyba v CPU se vůbec tou nejzávažnější chybou.

8.3.1 Meltdown

Pro zvýšení rychlosti se instrukce provádí paralelně (tzv. *out of order*). CPU se pak stará o synchronizaci a předpokládá, že instrukce skončí dobře. Po dokončení obsluhy výjimky můžeme postranním kanálem (konkrétně díky času) zjistit kam do paměti se přistupovalo a data si následně přečíst.

```
C++
unsigned char *ptr = /* 0x12..78 */; // adresa hledanych dat
unsigned char value = 0; // prectena hodnota
unsigned char probe_array[256 * 4096]; // pomocne (velke) pole

int main() {
  signal(SIGSEGV, handle_sigsegv);  // inicializace
  clflush();
  // precte byte z pameti a muze skoncit vyvolanim vyjimky
  unsigned char t = *ptr;
  // pristup do probe_array, bude proveden mimo poradi
  unsigned char x = probe_array[t * 4096];
  value = t; // pokud nedoslo k vyjimce
:read_complete
  // na toto misto se vrati kod z obsluhy vyjimky
  printf("Hodnota: %i\n", value);
  return 0;
}
```

8.3.2 Spectre

Velmi podobné jako Meltdown, ale hůř se využívá. Spoléhá se na spekulativní vykonání `if` větví. Pokud se mu spekulace povede, tak se přístup zrychlí, pokud ne tak výsledek zahodí. Jenže i zahozený výsledek je možné v L1 či L2 cache procesoru najít.

Pro provedení musí nejdřív útočník donutit procesor si myslet, že danou instrukci je dobré spekulativně provádět, protože podmínka platí. Poté podstrčí čtení z paměti, kde mohou být uloženy útočníkem chtěná data a procesor spekulativně proveden operaci. Jenže realita je jiná

a podmínka neplatila a data je nutné zahodit. Díky postrannímu kanálu je však možné (podle rychlosti/času) zjistit kam se přistupovalo a data nějakým způsobem z daného místa získat.

Má 2 verze, druhá se zaměřuje na *branch prediction* a *branch target buffer*, kde jsou adresy pro nepodmíněné skoky.

9 Bezpečnost síťových technologií

Ústředním tématem je protokol *ethernet*, který slouží ke komunikaci mezi sousedními uzly a funguje na linkové vrstvě ISO/OSI modelu. Využívá MAC adresy. Rámce se přenáší všesměrově a na síťových kartách je možné využít **promiskuitní režim**, který umožní zachytit a zobrazit rámce, které nejsou mě určené. Rámce mají sice kontrolní součet, ale slouží pouze ke kontrole technických chyb. Útočník může data pozměnit, kontrolní součet přepočítat a nic se nepozná.

Existují 2 varianty: nepřepínaný a přepínaný ethernet. U nepřepínaného jsou uzly spojeny hubem a tím pádem vidí všechno (dnes se již nepoužívá), u přepínaného jsou uzly spojené switchem, který řídí komunikaci mezi nimi a data přeposílá jen tam, kam patří.

9.1 ARP protokol

Pro IP komunikaci musíme logicky znát IP adresu dané síťové karty. Je nutné přiřazení IP a MAC adres, k čemuž slouží **protokol ARP**. Ten používá všesměrový rámce, kde se ptá kdo má danou MAC adresu. Stroj poté odpoví správnou IP adresou.

> **Poznámka 7**
>
> Uzly si udržují vazby v ARP cahce. Příkaz `arp`.

> **Poznámka 8**
>
> Odhalení uzlů v síti se dá docílit nástrojem `nmap`.

9.1.1 ARP spoofing

Jde o podvržení MAC adresy routeru (brány) v ARP cache oběti a zároveň podvržení MAC adresy oběti v ARP cache routeru. Tím docílíme MITM útoku, jelikož daná komunikace uzlu s routerem poteče přes nás. Dokonce existuje nástroj `ettercap`, která útok umožní realizovat i s GUI.

Obranou je používat statickou ARP cache či filtrovat odpovědi bez dotazu na speciálních typech switchů. Případně následná kontrola MAC a IP adres u všech paketů – tzv. *dynamic IP lockdown*.

9.1.2 MAC flooding

Switche a routery si udržují CAM (content addressavle memory) tabulku, kde jsou vazby MAC adresa a k ní náležící port. Útočník generuje pakety s náhodnými MAC adresami a tím tuto tabulku přeplní a dojde k jejímu resetování. Každý router však může mít reakci mírně odlišnou.

Obranou je detekce IP provozu nebo statické nastavení managovatelného switche.

9.1.3 Port stealing

Jedná se o MITM útok. Útočník bude simulovat přepojení oběti do jiného fyzického portu (musí dojít k aktualizaci CAM tabulky). Nevýhodou pro útočníka je, že každá komunikace od oběti útok naruší a útočník musí neustále reagovat.

Obranou je možnost statického nastavení CAM nebo aktivní obrana oběti. Otázkou je jak správně odlišit oběť a útočníka.

9.1.4 Další útoky

- ICMP redirect – generování falešných ICMP[5] zpráv
- DHCP spoofing – vytvoření falešného DHCP serveru v síti
- DNS spoofing – podvržení odpovědi na DNS dotazu

9.2 Protokol PPP a PPTP

Celým názvem *Poin to Point Protocol* a *Point to Point Tunneling Protocol*. Tyto protokoly se používají k pro připojení k internetu přes telefonní linku (i dnes – xDSL nebo optika).

- PPP funguje na L2 vrstvě a používá se ke spojení mezi 2 uzly. Umožňuje autentizaci (PAP) a zapouzdření.
- PPTP slouží k vytváření VPN tunelů. Autorem je Microsoft. Používá šifrování MPPE a protokol GRE k přenosu paketů. Dnes je považován za nebezpečný.

Aktuálně je jejich funkcionalita nahrazena jinými a lepšími. Tyto obsahují známe zranitelnosti.

9.3 Extensible Auth Protocol (EAP)

Zajišťuje že zařízení se podle něj budou autentizovat v budoucnu. Nejedná se o jediný autentizační protokol, ale rámec, kde se konkrétní metoda dohodne později.

- EAP-MD5 – obdoba CHAP[6]
- EAP-TLS – využívá Certifikáty
- EAP-PSK – využívá pre-shared key

9.4 Filtrace

Funguje na různých úrovních (vrstvách). Tím můžeme docílit různých pravidel i zabránění různých útoků. Většinou to co je přísnější bývá i pomalejší a může omezit i některé chtěné scénáře.

9.5 Firewall

Jedná se o aplikaci či samostatné HW zařízení, které provádí filtraci provozu a případné logování. Může aktivně bránit různým vstupům. Pro větší provoz je potřeba obecně vysoký výkon, protože přes to jde celý provoz. Jeho nastavení probíhá formou pravidel `if ... then` (těch mohou být až tisíce).

Některé podporují i inspekci SSL provozu (brání MITM útoku). Tento přístup je z hlediska etiky a práva spíše hraniční.

[5] Protokol pro odesílání služebních informací o komunikaci.
[6] Nástupce PAP protokolu.

Pravidla je dobré komentovat mít logicky rozdělené. S tím souvisí i dělení sítě a jejich pojmenování. Dobrá ochrana i proti *bus factoru*. Pravidla se prochází shora a jakmile se narazí na první, které se může použít, tak procházení končí.

Obrázek 2: Fyzický firewall

Ukázka firewall pravidel z Mikrotiku **Příklad 9**

9.6 Skenování portů

Na specifických portech běží specifické služby. Naskenováním můžeme zjistit zda ta služba na daném stroji běží (dostaneme odpověď).

Při etickém použití velmi užitečné a je nutné používat. Pokud se toho chopí útočník dá mu to celkem značný přehled o síti a různých službách (počet stanic, bežící služby, verze služeb, …).

Poznámka 10

Masivní skenování sítě může být považováno za útok!

Nástroje ke skenování sítí **Poznámka 11**

Služby jako `nmap` či `shodan`.

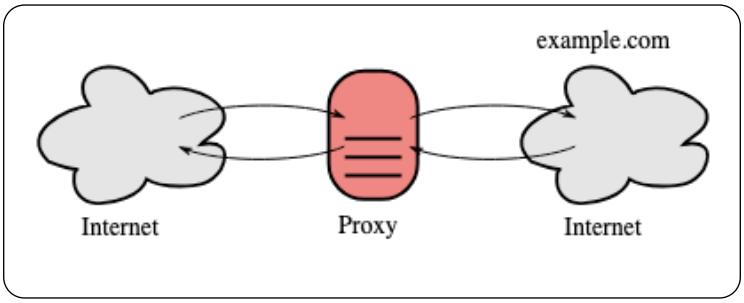
9.6.1 Typy skenování portů

1. TCP connect – vykoná celý třífázový handshake; trvá dlouho, nejspíš ho systém zaloguje
2. TCP SYN – odesílá se pouze `SYN` paket a `ACK` už se neposílá
3. TCP FIN – na port odešle pouze `FIN` paket a na základě (ne)dodržení RFC 793 se můžeme dočkat zajímavé odpovědi
4. TCP Null – obdobné jako TCP FIN
5. UDP – odešle paket na cílový port, na UDP ale není vytvářeno spojení

9.7 Proxy

Jedná se o aplikaci, které poskytuje službu počítačům ve vnitřní síti. Tím klienty zastupuje a něco za ně vykonává. Obvykle jde o prostředníka mezi sítěmi. Musí znát aplikační protokoly, které používá. Slouží k filtraci a cachování. V dnešní době se už tolik nepoužívá

Některé protokoly se nativně chovají jako proxy – DNS nebo SMTP.



Obrázek 3: Proxy server

Fungování: Při běžném fungování jsou pravidla na staveny na klientovi a ten kontaktuje proxy server s tím co chce udělat. Proxy tento požadavek sprostředkuje. Typicky se může jednat o HTTP připojení.

9.7.1 Generická proxy

Generická proxy je typ síťového prostředníka, který zprostředkovává připojení mezi klientem a pevně definovaným cílovým serverem, aniž by analyzoval obsah přenášených dat. Na rozdíl od běžné proxy, klient nemusí znát nebo specifikovat cílovou adresu; proxy je „natvrdo“ nasměrována na jeden cíl.

9.7.2 Transparentní proxy

Klient musí umět adresovat daný server v internetu (vytvořit pakety a normálně odeslat). Paket je potom automaticky přesměrován na transparentní proxy, ta vykoná dotaz a vrací odpověď. Nepřichází o možnost filtrace a cache. Klient o proxy neví.

9.8 Gateway (brána)

Podobný jako proxy, ale mění aplikační protokol mezi serverovou a klientskou částí. Nejčastější je brána HTTP ↔ FTP (brána generuje webovku zobrazující obsah FTP).

9.9 SOCKS

Jedná se o společnou proxy pro všechny aplikační protokoly. Nemusíme tedy pro každý vytvářet zvláštní proxy. Definován v RFC 1928.

9.10 Remote Authentication Dial-In User Service (RADIUS)

Jedná se o protokol, který spravuje uživatele v síti a zajišťuje jejich autorizaci a autentizaci. Existuje pro něj řada implementací, které jsou jak placená tak zdarma a poskytují různou míru komplexnosti. Je zde možnost *roamingu* – spolupracující organizace umožní autentizaci uživatelů (např. náš eduroam).

Využívá porty 1812 a 1813 na UDP.

9.10.1 Autentizace v RADIUS

Berme modelovou situaci kde uživatel přistupuje do Wi-Fi sítě na UPOLu. V roli RADIUS klienta není počítač uživatele, ale AP ke kterému se připojuje. AP si od klienta vyžádá jméno a heslo a vyřídí za něj komunikaci s RADIUS serverem.

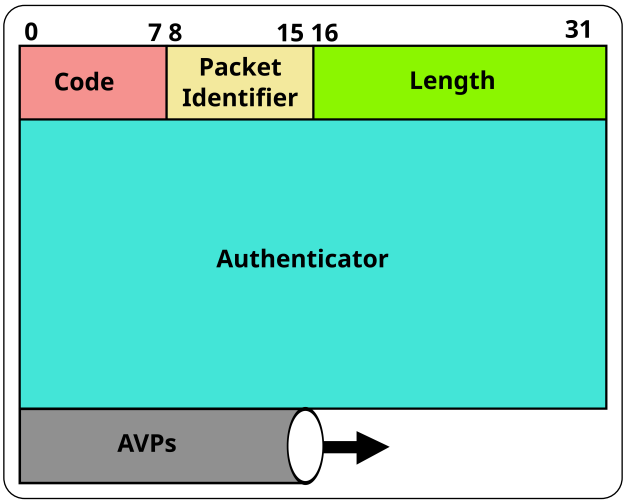
1. AP vytvoří *access request* s ID AP, autentizačními daty uživatele
2. request šifrovaně (pomocí PAP, CHAP, EAP) odešle na RADIUS server
3. RADIUS server ověří správnost údajů a dá AP vědět jaké služby může uživatel používat

4. AP umožní přístup uživateli do sítě

9.10.2 Struktura paketu

Pro všechny typy zpráv má paket stejnou strukturu. Složení je následující:

- 1B pro kód zprávy
- 1B pro ID zprávy
- 2B pro délku zprávy
- 16B pro autentikátor požadavku a odpovědi
- volitelná délka pro atributy (64 jich je předdefinovaných)



Obrázek 4: RADIUS paket

10 Bezpečnost bezdrátových technologií

Elektromagnetické vlny (vzduch) fungují jako sdílené médium. To má řadu pozitiv (nižší náklady na budování sítě, …) i negativ (obtížné zamezení šíření, lehkých odposlech, ..). Standard IEEE 802.11[7] stanovil řadu parametrů – pásma (2,4 a 5 GHz), kanály, národní regulace, rychlosti, … Sítě byly původně nezabezpečené, později se přidalo (WEP, WPA, WPA2, …). Architektura sítě může být dvojí – ad-hoc nebo infrastrukturní (AP ↔ klient).

AP vysílá SSID, které klient zachytí a může odeslat požadavek na asociaci. SSID musí klient znát. Některé karty umožňují *monitor mode*, který umožní pasivní odchytávání paketů bez nutnosti asociace. AP pravidelně posílá beacon rámce, které obsahují potřebné info (SSID nemusí být).

10.1 Wired Equivalent Privacy (WEP)

Pouze jednostranná autentizace klienta vůči síti. Používá sdílený klíč (40-104 bitů). Jako šifra je použita symetrická proudová RC4. Klíčový proud se generuje ze sdíleného klíče a asociačního vektoru, který je posílán otevřeně. RC4 je slabá a při odposlechu většího množství dat ji lze lehce prolomit.

10.1.1 Prolomení WEP klíče

Iniciační vektor (24 bitů) pro proudovou šifru se generuje pro každý paket a je obsažen v hlavičce. Jelikož je velká šance, že se bude opakovat je možná pak uhádnout klíčový proud.

[7] Ecistuje ve více verzích 802.11a / b / ac / ax / …

Alternativně jde odhadnout z velkého množství ARP paketů, kde se dá domyslet chybějící informace.

Poznámka 12

Pro prolomení WEP klíče je potřeba cca 60 000 inicializačních vektorů.

10.2 Wi-Fi Protected Access (WPA)

Nástupce WEP z roku 2002. Rozšířili se možnosti autentizace, ale šifrování zůstalo stejně slabé, jen se přidal proměnlivý klíč. Byla přidána i integrita dat (něco jako digitální podpis), při jejímž narušení dojde k deasociaci klienta. Šifrování používá TKIP – RC4 šifra se 128b klíčem.

WPA-PSK je použití předsdíleného klíče a SSID. Velmi nebezepčný způsob nebo při odchycení handshake lze snadno dopočítat.

10.3 WPA2 / 802.11i

Jedná se o plnohodnotnou náhradu za WEP a WPA. Používá silné šifrování s proměnlivým klíčem či stálý klíč (PSK). Umožňuje předběžnou autentizaci k jinému AP, což je vhodné při pohybu a přepojení klienta. Bezpečné odhlášení a deasociace, které brání MITM útoku. Běžně routery podporují WPA i WPA2 na jediném rozhraní.

Pro šifrování používá *Cipher Block Chaining*, což jsou šifrované bloky, které zavisí na předešlých. Šifrou je AES se 128b klíčem.

10.4 WPA3

Nejnovější standard z roku 2018. Má 2 varianty *personal* (AES-128) a *enterprise* (AES-192 a SHA-384). Pre-shared key byl nahrazen *Dragonfly Handshake*, který je znemožňuje úvodního handshake a je odolný proti offline slovníkovým útoků.

10.5 Skenování a odposlech sítí

Máme aktivní a pasivní skenování. Při aktivním (již se moc nepoužívá) klienti odesílají broadcastově *probe requesty* a pamatují si AP, které jim odpověděli. U pasivního klient odposlouchává kanály, kde ze zachycených dat získá MAC adresu AP a pokud zachytíme pokus o připojení jiného klienta máme i SSID.

Nástroje pro zjišťování sítí **Poznámka 13**

`kismet` – komplexní nástroj pro skenování/odposlech Wi-Fi `airodump-ng` – obsahuje nástroj pro útoky i skenování

U nešifrovaných sítích je odposlech a zachycení dat triviální[8]. Neměli by ani existovat, ale na veřejných místech se často používají kvůli složité distribuci klíčů. Ale odposlouchávat tuto (cizí) komunikaci je v ČR nelegální.

10.6 Virtuální privátní sítě (VPN)

Virtuální sítě využívající infrastrukturu větší sítě sloužící k přidání bezpečnostních prvků pro přenos přes nezabezpečené kanály.

[8]Pokud data nejsou šifrována na vyšších vrstvách je jednoduché rozluštit o co jde.

Tunelový mód umožní zapouzdření IP paketů do paketů transportní sítě podle protokolu GRE RFC-1701. Jedná se o propojení geograficky oddělených lokací do jedné sítě. Máme 3 základní módy:

1. počítač ↔ počítač : WireGuard
2. router ↔ router : spojení dvou sítí přes nezabezpečenou síť (klient často neví)
3. počítač ↔ router : připojení 1 klienta do interní sítě

Na VPN klademe požadavky jako: řízení přístupu, zajištění integrity dat, zajištění důvěryhodnosti dat, zajištění původu paketů.

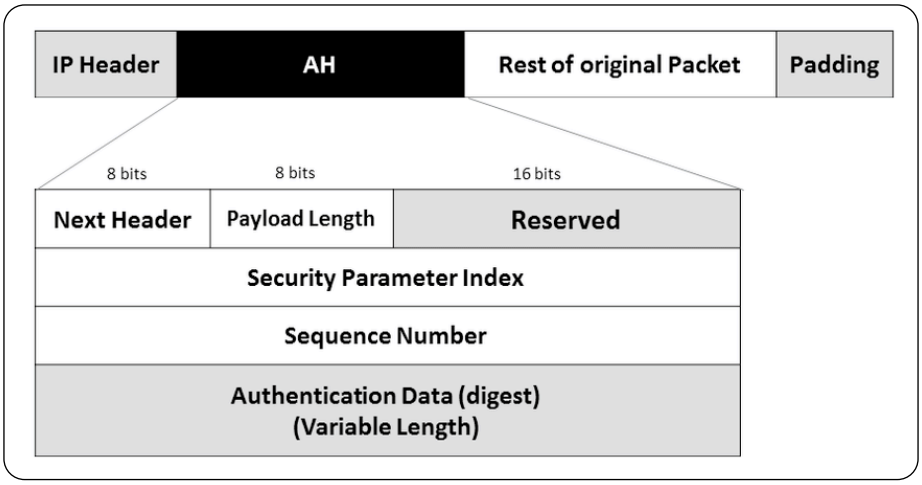
10.6.1 L2TP

Nástupce PPTP. Spolupracuje s protokolem PPP na autentizaci. Hlavička a data se posílají UDP. Neřeší důvěryhodnost a autentizaci. Často v kombinaci s IPsec (bez něj je nebezpečný). Široce podporován, ale dnes už máme lepší a bezpečnější protokoly. Obvykle pro přístup klienta do firemní sítě.

10.6.2 IPsec

Jedná se o soustavu protokolů zabezpečující komunikaci na IP vrstvě. Řeší komunikaci na úrovni počítač-počítač (aplikace o tom nemusí vědět). Původně byl nativní součástí IPv6 a později přidán do IPv4. Funguje ve 2 režimech: **transportní** a **tunelovací**. Transportní režim je jednodušší, mezi záhlaví IP a záhlaví vyšší vrství se vloží *bezpečnostní záhlaví* (specifikace jak jsou data zabezpečena). U tunelovacího režimu je zabezpečen celý IP datagram, který je vložen do nového a až potom putuje nezabezpečenou sítí. Samotné zabezpečení zajišťují 2 protokoly (AH a ESP).

Protokol IP Authentication Header (AH) zajišťuje integritu IP datagramů, autentizuje odesílatele a chrání proti útoku typu zopakování dat. Záhlaví má 6 částí viz obrázek.



Obrázek 5: IP authentication header hlavička

Protokol IP Encapsulating Security Payload (ESP) zajišťuje šifrování dat, integritu má jako volitelnou, ve vnějším IP paketu veden jako *protokol vyšší vrstvy – 50*, kvůli šifrování data zarovnána do bloku (komplikuje hlavičku)

Security parameter index (SPI) je jednoznačný identifikátor pro danou službu, zjednodušuje protokol, informace vedeny v *security policy database*, *security association (SA)* je trojice – IP adresa, SPI a protokol (AH nebo ESP)

Internet Security Association And Key Management Protocol (ISAKMP) je protokol pro dynamické naplnění SA databází, běží na portu 500/UDP, funguje na architektuře *initiator / responder*, obsahuje několik různých typů zpráv

Internet Key Exchange Protocol je samotný protokol výměny klíčů vystavěný nad ISAKMP, funguje ve 2 fázích – vytváření zabezpečeného ISAKMP kanálu a vytváření SA pro AH a ESP

Tyto protokoly jsou celkem komplikované. Systémy nemusí podporovat všechny najednou. Nastavování může být složité a připojování pomalejší. I z těchto důvodu tu máme WireGuard.

10.7 WireGuard

Zjednodušeně jde o odlehčeného nástupce IPsec. Od počátku je v linuxovém jádře, později implementace pro všechny OS. Samotný jeho kód je krátký a jednoduchý. OS ho vnímá jako síťové rozhraní. Pro zahájení komunikace jsou potřeba jen 2 zprávy (každá jedním směrem).

Ukázka nastavení wireguard **Příklad 14**

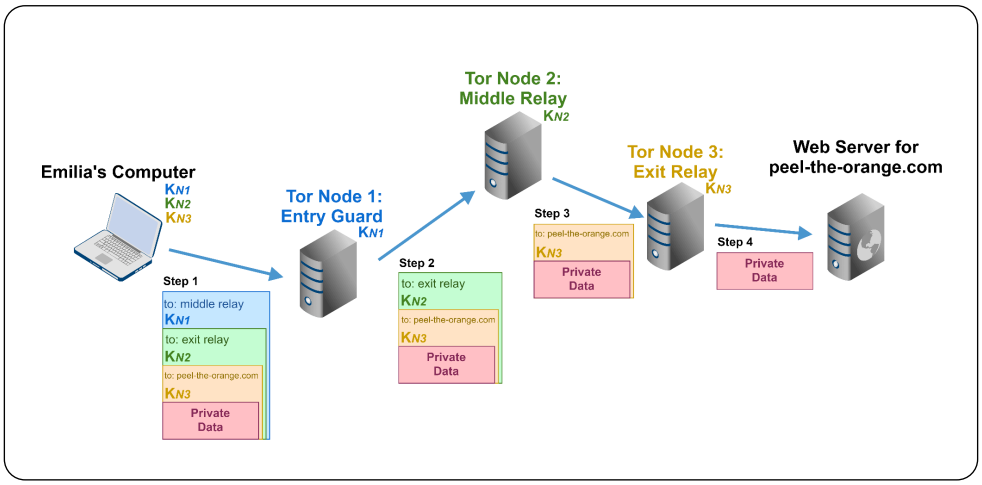
10.8 Tor (The Onion Router)

Jedná se o anonymní šifrovanou komunikaci v internetu. Efektivita je hodně upozaděna (velmi pomalé), ale klade se důraz na to být anonymní. Funguje jako vrstvy cibule (nabaluje se na sebe). Pro provoz jsou nutné mezilehlé uzly provozované dobrovolníky. Jsou 3 typy – *guard relay*, *middle relay* a *exit relay*. Pro šifrování se používá vícero veřejných a soukromých klíčů a data se nabalují na sebe.

Nejjednodušší je používat specializovaný prohlížeč. Samotné aplikace vůbec nemusí vědět, že běží přes Tor.

Poznámka 15

Využívá doménu `.onion`, která funguje pouze v této síti



Obrázek 6: Tor komunikace

S provozováním uzlů mohou být spojena bezpečnostní rizika. Musíme provozovatelům věřit. Ten kdo „páchá zlo“ je v podstatě relay. Adresy vstupních uzlů jsou veřejné, takže je relativně snadné je blokovat. Zvýší vůbec počet mezilehlých uzlů bezpečnost.

11 Šifrování a ochrana dat

Proč šifrovat ani nemusíme vysvětlovat. K nešifrovaným datům je možné získat přístup triviálně. Můžeme použít různé šifry (nejčastěji AES-192 a AES-256). Spíše než vlastní řešení je lepší využít již existující služby, které pouze voláme v naší aplikaci nebo přímo používáme.

Útočník by teoreticky mohl použít offline brute-force útok, ale ten je relativně pomalý.

Šifrovat můžeme jak jednotlivé soubory, tak celé adresáře či diskové oddíly. Přístupy jak to udělat jsou také různé (open-SSL knihovna, 7zip, šifrování přímo v aplikacích, …). Na Windows je možné zašifrovat celý adresář (využívá šifru RSA). V Linuxu se často využívá nástroj **`cryfs`**, který má jak CLI rozhraní tak UI. Výhodou je transparentní fungování a poměrně široká škála nastavení. V macOS funguje **`cryfs`** taky, případně je možné využít nativní nástroj jako DiskUtility.

Je možné jít až na úroveň šifrování při bootování. Tam je pak nutná spolupráce s UEFI. Avšak při recovery systému můžeme narazit na komplikace.

11.1 Bitlocker

Nástroj, který se používá ve Windows. Šifruje celý disk. Bitlocker spolupracuje s TPM čipem a UEFI secure boot. Je nutné myslet na uložení klíčů pro obnovu. Výhodou je nízká ztráta výkonu (max 10 %) a transparentnost pro aplikace.

11.2 Bezpečné mazání dat

Při běžném smazání dat se ve file systému obvykle jen nastaví flag, že místo je volné a může být přepsáno. Tím pádem je data možné s různými nástroji obnovit (celkově nebo alespoň částečně). Pro opravdové vymazání je nutné data přepsat (ideálně vícekrát). K tomu slouží nástroje jako **`shred`**, **`sfill`** či **`srm`**.

> **Poznámka 16**
>
> Pozor na to, že dešifrovaná data se někdy mohou nacházet v paměti nebo swap prostoru.