VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

POROVNÁNÍ A VYUŽITÍ NÁSTROJŮ SIMULUJÍCÍCH SÍŤ TOR

TOR NETWORK SIMULATORS: COMPARISON AND USABILITY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PETR MEDEK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. LIBOR POLČÁK, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Medek Petr**
Program: Informační technologie
Název: **Porovnání a využití nástrojů simulujících síť Tor**
Tor Network Simulators: Comparison and Usability
Kategorie: Počítačové sítě

Zadání:

1. Seznamte se s anonymizační sítí Tor a jejím bezpečnostním modelem.
2. Najděte nástroje umožňující simulovat síť Tor, popište je a porovnejte.
3. Navrhněte a realizujte sadu případů užití těchto nástrojů pro účely demonstrace principů sítě Tor.
4. Navrhněte a realizujte sadu případů užití těchto nástrojů pro účely simulace možných útoků na síť Tor.
5. Implementované případy užití otestujte a ukažte jejich využití pro výukové a demonstrační účely.
6. Práci vyhodnoťte a navrhněte možná zlepšení.

Literatura:

- JANSEN, Rob a HOPPER, Nicholas. *Shadow: Running Tor in a Box for Accurate and Efficient Experimentation*, Proceedings of the 19th Symposium on Network and Distributed System Security (NDSS), 2012.
- JOHNSON, Aaron aj. *Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries*, Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS), 2013.
- POLČÁK Libor. *Základní informace o síti Tor*. FIT-TR-2017-01, Brno, 2017. Dostupná online, URL https://www.fit.vutbr.cz/~polcak/

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Polčák Libor, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 29. října 2019

Abstrakt

Tato práce se zabývá simulací sítě Tor prostřednictvím simulátorů Shadow a Tor Path Simulator. V rámci práce vznikla webová aplikace využívající simulátor Tor Path Simulator, která demonstruje principy fungování sítě Tor a útoků na ni. Vytvořená aplikace provede simulace podle parametrů zadaných uživatelem a výsledky budou zobrazeny pomocí interaktivních grafů a tabulek. Aplikace sloužila jako podklad pro výukové materiály. V práci se také zaměřuji na simulátor Shadow a jeho využití.

Abstract

This thesis talks about simulations of Tor network through simulators Shadow and Tor Path Simulator. As a part of the thesis was created web application using Tor Path Simulator that demonstrates principles of Tor network operation and attacks on it. Created application will run simulations according to parameters input by user and results will be shown using graphs and tables. Application was used as a basis for teaching materials. In my thesis I also focus on Shadow simulator and its usage.

Klíčová slova

Tor simulace, anonymizační síť, Shadow, Tor Path Simulator

Keywords

Tor simulation, anonymization network, Shadow, Tor Path Simulator

Citace

MEDEK, Petr. *Porovnání a využití nástrojů simulujících síť Tor*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Libor Polčák, Ph.D.

Porovnání a využití nástrojů simulujících síť Tor

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Libora Polčáka, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Petr Medek
28. května 2020

Poděkování

Tímto bych chtěl poděkovat Ing. Liboru Polčákovi, Ph.D za vedení této práce, trpělivost a dobré rady.

Obsah

Kapitola 1

Úvod

Uživatelé vyhledávají možnosti, jak zůstat na internetu v anonymitě. Jedním z takovýchto nástrojů je síť Tor. Uživatelé mohou použít síť Tor pro skrytí své identity, anonymní komunikaci a předávání dokumentů. Hlavním důvodem je, že anonymita a bezpečnost na internetu je velmi důležitá, s tím jak roste shromažďování dat o uživatelích ze strany poskytovatelů internetu, vlády či soukromých společností. Získané anonymity lze ale zneužít i pro ilegální činnost, například pro gamblerství a nelegální tržiště. Základním principem sítě Tor je, že cílový server nezná identitu uživatele, pokud se uživatel neprozradí svým neopatrným chováním. Anonymity je docíleno hlavně tím, jakým způsobem je Tor navržen. Komunikace v síti Tor je směrovaná přes okruh, který se skládá ze tří uzlů: vstupní, prostřední a výstupní uzel. Komunikace procházející sítí Tor je šifrovaná, a to třemi různými veřejnými klíči získanými od uzlů v aktuálním okruhu [17]. Takto zašifrovaná data postupně prochází sítí Tor. Tímto dochází ke směrování sítí Tor. Výstupní uzel pak vytvoří běžné TCP spojení s cílovým serverem.

Cílem této práce je seznámení se se sítí Tor a nástroji pro její simulaci. Tyto nástroje jsou využity pro demonstraci fungování sítě Tor a útoků na ni. Pracoval jsem se dvěma simulátory sítě Tor: Tor Path Simulator (TorPS) a Shadow. Navrhl jsem webovou aplikaci a provedl několik simulací demonstrujících například výběr cesty v síti Tor a připojení ke skrytým službám. Také demonstruji fungování korelačního útoku, útoku Dropmark a útok, při kterém útočník odposlouchává výstupní uzel. Dalším cílem práce bylo ukázat využití navržené aplikace pro výukové a demonstrační účely, v rámci práce vznikly výukové materiály.

Informace o fungování a architektuře sítě Tor jsou uvedeny v kapitole 2. Tato kapitola také obsahuje bezpečnostní model a popis některých útoků na síť Tor. V kapitole 3 jsou uvedeny simulátory sítě Tor. Nejprve se věnuji simulátoru TorPS, k čemu slouží a jakým způsobem modeluje uživatele a útočníky. Dále se věnuji simulátoru Shadow, jeho funkčnosti a pluginu shadow plugin tor, který slouží pro simulování sítě. Na závěr této kapitoly oba dva simulátory porovnávám. Kapitola 4 navrhuje webovou aplikaci, ve které si uživatel může spustit simulace. Implementaci webové aplikace popisuji v kapitole 5, kde uvádím použité nástroje a jak jsem při implementaci postupoval. Kapitola simulace a testování 6 ukazuje využití implementované aplikace pro výukové a demonstrační účely. Kapitola také popisuje některé simulace provedené mimo aplikaci. Na závěr v kapitole 7 vyhodnocuji a shrnuji práci.

Kapitola 2

Anonymizační síť Tor

V této kapitole jsou popsány základní pojmy a definice, které tato práce využívá. Měla by čtenáři pomoci porozumět problematice fungování sítě Tor. Nejprve je krátce představena síť Tor. Je nastíněno, k čemu slouží a kdo ji nejčastěji využívá, viz sekce 2.1. Následuje popis architektury, viz sekce 2.2 a fungování sítě Tor, viz sekce 2.3. Konkrétně jsou vysvětleny jednotlivé části sítě Tor a hlavní principy, na kterých síť Tor funguje. Další sekce se zabývá bezpečnostním modelem, viz sekce 2.8. Závěr kapitoly se zaměřuje na možné útoky na síť Tor: analýza síťového provozu, korelační útok, útok Dropmark, viz sekce 2.9. Informace v této kapitole slouží k základnímu pochopení fungování sítě Tor a jsou důležité pro potřeby této práce. Nejedná se o detailní popis sítě Tor.

2.1 Základní informace o síti Tor

Síť Tor je anonymizační síť, která má za úkol chránit soukromí uživatelů [9]. Tor přenáší internetovou komunikaci přes distribuovanou síť uzlů, které jsou provozované dobrovolníky po celém světě. Tor byl originálně vytvořen v U.S. Naval Research Laboratory[1], jeho primárním účelem byla ochrana vládní komunikace. V dnešní době je udržován a vyvíjen neziskovou organizací The Tor project [2]. Software využívaný sítí Tor je volně dostupný včetně zdrojového kódu. Tor je nejčastěji používán jako Torbrowser[2] a je založen na Mozilla Firefox[3].

Obyčejným lidem nabízí síť Tor anonymitu při vyhledávání na internetu. Síť Tor používají také lidé, jako novináři, bezpečnostní složky a lidé, kteří potřebují ke své práci anonymní přístup k internetu. Nemalou součástí uživatelů jsou lidé zapojení do nelegálních aktivit, aby se chránili před dopadením. Dle stránky Tor Users [4] je síť Tor využívána následovně:

- Novináři se přes síť Tor připojují k internetu pro překonání restrikcí (cenzura některých témat). Některé noviny jako The Washington Post nebo The New York Times používají Secure Drop, jehož je Tor součástí, pro bezpečné předávání dokumentů a anonymní komunikaci.

[1] https://www.nrl.navy.mil/
[2] https://www.torproject.org/download/
[3] https://www.mozilla.org/en-US/firefox/

- Bezpečnostním složkám zajišťuje síť Tor sledování podezřelých webových stránek spojených s nezákonnou činností bez toho, aniž by byly odhaleny. Obdobně jako u novinářů mohou bezpečnostní složky použít Tor pro anonymní komunikaci.
- Aktivisté a takzvaní „Whistleblowers“ chrání svoji bezpečnost připojováním se přes síť Tor, protože ta umožňuje skrýt jejich identitu.
- Příkladem nelegální činnosti prováděné přes síť Tor je: nelegální tržiště, gamblerství, prodej kradených čísel a platebních karet [6].

2.2 Architektura sítě Tor

Síť Tor využívá existující strukturu internetu, kde komunikace mezi jednotlivými prvky sítě probíhá pomocí protokolu TCP/IP, jedná se tedy o takzvanou overlay síť. Síť se skládá z několika druhů uzlů plnících konkrétní úlohy, uzly lze rozdělit do následujících skupin:

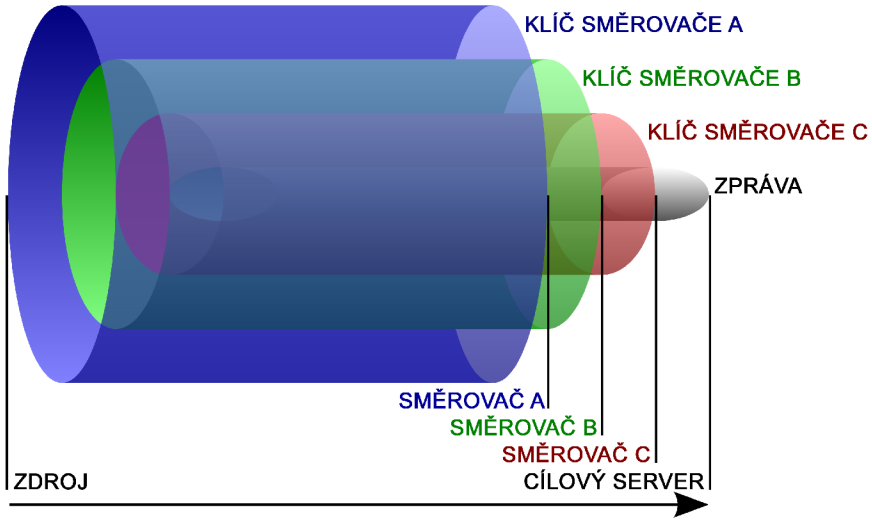
- **Adresářové servery (directory servers):** spravují seznam a stav aktivních uzlů v síti Tor, ze kterých si klienti mohou vybrat uzly pro vytvoření okruhů.
- **Uživatelská proxy (onion proxy – OP):** typicky běží na počítači koncového uživatele. Z adresářových serverů získává informace o topologii a stavu sítě. Zajišťují tvorbu virtuálních okruhů a řízení přenosu datových toků mezi aplikacemi a sítí Tor.
- **Směrovače (onion routers – OR):** směrovače jsou odpovědné za přeposílání dat v rámci sítě Tor. Vlastní směrovač si může vytvořit každý, kdo má rychlé a stabilní internetové připojení, může běžet jako uživatelský proces (user-level). Směrovače umožňují uživatelům vytvářet okruhy. Dle pozice v okruhu jde dělit směrovače na tři druhy:
  - **Strážci (guards):** jsou stabilní dlouho běžící uzly, ze kterých OP vybere jeden OR. Ten je používán jako vstupní OR pro všechny vytvářené okruhy a je obvykle obměňován každé dva až tři měsíce [16].
  - **Prostřední uzly (middle nodes):** uzel je druhý v pořadí, předává komunikaci mezi sousedními uzly.
  - **Výstupní uzly (exit nodes):** umožňují výstup ze sítě Tor k cílovému serveru, každý výstupní uzel má vlastní pravidla specifikující rozsah IP adres a portů, ke kterým je ochoten se připojit.
- **Skryté služby (hidden services – HS)**: Tor poskytuje skryté služby, což jsou servery dostupné pouze ze sítě Tor. Skryté služby jsou dostupné pomocí doménového jména s koncovkou **.onion** nejčastěji prostřednictvím Torbrowser[4]. Ke službám se přistupuje anonymně bez toho, aniž by byla prozrazena IP adresa klienta nebo serveru [3].

2.3 Fungování sítě Tor

Komunikace v síti Tor je směrovaná přes okruhy, normální okruh se skládá ze tří uzlů: vstupní uzel (guard), prostřední uzel (middle) a výstupní uzel (exit). Tor využívá asymetrické kryptografie k šifrování komunikace. Zasílaná zpráva je zašifrována třemi veřejnými

[4] https://www.torproject.org/download/

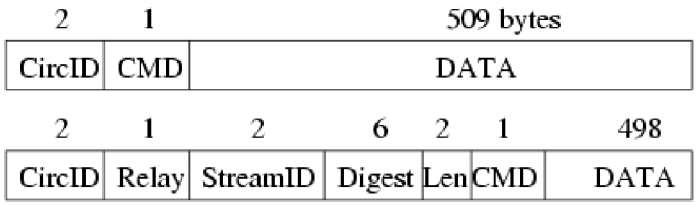
klíči získanými od uzlů v aktuálním okruhu [17]. Takto zašifrovaná data postupně prochází sítí Tor, každý OR na cestě dešifruje jednu vrstvu buňky. Tímto dochází k směrování sítí Tor. Výstupní uzel pak vytvoří běžné TCP spojení s cílovým serverem. Na obrázku 2.1 jsou vidět vrstvy šifrované zprávy.



Obrázek 2.1: Vrstvy šifrování zprávy, převzato z [14].

2.4 Buňky

Buňky jsou používány pro vzájemnou komunikaci OR a OP v síti Tor. Buňka má pevnou velikost[5] 512 B nebo 514 B [19] (podle verze použitého protokolu). Při přenášení dat dochází uvnitř buněk k zarovnání na požadovanou velikost. Buňky jsou rozděleny na **řídící (control cells)**, které jsou vždy interpretovány přijímající uzlem, a **přenosové (relay cells)**, které přenáší data po síti Tor mezi koncovými uživateli [9].



Obrázek 2.2: Struktura buňky, převzato z [9, 8].

Hlavička řídící buňky se skládá z ID (CircID) a příkazu (CMD). Přenosová buňka má navíc identifikátor datového toku TCP (StreamID), kontrolní součet (Digest) a délku dat (Len). Některé buňky obzvláště důležité pro tuto práci jsou:

- **Řídící buňky pevné délky:** create (založení okruhu), destroy (zrušení okruhu).
- **Přenosové buňky:** begin/end (otevření/uzavření spojení), connected (oznámení OP o úspěšném navázání spojení), resolve/resolved (anonymní DNS).

[5] V pozdější verzi Toru byly představeny některé řídící buňky s proměnlivou velikostí

2.5 Okruhy

Okruh je cesta v síti Tor z OP k cílovému serveru. Uživatel nejprve kontaktuje adresářový server, aby zjistil seznam dostupných uzlů. Poté uživatelský OP vytváří okruhy inkrementálně, postupně vyjednává symetrický klíč s každým OR v okruhu jeden po jednom [8]. Každý částečně vytvořený okruh se používá ke komunikaci s dalším OR na cestě a výstupní uzel vytvoří běžné TCP spojení s cílovým serverem. Díky tomuto postupu je zajištěno perfektní dopředné utajení (forward secrecy). Tor dále aplikuje multiplexování: více TCP spojení může sdílet jeden okruh [8].

Popis konstrukce okruhu

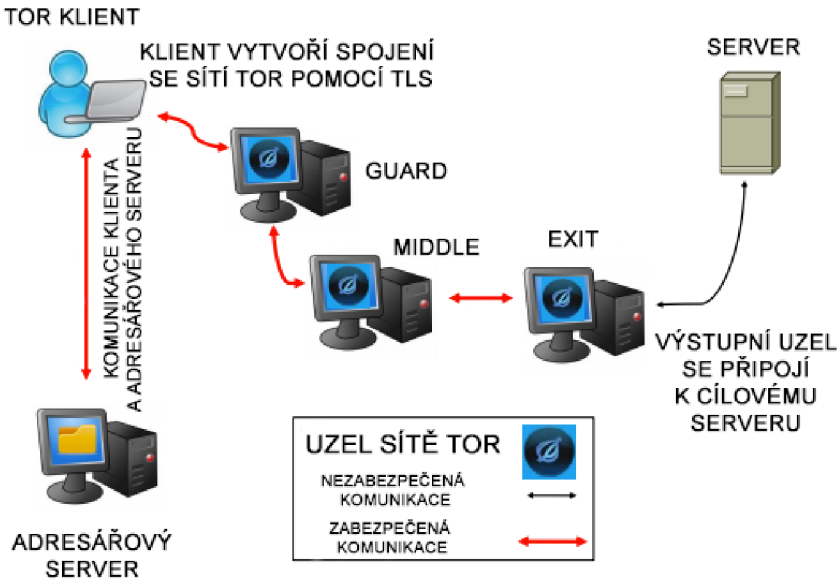
Uživatelská OP aplikuje preemptivní konstrukce okruhů (snaží se tedy odhadnout užitečné okruhy), OP vybírá OR na základě dat z konsensu (adresáře). OP se rozhoduje podle statusů a vah, ty odráží aktuální přenosovou kapacitu, spolehlivost, vstupní a výstupní politiku uzlů. Tor měří reálnou šířku pásma (throughput over time), kterou každý uzel poskytuje do sítě, a podle poskytované šířky pásma přiřazuje uzlu váhu. Váha je důležitá při vytváření okruhu, Tor podle váhy rozděluje zátěž [12]. To znamená, že přes uzel s větší šířkou pásma bude procházet více okruhů.

Vytváření okruhu probíhá tak, že OP inkrementálně vyjednává symetrické klíče s každým OR v okruhu, a to od prvního (guard) až po výstupní (exit). Při každém rozšíření okruhu o další OR komunikuje OP s daným OR přes již částečně vytvořený okruh. Před vlastním přenosem dat je buňka zašifrována sdílenými symetrickými klíči, a to v pořadí od posledního OR k prvnímu. OP zná všechny klíče, zatímco každý OR zná vždy jen jeden. Tím je zajištěno, že žádný OR nezná destinaci zprávy a uživatele. Při průchodu buňky sítí Tor dochází k postupnému dešifrování vrstev buňky, viz obrázek 2.1.

Komunikace OP s cílovým serverem probíhá tak, že jsou data přenášena nad vrstvou TCP. Datové pakety jsou přenášeny v buňkách o pevné velikosti 512 B. Výstupní uzel pak vytvoří běžné TCP spojení s cílovým serverem. Ten odpovídá uživatelské aplikaci analogicky. Při zpětném průchodu sítí Tor každý OR na cestě šifruje buňku svým sdíleným symetrickým klíčem. Buňka je šifrována ve směru od posledního OR k prvním OR v okruhu. Uživatelská OP potom buňku opakovaně dešifruje.

Na obrázku 2.3 je zakreslen sestavený okruh v síti Tor. Lze pozorovat, že spojení s cílovým serverem nemusí být šifrované. Záleží na tom, jestli uživatelská aplikace používá šifrovaný datový tok. Pokud uživatel vyžaduje šifrovanou komunikaci mezi aplikací a cílovým serverem, musí použít například protokol TLS. Tor sám o sobě nezajišťuje šifrování mezi koncovými uživateli. Když chce uživatel zůstat anonymní, je důležité neposílat cílovému serveru žádné identifikační údaje.

Pokud by byl vstupní uzel (guard) vybírán náhodně, pravděpodobnost toho, že bude uzel kompromitován je $(\mathbf{c}/\mathbf{n})^{\mathbf{2}}$, kde c představuje počet uzlů kontrolovaných útočníkem a n je celkový počet uzlů. V minulosti to fungovalo tak, že si uživatel sítě Tor vybral několik uzlů (obvykle tři) jako vstupní uzly pro všechny vytvářené okruhy. To znamenalo, že pokud si uživatel vybral uzel kontrolovaný útočníkem jako vstupní uzel (guard), byla pravděpodobnost $\mathbf{c}/\mathbf{n}$, že jeho okruh byl kompromitován [8]. Pokud k tomu ale nedošlo, byly okruhy bezpečné. Uživatel střídal každý z těchto uzlů po 30 až 60 dnech. V dnešní době se využívá pouze jeden vstupní uzel (guard), který se mění po dvou až třech měsících [16].



Obrázek 2.3: Okruh v síti Tor, převzato z [1].

2.6 Plánovače v síti Tor

Síť Tor kvůli svým anonymizačním technikám přináší značné zpoždění. Část tohoto zpoždění je způsobena algoritmem, který má na starosti plánování okruhů (scheduling algorithm). Každý z těchto algoritmů má k plánování jiný přístup. Algoritmus KIST (Kernel-Informed Socket Transport) byl navržen, aby vyřešil problém se správou soketů. Dříve používaný algoritmus Vanilla sekvenčně zapisuje do soketů, ale ignoruje stav všech soketů, do kterých zrovna nepíše. To způsobuje, že data s menší prioritou jsou někdy posílána dříve než data s vysokou prioritou [11]. KIST má dvě hlavní vlastnosti, které vylepšují kontrolu sítě Tor nad zahlcením sítě: KIST vybírá ze všech okruhů (ne pouze z těch, které patří pod jeden TCP soket). Dále na základě informací z jádra kontroluje množství dat zapsaných do soketu, aby nedošlo k zahlcení [11]. KIST funguje pouze na zařízeních s verzí Linux jádra 2.6.39 a vyšší. V aktuální verzi Tor je algoritmus KIST výchozím pro plánování, do Toru byl přidán v roce 2017[6] jako náhrada za algoritmus Vanilla, který běží na všech operačních systémech. Tor podporuje oba algoritmy z důvodu zpětné kompatibility. Algoritmus KIST-Lite je jednoduší verze algoritmu KIST, ale bez podpory jádra (poběží na všech operačních systémech).

2.7 Skryté služby

Skryté služby neboli tzv. „Onion services“ jsou služby dostupné pouze ze sítě Tor, ten dává možnost uživatelům připojit se ke skrytým službám. Ke službám se připojuje anonymně bez toho, aniž by byla prozrazena IP adresa klienta nebo serveru. Toto připojení je možné přes tzv. „rendezvous points“ (RP). Základním principem je, že klient zná výhradně lokaci RP, nezná lokaci skryté služby, stejně tak skrytá služba zná pouze lokaci RP.

Připojení ke skrytým službám lze popsat v několika krocích:

[6] https://blog.torproject.org/kist-and-tell-tors-new-traffic-scheduling-feature

1. Skrytá služba nejprve vygeneruje soukromý a veřejný klíč, připojí se k vybranému OR a požádá ho, zda může OR vstupovat jako tzv. „introduction point“ (InP). Skrytá služba se snaží získat požadovaný počet InP, s těmito InP poté udržuje otevřené okruhy.

2. Skrytá služba dává vědět o existujících InP přes adresářové servery pomocí deskriptoru, který obsahuje veřejný klíč a seznam InP, tento deskriptor je podepsán soukromým klíčem. Od této chvíle mohou klienti kontaktovat skrytou službu.

3. Klient, který chce kontaktovat skrytou službu, se zeptá adresářového serveru na existenci skryté služby. Při existenci služby s danou **.onion** adresou obdrží klient deskriptor z adresářového serveru. Tímto získá seznam InP a veřejný klíč.

4. Klient si zvolí náhodné OR jako tzv. „rendezvous point“ (RP) pro komunikaci se skrytou službou. K RP založí okruh a pošle mu náhodně zvolené rendezvous cookie (RC).

5. Klient vytvoří okruh k jednomu z InP skryté služby a předá mu zprávu obsahující (vybraný RP, RC a první část dohody algoritmu Diffie-Hellman (D-H)) zašifrovanou veřejným klíčem skryté služby. InP poté pošle zprávu skryté službě přes okruh vytvořený v bodě 1.

6. Pokud chce skrytá služba komunikovat s klientem, vytvoří okruh ke klientovu RP a odešle RC, druhou část dohody D-H na klíči a otisk klíče sezení (session key). RP spojí dohromady okruh, který sdílí s HS, a okruh, který sdílí s klientem. Důležitým principem je, že RP nezná identitu klienta, skryté služby ani data, která posílají.

7. RP pošle klientovi potvrzení, že spojení s HS bylo úspěšně navázáno. Nyní je možná komunikace mezi klientem a HS [3].

2.8 Bezpečnostní model

Většina útoků na síť Tor se zaměřuje na deanonymizaci uživatelů, tedy potvrzení vztahu mezi klientem a serverem. Jelikož klient vytváří okruh přes síť Tor k výstupnímu uzlu komunikujícímu se serverem, útočník se snaží potvrdit, že klient a server spolu opravdu komunikují.

Častou hrozbou je pasivní útočník, který sleduje vstupy a výstupy sítě a provádí mezi nimi korelaci, která se snaží potvrdit vztah mezi klientem a serverem. Na druhou stranu aktivní útočník může přidávat do síťového provozu vlastní charakteristiky, pomocí kterých lépe pozoruje průchod dat sítí [9]. Více o pasivním a aktivním útočníkovi je vysvětleno v sekci 2.9.1.

Je třeba připustit, že Tor se nezabývá ochranou proti těmto výše zmíněným útokům (traffic confirmation attacks), ale snaží se spíše zamezit útokům založených na charakteristikách síťového provozu (traffic analysis attacks): snaží se zabránit útočníkovi, aby určil, na jaké body v síti by měl provést útok [9].

Bezpečnostní model sítě Tor je navržen tak, aby chránil před útočníkem, který je schopný monitorovat část sítě. Je to založeno na předpokladu, že útočník může nejen vytvářet, upravovat, mazat a zpožďovat provoz, ale i být vlastníkem uzlu [8, 9].

Detekce komunikace: Bezpečnostní model sítě Tor se nijak nesnaží zakrýt, že se uživatel připojuje přes síť Tor. Potenciální útočník může provést analýzu síťového provozu na základě četnosti délek přenášených paketů bez znalosti dešifrované komunikace [17].

Neopatrné chování uživatelů: Chování uživatelů může ovlivnit jejich anonymitu. Doporučuje se použít pro připojení k síti Tor prohlížeč Torbrowser. Pokud uživatel komunikuje prostřednictvím nešifrovaného protokolu HTTP, může útočník pozorovat datový tok na cestě od výstupního uzlu k cílovému serveru. Může tak zjistit například uživatelské jméno nebo heslo. Dále mohou některé aplikace používající Tor vést k deanonymizaci uživatele: například BitTorrent, který přenáší informace o IP adrese[7].

2.9 Útoky na síť Tor

V této sekci jsou vysvětleny jednotlivé útoky na síť Tor: analýza síťového provozu, korelační útok a útok Dropmark. Uvedený popis útoků je důležitý pro simulace provedené v dalších částech práce. Simulace modelují právě fungování těchto útoků.

2.9.1 Analýza síťového provozu

Internetový provoz je často šifrován, proces šifrování znemožňuje přečíst obsah paketů. I když je provoz šifrovaný, je stále možné číst metadata z hlaviček paketů, časování paketů a velikost paketů. Díky těmto informacím je možné jednoznačně identifikovat datový tok. Právě získáváním informací ze síťových metadat se zabývá analýza síťového provozu. Pro analýzu provozu anonymizačních sítí jsou důležité informace jako časování, objem paketu a zpoždění dat při průchodu sítí [7]. Útočníky lze rozdělit do dvou skupin podle jejich chování: pasivní a aktivní.

Pasivní útočník: Tento útočník se příliš neprojevuje, nemá možnost měnit charakteristiku datových toků, tím pádem nemůže získat lepší podmínky pro korelaci [7]. Využívá pouze charakteristik síťového provozu, z toho důvodu je složitější takového útočníka vystopovat.

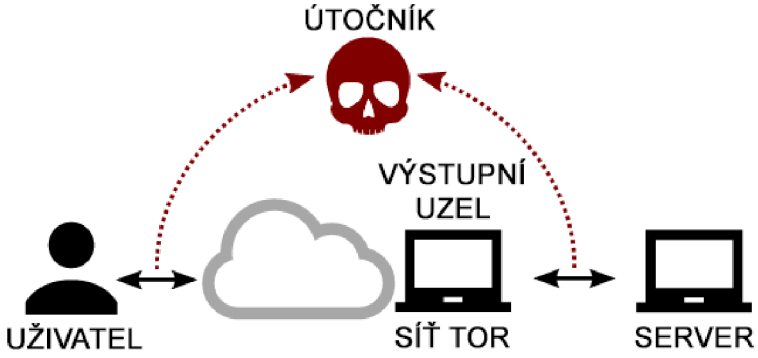
Aktivní útočník: Předpokládá se, že aktivní útočník má pod kontrolou některé OR, což mu umožňuje přidávat do síťového provozu vlastní charakteristiky, pomocí kterých lépe pozoruje průchod dat sítí. Útočník může upravovat charakteristiku datových toků procházejících sítí Tor, protože síť Tor kontroluje integritu zasílaných dat jen na vstupním a výstupním uzlu okruhu [7]. Aktivní útočník nemá výhodu oproti pasivnímu útočníkovi, protože nezískává žádné další informace [9].

2.9.2 Korelační útok

Korelační útok je jedna z deanonymizačních technik, jejímž cílem je zjistit závislost mezi vstupním a výstupním datovým tokem. To je založeno na předpokladu, že útočník má pod kontrolou vstupní (guard) i výstupní (exit) uzel okruhu. Útočník se dívá na provoz v síti a pomocí korelačních metod se snaží určit, zda existuje závislost mezi datovými toky na vstupu a na výstupu. Vstupní uzel sítě zná identitu klienta a výstupní uzel zná identitu serveru, tím pádem útočník může předpokládat, že spolu klient a server komunikují. Jak

[7] https://blog.torproject.org/bittorrent-over-tor-isnt-good-idea

bylo zmíněno v sekci o bezpečnostním modelu 2.8, Tor je vůči korelačním útokům zranitelný. Není ale jasné, jaké množství dat musí útočník zpracovat, aby výsledky korelační metody byly přesvědčivé [7]. S tím souvisí velikost anonymizační sítě. Větší síť znamená pro útočníka více kombinací vstupních a výstupních uzlů, z toho důvodu je realizace korelačního útoku složitější [21].



Obrázek 2.4: Princip korelačního útoku, převzato z [5].

Fungování korelačního útoku

Pro úspěšnost korelačního útoku je pro útočníka klíčové rozeznat síťový provoz Toru. Jak je uvedeno v sekci 2.4, síťový provoz je přenášen v buňkách o pevné velikosti 512 B přes protokol TCP. Korelační útok je typem útoku založeném na analýze síťového provozu, pro úspěšnost útoku je důležité množství nasbíraných dat a jejich charakteristiky. Z toho vyplývá, že s větším počtem dat jsou výsledky korelační metody přesvědčivější [7]. Korelace se v síti Tor může nacházet na různých úrovních abstrakce. Na vyšší úrovni abstrakce se síť Tor chová jako černá skříňka, kdy útočník pouze sleduje, když klient navazuje spojení a když datové toky opouští síť Tor. Na nižší úrovni se útočník zaměřuje na síťový provoz uvnitř sítě Tor, na vybraných linkách sleduje charakteristiky síťového provozu [13].

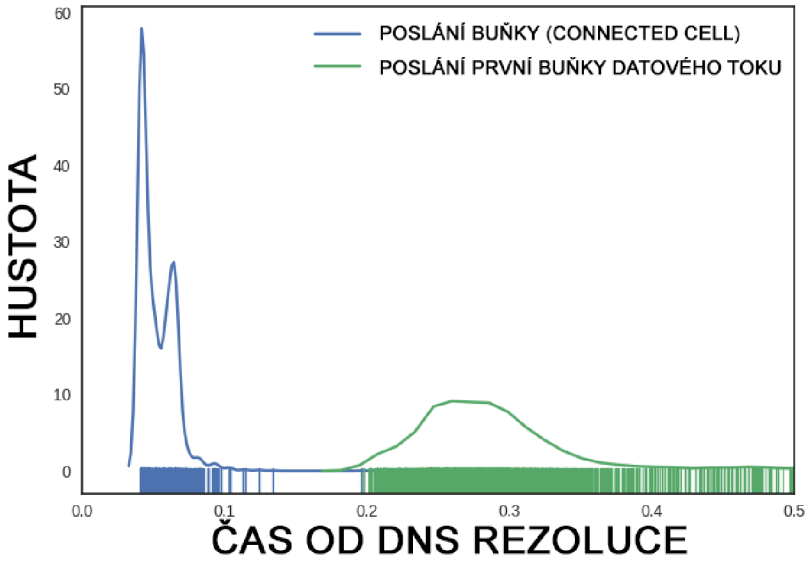
2.9.3 Útok Dropmark

Tento útok modifikuje tok dat na vstupním uzlu v odchozím směru, aby generoval tzv. integrity error. Tento error je detekovatelný po okruhu dalšími kompromitovanými uzly, ale také vyvolá uzavření okruhu na nekompromitovaných výstupních uzlech [18].

Zahazování paketů sítí Tor

Útok Dropmark využívá protokolu Tor a jeho chování k neznámým buňkám, kdy výchozí stav je ignorovat neznámé buňky z důvodu kladení důrazu na dopřednou kompatibilitu. To znamená, že buňky s neznámým příkazem CMD ať řídící, nebo přenosové, jsou tiše zahozeny. Buňky jsou detailněji popsány v sekci 2.4. Útok cílí na dobu, kdy je datový tok nečinný, a to z důvodu, aby nebylo zvýšeno zpoždění datového toku oběti a aby přidaná značka Dropmark byla jednoduše získána [18].

Když se klient například připojuje k webové stránce, zašle po okruhu buňku (begin cell), což vyvolá DNS rezoluci na okraji okruhu. Když je DNS rezoluce úspěšná a spojení je navázáno, výstupní uzel pošle buňku (connected cell) ve směru ke klientovi. Další zaslaná buňka je odpovědí na klientův požadavek GET. Doba nečinnosti, během níž po okruhu neprochází žádná buňka ve směru ke klientovi, koresponduje se součtem obousměrného zpoždění (round-trip time) okruhu a obousměrného zpoždění mezi výstupním uzlem a cílovým serverem. Toto platí pouze, pokud klient neposlal požadavek před obdržením buňky (connected cell) [18]. Tato doba nečinnosti slouží k zakódování značky Dropmark pomocí buněk (relay drop cell) nebo pomocí jiných buněk, které by byly tiše zahozeny na okraji okruhu. Na obrázku 2.5 je vidět doba nečinnosti, při které se posílá značka Dropmark.



Obrázek 2.5: Graf hustoty odesílaných buněk, převzato z [18].

Fungování útoku Dropmark

Útok Dropmark umožňuje přenos značky Dropmark po okruhu bez přidání latence k toku dat oběti. Tento útok potřebuje k funkčnosti tyto předpoklady:

1. Protokol by měl tiše zahazovat neznámé pakety.
2. Okruh by měl být v určitém okamžiku nečinný (bez toku dat).
3. Útočník musí vlastnit dva uzly sítě Tor, jeden z nich v pozici výstupního uzlu.

Kódování značky Dropmark nastává na výstupním uzlu sítě Tor. Jak už bylo zmíněno, útok Dropmark využívá doby nečinnosti okruhu, buňka (relay begin cell) je poslána směrem k výstupnímu uzlu. Pokud nese požadovanou IP adresu nebo doménové jméno, je poslána značka Dropmark. Výstupní uzel přijme buňku a po provedení DNS rezoluce si zapíše IP adresu a pošle zpátky tři buňky (drop cells) [18]. V případě simulace útoku Dropmark to probíhá tak, že jsou specifikovány IP adresy. Pokud se na některou z nich chce klient připojit, je výstupním uzlem poslána značka Dropmark.

Potvrzení přítomnosti značky Dropmark

Je potřeba potvrdit, že značka Dropmark je v komunikaci přítomna. Vstupní uzel pozoruje prvních pár příchozích buněk v okruhu. Pokud mezi prvními čtyřmi buňkami mají tři buňky stejný čas příchodu, je komunikace označena jako obsahující značku Dropmark [18]. Obrázek 2.6 znázorňuje, jak vypadá situace se značkou Dropmark a bez ní.

(a) Normální chování buněk, když není poslána značka Dropmark.

(b) Značka Dropmark.

Obrázek 2.6: Tok buněk z pohledu vstupního uzlu. Buňky jdou ve směru od výstupního uzlu směrem ke klientovi, převzato z [18].

Kapitola 3

Simulace sítě Tor

Tato kapitola popisuje a srovnává dva různé simulátory sítě Tor. Nejprve je vysvětleno fungování simulátoru TorPS, viz sekce 3.1. Johnson et al. [12] vytvořili simulátor TorPS k simulaci zranitelnosti sítě vůči korelačnímu útoku. Dále se zabývám simulátorem Shadow, viz sekce 3.2. Na závěr této kapitoly jsou porovnány oba simulátory navzájem, jejich slabé a silné stránky, viz sekce 3.3. Rozdíly a společné znaky obou simulátorů jsou shrnuty v tabulce, viz tabulka 3.1.

3.1 Simulátor TorPS

Simulátor TorPS simuluje proces výběru uzlu pro vytváření okruhu [12]. Před samotnou simulací TorPS zpracovává historická data z portálu Tor Metrics[1], konkrétně zpracovává server deskriptory (Relay Server Descriptors) a konsensus soubory (Network Status Consensuses). Tato data jsou konvertována do souborů obsahujících stav sítě v daném čase, ty slouží jako vstupní data pro simulaci. TorPS využívá pro simulaci výběru okruhu jeden z pěti možných modelů uživatelů. Výstupem simulace je soubor se seznamem okruhů, každý okruh se skládá ze tří IP adres. TorPS implementuje také simulátor zabývající se alternativním výběrem cesty tzv. Congestion Aware Tor (CAT) [20]. Simulátor pracuje s dvěma druhy útočníků: Relay Adversary, který má pod kontrolou některé z uzlů sítě, a Network Adversary, který sleduje část sítě například autonomní systémy AS nebo Internet Exchange Points IXP. Oba druhy útočníků jsou vysvětleny níže, viz sekce 3.1. Simulátor byl vytvořen proto, aby otestoval zranitelnost sítě Tor proti korelačním útokům, a využívá modely chování klienta (typický, BitTorrent, Nejlepší/Nejhorší port) a modelu útočníka [12].

Path Simulator

K vytvoření cest využívá TorPS historická síťová data, a to z Tor Metrics, aby mohl modelovat stav sítě Tor. Tor Metrics poskytuje archiv deskriptorů serverů (Relay Server Descriptors) a konsensus (Network Status Consensuses). Ty jsou důležité pro určení: statusu (flag), výstupních pravidel, stavu OR v průběhu času, stav hibernace a další charakteristiky. Na základě těchto dat simuluje stav sítě a na základě těchto informací spustí algoritmus na vytvoření cesty. TorPS zahrnuje model OR a jejich předchozích stavů, model chování uživatelů a model klienta sítě Tor [12].

[1] https://metrics.torproject.org/collector.html

Modely uživatelů

Pro lepší pochopení bezpečnosti opravdových uživatelů bylo vytvořeno pět modelů uživatelů sítě Tor. Každý model zahrnuje sekvenci toků dat a časy, kdy k nim dochází. Datové toky zahrnují výsledky DNS rezoluce spolu s TCP spojeními ke specifickým destinacím [12].

- **Typický** – tento model by měl reprezentovat typického uživatele sítě Tor. Připojuje se k Gmail, Google Calendar, Facebook a vyhledávání. Toto vyhledávání se opakuje pětkrát každý den v předem určené časy.
- **IRC** – reprezentuje uživatele používající pouze IRC (Internet Relay Chat) chat, od pondělí do pátku 27× denně.
- **BitTorrent** – tento model popisuje využívání BitTorrentu přes síť Tor během stahování jednoho souboru, a to v sobotu a v neděli dohromady 18× za den.
- **Nejhorší port** – upravuje typický model nahrazením čísla portu číslem 6523. Johnson et al. [12] ve své práci označili tento port jako port s druhou nejmenší výstupní kapacitou.
- **Nejlepší port** – upravuje typický model nahrazením čísla portu portem HTTPS 443 [12].

Model nejlepší port a typický uživatel si mohou být podobné, protože typický uživatel také často používá HTTPS port 443.

Relay Adversary: Útočníci, kteří provozují OR, jsou známou hrozbou pro klienty sítě Tor. Klienti si vybírají OR pro vytvoření okruhů zhruba v poměru k šířce pásma, a proto se schopnost útočníka deanonymizovat provoz sítě odvíjí od jeho šířky pásma. Šířka pásma je drahá a zároveň je síť Tor relativně velká, proto by zahlcení sítě bylo velmi nákladné [12]. Útočník musí rozhodnout, jak nejlépe rozdělit svou šířku pásma, aby měl největší šanci zachytit datové toky. Protože jeden OR nemůže být zároveň v pozici vstupního (guard) a výstupního (exit) uzlu v okruhu, útočník musí provozovat alespoň dva OR, aby byl schopen provést korelační útok. Útočníkův vstupní uzel musí získat status guard (guard flag). Aby zvýšil šanci na úspěšnou korelaci, pravidla výstupního uzlu by měla podporovat výstup na všechny adresy a porty. Výstupní uzel nemá status guard a oba uzly získaly status fast (fast flag). Princip korelačního útoku je podrobněji vysvětlen v sekci 2.9.2.

Network Adversary: Network adversary nepoužívá na rozdíl od Relay adversary OR, ale využívá své pozice poskytovatele síťového provozu ke korelaci datových toků procházejících sítí v některém bodě mezi klientem a vstupním uzlem (guard), a mezi výstupním uzlem (exit) a cílovým serverem [12]. Potenciální útočník může využít kontroly nad jedním či více autonomními systémy (AS – základní organizační jednotka pro správu internetového směrování) nebo IXP (Internet Exchange Point) pro sledování datových toků.

3.2 Simulátor Shadow

Simulátor Shadow byl vytvořen, aby zvýšil přesnost, konzistenci a škálovatelnost experimentů na síti Tor. Výsledky experimentů jsou opakovatelné a ověřitelné nezávislou analýzou. Shadow simuluje síťovou vrstvu, připojuje se k originálnímu software sítě Tor. Díky

tomu mohou být nové prvky implementovány přímo do zdrojového kódu. Shadow je simulátor s diskrétním časem, který dovoluje spouštět reálné aplikace v simulovaném prostředí. Aplikace jsou k Shadow připojeny pomocí pluginů obsahujících potřebné funkce k tomu, aby Shadow mohl komunikovat s aplikací [10]. Jak už bylo zmíněno, Shadow je simulátor s diskrétním časem, který spouští reálné aplikace jako pluginy s minimálními modifikacemi aplikace.

3.2.1 Design

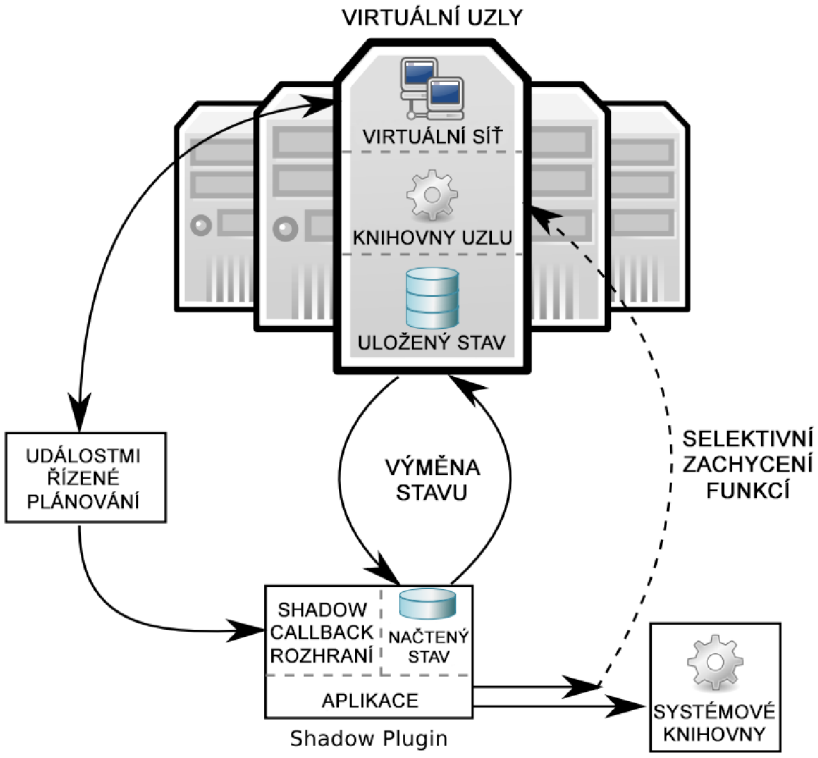
Shadow dynamicky načítá pluginy a vytváří instance virtuálních uzlů podle specifikace v simulačním skriptu. Plugin obsahuje spojení aplikace s knihovnami Shadow. Simulátor Shadow dynamicky načítá a spouští kód aplikace, zatímco simuluje komunikaci na síťové vrstvě [15]. Komunikace mezi Shadow a pluginem probíhá prostřednictvím rozhraní zpětného volání (callback interface) implementovaného pluginem. Shadow zachycuje systémová volání aplikace. Jak je vidět na obrázku 3.1, systémové volání je zachyceno simulátorem Shadow a přesměrováno na knihovní funkci. Díky principu zachycení systémového volání integruje Shadow aplikace bez potřeby modifikace jejich kódu. Virtuální uzly spolu poté komunikují přes virtuální síť, která zařazuje pakety a síťové události do plánovače. Každý virtuální uzel uchovává jen stav aplikace, který se načítá, nebo uvolňuje dle potřeb simulace. Shadow modeluje geografické rozložení internetu podle dat z příkazu ping na PlanetLab[2]. Data získaná z příkazu ping prozrazují zpoždění mezi jednotlivými uzly, které pak lze přiřadit do regionů [10].

Simulační skript: Každá simulace je spuštěna simulačním skriptem. Ten specifikuje, které pluginy spustit a kdy spustit každý uzel. Pluginy jsou načteny zadáním cesty, ale parametry (latence, šířka pásma, rychlost CPU) jsou načítány ze souboru s CDF (Common Data Format), nebo generovány pomocí vestavěného CDF generátoru. Autoři [10] popisují, že Shadow extrahuje události ze simulačního skriptu a zařadí je do plánovače událostí podle časů specifikovaných pro každý příkaz. Simulace postupně zpracovává události a běží, dokud jsou v plánovači události, nebo dokud nedosáhne specifikovaného času pro konec simulace.

Shadow plugin: Shadow plugin obsahuje aplikaci, kterou uživatel chce simulovat. Každý obal pluginu (Shadow plug-in wrapper) umožňuje integraci se simulátorem Shadow tím, že implementuje rozhraní zpětného volání (callback interface), přes které Shadow komunikuje s pluginem [15]. Shadow může přes rozhraní zpětného volání nejen alokovat a dealokovat zdroje aplikací, ale také upozornit plugin, kdy je možné provést operace čtení a zápis. Shadow může spustit pouze asynchronní aplikace, aby se zabránilo deadlocku při běhu aplikace. Dále aplikace musí běžet v jednom procesu a v jednom vlákně. Více virtuálních uzlů může mít spuštěnou jednu aplikaci, z toho důvodu pluginy předávají knihovním funkcím Shadow ukazatele na alokovanou paměť uzlu (každý uzel má svou vlastní paměť) [10].

Virtuální uzly: Každý virtuální uzel představuje jednoho simulovaného hosta. Virtuální uzel zahrnuje data o charakteristice hosta: například informace o síti, což umožňuje komunikaci s ostatními hosty v síti. Virtuální uzly ukládají vlastní stav aplikace, který přenesou do adresářového prostoru pluginu, než předají pluginu kontrolu nad řízením. Virtuální plugin může běžet na několika virtuálních uzlech zároveň [10]. Shadow pro

[2] https://www.planet-lab.org/



Obrázek 3.1: Komunikace mezi Shadow a pluginem, zachycení systémových volání, převzato z [10].

každý virtuální uzel ukládá do paměti jenom stav pluginu, neduplikuje celý plugin. Díky tomuto není paměť zahlcována a při velkých simulacích dochází k úsporám paměti.

Virtuální síť: Virtuální síť je v simulátoru Shadow hlavním rozhraním pro komunikaci mezi virtuálními uzly. Při svém vzniku dostane každý uzel přiřazenou IP adresu a šířku pásma podle konfigurace v simulačním skriptu. Každá virtuální síť implementuje model tekoucího vědra (Leaky Bucket model), který dovoluje rozložení datového provozu. Tok v síti je řízen tak, že pakety nemohou být přeposílány větší než námi konfigurovanou rychlostí. Pomocí knihovny virtuálních soketů implementuje Shadow funkcionality soketu jako například vytváření, otevíraní a zavírání soketu, posílání a přijímání dat a také síťové protokoly jako UDP a TCP. Funkce ze systémového rozhraní soketu jsou přesměrovány na jejich protějšky v simulátoru Shadow obdobně jako níže v sekci knihovny. Když aplikace posílá data do virtuální knihovny soketu, data jsou uložena jako pakety [10]. V průběhu simulace, kdy paket prochází různými sokety a vyrovnávací pamětí, se nepřenáší pakety, ale pouze ukazatele na daný paket.

Knihovny: Systémové a knihovní funkce jsou přesměrovány na jejich protějšky v simulátoru Shadow. Probíhá to tak, že při volání funkce simulátorem Shadow se nejprve zkontroluje přednačtená knihovna, zda obsahuje zavolanou funkci. V případě, že knihovna má tuto funkci, je tato přednačtená funkce zavolána. V opačném případě je využit standardní mechanismus pro volání funkce. V simulačním skriptu se používá atribut preload pro konfiguraci cesty ke knihovně obsahující přednačtené funkce.

Virtuální systémy: Virtuální systém zahrnuje virtuální model CPU, aby zohlednil zpoždění způsobené aplikací. Využívání virtuálního CPU a zohlednění zpoždění zvyšují přesnost simulátoru Shadow, bez toho by byla všechna data zpracována aplikací v jediném diskrétním okamžiku simulace [10].

Shadow plugin tor: Shadow-plugin-tor integruje Tor do simulátoru Shadow zabalením zdrojového kódu Tor s funkcemi potřebnými pro komunikaci se simulátorem Shadow. Shadow-plugin-tor také zahrnuje skripty pomáhající při analýze výsledků, generování síťových topologií a běhu experimentů [15]. Simulace většinou běží se zmenšenými verzemi topologií a menším počtem klientů. Správné škálování simulované sítě může být komplikované. Například více uzlů s malou šířkou pásma nebude mít stejnou propustnost jako méně uzlů s větší šířkou pásma, i když se celkové kapacity shodují [10].

3.3 Porovnání simulátorů

V této sekci je shrnuto fungování obou simulátorů. Jsou zmíněny společné znaky a také popsány jejich rozdíly. Simulátor TorPS simuluje výběr cesty v síti Tor. Na rozdíl od simulátoru Shadow se zaměřuje na specifický problém: TorPS byl původně vytvořen pro práci zabývající se korelačními útoky na síť Tor a jejich hrozbou. Autoři simulátoru tvrdí, že díky použitým metodám byli schopni realisticky odhadnout zranitelnost sítě Tor vůči korelačním útokům [12]. Pro simulátor jsou důležité dva modely útočníka: Relay Adversary, kdy má útočník pod kontrolou některé z uzlů sítě, nebo Network Adversary, kdy útočník sleduje část sítě. Simulátor pracuje s modely uživatelů, které rozděluje do skupin podle jejich aktivity (typický, BitTorrent, IRC atd.). TorPS nebere v potaz efekty sítě, jako je například přetížení [20]. Simulátor TorPS stejně jako Shadow pracuje s historickými daty dostupnými na portálu Tor Metrics[3], TorPS ale nevyužívá originální zdrojový kód Tor a je napsán jako několik Python skriptů. TorPS obsahuje také simulátor zabývající se alternativním výběrem cesty tzv. Congestion Aware Tor (CAT).

Shadow byl vytvořen pro zvýšení přesnosti, konzistence a škálovatelnosti experimentů na síti Tor. Stejně jako TorPS umožňuje Shadow simulovat síť Tor jako proces s uživatelskými právy, a to na jednom zařízení [10]. To může limitovat některé větší simulace, protože není možné rozprostřít zatížení na více zařízení. Shadow dále snižuje složitost a trvání simulace tím, že neprovádí kryptografické operace [20]. Shadow je dostupný a jednoduchý na instalaci, proto je využíván studenty, vývojáři a výzkumníky po celém světě. Oproti TorPS využívá Shadow originální zdrojový kód Tor. Díky tomu je možné provádět změny a experimenty přímo ve zdrojovém kódu sítě Tor. Aplikace běží v simulátoru pomocí pluginů, například shadow plugin tor nebo také Tgen[4], který slouží k simulaci datových toků. Na rozdíl od TorPS využívá Shadow snižování velikosti sítě (downscaling), a to jak pro počet uživatelů, tak i serverů. Vývoj na simulátoru stále probíhá a je udržován, na Githubu[5] má aktivní komunitu uživatelů a technickou podporu.

Souhrn

Tabulka 3.1 shrnuje rozdíly mezi simulátory dle daných charakteristik:

[3] https://metrics.torproject.org/collector.html
[4] https://github.com/shadow/tgen
[5] https://github.com/shadow/shadow

- **Velikost / Počet uzlů:** jak se simulátor staví ke snižování velikosti sítě (downscaling).
- **Směrování:** jak se simulátor staví k směrování v síti Tor a jak se jej snaží napodobit.
- **Topologie:** jestli simulátory berou v potaz geografické rozložení sítě a jak se modeluje distribuce šířky pásma.
- **Efekty sítě:** například přetížení.

	Shadow	**TorPS**
Velikost / Počet uzlů	Ano	Ne
Směrování	nepoužívá váhy při výběru uzlu	–
Topologie	Stejné jako Tor	Stejné jako Tor
Efekty sítě	Ano	Ne
Počet uživatelů	Ano (downscaling)	Ne
Modely uživatelů	5 modelů	5 modelů
Modely útočníků	Umožňuje modelovat	2 modely
Aktuálně je udržován	Ano	Nevím, poslední aktualizace 2017
Originál kód Tor	Ano	Ne, Python aplikace

Tabulka 3.1: Porovnání simulátorů, převzato z [20].

Kapitola 4

Návrh

V této kapitole se nejprve zaměřuji na simulace v simulátoru TorPS, viz sekce 4.1. V této sekci je popsán návrh samotné webové aplikace, která nejen demonstruje základní fungování sítě Tor, ale také některé útoky na síť Tor, viz sekce 4.1.1. Následuje simulace korelačního útoku přímo pomocí simulátoru TorPS, viz 4.1.2. V druhé části kapitoly uvádím simulace provedené v simulátoru Shadow, viz sekce 4.2.

4.1 Simulace v simulátoru TorPS

V této sekci se nejprve zaměřuji na webovou aplikaci využívající simulátor TorPS. V této části jsou představeny jednotlivé simulace, které lze přes webovou aplikaci spustit. V druhé části je simulace korelačního útoku uskutečněna mimo navrženou aplikaci přímo pomocí simulátoru TorPS kvůli náročnosti na zdroje.

4.1.1 Návrh webové aplikace

Jedním z cílů této práce je využití dostupných simulačních nástrojů sítě Tor pro účely demonstrace fungování sítě a útoků na ni. Proto jsem se rozhodl navrhnout webovou aplikaci, která umožňuje uživateli simulovat základní principy fungování sítě Tor. Protože cílem práce bylo využití aplikace pro výukové a demonstrační účely, výsledná aplikace by tedy měla být přehledná, jednoduchá na ovládání a měla by uživateli pomoci v pochopení problematiky sítě Tor. Navržená aplikace dává možnost uživateli spustit několik druhů simulací dle zadaných parametrů a výsledky zobrazí pomocí grafů a tabulek.

Aplikace využívá pouze jeden ze simulátorů, a to TorPS. Pro TorPS se takováto aplikace hodí, zjednodušuje zadávání parametrů simulace, podle kterých generuje data. Na základě vygenerovaných dat se daná simulace spustí. Výsledek simulace se pak zobrazí v podobě grafu a tabulek pro jednoduchost a srozumitelnost. Simulátor Shadow by ale z vytvoření uživatelského rozhraní nic nového nezískal, protože simulace jsou v něm náročné a trvají i několik hodin. Shadow byl vytvořen jako aplikace pro příkazový řádek, parametry se zadávají prostřednictvím simulačního skriptu nebo upravením souborů simulace. Uživatelské rozhraní pro simulátor Shadow by nijak nezjednodušilo ani práci se simulátorem, ani proces simulace, proto je zbytečné. Z tohoto důvodu jsem se soustředil na simulátor TorPS.

Smyslem aplikace je provedení simulace zadané uživatelem, zpracování jejích výsledků ve formě grafu a tabulek s daty. Aplikace by měla být schopna vytvořit vlastní soubory server deskriptor a konsensus, které slouží jako vstupní soubory pro simulaci. Soubory by

měly odpovídat server deskriptorům a konsensus souborům používaných sítí Tor, formát souboru je podrobně vysvětlen na webové stránce[1].

Druhy simulací

Aplikace by měla simulovat tyto simulace:

- Path – simuluje výběr cesty v síti Tor.
- Onion Service – simuluje princip připojení uživatele ke skryté službě (Onion service).
- Correlation Attack – simuluje fungování korelačního útoku.
- Exit Attack – simuluje útočníka, který odposlouchává datové toky vycházející ze sítě Tor.
- Multiple simulations – spustí zaráz několik simulací, výsledný graf znázorňuje všechny simulace. Aplikace také vytvoří statistiky popisující každou ze simulací.

Parametry

V navržené aplikaci je možné zadat parametry simulace:

- Remove duplicite paths – odstraní duplicitní cesty vygenerované simulátorem.
- Same bandwidth – každý vytvořený uzel bude mít stejnou šířku pásma.
- Guard – počet vstupních uzlů.
- Middle – počet uzlů na prostřední pozici v okruhu.
- Exit – počet výstupních uzlů.
- Number of simulations - počet vygenerovaných cest.
- Simulation size – druh a velikost grafu (pouze pro simulaci Path).
- Path selection – druh výběru cesty (pouze pro simulaci Path).
- Encryption (%) – podíl zašifrované komunikace mezi koncovými uživateli v procentech (end-to-end encryption).
- ID occurrence (%) - pravděpodobnost v procentech, že se při komunikaci objeví nějaké ID (v reálné síti například heslo, emailová adresa atd.).
- Adversary guard – počet nepřátelských vstupních uzlů.
- Adversary exit – počet nepřátelských výstupních uzlů.
- Adversary guard bandwidth (KB/s) – šířka pásma nepřátelských vstupních uzlů.
- Adversary exit bandwidth (KB/s) – šířka pásma nepřátelských výstupních uzlů.

[1] https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt

Pro každý typ simulace by měla aplikace obsahovat několik příkladů, které si uživatel může spustit. Uživatel by měl v aplikaci také zadávat vlastní uzly, a to vepsáním názvu, IP adresy, šířky pásma a zvolením pozice uzlu v okruhu. Všechny tyto parametry lze zadat buď přímo v aplikaci, nebo nahráním konfiguračního souboru s příponou `.ini`. Program zkontroluje hodnoty zadané uživatelem. Pokud jsou hodnoty zadané správně, aplikace vygeneruje data určená k simulaci, v opačném případě vypíše chybové hlášení. Z parametrů zadaných uživatelem vygeneruje aplikace server deskriptor a konsensus soubory, ty pak slouží jako vstupní data pro samotnou simulaci. Poté, co aplikace dokončí simulaci, jsou výsledná data zpracována do statistik a tabulek. Z dat jsou vytvořeny grafy, jejich podoba závisí na zvoleném typu simulace. Některé grafy jsou interaktivní a lze na nich sledovat kroky simulace. Aplikace umožňuje stažení vygenerovaných grafů i se skripty potřebnými, aby graf byl interaktivní.

Aplikace Graphviz

V navržené aplikaci se používá pro generování grafů aplikace Graphviz (Graph Visualization Software) a python knihovna Graphviz. Aplikace Graphviz je dostupná jako open source a na zařízeních Linux je dostupná přes příkazový řádek. Graphviz využívá formátu .dot pro specifikaci grafu, ty pak umožňuje generovat ve formátu SVG. Grafy mohou být rozděleny na několik vrstev, pomocí jednoduchých skriptů pak mohu určité vrstvy zobrazit nebo skrýt. Díky tomuto principu lze vytvářet interaktivní grafy.

Simulace výběru cesty (Path selection)

Simulace výběru cesty má za cíl demonstrovat, jak probíhá výběr cesty v síti Tor, což jsem popsal v sekci 2.5. Vybrané cesty jsou znázorněny grafem. Lze si vybrat ze dvou typů grafů. První z nich slouží pro jednoduchou demonstraci fungování sítě Tor, simulace obsahuje pro přehlednost jen malý počet uzlů. Graf má zobrazovat uzly, jejich pozici v okruhu a jejich IP adresy. Druhý graf funguje obdobně jako ten první, akorát umí znázornit větší síť, například 150 uzlů. Uzly jsou pro přehlednost barevně odlišeny podle své pozice v okruhu. Grafy ukazují vytvořené okruhy mezi uživatelem a serverem přes jednotlivé uzly. Každý graf je vytvořen ve formátu SVG s několika vrstvami, každá vrstva znázorňuje jeden okruh v síti Tor. Skript umožňuje přepínání mezi jednotlivými vrstvami. Výsledný graf je tím pádem interaktivní. Kromě výsledného grafu také aplikace zobrazuje v tabulce přehled všech cest, které byly vytvořeny. Dále aplikace generuje druhou tabulku s jednotlivými uzly, jejich šířkou pásma a kolikrát byly uzly použity.

Tato simulace názorně demonstruje, jak probíhal výběr okruhu v minulosti a jak se liší od současnosti. Nejprve výběr vstupního uzlu probíhal náhodně, od této možnosti se ale upustilo, protože potenciální útočník měl větší šanci provést korelaci. Poté následovala metoda, kdy si uživatel vybral tři až čtyři uzly jako vstupní uzly do sítě Tor pro všechny okruhy, tyto uzly se obměňovaly každých 30 až 60 dnů. Od této metody se také upustilo, v dnešní době si uživatel vybírá pouze jeden uzel pro vytvoření všech okruhů v síti Tor. Zadáním parametru `Path selection` lze vybrat, který z těchto tří výběrů cesty bude aplikace simulovat:

- `Random` – vybírá vstupní uzel ze všech uzlů - v minulosti takto fungoval výběr vstupního uzlu.
- `3 guards` – vybere tři uzly a ty používá jako guard pro všechny vytvářené okruhy.

- `1 guard` – vybere jeden uzel a ten používá jako vstupní uzel pro všechny vytvářené okruhy (aktuálně v síti Tor).

Skryté služby (Onion services)

V případě simulace skrytých služeb má výsledný graf za cíl ukázat navázání komunikace mezi uživatelem a skrytou službou. V grafu se zobrazuje, jak probíhá komunikace mezi uživatelem, tzv. „rendezvous point“, tzv. „introduction point“, skrytou službou a adresářovým serverem. Princip této komunikace je detailně popsán v sekci 2.7. Stejně jako v minulém případě užití je graf interaktivní a je vygenerován programem Graphviz. Graf má několik vrstev, každá z nich znázorňuje jednu cestu v síti Tor. Cesty jsou barevně rozlišeny, aby nedošlo k záměně jednotlivých cest.

Simulace korelačního útoku (Correlation attack)

Simulace korelačního útoku má demonstrovat, jak probíhá korelační útok, který je podrobně rozebrán v sekci 2.9.2. Výsledný graf simulace korelačního útoku ukazuje pouze cesty, při kterých útočník ovládá vstupní a výstupní uzel. To znamená, že může korelovat datové toky a deanonymizovat uživatele.

Výsledný graf znázorňuje, kolik cest prošlo přes daný uzel. Předpokládejme, že jsme si vytvořili 50 uzlů a simulovali 1000 cest. Přes uzel A prošlo 240 cest a přes uzel B prošlo pouze 15 cest. Tak v tomto případě bude uzel A vyznačen sytější barvou než uzel B, protože byl použit vícekrát. Barva uzlu je určena podle toho, jestli je v simulaci označen jako přátelský nebo nepřátelský (modrá a červená). Tvar uzlu pak koresponduje s jeho pozicí v okruhu, vstupní uzel je znázorněn jako čtvereček a výstupní uzel jako kolečko.

U grafu je tabulka ukazující, kolikrát byl který uzel použit, jakou má šířku pásma a kolik procent komunikace procházející přes uzel bylo šifrováno (například pomocí HTTPS, nebere ohled na šifrování v síti Tor) pouze šifrováním mezi koncovými uživateli (end-to-end encryption).

Simulace odposlechu výstupního uzlu (Exit attack)

Simulace odposlechu výstupního uzlu je založena na tom, že útočník odposlouchává datové toky vycházející ze sítě Tor. Útok se zaměřuje na možnost, že útočník ovládá nebo odposlouchává výstupní uzel sítě. Výstupní uzel sítě Tor komunikuje s cílovými servery. Jak bylo zmíněno v sekci zabývající se okruhy 2.5, komunikace nemusí být zabezpečená. Cílem útočníka tohoto typu je odposlouchávat komunikaci a hledat, zda uživatel posílá nějaké citlivé údaje: například hesla, e-mailové adresy atd. Neopatrné chování uživatelů je popsáno v sekci 2.8. Šifrování komunikace na internetu v průběhu let dokumentuje Google Transparency report. Například v lednu 2014 to bylo 51 % komunikace, oproti tomu v lednu 2020 to už bylo 94 %[2].

Tento případ je modelován v práci tak, že před začátkem simulace uživatel zadá parametr `ID_occurrence`. Pro účely naší simulace předpokládejme, že parametr `ID_occurrence` představuje šanci, že přes danou cestu prošlo nějaké ID. Identifikační údaje se v reálné síti vyskytují ve zprávách procházejících sítí Tor. TorPS ale nemůže simulovat posílání zpráv, simuluje pouze výběr cesty, což je pro účel demonstrace dostatečné. Dalším parametrem je, kolik procent komunikace procházející sítí je šifrované (parametr encryption). V tomto případě se jedná o šifrování mezi koncovými uživateli, ne o šifrování v síti Tor.

[2] https://transparencyreport.google.com/https/overview

Například máme simulaci, kde parametr `ID_occurrence` je 10 %. To znamená, že 10 % cest obsahuje nějaký typ ID. Při zadání parametru `Encryption` na 60 % to znamená, že 60 % celkové komunikace je šifrováno mezi koncovými uživateli. Předpokládejme, že pokud je komunikace šifrovaná, nebude se útočník snažit zprávu dešifrovat a ID je v pořádku. V tomto případě se tedy u každé cesty s pravděpodobností 10 % rozhodne, zda obsahuje nějaké ID, a s pravděpodobností 60 %, zda je tato cesta šifrovaná. Pokud cesta prochází přes nepřátelský uzel a není šifrovaná, tak útočník získá ID, pokud ale šifrovaná je, útočník ID nezíská.

Výsledná tabulka zahrnuje uzly, jejich šířku pásma, kolik ID prošlo přes tento uzel a kolik ID z toho bylo ukradeno. Poslední sloupec tabulky ukazuje předchozí hodnotu v procentech. Za situace, že je uzel na vstupní pozici v okruhu a dané ID neukradl, a zároveň výstupní uzel je nepřátelský a ID ukradne, do statistik všech tří uzlů v okruhu se počítá, že ID bylo ukradeno.

4.1.2 Simulace korelačního útoku mimo navrženou aplikaci

Na rozdíl od předchozího korelačního útoku popsaného v sekci 4.1.1 tato simulace nedemonstruje výhradně princip fungování, ale simuluje korelační útok na síť Tor. Pro provedení korelačního útoku je nutné použít dva nepřátelské uzly, jeden v pozici guard a druhý v pozici exit. Samotný jeden uzel by nestačil, protože by nemohl být ve stejném okruhu použit dvakrát. Data jsou brána z Tor Metrics. Výsledkem simulace jsou grafy zobrazující procentuální šanci na deanonymizaci uživatele a kolikrát byly nepřátelské uzly vybrány v okruhu.

4.2 Simulace v simulátoru Shadow

V rámci práce jsou uskutečněny simulace mimo navrženou aplikaci pro demonstraci využití simulátoru Shadow. Simulátor Shadow není zařazen do navržené aplikace, protože by z vytvoření uživatelského rozhraní nic nového nezískal. Je to z toho důvodu, že simulace v Shadow jsou časově náročné a potřebují hodně zdrojů.

Simulace plánovačů a jejich vliv na síť Tor: Simulace využívá simulátoru Shadow. V této simulaci plánovačů jsou simulovány tři algoritmy: KIST, KISTLite, Vanilla. Simulace se zaměřuje na to, jaký vliv mají tyto algoritmy na výkon sítě Tor. Pro simulaci algoritmů jsou využita historická data z Tor Metrics a na základě těchto dat probíhá simulace všech tří plánovačů KIST, KISTLite a Vanilla. Výsledky simulace v podobě grafů ukazují vliv jednotlivých plánovačů na síť Tor.

Simulace útoku Dropmark: Simulace využívá simulátoru Shadow. Útok Dropmark je založen na tom, že Tor tiše zahazuje neznámé pakety. Více o útoku Dropmark je uvedeno v sekci 2.9.3. Tato simulace používá upravenou verzi Toru, která umí zachytit značku Dropmark a historická data z Tor Metrics.

Kapitola 5

Implementace

V této kapitole je popsána implementace navržené webové aplikace pro simulátor TorPS. V první sekci kapitoly uvádím implementaci webové aplikace, viz 5.1. Je vysvětlen konfigurační skript, generování souborů server deskriptor a konsensus. Další důležitou částí je zpracování dat a využití programu Graphviz pro generování interaktivních grafů z výsledných dat simulace. Dále zmiňuji nástroj Docker, který jsem použil pro lepší kompatibilitu a jednodušší nasazení. Závěr kapitoly se zaměřuje na některé problémy při práci se simulátory, viz sekce 5.2.

5.1 Webová aplikace

Aplikaci jsem implementoval v jazyce Python3, pro vytvoření webového rozhraní jsem použil PHP7.2, Bootstrap 4.3 a Bootstrap Table 1.15. Samotný simulátor TorPS využívá jazyk Python2, který už ale v této době není podporován. Z toho důvodu jsem využil aplikace Docker, protože tato aplikace řeší problémy s kompatibilitou, jak vysvětluji níže.

Uživatel může v aplikaci zadat parametry pro různé simulace. Daný formulář se zadanými parametry je zpracován pomocí PHP a data jsou zapsána do konfiguračního souboru `.ini`. PHP následně zavolá Python skript `sim.py` s argumentem `-i`, který popisuje cestu ke konfiguračnímu souboru. Skript `sim.py` zkontroluje, zda jsou zadané údaje validní. Podle informací v konfiguračním souboru vytvoří za pomoci knihovny `stem` vlastní server deskriptory a konsensus soubory. Poté spustí zadanou simulaci simulátorem TorPS. Po skončení simulace `sim.py` zpracuje výsledná data a zavolá `graph.py`, aby z dat vytvořil graf pro danou simulaci. Soubor `graph.py` využívá knihovny `graphviz`, která slouží k vytváření souborů `.dot`. Tento soubor je předán nástroji `Graphviz` a ten vytvoří zadaný graf ve formátu `SVG`. Po vytvoření grafu předá skript webové aplikaci `sim.py` graf a soubory s daty ve formátu `JSON` (`usage.json`, `output.json`, `statistic.json`). Webová aplikace pak zobrazí výsledný graf a data. Skript `sim.py` může fungovat samostatně přes příkazovou řádku, stačí pouze pomocí parametru `-i` zadat cestu ke konfiguračnímu souboru.

Konfigurační soubor

Skript `sim.py` spouští simulace pomocí parametrů zadaných v konfiguračním souboru s příponou `.ini`. Celý formát konfiguračního souboru je uveden ve webové aplikaci. Tento soubor je rozdělen do několika sekcí:

- `general` – sekce s obecnými parametry simulace

- `path_simulation`, `hiden_service_simulation`, `attack_simulation`, `exit_attack` – sekce s parametry k jednotlivým typům simulacím
- `multiple_sim` – možnost spuštění více simulací
- `sim_N` – podsekce k `multiple_sim` a definuje jednu simulaci, kde N definuje číslo simulace
- `nodeN` – uzly zadané uživatelem, kde N specifikuje číslo uzlu

Generování vlastních server deskriptorů

Pro generování vlastních server deskriptorů jsem se rozhodl hned z několika různých důvodů. Jedním z nich je, že TorPS nepodporuje snižování velikosti sítě (downscaling), což znamená, že v případě použití dat z reálné sítě Tor by graf obsahoval několik tisíc uzlů. Dalším důvodem je, že zpracování dat i simulace sama o sobě by trvaly několik hodin. Pro fungování simulace potřebuji dva typy vstupních souborů, a to server deskriptor a konsensus soubor. Soubory jsou vytvořeny podle specifikací uvedených na stránce[1]. Od každého typu těchto souborů generuji pouze jeden soubor, který obsahuje všechny záznamy. Každý záznam reprezentuje jeden uzel zadaný uživatelem. Simulátor TorPS před simulací kontroluje, zda jsou server deskriptory a konsensus validní, proto musí obsahovat klíče, otisky a vstupní nebo výstupní politiky jednotlivých uzlů. Klíče a otisky jsou generovány přes knihovnu `stem`. Uživatel si v parametrech může zadat vlastní šířku pásma. Pokud není zadána, je šířka pásma generována automaticky, a to v rozmezí od 1KB/s do 2,5MB/s. V souboru konsensus používám pro všechny uzly stavy `Fast`, `Running`, `Stable`, `Valid`. Pro výstupní uzly přidávám `exit` a pro vstupní uzly `guard`. V souborech konsensus se také nachází váhy uzlů `bandwidth-weights`, ty jsem nastavil na výchozí hodnotu 10 000. Aby byly jednotlivé záznamy validní, jsou důležité následující parametry (bandwidth, reject, onion-key, signing-key, ntor-onion-key, router-signature, fingerprint). Celý záznam jednoho uzlu v souborech server deskriptoru a konsensus je v příloze A.

Zpracování dat

Po skončení simulace skript `sim.py` vytvoří statistiky podle dat z TorPS a parametrů zadaných uživatelem. Zpracování dat záleží na typu simulace. Zpracovaná data ukládá skript do tří souborů ve formátu json. Prvním souborem je `output.json`, který pro každý vygenerovaný okruh v TorPS obsahuje uzly okruhu (guard, middle, exit), zda došlo ke korelaci (correlation) a zda některý uzel je nepřátelský (affiliation). Druhým souborem je `usage.json` obsahující jeden záznam pro každý uzel. V souboru je vidět šířka pásma (bandwidth), kolikrát byl daný uzel použit (usage), kolik ID prošlo přes uzel (id), kolik ID procházejících přes uzel bylo ukradeno v procentech (id_stolen_percentage), kolik komunikace procházející přes uzel bylo šifrováno v procentech (encryption) a jestli je uzel nepřátelský (affiliation). Třetí soubor počítá statistiky pro celou simulaci: například kolikrát byl použit nepřátelský vstupní uzel (bad_guard_used) nebo kolik procent komunikace procházející přes výstupní uzly bylo šifrované (bad_exit_encrypt). Tyto soubory slouží jako data pro tabulky ve webové aplikaci, skript `sim.py` předá data skriptu `graph.py`, aby vygeneroval výsledný graf.

[1] https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt

Graphviz

Nástroj `Graphviz` slouží pro generování výsledných grafů, bere jako vstup dokument ve formátu `dot` a generuje výsledný graf ve formátu `SVG`. Pro práci s nástrojem `Graphviz` jsem použil python knihovny `Graphviz`, která umožňuje práci s formátem `dot`.

V souboru `dot` jsou definovány jednotlivé uzly a jejich IP adresy. Další součástí souboru jsou okruhy vygenerované simulátorem TorPS. Každý okruh je označen jako `path N`, kde N označuje číslo daného okruhu. Okruhy jsou vytvořeny pomocí cest mezi jednotlivými uzly. Protože cesty se mohou opakovat, každá cesta má v parametru `layer` záznam `path N` pro každý okruh, ve kterém se cesta objevila. Díky parametru `layer` a skriptu `animation.js` jsou výsledné grafy interaktivní. Ve webové aplikaci je možné přepínat postupně mezi jednotlivými okruhy, které TorPS vygeneroval. Vždy se zobrazí pouze jeden okruh.

Graphviz dovoluje generování grafů v různých rozloženích. Pro tuto práci jsem použil dva druhy rozložení (Dot, Neato). Rozložení Dot je hierarchické a hodí se pro malé grafy. Na rozdíl od toho rozložení Neato se hodí pro velké grafy s více jak 30 uzly.

Docker

Pro nasazení aplikace byl zvolen nástroj Docker, který umožňuje jednoduchou izolaci aplikace do kontejnerů a tím odstraňuje problémy s kompatibilitou. Protože mnou navržená aplikace využívá hned několik programovacích jazyků a knihoven, je program Docker ideální. Výsledný Docker kontejner je jednoduše spustitelný a přenosný. Výsledná aplikace využívá dva Docker kontejnery: prvním je `Nginx` web server, druhým je kontejner postavený na PHP obsahující všechny potřebné knihovny, Python3.6, Python2.7, Graphviz a data výsledné aplikace. Tento kontejner je dostupný přes službu Docker Hub[2]. Oba kontejnery jsou vytvořeny soubory `Dockerfile`, celá aplikace se poté spouští souborem `docker-compose-2.yml`, kde specifikuji oba použité kontejnery a jak na sobě závisí. V tomto souboru také uvádím datové svazky (data volumes) a porty, na kterých má výsledná aplikace fungovat.

5.2 Problémy při práci se simulátory

Při práci se simulátory se vyskytlo hned několik problémů, které ztěžovaly práci se simulátory. Jednalo se především o simulátor TorPS. Simulátor už pravděpodobně není udržován, jelikož poslední příspěvek na github byl v roce 2017. TorPS neobsahuje žádný typ dokumentace, jediný popis k simulátoru je soubor `README.md` na githubu[3]. Práce napsaná autory Johnson et al. [12], ke které byl TorPS vytvořen, neposkytuje mnoho informací o funkčnosti simulátoru.

Jedním z problémů bylo, že se nepodařilo zprovoznit simulátor zabývající se alternativním výběrem cesty tzv. Congestion Aware Tor (CAT). Bylo to z toho důvodu, že tato simulace potřebuje jako vstupní parametr soubor `congestion.cator.pickle`. Jelikož neexistuje skoro žádná dokumentace formátu tohoto souboru a není nikde jeho specifikace nebo způsob jak jej získat, simulaci jsem nezprovoznil. Na stejný problém narazilo už více lidí[4].

Dalším problémem byla chyba TorPS při zpracovávání souboru. Autoři doporučují použít argument `-initial_descriptor_dir`, který specifikuje složku se server deskriptory

[2] https://hub.docker.com/r/meda10/bp_tor_php/tags
[3] https://github.com/torps/torps
[4] https://github.com/torps/torps/issues/10

z měsíce před začátkem simulace. Například pokud bych chtěl simulovat síť od května do srpna, bude `-initial_descriptor_dir` obsahovat server deskriptory i za březen, aby několik prvních souborů o stavu sítě obsahovalo všechny uzly sítě. Bohužel i s tímto argumentem dochází někdy k chybám. Chyba nenastane při zpracování dat, ale až při samotné simulaci. Jako řešení fungovalo odstranit první až první tři soubory stavu sítě a provést simulaci bez nich.

Kapitola 6

Simulace a testování

V této kapitole se zabývám simulacemi, které jsem provedl mimo vytvořenou aplikaci, a testováním implementované webového aplikace. Nejprve simuluji útok Dropmark, viz sekce 6.1, poté simuluji korelační útok pomocí simulátoru TorPS, viz sekce 6.2. V další části jsem se zabýval simulací plánovačů a jakým způsobem ovlivňují výkon sítě Tor, viz sekce 6.3. Dále jsem využil implementovanou aplikaci pro simulaci výběru cesty, viz sekce 6.4, simulaci skrytých služeb, viz sekce 6.5, korelačního útoku, viz sekce 6.6, a odposlechu výstupního uzlu, viz sekce 6.7.

6.1 Útok Dropmark

V rámci této práce jsem simuloval útok Dropmark, ten využívá chování sítě Tor k neznámým buňkám, které Tor tiše zahazuje. Samotné fungování útoku bylo podrobně popsáno v sekci 2.9.3. K simulaci jsem využil simulátor Shadow společně s pluginem Tgen[1], který v Shadow generuje síťový provoz. Pro tuto simulaci jsem použil upravenou verzi Toru, která umí zachytit značku Dropmark[2].

Získání dat

Pro simulaci útoku Dropmark jsem musel nejprve získat data. Z Amazon AWS[3] jsem stáhl archiv nejpoužívanějších serverů. Tato data sloužila jako vstup pro skript `parsealexa.py` dostupného v shadow plugin tor. Výsledkem tohoto skriptu byl soubor `top-1m.csv`. Zbylé soubory byly staženy užitím příkazu `wget`:

- `server-descriptors-2017-03.tar.xz`
- `extra-infos-2017-03.tar.xz`
- `consensuses-2017-03.tar.xz`
- `userstats-relay-country.csv`

Všechna tato data slouží jako vstup pro skript `generate.py`, který je dostupný jako součást shadow plugin tor. Tento skript generuje síť Tor. Moje síť se skládá z 20 serverů,

[1] https://github.com/shadow/tgen
[2] https://github.com/frochet/dropping_on_the_edge/tree/master/dropmark/tor
[3] http://s3.amazonaws.com/alexa-static/top-1m.csv.zip

150 webových klientů a 50 uzlů sítě Tor. Celkově jsem provedl dvě simulace. První simulace odhalila počet falešně pozitivních případů (false positives), druhá simulace se už zaměřovala na samotný útok a počet falešně negativních případů (false negatives).

Simulace – neposlána značka Dropmark

První simulace měla za cíl odhalit falešně pozitivní případy, tedy zda může být standardní komunikace v síti Tor neprávem označena, že obsahuje značku Dropmark. Simulaci jsem provedl tak, že jsem vzal svoji vygenerovanou síť a před začátkem simulace upravil soubory `.torrc` v adresáři `conf`.

`tor.guard.torrc`
`ActivateSignalAttackListen 1` – zapnutí funkce pro pozorování, zda je přítomna značka Dropmark.

`tor.client.torrc`
`NewCircuitPeriod 1` – každou sekundu rozhoduje, zda vytvořit nový okruh.
`IsolateDestAddr` – nesdílí okruhy s datovými toky, které mají jinou cílovou adresu.
`MaxCircuitDirtiness 1` – znovu použití okruhu, který byl použit před jednou sekundou.

Po skončení samotné simulace jsem užitím následujícího příkazu analyzoval soubory, abych našel počet falešně pozitivních případů:

```
grep "Spotted watermark" shadow.data/hosts/relayguard*/
stdout-relayguard*.tor.1000.log | wc -l
```

Výsledkem simulace byl jeden případ falešně pozitivní. Autorům práce nastala podobná situace. Vysvětlují si to tak, že dvě buňky (connected cell) dorazily ve stejný čas jako buňka přenosová [18].

Simulace – poslána značka Dropmark

Druhá simulace měla za cíl simulovat útok Dropmark. Simulace proběhla na stejném modelu sítě. Útok Dropmark posílá značku Dropmark všem okruhům, které se chtějí připojit na specifikovanou IP adresu. Tyto IP adresy jsem získal ze souboru `shadow.config.xml` nebo pomocí následujícího příkazu, který je získá z `shadow.log`.

```
grep server shadow.log | tail -n 20 | cut -d " " -f 5 |
cut -d "~" -f 2 | cut -d "]" -f 1
```

Obdobně jako v předchozí simulaci bylo potřeba upravit soubory v adresáři `conf`. Provedl jsem stejné změny na souborech jako v první simulaci. Navíc jsem změnil následující soubory, pro každou IP adresu, kterou jsem chtěl sledovat, jsem přidal řádek `WatchAddres IP`.

`tor.exit.torrc` a `tor.exitguard.torrc`
`ActivateSignalAttackWrite 1` – spustí útok Dropmark.
`WatchAddres IP` – IP adresa, kterou chci sledovat.

V mém případě jsem poslal značku Dropmark, pokud se klient připojil k jakémukoli serveru. Síť obsahovala 20 serverů, v souborech `tor.exit.torrc a tor.exitguard.torrc` jsem definoval adresy všech 20 serverů. Pro získání výsledků simulace jsou důležité dva příkazy, první získá počet falešně negativních případů a druhý počet opravdu pozitivních případů (true positives).

```
grep "No watermark" shadow.data/hosts/relayguard*/
stdout-relayguard*.tor.1000.log | wc -l

grep "Spotted watermark" shadow.data/hosts/relayguard*/
stdout-relayguard*.tor.1000.log | wc -l
```

Výsledkem simulace bylo 164 případů falešně negativních a 7105 případů opravdu pozitivních. Tato metoda je velice úspěšná, v mé simulaci bylo procento úspěšnosti 97 %. Rochet, F. et al. [18] tvrdí, že pro potenciálního útočníka stačí, aby si klient a server vyměnili jen několik bajtů, a korelace bude úspěšná.

6.2 Korelační útok na síť Tor pomocí TorPS

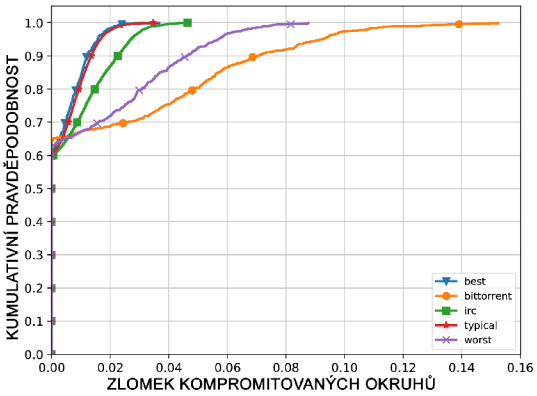
Simulace probíhá na datech z roku 2013 od února do června, k tomu jsem se rozhodl z důvodu náročnosti simulátoru. V roce 2013 byla síť Tor menší, což znamená, že zpracování dat nezabere tolik času a zdrojů počítače, především paměti RAM. Náročné je hlavně zpracování vstupních dat, v mém případě jsem na zpracování jednoho měsíce dat využil proces asi 8 GB RAM. Data jsem zpracovával postupně po měsících. Autoři simulátoru doporučují použít server deskriptory i z minulého měsíce, aby zpracované soubory byly kompletní. OR totiž publikuje nový server deskriptor každých 18 hodin, to znamená, že deskriptory pro některé OR jsou v archivu server deskriptorů z minulého měsíce. I tak jsem narazil na problém, že simulace končí chybou v kódu. Tuto chybu jsem vysvětlil v sekci problémy se simulátory 5.2.

Cílem této simulace je provést korelační útok na síť Tor. Pro simulaci jsem využil dva nepřátelské uzly, jeden v pozici guard a druhý v pozici exit. Zvolil jsem šířku pásma 100 Mib/s, tu jsem poté rozdělil mezi uzly v poměru 5:1, tedy vstupní uzel (guard) 83.3 MiB/s a výstupní uzel (exit) 16.7 Mib/s. Johnson et al. [12] testovali, jak nejlépe rozdělit šířku pásma mezi dva uzly, a doporučují poměr 5:1 (guard:exit) pro maximalizaci šance kompromitace. Data pro simulaci jsem získal z Tor Metrics, potřeboval jsem server deskriptory a konsensus soubory. Data jsem zpracoval užitím skriptu `process.sh`.

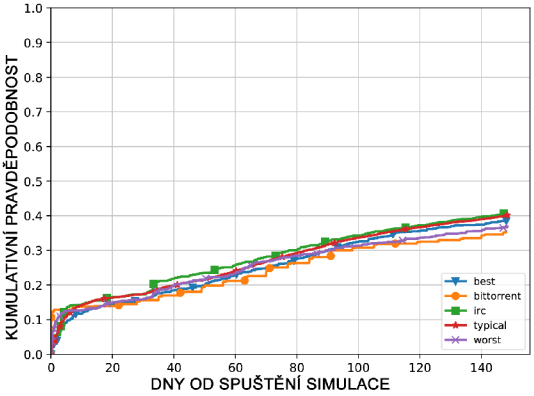
Simulace korelačního útoku

Celkově jsem provedl pět simulací korelačního útoku, pokaždé s jiným modelem uživatele. Některé simulace trvaly až 20 hodin. Výsledné soubory pak dosahovaly velikosti až 5,8 GB. Pro analýzu souborů jsem použil skript `do_analysis.sh` a `pathsim_plot_v2.py`.

Graf 6.1b ukazuje, že v průběhu pěti měsíců je 40 % šance deanonymizace uživatele. Z grafu také vyplývá, že risk deanonymizace se v průběhu času zvětšuje. Graf 6.1a ukazuje závislost vybraného uživatelského modelu na tom, jaké procento okruhů je útočník schopen kompromitovat. To závisí na tom, kolik je schopen kompromitovat vstupních uzlů, viz graf 6.3a a kolik výstupních uzlů, viz graf 6.2a. Například v grafu 6.2a je vidět, že útočník kompromitoval častěji model BitTorrent než například Typický model. Může to být způsobeno tím, na jaké porty se připojují, protože porty používané BitTorrentem nepodporuje velké

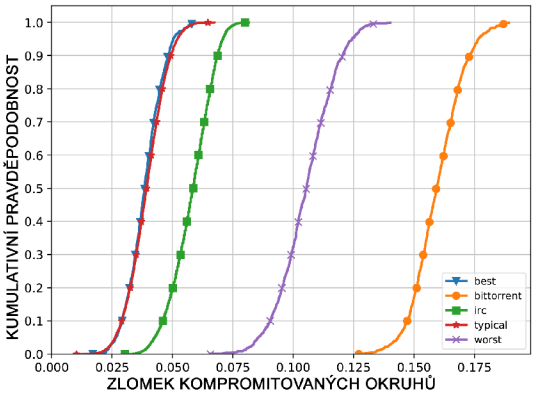


(a) Kolik okruhů byl schopen útočník kompromitovat.

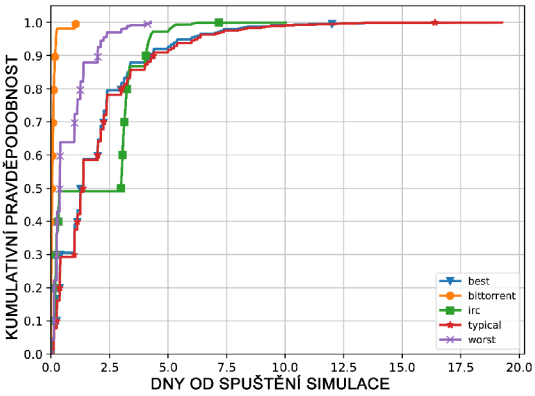


(b) Kumulativní pravděpodobnost jak dlouho trvá vybrání útočníkova uzlu guard a exit zároveň.

Obrázek 6.1: Výsledky simulace pro korelační útok.



(a) Kolik uzlů exit byl schopen útočník kompromitovat.

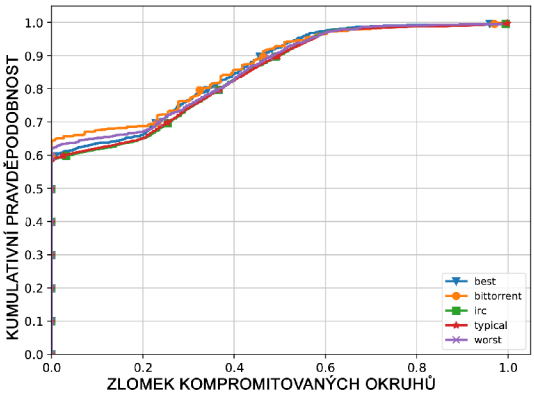


(b) Kumulativní pravděpodobnost jak dlouho trvá vybrání útočníkova uzlu Exit.

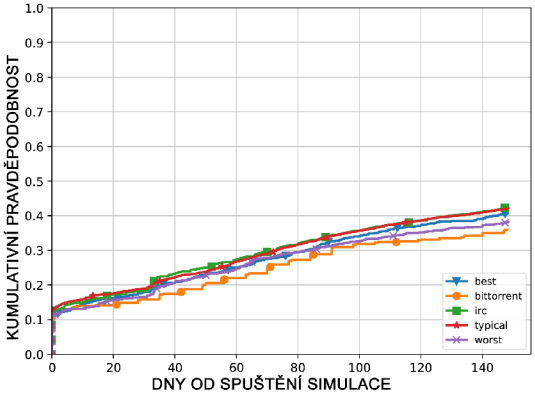
Obrázek 6.2: Výsledky simulace pro výstupní uzly.

množství výstupních uzlů. To znamená, že útočník má větší šanci jej kompromitovat [12]. V modelu s vysokou uživatelskou aktivitou se pravděpodobnost výběru kompromitovaného uzlu také zvyšuje.

Oproti typickému modelu je výstupní uzel při modelu BitTorrent rychleji a častěji kompromitován. To se pak odráží i v grafu 6.1a, kde vidíme, že uživatel BitTorrent měl nejvíce kompromitovaných okruhů. Na grafu 6.3b lze pozorovat, jak dlouho trvá, než je vybrán útočníkův uzel guard. Jak je vidět, počet kompromitovaných vstupních uzlů je zhruba stejný pro všechny uživatele. Nové vstupní uzly jsou vybrány vždy po expiraci, kterou jsem v simulaci nastavil na 30 dní. Tento princip je popsán v sekci okruhy 2.5. V grafu 6.2b je pak zobrazeno, jak dlouho trvá výběr útočníkova vstupního uzlu. Výstupní uzel je naopak vybírán nový pro každý vytvářený okruh, proto jsou tyto dva grafy tak rozdílné. Podle simulace vychází, že model nejlepšího portu je téměř identický s modelem typického uživatele. To jsem popsal v sekci 3.1.



(a) Kolik uzlů guard byl schopen útočník kompromitovat.



(b) Kumulativní pravděpodobnost jak dlouho trvá vybrání útočníkova uzlu guard.

Obrázek 6.3: Výsledky simulace pro vstupní uzly.

Autoři simulátoru TorPS tvrdí, že útočník, který má možnost sledovat síťový provoz mezi klientem a vstupním uzlem (jako například ISP), může deanonymizovat cílovou destinaci několikrát rychleji než útočník, který sleduje komunikaci vycházející ze sítě Tor a chce deanonymizovat zdroj [12].

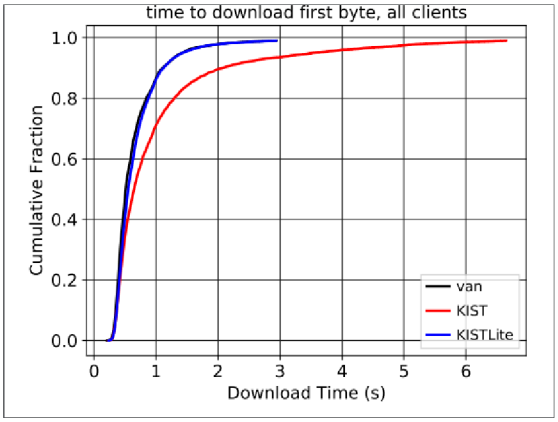
6.3 Simulace plánovačů

V této simulaci jsem se zaměřil na to, jak může výběr plánovače ovlivnit výkon sítě Tor. Plánovače jsem popsal v sekci 2.6. Simulaci jsem provedl v simulátoru Shadow, k jejímu provedení jsem použil shadow plugin tor a Tgen. Data jsem získal z Tor Metrics stejným způsobem, jak jsem popsal v útoku Dropmark, viz sekce 6.1. Ze získaných dat jsem vytvořil použitím skriptu `generate.py` topologii sítě obsahující:

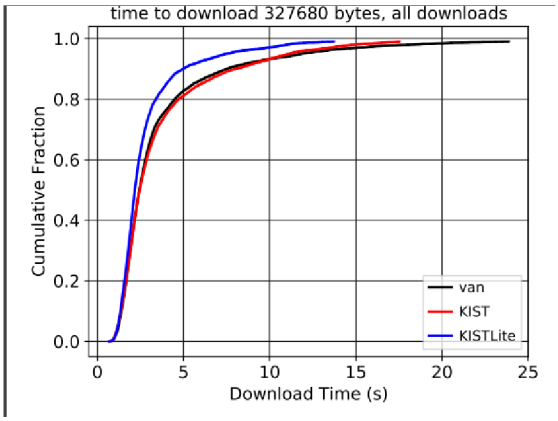
- `server` – 20× servery, na které se klienti mohou připojovat
- `webclient` – 135× webový klient
- `bulkclient` – 10×
- `relay` – 20× uzly sítě Tor
- `perf50kclient` – 10× klienti stahující soubory o velikosti 50kB
- `perf1mclient` – 10× klienti stahující soubory o velikosti 1MB
- `perf5mclient` – 10× klienti stahující soubory o velikosti 5MB

Celkově jsem provedl tři simulace, u každé z nich jsem v souborech `.torrc` pomocí parametru `Schedulers` upravil, jaký plánovač má být použit. Soubory `.torrc` jsou popsány na webové stránce[4]. Ze simulace byly vygenerovány grafy ukazující, jaký mají vliv algoritmy plánovačů na výkon sítě Tor. První graf 6.4a by měl indikovat, jaký má vybraný algoritmus vliv na zpoždění okruhu a to tím, že ukazuje čas, než klient obdrží první byte. V grafu lze

[4] https://2019.www.torproject.org/docs/tor-manual.html.en

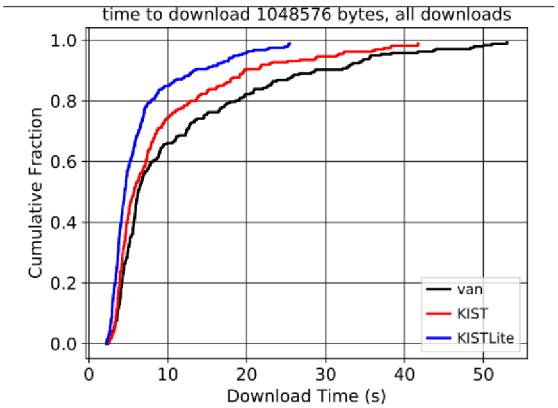


(a) Čas, než klient obdrží první byte.

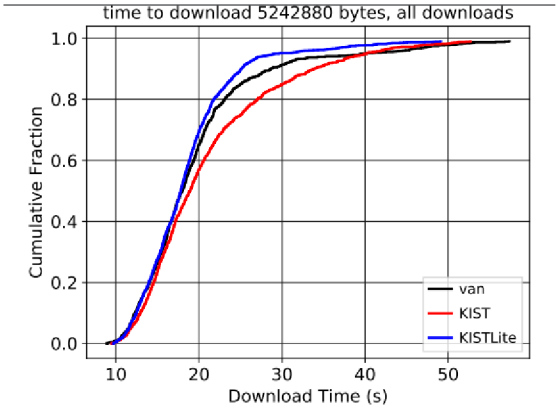


(b) Čas na stažení souboru o velikosti 320 kB.

Obrázek 6.4: Vliv plánovačů na výkon sítě.



(a) Čas na stažení souboru o velikosti 1 MB.



(b) Čas na stažení souboru o velikosti 5 MB.

Obrázek 6.5: Porovnání plánovačů, čas na stažení souborů o velikosti 1MB a 5MB.

vidět, že algoritmy Vanilla a KISTLite měly výsledky téměř totožné. Oproti tomu výsledky algoritmu KIST ukázaly větší zpoždění. To je překvapující, protože KIST je v síti Tor aktuálně používaný plánovač a jeho cílem bylo snížit zpoždění, viz sekce 2.6.

Následující graf 6.4b zobrazuje, jak dlouho klientovi trvá stažení souboru o velikosti 320 kB. V tomto případě se ukázalo, že nejlepší výsledek měl algoritmus KISTLite, následoval KIST a nejdelší čas stažení měl algoritmus Vanilla. Obdobně pak grafy 6.5a a 6.5b ukazují čas potřebný ke stažení souborů o velikosti 1 MB a 5 MB. V obou případech má nejkratší čas stažení algoritmus KISTLite a Vanilla je v obou případech nejhorší.

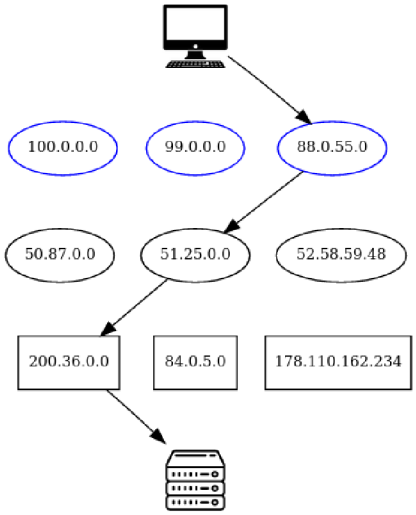
Výsledky můžeme srovnat s reálnými daty sítě Tor z daného období, viz příloha B, kdy graf ukazuje výkon sítě Tor při stahování souboru o zadané velikosti. V grafu lze vidět minima, maxima a medián času potřebného ke stažení souboru. Algoritmus KIST prioritizuje tzv. „bursty clients“ před tzv. „bulk clients“. Jelikož tzv. „bursty clients“ mají přestávky mezi stahováním dat, jejich datové toky mají tedy přednost [11]. Tento jev ukazují grafy 6.5a, 6.5b, kdy KIST měl lepší výsledky pro klienty stahující 1 MB než pro klienty stahující 5 MB.

Simulací plánovačů jsem porovnal, jak jednotlivé algoritmy ovlivňují síť Tor. Zjistil jsem, jak se chovají ke klientům stahujícím různě velké soubory. Simulace byla také porovnána s reálnými statistikami výkonu sítě Tor.

6.4 Simulace výběru cesty

Experiment 1

Prvním experimentem bych chtěl ukázat fungování výběru cesty v síti Tor. Pro tento experiment jsem si vytvořil síť s následujícími parametry: 3 uzly guard, 3 uzly exit, 3 uzly middle, 100 simulací cest a velikost simulace jsem nastavil na `small`. Pro tuto simulaci jsem si vytvořil vlastní uzly, všechny se šířkou pásma 1500 KB/s. Výsledkem simulace je následující obrázek 6.6, který ilustruje výběr cesty v síti Tor. Prvním uzlem musí být vždy uzel se statusem guard, posledním musí být vždy uzel se statusem exit. V implementované aplikaci je graf interaktivní a můžeme přepínat mezi jednotlivými okruhy.



Obrázek 6.6: Obrázek ukazuje jednoduchý graf výběru cesty.

Experiment 2

V tomto experimentu jsem zadal stejné parametry jako v předchozím experimentu, ale nyní jsem se zaměřil na tabulku v sekci Usage, která ukazuje IP adresy uzlů, jejich šířku pásma a kolikrát byly použity. V mém případně byl nejvíce používaný uzel v prostřední pozici s IP adresou 50.87.0.0 a nejméně používaný uzel v pozici guard s IP adresou 100.0.0.0., který měl 32 využití, viz obrázek 6.7a. Poté jsem přešel do konfiguračního souboru a změnil šířku pásma u uzlu 100.0.0.0 z 1500 KB/s na 2500 KB/s. Simulaci jsem spustil znovu a podíval se na výsledky. Uzel 100.0.0.0 je v tomto případě využit 51×, viz obrázek 6.7b. To znamená, že více než polovina všech generovaných cest šla přes tento vstupní uzel.

Stejný princip platí i u větší sítě. Při vytvoření sítě o velikosti 85 uzlů jsem nechal simulátor, aby jim náhodně přiřadil šířku pásma. Přidal jsem uzel guard s IP adresou 100.0.0.0 a se šířkou pásma 1500 KB/s. Zadal jsem 2000 simulací, viz obrázek 6.8a. Poté jsem stejný experiment zopakoval, ale se šířkou pásma uzlu 100.0.0.0 2500 KB/s, viz obrázek

IP	KB/s	Usage
100.0.0.0	1500	32

(a) Využití uzlu 100.0.0.0 při 1500 KB/s.

IP	KB/s	Usage
100.0.0.0	2500	51

(b) Využití uzlu 100.0.0.0 při 2500 KB/s.

Obrázek 6.7: Síť s šesti uzly a 100 simulacemi.

IP	KB/s	Usage
100.0.0.0	1500	88

(a) Využití uzlu 100.0.0.0 při 1500 KB/s.

IP	KB/s	Usage
100.0.0.0	2500	149

(b) Využití uzlu 100.0.0.0 při 2500 KB/s.

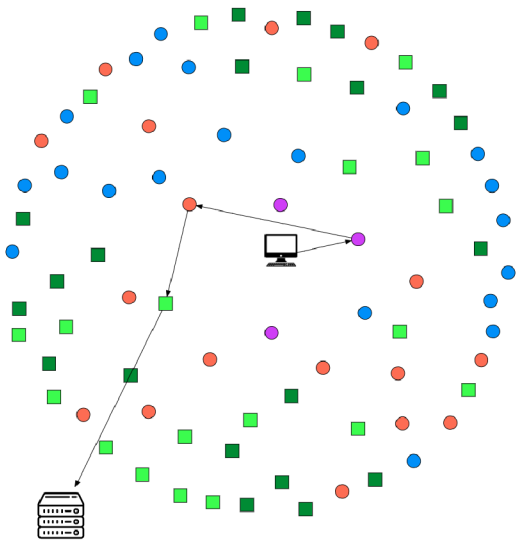
Obrázek 6.8: Síť s 85 uzly a 2000 simulacemi.

6.8a. Při prvním pokusu byl uzel využit 88×, v druhém případě 149×, což je nárůst o 69 %. Aplikace přiřazuje generovaným uzlům šířku pásma v rozpětí od 1 KB/s do 2,5 MB/s. To znamená, že mnou vytvořený uzel měl v simulaci nejvyšší šířku pásma ze všech uzlů.

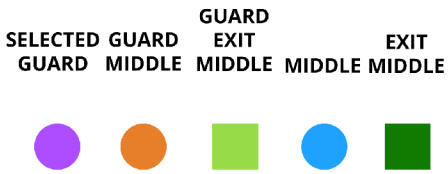
Experiment 3

V tomto experimentu jsem se podíval, jak fungují uzly se statusem guard, a porovnal jsem, jak fungoval výběr okruhu v minulosti a jak funguje dnes.

V první simulaci jsem si vytvořil síť, která obsahuje 20 uzlů od každého typu. Zadal jsem parametr `Path selection: random`. Výsledný graf a tabulka zobrazují, že každá vytvořená cesta má jiný vstupní uzel. Takto fungovala síť Tor v minulosti. Později se z důvodu bezpečnosti přešlo na výběr tří uzlů, které figurují jako uzel guard ve všech vytvořených okruzích. Toto lze znázornit parametrem `Path selection: 3 guards`, viz obrázek 6.9. V současnosti se používá jeden uzel jako guard, což lze simulovat parametrem `Path selection: 1 guard`.



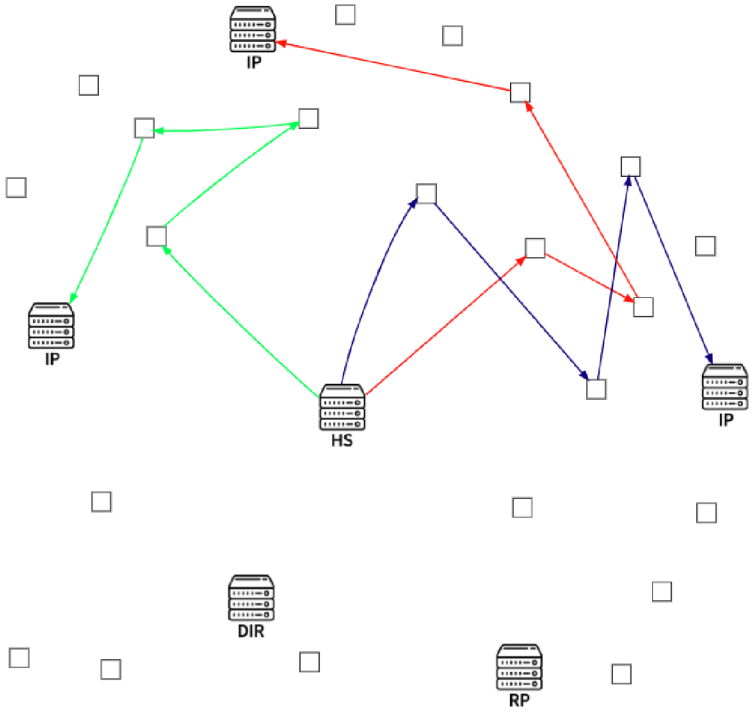
Obrázek 6.9: Obrázek ukazuje výběr cesty s parametrem `Path selection: 3 guards`.



Obrázek 6.10: Legenda pro obrázek 6.9

6.5 Simulace skrytých služeb

Simulace skrytých služeb slouží k pochopení využití tzv. „rendezvous point“ a tzv. „introduction points“ v komunikaci mezi uživatelem a skrytým serverem. Graf názorně ukazuje, v jakých krocích probíhá navázání komunikace. Výsledný graf je interaktivní. Následující obrázek 6.11 ilustruje připojení skryté služby k tzv. „introduction points“.

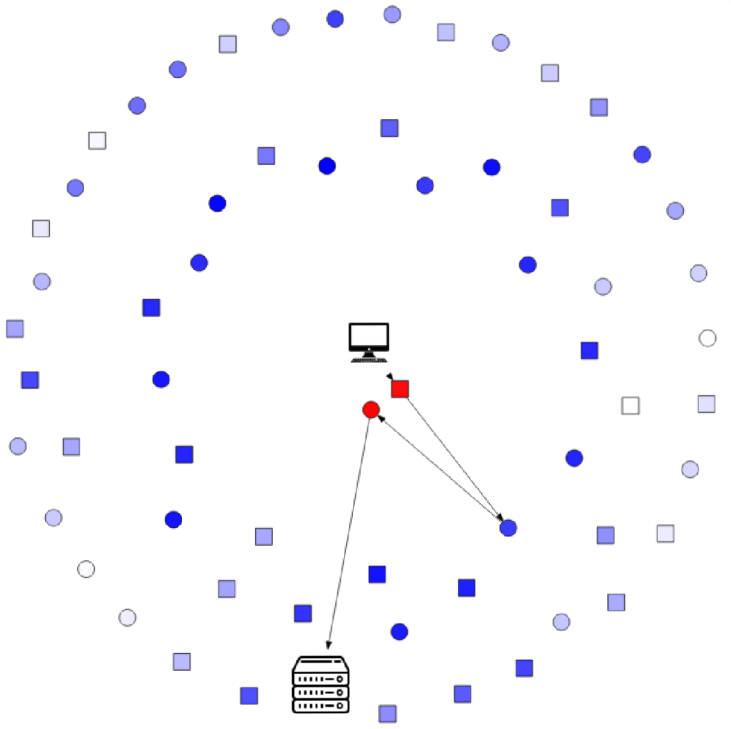


Obrázek 6.11: Připojení skryté služby k tzv. „introduction points“.

6.6 Simulace korelačního útoku

V této simulaci jsem se podíval na průběh korelačního útoku. Cílem korelačního útoku je deanonymizování uživatele útočníkem. Je možné i zjistit, kolik výstupní komunikace může útočník sledovat, pokud zadáme parametr `Encryption`. Vytvořil jsem síť obsahující 30 vstupních a 30 výstupních uzlů. Útočník ovládá jeden vstupní i výstupní uzel se šířkou pásma 2400 KB/s. Zadal jsem 10 000 simulací. Sytost barvy v grafu 6.12 znázorňuje, kolikrát byl daný uzel použit. Podle statistik je vidět, kolikrát byly nepřátelské uzly použity, například útočníkův vstupní uzel byl použit jako guard 550× a výstupní uzel jako exit

578×, viz obrázek 6.13a. V tomto případě provedl útočník korelaci okruhu 33x. Když jsem změnil parametr exit bandwidth na 1000 KB/s, tak se počet korelací snížil na 17. Útočníkův vstupní uzel byl použit na pozici guard 624× a výstupní uzel na pozici exit 242×, viz obrázek 6.13b. Z tohoto experimentu vyplývá, že útočník, který je schopen provozovat uzly s větší šířkou pásma, je schopen korelovat více datových toků.



Obrázek 6.12: Graf korelačního útoku.

ADV as guard	550
ADV as middle	583
ADV as exit	578
ADV correlation	33

(a) Statistiky první simulace.

ADV as guard	624
ADV as middle	423
ADV as exit	242
ADV correlation	17

(b) Statistiky druhé simulace.

Obrázek 6.13: Statistiky ukazující počty užití nepřátelských uzlů.

6.7 Simulace odposlechu výstupního uzlu

V simulaci odposlechu výstupního uzlu jsem se zaměřil na to, kolik uzlů a s jakou šířkou pásma musí útočník vlastnit, aby maximalizoval počet ukradených ID v komunikaci.

Experiment 1

Vytvořil jsem si síť s 30 guard a 30 exit uzly. Útočník vlastnil dva uzly se šířkou pásma 2400 KB/s. V první simulaci jsem zadal parametr `identification_occurrence` 25, počet simulací 10 000 a `Encryption` 0. V druhé simulaci jsem změnil dva parametry útočníka: šířku pásma na 1200 KB/s a počet výstupních uzlů na čtyři. V třetí simulaci změnil parametry útočníka: šířka pásma 800 KB/s a počet uzlů šest. V poslední simulaci byla šířka pásma 600 KB/s a počet uzlů osm. Výsledky simulací jsou v příloze C. Když jsem sečetl počet ID, které byl útočník schopen odcizit, došel jsem k výsledku, že útočník získal nejvíce ID při čtyřech uzlech a šířce pásma 600 KB/s.

Experiment 2

Pro tyto simulace jsem využil parametry předchozí simulace, která byla nejúspěšnější, a vytvořil jsem tři simulace: první s parametrem `Encryption` na 30 %, druhou s parametrem `Encryption` na 60 % a třetí na 90 %. Lze pozorovat, že se zvyšujícím se parametrem `Encryption`, je útočník schopen získat méně ID.

IP	KB/s	ID's	Stolen	Stolen %
10.3.0.0	1200	107	84	79
10.2.0.0	1200	122	89	73
10.4.0.0	1200	115	89	77
10.1.0.0	1200	105	71	68

(a) Počet ukradených ID při šesti uzlech a šířce pásma 800 KB/s.

IP	KB/s	ID's	Stolen	Stolen %
10.2.0.0	1200	106	45	42
10.4.0.0	1200	109	46	42
10.3.0.0	1200	119	44	37
10.1.0.0	1200	102	48	47

(b) Počet ukradených ID při osmi uzlech a šířce pásma 600 KB/s.

Obrázek 6.14: Tabulky s počtem ukradených ID.

IP	KB/s	ID's	Stolen	Stolen %
10.1.0.0	1200	111	10	9
10.4.0.0	1200	121	16	13
10.3.0.0	1200	104	13	12
10.2.0.0	1200	102	9	9

Obrázek 6.15: Počet ukradených ID při osmi uzlech a šířce pásma 600 KB/s.

Kapitola 7

Vyhodnocení

V této kapitole bych rád vyhodnotil, jak se povedlo implementovat webovou aplikaci a simulace mimo ni. Tyto simulace sloužily nejen pro demonstraci, jak simulátory pracují a čeho jsou schopny, ale také pro simulaci útoků na síť Tor a její fungování.

Cílem práce bylo demonstrovat fungování sítě Tor. Z toho důvodu jsem navrhl a implementoval webovou aplikaci. Vytvořená webová aplikace je založena na simulátoru TorPS a umožňuje simulovat pět různých případů užití. Data jsou v aplikaci generována podle zadaných parametrů. Pomocí aplikace jsem demonstroval výběr cesty v síti Tor a připojení uživatele ke skrytým službám. Aplikace znázorňuje princip následujících útoků: korelační útok a útok odposlechu výstupního uzlu. Aplikace demonstruje pouze principy fungování sítě Tor, ale nemá být považována za simulaci samotné sítě Tor, protože nevyužívá historická data. Naopak simulace, které jsem dělal mimo aplikaci, využívají historická data sítě Tor a měly by reprezentovat stav reálné sítě Tor v daném čase. Aplikace generuje interaktivní grafy, které umožňují uživateli lépe pochopit mimo jiné princip výběru uzlů nebo komunikaci uživatele se skrytými službami. Ke každé simulaci jsou dostupné tabulky a statistiky popsané v sekci 4. U každého případu použití je v aplikaci popis, jak daný případ užití ve skutečnosti funguje. Pro uživatele je takto jednoduché pochopit princip daného případu užití. Společně s prací vznikly přehledné pracovní listy, které využívají vytvořenou aplikaci k demonstraci funkčnosti sítě Tor a principu útoků na síť Tor.

Přestože v nynějším stavu aplikace funguje, je tu stále prostor pro vylepšení, jako třeba možnost nahrání historických dat sítě Tor uživatelem nebo by data mohla být obsažena již v aplikaci. Jedním z vylepšení by mohlo být přidání databáze, díky které by si každý uživatel mohl uchovávat proběhlé simulace. Další přidanou funkcí by mohlo být umožnění náročnějších simulací, přičemž uživateli by pouze stačilo zadat parametry a aplikace by ho poté upozornila o skončení simulace.

Nabízí se tu možnost využít simulátor Shadow v aplikaci. K tomu jsem se ale nerozhodl z důvodu, že simulace v Shadow jsou náročné na zdroje, trvají několik hodin a výstupní soubory mají velikost několika stovek MB. Tyto vlastnosti by způsobovaly řadu problémů. Některé z těchto problémů by šlo vyřešit, ale zabralo by to nespočetně více hodin práce a výsledek by měl minimální přidanou hodnotu, protože simulátor Shadow funguje dobře jako aplikace pro příkazovou řádku.

Pro demonstraci simulátorů jsem provedl několik dalších simulací přímo v simulátorech Shadow a TorPS. Simulátor Shadow umožňuje simulovat celou síť Tor pomocí historických dat z reálné sítě Tor. Díky dobré dokumentaci a komunitě uživatelů bylo zprovoznění simulátoru Shadow a práce s ním relativně jednoduché. Shadow jsem využil k simulaci plánovačů pro ukázku, jakým způsobem ovlivňují výkon sítě Tor. Výsledná data jsem porovnal s re-

álnými daty sítě Tor. Druhá simulace v Shadow, což byl útok Dropmark, proběhla také úspěšně. Objevil jsem 7105 případů opravdu pozitivních, což znamená, že úspěšnost útoku byla 97 %. Simulátor Shadow umožňuje zobrazení výsledků simulace v grafech, což je vhodné pro znázornění funkčnosti sítě Tor. Po simulátoru Shadow jsem se zaměřil na TorPS, kde jsem provedl simulaci korelačního útoku. Výsledkem mé simulace bylo, že v průběhu pěti měsíců dojde s pravděpodobností 40 % k deanonymizaci uživatele. Při práci s TorPS jsem se potýkal s jistými problémy, které jsou spolu s jejich řešením vysvětleny v sekci 5.2.

Kapitola 8

Závěr

Cílem této práce bylo porovnání a využití nástrojů simulujících síť Tor, pomocí těchto nástrojů jsou demonstrovány principy fungování sítě Tor a útoky na ni.

V rámci bakalářské práce jsem se nejprve seznámil s architekturou a fungováním sítě Tor. Zaměřil jsem se na vytváření okruhů a jakým způsobem probíhá komunikace mezi uzly v okruhu. Další důležitou částí je bezpečnostní model sítě Tor, který se snaží zabránit tomu, aby byl uživatel deanonymizován. S tím souvisí útoky na síť Tor. V mé práci jsem se věnoval korelačnímu útoku, útoku Dropmark a odposlechu výstupního uzlu. Nejdůležitějšími nástroji se kterými pracuji, jsou simulátor Tor Path simulator a simulátor Shadow. Seznámil jsem se s fungováním simulátoru TorPS, který pracuje s modely útočníků a uživatelů, a nastudoval jsem fungování simulátoru Shadow a shadow plugin tor, který umožňuje simulaci sítě Tor. Oba simulátory jsem následně porovnal a ukázal jejich slabé a silné stránky.

Jedním z cílů práce bylo pomocí nástrojů demonstrovat, jak fungují některé principy sítě Tor a také, jak fungují útoky na síť Tor. Z tohoto důvodu jsem vytvořil webovou aplikaci, která využívá simulátor TorPS. Aplikace dle parametrů zadaných uživatelem simuluje několik případů užití: vytvoření okruhu v síti Tor, komunikace uživatele a skryté služby, demonstrace principu fungování korelačního útoku a odposlech výstupního uzlu útočníkem. Aplikace ke každému případu užití generuje tabulky a interaktivní grafy. V rámci práce vznikly výukové materiály využívající vytvořenou aplikaci. Mimo vytvořenou aplikaci jsem provedl také několik simulací, prostřednictvím kterých ukazuji nejen fungování simulátorů TorPS a Shadow, ale i samotné sítě Tor. V kapitole simulace a testování představuji výsledky těchto simulací.

Aplikaci by bylo možné vylepšit přidáním databáze, která by umožňovala uživatelům ukládat simulace. Protože vytvořená aplikace využívá aplikaci Docker, nebylo by přidání dalších kontejnerů složité. Do aplikace by šlo také přidat další případy užití.

Literatura

[1] *Megamind: Tor Dark Web* [online]. 2020 [cit. 2020-27-05]. Dostupné z: https://megamindsin.blogspot.com/2016/06/tor-dark-web.html.

[2] *Tor Project: FAQ* [online]. 2020 [cit. 2020-27-05]. Dostupné z: https://2019.www.torproject.org/docs/faq.

[3] *Tor Project: Onion services* [online]. 2020 [cit. 2020-27-05]. Dostupné z: https://2019.www.torproject.org/docs/onion-services.

[4] *Tor Project: Who uses Tor* [online]. 2020 [cit. 2020-27-05]. Dostupné z: https://2019.www.torproject.org/about/torusers.

[5] *Zdnet: How DNS can be used to unmask Tor users* [online]. 2020 [cit. 2020-27-05]. Dostupné z: https://www.zdnet.com/article/how-dns-can-be-used-to-unmask-tor-users/.

[6] Biryukov, A., Pustogarov, I., Thill, F. a Weinmann, R.-P. Content and popularity analysis of Tor hidden services. In: IEEE. *2014 IEEE 34th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. 2014, s. 188–193.

[7] Coufal, Z. *Korelace dat na vstupu a výstupu sítě Tor*. Brno, CZ, 2014. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/15819/.

[8] Dingledine, R., Mathewson, N., Murdoch, S. a Syverson, P. Tor: The Second-Generation Onion Router (2014 DRAFT v1). *Cl. Cam. Ac. Uk.* 2014.

[9] Dingledine, R., Mathewson, N. a Syverson, P. *Tor: The second-generation onion router*. Naval Research Lab Washington DC, 2004.

[10] Jansen, R. a Hopper, N. Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. 2011. Dostupné z: https://www.nrl.navy.mil/itd/chacs/sites/www.nrl.navy.mil.itd.chacs/files/pdfs/11-1226-3413.pdf.

[11] Jansen, R., Traudt, M., Geddes, J., Wacek, C., Sherr, M. et al. KIST: Kernel-Informed Socket Transport for Tor. *ACM Transactions on Privacy and Security (TOPS)*. ACM New York, NY, USA. 2018, sv. 22, č. 1, s. 1–37.

[12] Johnson, A., Wacek, C., Jansen, R., Sherr, M. a Syverson, P. Users get routed: Traffic correlation on Tor by realistic adversaries. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 2013, s. 337–348.

[13] Murdoch, S. J. a Danezis, G. Low-cost traffic analysis of Tor. In: IEEE. *2005 IEEE Symposium on Security and Privacy (S&P'05)*. 2005, s. 183–195.

[14] Neal, H. *Onion diagram* [online]. 2020 [cit. 2020-27-05]. Dostupné z: https://en.wikipedia.org/wiki/File:Onion_diagram.svg.

[15] Neal, H. *Shadow design* [online]. 2020 [cit. 2020-27-05]. Dostupné z: https://github.com/shadow/shadow/blob/master/docs/0-Design-Overview.md.

[16] Neal, H. *Support Torproject* [online]. 2020 [cit. 2020-27-05]. Dostupné z: https://support.torproject.org/tbb/tbb-2/.

[17] Polčák, L. *Základní informace o síti Tor*. 2017. 18 s. Dostupné z: https://www.fit.vut.cz/research/publication/11513.

[18] Rochet, F. a Pereira, O. Dropping on the edge: Flexibility and traffic confirmation in onion routing protocols. *Proceedings on Privacy Enhancing Technologies*. Sciendo. 2018, sv. 2018, č. 2, s. 27–46.

[19] Roger Dingledine, N. M. *Tor Protocol Specification* [online]. 2020 [cit. 2020-27-05]. Dostupné z: https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt.

[20] Shirazi, F., Goehring, M. a Diaz, C. Tor experimentation tools. In: IEEE. *2015 IEEE Security and Privacy Workshops*. 2015, s. 206–213.

[21] Syverson, P., Tsudik, G., Reed, M. a Landwehr, C. Towards an analysis of onion routing security. In: Springer. *Designing Privacy Enhancing Technologies*. 2001, s. 96–114.

Příloha A

Záznamy ze souborů server deskriptor a konsenzus

Příklad jednoho záznamu v souborech server desktiptor a konsenzus, viz sekce 5.1.

A.1 Server deskriptor

```
@type server-descriptor 1.0
router Unnamed370408321458 61.3.198.214 32867 0 0
published 2019-03-04 13:37:39
bandwidth 364192030 411695338 316688721
reject *:*
onion-key
-----BEGIN RSA PUBLIC KEY-----
9GRrIlWT6mTAYZahaFrmx6+mE/bkaFAplGDPVsNdUzwCWdNmZ3oM7tVJgPCJDLYB
R/UqLSNnPGXLwMb/wp5ti5cmMEStDdE4ssep5JPNqhaoAU8UUEtpg6jtwdUHeEyK
X8+CRYv2HxzGci1rVqSepoSPGT8zYekc0A3dLsSL0LgP28SOr8xYTX1nwGE=
-----END RSA PUBLIC KEY-----
signing-key
-----BEGIN RSA PUBLIC KEY-----
MIGJAoGBALr8A7z8eNG3QV7jSu0atd9nmT3BBy0tbsXb7rNp3g/mSUGJBWg0BWWp
2zEH4zDXp1k6xKQfNgXJDxDjCSExy/VU7zywcU888tLmB1Klk4f5fiF0i9U0VPB3
NTCFyOsA0tUyvj3YJ3LZp04J63mD7VsJATZ99kG4vNEY5apXRhi5AgMBAAE=
-----END RSA PUBLIC KEY-----
protocols Link 1 2 Circuit 1
platform Tor 0.2.4.8 on Linux
ntor-onion-key Yic4X19CcFx4ZmRceDg5XHgxND5ceGJjXHg5NFx4YjBWXHgxO
TBhXHgxZlx4N2ZceDk2N1x4MGNwXHgwMGZceGJhXHhlMz5ceGU1XHhlY1x4YzBce
GUzLSc
fingerprint 3E15 B5BF B1BA 3E66 9071 FFA0 2B44 9C6F 11D5 2026
router-signature
-----BEGIN SIGNATURE-----
YI0dwHueIovzu4rN2i/nZRB3f29W/PeaavnLmCR9ff1hY3wwqWZkNrwBpA85E6Om
MKO7Fc2SyzbxoScwv0+EFbvynu0gAdT3q5praQumZztAyWP7c7Cla7T5RhOcG9iX
9NcpvfG7I7QZsFGK1FlHR2TYxhYfrTfaY4hmW8u+vbQ=
```

```
-----END SIGNATURE-----
```

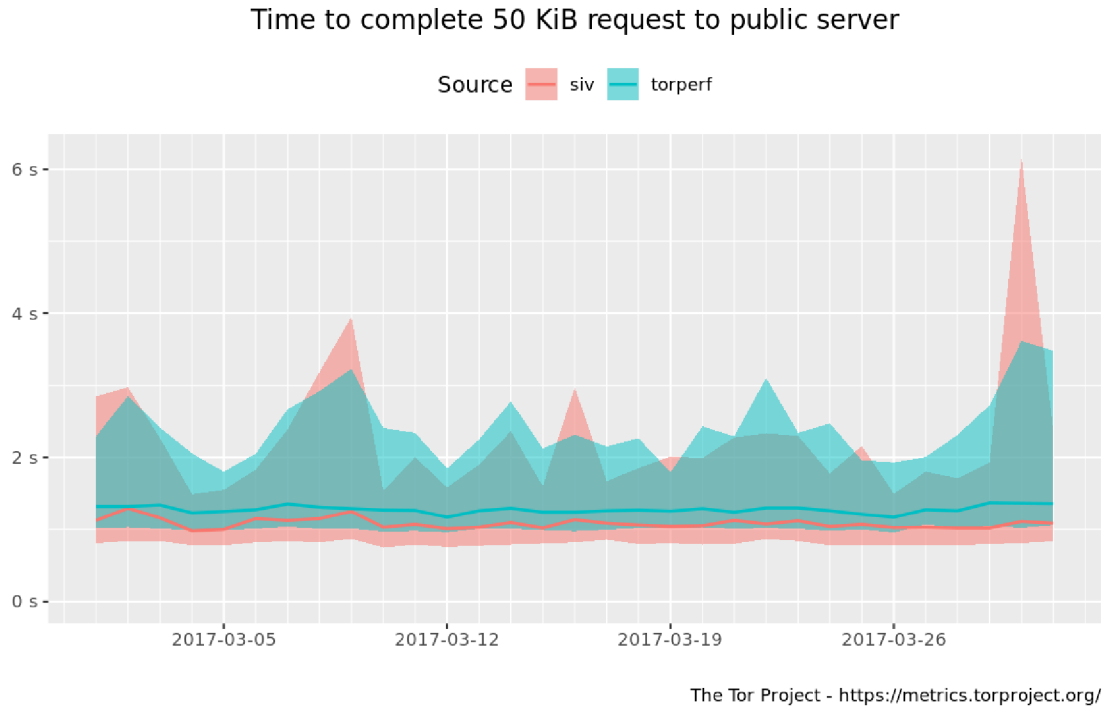
A.2 Konsenzus

```
r Unnamed370408321458 PhW1v7G6PmaQcf+gK0ScbxHVICY Wn4zQqxfJvDhPa
IcK+6pbzSvvvM 2019-03-04 13:37:39 61.3.198.214 32867 0
s Fast Guard Running Stable Valid
v Tor 0.2.4.8
w Bandwidth=364192030
p reject 1-65535
```

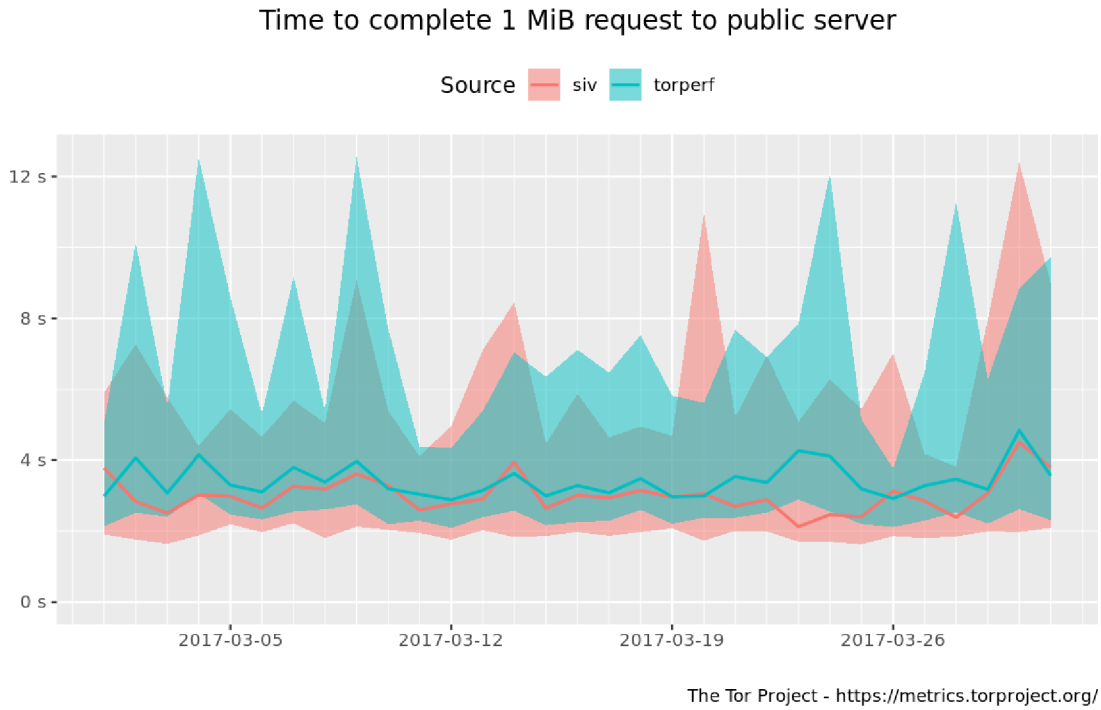
Příloha B

Výkon sítě Tor

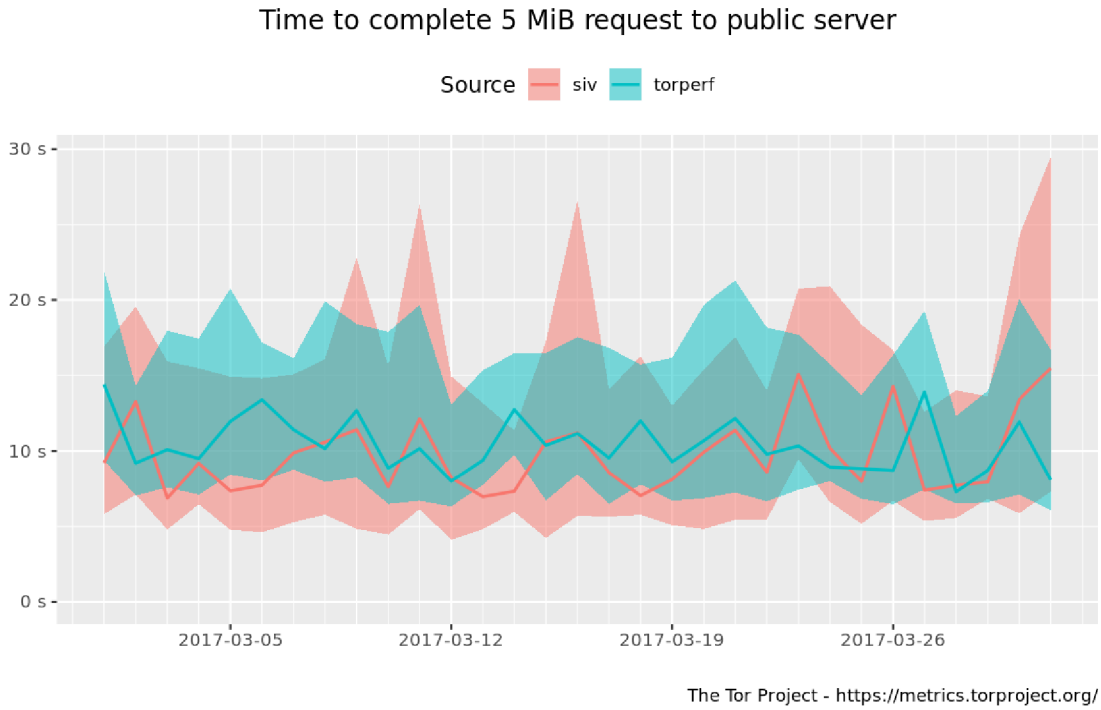
Výkon sítě Tor v březnu 2017, pro porovnání vlivu plánovačů na síť Tor, viz sekce 6.3.



Obrázek B.1: Čas ke stahování souboru velikosti 50 KiB březen 2017. Převzato z http://metrics.torproject.org



Obrázek B.2: Čas ke stahování souboru velikosti 1 MiB březen 2017. Převzato z http://metrics.torproject.org



Obrázek B.3: Čas ke stahování souboru velikosti 5 MiB březen 2017. Převzato z http://metrics.torproject.org

Příloha C

Simulace odposlechu uzlu

Tabulky k simulaci odposlechu výstupního uzlu, viz sekce 6.7.

C.1 Tabulky ukradených ID

IP	KB/s	ID's	Stolen	Stolen %
10.1.0.0	2400	238	238	100
10.2.0.0	2400	233	233	100

(a) Počet ukradených ID při dvou uzlech a šířce pásma 2400 KB/s

IP	KB/s	ID's	Stolen	Stolen %
10.2.0.0	1200	114	114	100
10.1.0.0	1200	123	123	100
10.3.0.0	1200	123	123	100
10.4.0.0	1200	125	125	100

(b) Počet ukradených ID při čtyřech uzlech a šířce pásma 1200 KB/s

Obrázek C.1: Počet odcizených ID při dvou a čtyřech nepřátelských uzelch.

IP	KB/s	ID's	Stolen	Stolen %
10.6.0.0	800	81	81	100
10.5.0.0	800	79	79	100
10.4.0.0	800	76	76	100
10.2.0.0	800	79	79	100
10.1.0.0	800	86	86	100
10.3.0.0	800	75	75	100

(a) Počet ukradených ID při šesti uzlech a šířce pásma 800 KB/s

IP	KB/s	ID's	Stolen	Stolen %
10.5.0.0	600	39	39	100
10.2.0.0	600	46	46	100
10.7.0.0	600	62	62	100
10.6.0.0	600	52	52	100
10.1.0.0	600	58	58	100
10.4.0.0	600	57	57	100
10.8.0.0	600	65	65	100
10.3.0.0	600	55	55	100

(b) Počet ukradených ID při osmi uzlech a šířce pásma 600 KB/s

Obrázek C.2: Počet odcizených ID při šesti a osmi nepřátelských uzelch.