Identifikace mobilního síťového provozu pomocí TLS otisků

Přenos dat, počítačové sítě a protokoly (PDS)

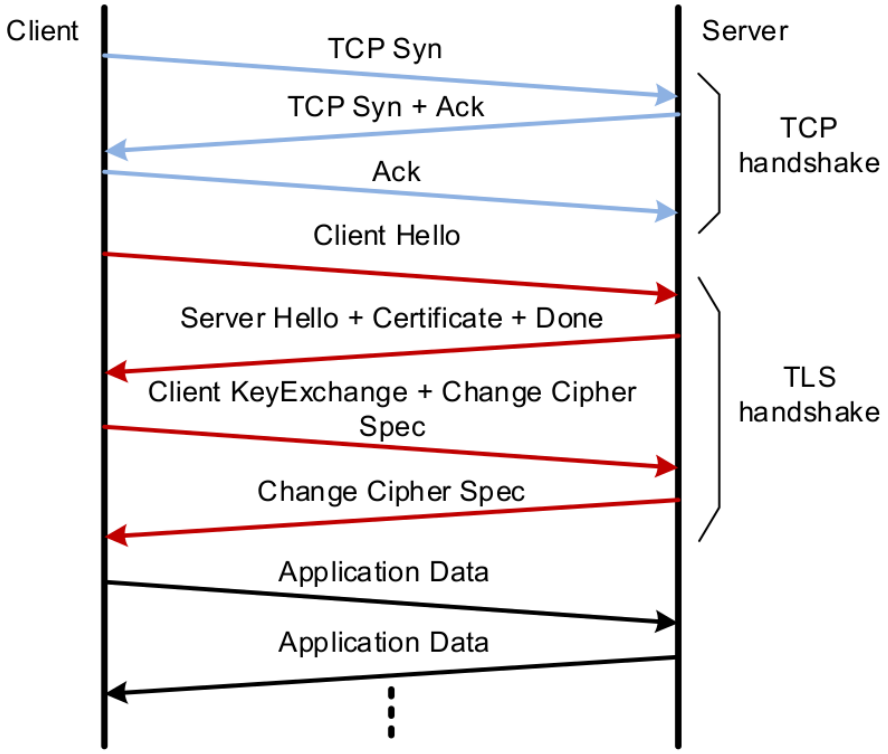
22. dubna 2021 Vladimír Dušek, xdusek27

Obsah

1 Popis problému

V současné době je již většina internetového provozu šifrována [1]. Jakkoliv šifrování zvyšuje bezpečnost a soukromí uživatelů, znesnadňuje na druhou stranu monitorovací a analytickou činnost. Tento projekt se zaměřuje na jednu z možných technik rozpoznávání aplikací v šifrovaném TLS (*Transport Layer Security*) provozu. Jsme omezeni výhradně na komunikaci mobilních platforem.

TLS je protokol, který pracuje nad TCP, kde zajišťuje soukromí a integritu dat pro komunikaci aplikací. Navázání TLS spojení probíhá pomocí tzv. TLS *handshake*. V této fázi komunikace jsou mezi klientem a serverem vyjednány informace o způsobu šifrování – metody pro výměnu klíčů, šifrovací algoritmy, verze, výměna šifrovacích klíčů, autentizace, apod [1]. Tato fáze komunikace probíhá pochopitelně nešifrovaně, veškerý další provoz již šifrovaný je.



Obrázek 1: Ustanovení TLS spojení. Obrázek je převzatý [1].

2 Analýza problému

Použitá technika rozpoznávání aplikací v šifrovaném TLS provozu využívá tzv. JA3 otisky (*fingerprints*). Ty byly představeny pány John B. Althouse, Jeff Atkinson a Josh Atkins v roce 2015.[1] Technika spočívá v tom, že ze síťové komunikace jsou vyfiltrovány TLS *handshake* packety. Z packetů *Client Hello* a *Server Hello* jsou vyextrahovány následující položky: *Handshake Version*, *Cipher Suites*, *Extensions*, *Supported Groups* a *Elliptic Curve Point Format*. Z těchto informací je spočítán MD5 hash (*Message-Digest algorithm*) – JA3 otisk.

Obrázek 2: Vytvoření JA3 otisku. Obrázek je převzatý [1].

Limitace tohoto řešení spočívá v poměrně nízkém počtu možných otisků. Možností ustanovení TLS spojení není příliš velké. Porovnáváním pouze JA3 otisku pro danou aplikaci by nebylo dostatečné. Pravděpodobně pokud dvě aplikace využívají stejnou TLS knihovnu, jejich otisk bude stejný. Proto jako příznak pro rozpoznávání je třeba využít více informací, například SNI (*Server Name Indication*).

[1] `https://github.com/salesforce/ja3`

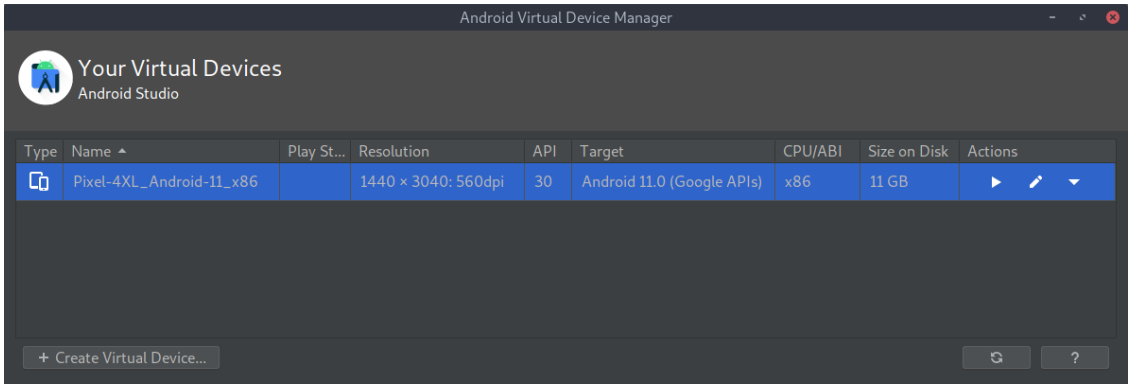
3 Řešení

3.1 Vytvoření trénovací sady

Pomocí nástroje pro vývojáře mobilních aplikací Android Studio[2] byl vytvořen virtuální mobilní telefon Google Pixel 4XL s operačním systémem Android 11 a procesorovou architekturou x86.

Jelikož na takto vytvořený virtuální telefon není možné instalovat aplikace přes vestavěný klient Google Play, bylo nutné využít repozitáře třetích stran ApkMirror[3] nebo ApkPure[4] a aplikace nainstalovat manuálně. Byl nainstalován následující seznam aplikací:

1. *Yr* v5.13.2,
2. *Settle Up: Group Expenses* v10.0.2030,
3. *Forest: Stay focused* v4.35.1,
4. *Forza Football: Live soccer scores* v5.1.13,
5. *Revolut* v7.43.1,
6. *GitHub* v1.7.4,
7. *Netflix* v7.97.1,
8. *Twitter* v8.88.0,
9. *YouTube* v16.12.34,
10. *Phoenix: The Bitcoin wallet from the future* v1.4.8.



Obrázek 3: Program Android Virtual Device Manager pro správu virtuálních mobilních zařízení.

[2] https://developer.android.com/studio
[3] https://apkmirror.com
[4] https://apkpure.com

Listing 1: Spuštění virtuálního mobilního telefonu přes příkazový řádek na konkrétním portu.

```
$ emulator -avd Pixel-4XL_Android-11_x86 -port 12345
```

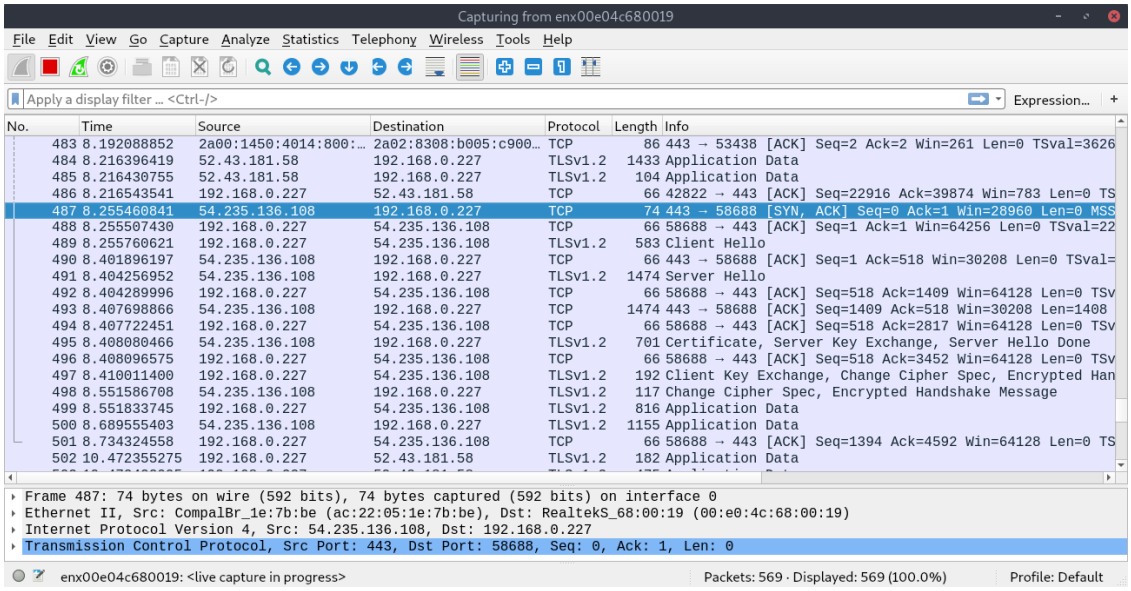
Obrázek 4: Virtuální mobilní telefon Google Pixel 4XL s operačním systémem Android 11 a nainstalovanými testovacími aplikacemi.

Pomocí klienta `adb` (*Android Debug Bridge*)[5] je možné komunikovat s virtuálním telefonem skrze příkazový řádek. Pro simulaci provozu jednotlivých aplikací byl použit nástroj `monkeyrunner`.[6] Ten lze přes `adb` spustit uvnitř telefonu. Programem Wireshark[7] je možné poté odposlechnout síťový provoz a uložit jej jako `pcap` soubor.

Listing 2: Spuštění 1000 náhodných událostí aplikace Phoenix.

```
$ adb -s emulator-12345 shell monkey -p fr.acinq.phoenix.mainnet \
    -v 1000
```

[5] `https://developer.android.com/studio/command-line/adb`
[6] `https://developer.android.com/studio/test/monkeyrunner`
[7] `https://www.wireshark.org`



Obrázek 5: Zaznamenávání síťového provozu na virtuálním mobilním zařízení pomocí aplikace Wireshark.

Tímto způsobem byla postupně zaznamenána komunikace každé z vybraných aplikací. Celý postup byl opakován 6×. Celkově tedy bylo sesbíráno 60 trénovacích `pcap` souborů. V každém z nich byla zaznamenána síťová komunikace během 1000 náhodných událostí aplikace vyvolaných nástrojem `monkeyrunner`.

3.2 Zpracování trénovací sady

Pro zpracování `pcap` souborů byl využit jazyk Python a knihovna Pyshark[8]. Pyshark pouze zaobaluje Tshark[9], což je terminálová verze Wiresharku, do Python knihovny.

Zachycený provoz aplikací neobsahuje pouze komunikaci mezi telefonem a serverem aplikace samotné, ale také komunikace aplikací na pozadí a spojení s reklamními a analytickými servery. Tyto spojení je nutné odfiltrovat. To bylo provedeno na základě sestavení *whitelistu* podřetězců serverů (doménové jméno serveru musí obsahovat daný podřetězec), pro jednotlivé aplikace. Doménová jména serverů byla vypozorována z komunikace jednotlivých aplikací.

Listing 3: *Whitelist* podřetězců komunikujících serverů pro jednotlivé aplikace.

```
{
  "revolut": ["revolut"],
  "twitter": ["twitter", "twimg"],
  "phoenix": ["phoenix", "acinq"],
  "netflix": ["netflix", "nflxext"],
  "forza": ["forza"],
  "youtube": ["youtube", "ytimg", "redirector.googlevideo"],
  "yr": ["yr"],
```

[8] https://github.com/KimiNewt/pyshark

[9] https://www.wireshark.org/docs/man-pages/tshark.html

```
  "forest": ["forest", "seekrtech"],
  "github": ["github"],
  "settleup": ["settleup", "settle-up"]
}
```

Z veškeré zachycené komunikace byly vyfiltrovány TLS *Client Hello* a TLS *Server Hello* packety. Z vybraných položek *Handshake Version*, *Cipher Suites*, *Extensions*, *Supported Groups* a *Elliptic Curve Point Format* byl spočítán JA3 otisk pomocí hashovací funkce MD5. Takto byl spočítán otisk každého vyfiltrovaného packetu. Ze seznamů *Cipher Suites*, *Supported Groups* a *Extensions* byly vymazány hodnoty GREASE (*Generate Random Extensions And Sustain Extensibility*), *renegotiation* a *padding*. Které se objevují náhodně, případně jsou redundantní a do komunikace zavádějí nedeterminismus [1].

Listing 4: Příklad výpočtu JA3 otisku z TLS *Client Hello* packetu.

```
{
  "handshake_version": "771",
  "cipher_suites": "
   49195,49196,52393,49199,49200,52392,49171,49172,156,157,47,53",
  "extensions": "0,23,10,11,5,13",
  "supported_groups": "29,23,24"
  "ec_point_format": "0",
  "ja3_fingerprint": "2595ab65e691eb1d942c6094ff92c933"
}
```

Příznaky aplikací tvoří trojice následujícího formátu: JA3 otisk spočítaný z TLS *Client Hello*, JA3 otisk spočítaný z TLS *Server Hello* a SNI (*Server Name Indication*), tedy doménové jménu serveru. Každé aplikaci připadá seznam příznaků. Všechny příznaky byly uloženy do souboru formátu `json`.

Odpovídající packety *Client Hello* a *Server Hello* byly propojeny na základě zdrojové/cílové IP adresy a zdrojového/cílového portu.

Listing 5: Ukázka z trénovací sady, příznaky aplikace Phoenix.

```
{
  ...
  "phoenix": [
    {
      "client_ja3": "709c6b4bc7678a83ccc13a529103c873",
      "server_ja3": "77f87cbba1f72fcf7fbc308d9dcb561d",
      "sni": "phoenix.acinq.co"
    },
    {
      "client_ja3": "2595ab65e691eb1d942c6094ff92c933",
      "server_ja3": "b5c7a4c92e1f26c18a37bc9c1acd64b5",
      "sni": "acinq.co"
    },
    {
```

```
      "client_ja3": "cb533a60549e1e6021bd620b77ffce72",
      "server_ja3": "9758f7a0e1460a60e317ae2131b733a5",
      "sni": "electrum.acinq.co"
    }
  ],
  ...
}
```

4 Testování a vyhodnocení

Pro otestování a vyhodnocení úspěšnosti detekce byla vytvořena testovací datová sada. Byly sesbírány 3 `pcap` soubory s komunikací každé z natrénovaných aplikací a 10 `pcap` souborů s neznámou komunikací. Testovací datová sada byla vytvořena stejným způsobem jako trénovací.

Z každého testovacího pcap souboru byly vyextrahovány všechny TLS *Client Hello* a TLS *Server Hello* packety. Byly spočítány jejich JA3 otisky a přes IP adresy a čísla portů byly propojeny. Takto byl sestaven příznak komunikace. Tímto způsobem byly z konkrétního pcap souboru vyextrahovány všechny příznaky. Ty byly následně porovnávány se známými příznaky v databázi. Aplikace která měla nejvíce shod byla vybrána.

Byly spočítány hodnoty TP (*true positive* – skutečně pozitivní), FP (*false positive* – falešně pozitivní), TN (*true negative* – skutečně negativní) a FN (*false negative* – falešně negativní) a z nich metriky *accuracy*, *recall*, *precision*. Kompletní výsledky testování jsou zaneseny v tabulce 1.

- $TP = 30,\ FP = 1,\ TN = 9,\ FN = 0$
- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = 0,975$
- $Recall = \frac{TP}{TP+FN} = 1$
- $Precision = \frac{TP}{TP+FP} \approx 0,968$

Jméno souboru	Detekovaná aplikace	Výsledek
forest-01.pcapng	Forest	✓
forest-02.pcapng	Forest	✓
forest-03.pcapng	Forest	✓
forza-01.pcapng	Forza	✓
forza-02.pcapng	Forza	✓
forza-03.pcapng	Forza	✓
github-01.pcapng	GitHub	✓
github-02.pcapng	GitHub	✓
github-03.pcapng	GitHub	✓
netflix-01.pcapng	Netflix	✓
netflix-02.pcapng	Netflix	✓
netflix-03.pcapng	Netflix	✓
phoenix-01.pcapng	Phoenix	✓
phoenix-02.pcapng	Phoenix	✓
phoenix-03.pcapng	Phoenix	✓
revolut-01.pcapng	Revolut	✓
revolut-02.pcapng	Revolut	✓
revolut-03.pcapng	Revolut	✓
settleup-01.pcapng	Settle Up	✓
settleup-02.pcapng	Settle Up	✓
settleup-03.pcapng	Settle Up	✓
twitter-01.pcapng	Twitter	✓
twitter-02.pcapng	Twitter	✓
twitter-03.pcapng	Twitter	✓
uknown-bitcoin.pcapng	×	✓
uknown-gmail.pcapng	Twitter	×
uknown-seznam.pcapng	×	✓
unknown-damejidlo.pcapng	×	✓
unknown-facebook.pcapng	×	✓
unknown-google_maps.pcapng	×	✓
unknown-google_photo.pcapng	×	✓
unknown-linkedin.pcapng	×	✓
unknown-livesport.pcapng	×	✓
unknown-stackoverflow.pcapng	×	✓
youtube-01.pcapng	YouTube	✓
youtube-02.pcapng	YouTube	✓
youtube-03.pcapng	YouTube	✓
yr-01.pcapng	Yr	✓
yr-02.pcapng	Yr	✓
yr-03.pcapng	Yr	✓

Tabulka 1: Tabulka s kompletními výsledky testování úspěšnosti detekce.

5 Závěr

Natrénový detektor využívající kombinace JA3 otisku pro *Client Hello* packet, JA3 otisku pro *Server Hello* packet a doménového jména serveru dosáhl velmi dobrých výsledků na testovacích datech. Byla zaznamenána pouze jediná chyba, kdy provoz neznámé aplikace Gmail byl detekován jako aplikace Twitter.

Detekce neznámého TLS provozu pouze na základě JA3 otisků nemusí být příliš úspěšná [1]. Tato úspěšnost může být zvýšena přídáním dalších informací do příznaků. Přidání doménového jména serveru, vyzkoušená v tomto projektu, se ukázala jako velmi spolehlivá.

6 Reference

[1] Petr Matoušek, Ivana Burgetová, Ondřej Ryšavý, Malombe Victor: *On Reliability of JA3 Hashes for Fingerprinting Mobile Applications*, 7. 2. 2021. Dostupné online na `https://rdcu.be/ci7mV`.