VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

TESTOVÁNÍ SOND PRO MONITOROVÁNÍ SÍŤOVÉHO PROVOZU

TESTING OF PROBES FOR NETWORK TRAFFIC MONITORING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. JAN SOBOL**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. JAN KOŘENEK, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2021/2022

Zadání diplomové práce

Student: **Sobol Jan, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Počítačové vidění
Název: **Testování sond pro monitorování síťového provozu**
Testing of Probes for Network Traffic Monitoring
Kategorie: Počítačová architektura
Zadání:

1. Seznamte se s charakteristickými vlastnostmi sond pro monitorování síťového provozu na bázi specializovaných síťových sond (sonda IPFIX probe, sonda FlexProbe).
2. Navrhněte vhodný způsob automatického testování funkčnosti a výkonových parametrů sond. Zaměřte se primárně na dostupné sondy IPFIXProbe a FlexProbe.
3. Navržený systém automatického testování implementujte.
4. Ověřte správnou funkci a parametry vytvořené implementace.
5. Diskutujte dosažené výsledky a navrhněte další rozšíření.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 2 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Kořenek Jan, doc. Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 18. května 2022
Datum schválení: 29. října 2021

Abstrakt

Pro účel zajištění bezpečného a stabilního internetu potřebují administrátoři nástroje k monitorování sítě, které jim umožní analyzovat probíhající provoz a reagovat včas na nastalé situace. Jedním z prostředků, jak provoz monitorovat je nasazení síťových sond. Tato práce se zaměřuje na důkladné ověření parametrů existujících sond IPFIX probe a FlexProbe. FlexProbe je sonda určená k realizaci zákonných odposlechů vyvíjená na FIT VUT v Brně ve spolupráci s Policií ČR. Sondu IPFIX probe vyvíjí sdružení CESNET a v rámci sondy FlexProbe se využívá k monitorování síťových toků. Aby bylo možné sondy dlouhodobě provozovat v cílovém prostředí, je nezbytné zařízení důkladně otestovat. Přesné chování sondy bylo definováno specifikací požadavků, které jsou vypracovány pro obě sondy. Na základě požadavků byl navržen ucelený systém testů pokrývající funkční i výkonnostní aspekty sond. Testy jsou sjednoceny pomocí testovacího frameworku a zařazeny do automatizovaných scénářů implementovaných v aplikaci Jenkins. V závěru práce je vyhodnoceno pokrytí požadovaných vlastností sond a jejich výkon.

Abstract

In order to ensure a secure and stable Internet, administrators need tools for network monitoring which will allow them to analyze ongoing network traffic and respond to situations in a timely manner. One way to monitor traffic is to use monitoring probes. This thesis focuses on a thorough verification of the parameters of existing probes IPFIX probe and FlexProbe. FlexProbe is a network probe designed for the implementation of lawful interceptions developed at FIT BUT in cooperation with the Police of the Czech Republic. The IPFIX probe is developed by the CESNET association and is used for flow monitoring within the FlexProbe probe. In order to be able to operate the probes in the target environment for a long time, it is necessary to thoroughly test the device. The exact behavior of the probe is defined by the specification requirements that are developed for both probes. Based on the requirements, a comprehensive test system covering functional and performance parameters of the probes was designed. The tests are unified using a test framework and included in automated scenarios implemented in system Jenkins. At the end of the thesis, the coverage of the required properties of the probes and their performance is evaluated.

Klíčová slova

monitorování síťového provozu, monitorování síťových toků, síťový tok, záchyt zájmového provozu, zákonné odposlechy, ETSI, NetFlow, IPFIX, FlexProbe, IPFIXprobe, specifikace požadavků, testovací scénář, pokrytí testy, SNMP, Ping, Traceroute

Keywords

traffic monitoring, flow monitoring, flow export, packet capture, lawful interceptions, ETSI, NetFlow, IPFIX, FlexProbe, IPFIXprobe, test requirements, test case, test coverage, Ping, Traceroute, SNMP

Citace

SOBOL, Jan. *Testování sond pro monitorování síťového provozu.* Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Jan Kořenek, Ph.D.

Testování sond pro monitorování síťového provozu

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně pod vedením doc. Ing. Jana Kořenka, Ph.D. Další informace mi poskytli řešitelé projektu flexibilní sondy pro realizaci zákonných odposlechů. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jan Sobol
24. května 2022

Poděkování

Rád bych poděkoval svému vedoucímu, doc. Ing. Janu Kořenkovi, Ph.D., za vedení práce a pomoc při tvorbě textové zprávy. Nemalý dík patří všem členům projektu FlexProbe, kteří mě do problematiky uvedli a poskytli odborné vysvětlení funkce i testů existující sondy FlexProbe.

Obsah

Kapitola 1

Úvod

Na poli informačních a komunikačních technologií lze sledovat postupný nárůst významu internetu. Každým rokem se k síti připojuje větší množství zařízení, internet je díky bezdrátovým technologiím všude kolem nás. Infrastruktura se stává složitější, přenáší se více dat. Vývoj síťových technologií přináší robustnější protokoly, které ale kladou vyšší nároky na síť. Proto je nutné uživatelům nabídnout stabilní a dostatečně bezpečný internet. Aby tohoto cíle bylo dosaženo, používají administrátoři nástroje k monitorování síťového provozu, které jim umožňují sledovat výpadky tak, aby na ně mohli včas reagovat a nastalou situaci analyzovat. Statistická analýza provozu přináší cenné informace o druhu přenášené komunikace, čehož se využívá při posilování infrastruktury nebo pro detekci a následné zablokování síťových útoků (DoS). Analýzou obsahu lze zachytit komunikaci osob podezřelých z trestného činu, přičemž se nejedná pouze o samotné kybernetické hrozby. Internet také může sloužit jako nezanedbatelný komunikační kanál pro pachatele trestných činů.

Monitorování síťového provozu se dělí na aktivní a pasivní. Aktivní nástroje navazují spojení se síťovými prvky a zjišťují jejich stav. Pasivní do provozu nijak nezasahují, ale pouze získávají informace o probíhající komunikaci. Poměrně rozšířené je k pasivnímu monitoringu použít specializované zařízení – síťovou sondu, která disponuje softwarem optimalizovaným k monitorování síťového provozu. Pro dosažení požadované výkonnosti je často potřeba využít akceleraci pomocí vhodné hardwarové platformy, na kterou jsou delegovány časově náročné procesy zpracování síťového provozu. Hardwarová akcelerace je prakticky nutností při nasazení sond do prostředí vysokorychlostních sítí. Páteřní linky internetu dosahují rychlosti ve stovkách gigabitů za sekundu, stejně tak přípojky koncových uživatelů se neustále zrychlují.

V rámci řešené práce se primárně zaměříme na monitorování síťových toků a záchyt zájmové komunikace. Síťový tok vyjadřuje spojení mezi dvěma účastníky (aplikacemi) přes síť. Monitorováním síťových toků se zabývá sdružení CESNET, které vyvíjí volně dostupný softwarový nástroj pro sledování a export síťových toků, sondu IPFIX. Tento nástroj lze zabudovat do monitorovacích zařízení a při využití vhodných akceleračních karet i urychlit. Ve výchozím režimu IPFIX probe pracuje na síťové a transportní vrstvě. Aktivací rozšiřujících modulů lze statistiku obohatit o položky aplikační vrstvy, např. DNS, HTTP, SMTP a další.

Záchyt zájmového provozu se provádí zejména pro analýzu detekovaných bezpečnostních incidentů, ale také při realizaci zákonného odposlechu komunikace podezřelých osob. Bezpečnostní složky provádějící zákonný odposlech potřebují prostředky, pomocí kterých získají kvalitní data použitelná jako důkazní materiál. Je nutné zachytit síťový provoz s minimem ztrát a o rozsahu ztrát poskytnout přesné informace. Odposlech musí být tajný

a izolovaný, aby nedocházelo k záchytu dat osob, které nejsou podezřelé. Současně musí být zachycená komunikace bezpečně uložena.

Projekt FlexProbe [9] si bere za cíl vyvinout a uvést do rutinního provozu sondu, která umožní záchyt komunikace na základě IP adres, ale také L7 identifikátorů aplikačních protokolů (např. e-mailová adresa). V rámci projektu byla navržena vlastní hardwarová platforma, která akceleruje záchyt a předává data do softwaru optimalizovaného pro tuto platformu. Sonda dále poskytuje statistiku o provozu na připojené lince, která je vytvořena prostřednictvím monitorování síťových toků. Orgány činné v trestním řízení mohou využít informace o zastoupení a struktuře protokolů, variabilitě jejich zapouzdření, rozpoznání šifrovaného provozu a podobně.

K zajištění spolehlivého fungování síťových sond v produkčním prostředí je nutné zařízení důkladně otestovat a odhalit tak jeho chyby a limity ještě před nasazením. Jedná se o sofistikovaný systém složený z mnoha modulů, které spolu kooperují. V případě vysokorychlostní sondy je nutné zajistit součinnost komponent implementovaných v hardware a softwarových subsystémů, které hardware ovládají. Je tak potřeba zaměřit se nejen na testy funkční, ale i na důkladné ověření požadovaného výkonu. Sonda musí splňovat požadavky, které jsou definovány implementovaným protokolem – IPFIX. V případě specializované sondy FlexProbe udává požadavky Policie České republiky, která je pověřena prováděním zákonných odposlechů. Nesplnění požadavků je pro funkci sondy fatální. Projevit se může neúspěšným zachycením síťových hrozeb a výpadků nebo ztrátou dat při zákonném odposlechu, která znehodnotí důkaz trestné činnosti.

Cílem této práce je navrhnout systém testů, kterým bude možné ověřit korektní chování a výkon sond pro monitorování síťového provozu. Součástí systému je prostředí pro testování, automatizace testů a testovací framework, který jednotlivé testy sjednocuje. Testy jsou navrženy tak, aby byly deterministické, automatizovatelné a opakovatelné. Testovací scénáře jsou odvozeny z definovaných požadavků na sondu a mají za cíl všechny požadavky ověřit. Výsledkem práce je vyhodnocení pokrytí specifikovaných požadavků testy, které byly implementovány a automatizovány. Vyhodnocení se zabývá také dobou běhu testovacích scénářů.

Dokument se nejprve věnuje teoretickému rozboru monitorování síťového provozu. Kapitola 2 pohlíží na metody monitorování sítě, které dělí na aktivní a pasivní. Následně se text zaměří na monitorování síťových toků a technologie NetFlow a IPFIX. Jako zástupce exportéru síťových toků je uveden software IPFIX probe, který je rozebrán z pohledu architektury, způsobu konfigurace i možnosti akcelerace. Kapitola 3 se věnuje zákonným odposlechům. Nejprve je popsána architektura systému pro zákonné odposlechy, následně sonda FlexProbe s touto funkcionalitou. Cílem kapitoly 4 je uvést čtenáře do teorie testování hardwarových sond a popsat testování počítačového systému na různých úrovních. V kapitole 5 jsou specifikovány požadavky na obě existující sondy IPFIX probe a FlexProbe. Výsledná specifikace je tvořena číselně označenými požadavky, které musí být ověřeny systémem testů navrženým v kapitole 6. Kromě samotné testovací sady se návrh zabývá také automatizací a testovacím prostředím, ve kterém jsou testy spouštěny. Kapitola 7 je věnována vyhodnocení kvality systému testů, kterou určuje dosažené pokrytí specifikace požadavků. Současně jsou v kapitole diskutovány nalezené chyby a naměřený výkon sondy FlexProbe.

Kapitola 2

Monitorování síťového provozu

Již několikátou dekádu lze sledovat souvislý a nepřestávající vývoj globální sítě – internetu. Nejedná se pouze o pokrok síťových technologií, ale také o růst důležitosti ve společnosti. Na dostupnosti internetového připojení je v dnešní době více či méně závislá většina společnosti při výkonu práce, studiu i ve volném čase. Na základě různých událostí dokonce povinnost připojení byla zahrnuta i do legislativy. Rozvoj infrastruktury, do kterého se řadí např. přechod na nové generace bezdrátových sítí s rychlejším datovým přenosem, umožňuje nárůst počtu služeb, složitosti síťových protokolů a komplexity dat přenášených internetem.

Stále větší integrace internetu do běžného života motivuje k vylepšování prostředků pro správu a monitorování sítě. Administrátor usiluje o omezení výpadků a co nejvyšší výkon. Ke své práci potřebuje nástroje, pomocí kterých získá povědomí o síťovém provozu i stavu síťových prvků. Uživatelům je nutné zajistit také bezpečnost. Je proto snahou, aby síťové útoky byly odhalovány ještě dříve, než napáchají škody. Statistiky vytvořené v rámci monitorování síťového provozu poskytují užitečné informace nejen z pohledu povědomí, co se v síti děje a jestli nedochází k nějaké nekalé činnosti, ale také při posilování nebo výstavbě nové infrastruktury.

Cílem této kapitoly je představit problematiku monitorování sítě. V kontextu síťových sond jsou relevantní zejména dva přístupy: sledování síťových toků a záchyt zájmového provozu pro následnou analýzu obsahu komunikace. Na tyto přístupy je v teoretickém rozboru kladen největší důraz.

2.1 Metody sledování sítě

Metody je možné rozdělit do dvou základních skupin: aktivní a pasivní [11]. Aktivní monitorování sítě spočívá ve zjišťování stavu, dostupných služeb a dalších vlastností odesíláním dotazů na sledovaná zařízení v síti. Monitorovací systém tedy aktivně navazuje komunikaci a interpretuje přijaté odpovědi. Při pasivním monitorování je sledována probíhající komunikace na síti. Ze zpráv, které se v provozu objevují se sbírají informace o stavu sítě a zařízeních na síti, informace o chybách či nedostupnosti zdrojů, případně pokusy o neoprávněný přístup k síťovým službám.

Základním nástrojem pro aktivní monitorování je program Ping[1], který generuje zprávy protokolu ICMP a čeká na odpověď. Pokud zařízení odpoví korespondující ICMP zprávou, je dostupné. V opačném případě tazatel přijme ICMP typu *Destination Unreachable* od

[1] https://linux.die.net/man/8/ping

jiného síťového zařízení (např. směrovače). Podle kódu zprávy[2] lze navíc určit podrobnosti o důvodu, proč nebylo možné přenést paket do cíle. Jiný nástroj využívající ICMP je Traceroute[3]. Ten umožňuje pomocí sledování cesty kontrolních paketů zkoumat topologii sítě. Pokročilý protokol, který umožňuje dlouhodobý monitoring téměř libovolných parametrů zařízení připojeného do sítě je SNMP[4]. Pravidelným dotazováním je kontrolována dostupnost a další parametry určené ke sledování, například množství volné paměti, teplota čipu nebo počet zahozených paketů na síťovém rozhraní.

Nevýhodou aktivního sledování může být ovlivňování/zasahování do síťového provozu. V době přetížené sítě nelze zajistit, aby pakety kontrolních protokolů přicházely beze ztrát. Pro získání libovolné informace musí být možné navázat spojení mezi zdrojovým a cílovým zařízením. Pokud tomu tak není, výsledky mohou být špatně interpretovány jako špatná funkce koncového zařízení. Výpadek přitom mohl způsobit kterýkoli síťový prvek po cestě, například směrovač, přepínač a podobně.

Zařízení sbírající informace je při pasivním monitorování pouze pozorovatelem a neposílá do sítě žádný provoz. Na rozdíl od aktivního monitorování síťové prvky nejsou dotazovány, ale samy odesílají informace o svém stavu (např. SNMP traps) nebo nashromážděné statistiky. Druhou možností pasivního monitorování je záchyt probíhajícího provozu bez účasti a vědomí ostatních síťových zařízení. Záchyt je možné realizovat na koncových uzlech nebo uvnitř infrastruktury sítě. Na koncových stanicích se používají softwarové nástroje, např. tcpdump nebo Wireshark. Pro monitoring ve vnitřní síti se využívají funkce síťových prvků. O monitorovací funkce bývají obohaceny směrovače i přepínače. Zde ale nastává problém v případě přetížení prvku, kdy musí zařízení zajistit hlavně svou primární funkci. Informace tak mohou být nepřesné a neúplné. K zajištění přesného monitorování síťového provozu se proto používají specializované sondy, do kterých je směrována kopie veškerého provozu na konkrétní lince.

Analýzou zachyceného síťového provozu lze získat informace sloužící k mnoha rozličným účelům. Od rekonstrukce komunikace mezi subjekty a následné podrobení obsahu kontrole legálnosti, přes účtování poskytnutých služeb až po vytvoření modelu provozu na dané lince, například z hlediska zastoupení síťových protokolů.

Záchyt celé komunikace na odposlouchávané lince, tzv. *packet capture*, umožňuje nejpodrobnější analýzu. Pro dnes běžné rychlosti páteřních sítí od desítek po stovky gigabitů za sekundu, je ale velmi náročné takto vytvořené záchyty ukládat i zpracovávat. Proto jsou podle účelu výsledné analýzy data ihned po záchytu předzpracována a ukládána ve formě agregovaných metadat. Příkladem agregace jsou sondy realizující monitorování síťových toků (technika zvaná *flow monitoring* a *flow export*). Agregované výstupy jsou například počet paketů, počet bytů, čas příchodu prvního a posledního paketu, atd. Další způsoby zpracování dat, stejně jako definice síťového toku, popis architektury a procesů probíhajících při pasivním monitorováním síťového provozu budou popsány v dalších kapitolách.

Nástroje pro aktivní a pasivní monitoring plní odlišné potřeby správy sítě. Aktivní monitoring ověřuje vnitřní stav prvků zapojených do sítě a je schopen reagovat (varovat, notifikovat) při pádu do nefunkčního stavu. Pasivní monitoring analyzuje provoz v síti a nemá informaci, jaká změna v síti vedla k vývoji provozu na lince. Kombinace obou skupin může napomoct řešení problémů, ke kterým došlo při provozu sítě. Použití ilustruje situace úspěšného DDOS[5] útoku. Nástroj využívající SNMP zaregistroval změnu stavu vy-

[2] https://datatracker.ietf.org/doc/html/rfc792
[3] https://linux.die.net/man/8/traceroute
[4] https://datatracker.ietf.org/doc/html/rfc1157
[5] Distributed Denial of Service, https://doi.org/10.1016/S1353-4858(08)70086-2

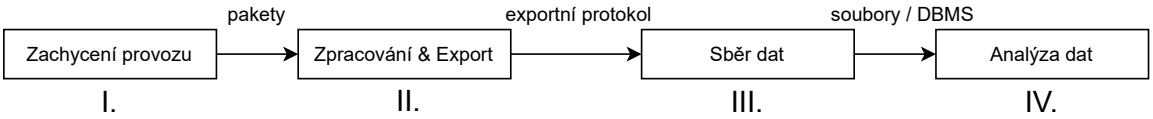
řazeného zařízení a vytvořil o tom záznam s aktuálním časem v logu. Díky zachycenému provozu s časovými značkami je možné analyzovat dění v síti před zahlcením síťového prvku a identifikovat, ze kterých IP adres byl útok veden.

2.2 Monitorování síťových toků

Monitorování síťových toků je pasivní forma sledování probíhajícího provozu. Jejím účelem je vytvořit ze sledované komunikace statistický model v podobě množiny záznamů o síťovém toku. Záznam reprezentuje jedno konkrétní spojení mezi dvěma aplikacemi komunikujícími srze síť, které proběhlo během monitoringu. K síťovému toku se sbírají informace, které vypovídají o některých vlastnostech komunikace, např. o délce spojení nebo použitých protokolech. Vybrány jsou vlastnosti podstatné pro analýzu sledovaného provozu, ostatní informace jsou zahozeny. Architektura popsaná v této sekci vychází z protokolu NetFlow[6] od firmy Cisco, který implementuje monitorování síťových toků. Díky své rozšířenosti se stal základem pro novější technologii IPFIX, kterou využívají sondy testované v rámci této práce.

Síťový tok je možné definovat jako spojení mezi bodem A a B, kde body A a B jsou aplikace umístěné na konkrétních uzlech sítě. Přesně je definován množinou atributů, nejčastěji pěticí zdrojová IP adresa, cílová IP adresa, zdrojový port, cílový port a číslo transportního protokolu. Všechny pakety náležící toku mají stejné hodnoty těchto atributů. Typicky se používají položky získatelné nebo odvozené ze síťových a transportních hlaviček paketu. Celý tok pak vyjadřuje úplnou síťovou konverzaci od jednoho účastníka k druhému. Například spojení protokolem TCP z pracovní stanice na webový server při čtení webové stránky. Navázání TCP spojení představuje začátek toku, uzavřením spojení dochází k jeho ukončení [10].

Často se také pracuje s obousměrným tokem (tzv. *Biflow* – *Bidirectional Flow*). Ten v sobě kombinuje dva toky, které tvoří konverzaci mezi dvojicí účastníků. V prvním toku vystupuje účastník A jako zdroj a v druhém jako cíl, opačné role zastává účastník B. NetFlow záznam pak obsahuje informace sesbírané k oběma tokům.



Obrázek 2.1: Stupně monitorování síťového provozu

Proces monitorování provozu lze rozdělit do několika fází (viz obrázek 2.1) [4]. Prvním stupněm je samotný záchyt paketů na příchozí lince. Místo v síti, kde k záchytu dochází je nazýváno pozorovacím bodem (anglicky *Observation Point*). Jako pozorovací bod často vystupuje rozhraní síťové karty nebo směrovače, kde jsou příchozí pakety analyzovány a předány dál. Od této operace se očekává přesnost záchytu a bezeztrátovost. S narůstající rychlostí vstupní linky je stále náročnější tyto podmínky splnit. Například pro rychlost 100 Gb/s je limit na zpracování jednoho paketu pouze 5 nanosekund. Proto se v praxi pro záchyt využívá hardwarová akcelerace. Jinou možností je již při záchytu provoz vzorkovat.

Vzorkování umožňuje efektivně snížit množství zpracovávaných dat. Při záchytu se ke zpracování vybere pouze 1 z N paketů a zbytek je zahozen. Tento přístup je možné použít,

[6] Cisco Systems NetFlow Services Export Version 9, https://datatracker.ietf.org/doc/html/rfc3954

pokud jsou data určená ke statistickému zpracování. Je ale potřeba počítat se zkreslením výsledků, případně mohou některé informace úplně chybět. Druhou používanou operací je filtrace. Při správné definici podmínek nedochází ke ztrátě zájmových dat. Navíc bývá filtrace akcelerovaná hardwarově, kdy software zpracovává jenom část provozu vybranou prostřednictvím filtrace.

Ve druhém stupni jsou data zpracována a výsledky exportovány. Zpracováním se myslí zejména agregace, kvůli které musí být rozpoznáno zařazení paketu do síťového toku. Následně jsou v záznamu příslušného toku aktualizována metadata (sbírané statistiky, např. počet paketů a počet bytů). Pomocí exportního protokolu jsou výsledky zpracování odeslány. Odeslání záznamu o síťovém toku proběhne typicky ve chvíli, kdy je tok považován za ukončený. Podmínky označení toku za ukončený jsou popsány v kapitole 2.3.

Další fází je sběr exportovaných dat. Sběrné místo má za úkol cyklicky přijímat od exportních procesů data, třídit je a ukládat na perzistentní úložiště pro následnou analýzu. Výběr způsobu perzistentního uložení ovlivňují dva parametry: rychlost ukládání a rychlost vyhledávání. Vysokou rychlost zápisu umožňuje uložení do běžných souborů tak, jak byly záznamy přijaty. Soubory na disku mohou být textové, většinou se ale používají binární, svázané s konkrétním analytickým nástrojem (např. formát nfcapd pro NfSen). Přístup optimalizovaný pro vyhledávání je ukládání do sloupcových nebo řádkových databází. Mezi rychlostí zápisu a vyhledávání je nutné najít kompromis v závislosti na případu užití.

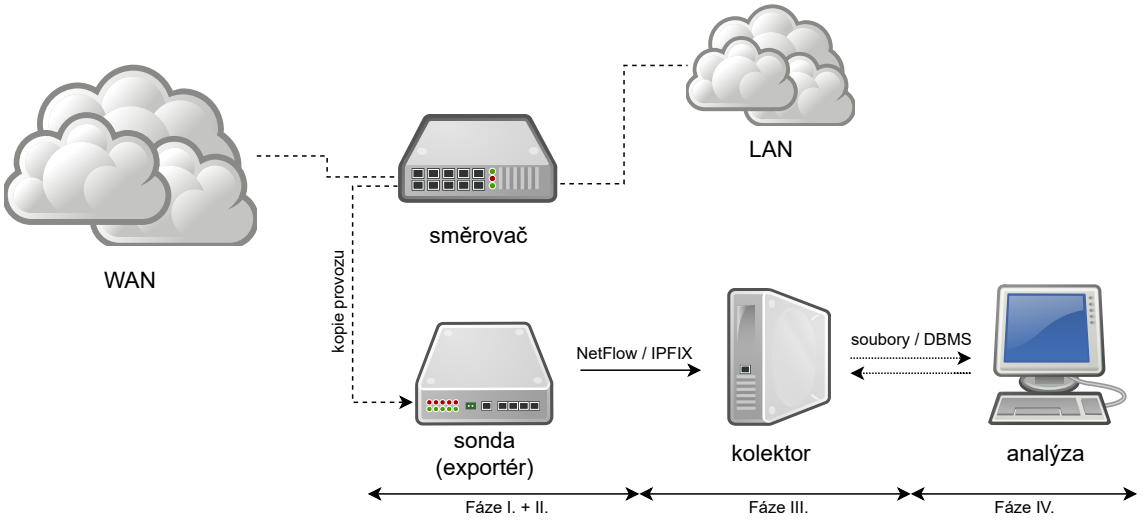
Netriviální dělá z úlohy sběru dat skutečnost, že exportních procesů bývá obvykle více. Současně se může stát, že každý exportní proces používá jiný protokol (např. jinou verzi). Záznamy ze sond mohou být ještě před uložením modifikovány. Pro příklad mohou být vyřazeny redundantní informace v případech, kdy byl tok exportován z více monitorovacích bodů. V rámci modifikace mohou být některé informace přidány. Příkladem může být čas přijetí dat. Dále mohou být sloučeny záznamy, které k sobě patří, nicméně musely být z důvodu omezené kapacity paměti sond rozděleny a exportovány zvlášť (důvody rozdělení jsou diskutovány v 2.3).

Poslední etapou monitorování síťového provozu je analýza dat. Může být automatická: detekce anomálií, vytvoření profilu provozu a jeho klasifikace. Často je analýza prováděna samotným uživatelem. Ten vytváří dotazy nad úložištěm buď přímo, nebo prostřednictvím nástrojů s možností vizualizace, například NfSen[7]. Typickými dotazy jsou další úrovně agregace, filtrace podle času nebo druhu provozu, atd. Ve fázi analýzy jsou z dříve exportovaných dat získávány znalosti.

Obrázek 2.2 zobrazuje systém zařízení, které provádějí monitorování síťového provozu na síťové lince. V tomto případě je monitorován provoz procházející přípojným místem lokální sítě do internetu. V praxi bývají fáze zachycení provozu, zpracování a exportu integrovány do jednoho zařízení, které se nazývá exportér. Jak již bylo zmíněno v předchozí kapitole, pokud je exportér dedikované zařízení, hovoříme o síťové sondě.

Na obrázku je exportér připojen na tzv. SPAN port síťového prvku. SPAN je port, speciálně nakonfigurovaný tak, aby kopíroval veškerý provoz procházející jiným portem. Jeho použití typicky umožňují přepínače nebo směrovače. Druhou možností je přímé napojení na monitorovací linku pomocí zařízení TAP, které umožňuje efektivně odbočit síťový provoz. Použití dedikovaného zařízení je vhodné pro zajištění kvalitních dat. V době přetížení síťového prvku totiž mohou být data na SPAN portu omezena na hlavičky paketů nebo jsou pakety zahazovány.

[7] http://nfsen.sourceforge.net



Obrázek 2.2: Architektura monitorování síťového provozu

Místo sběru dat se označuje jako kolektor. Mezi exportérem a kolektorem jsou data přenášena pomocí exportního protokolu, který je definován jako součást architektury NetFlow nebo IPFIX. S kolektorem komunikuje také uživatel, který má k dispozici nástroje pro tvorbu přehledů a vizualizací z uložených dat.

2.3 Architektura NetFlow/IPFIX sondy

Následující text se věnuje podrobněji jednomu druhu sledování síťového provozu – tzv. *flow export*, který probíhá na síťových sondách, exportérech. Architekturu definují protokoly NetFlow a IPFIX tak, jak je popsaná v předchozí sekci. Tato sekce je zaměřena na konkrétní problémy při záchytu a následné agregaci informací o síťových tocích.

Aby mohlo po zachycení provozu dojít k agregaci, musí být z příchozích paketů extrahovány sledované vlastnosti. Ty je možné rozdělit na dvě skupiny: atributy identifikující síťový tok a statistické informace, které se k danému toku agregují. První skupina kopíruje použitou definici toku, například pětici zdrojová IP adresa, cílová IP adresa, zdrojový port, cílový port a číslo transportního protokolu. V tabulce 2.1 je seznam základních atributů, které tvoří záznam toku v architekturách NetFlow a IPFIX. Různé implementace mohou skupinu atributů rozšiřovat, zejména agregovanou část za účelem získání většího množství informací k analýze.

Měření (agregace) toků je dlouhodobá záležitost už z toho důvodu, že pakety jednotlivých toků přicházejí postupně. Proto je potřeba, aby zařízení provádějící měření uchovávalo aktivní toky v paměti po celou dobu existence. Anglicky se takto specializovaná paměť nazývá *flow cache* a bývá implementována jako hashovací tabulka indexovaná právě unikátní kombinací atributů identifikující síťový tok [4]. Při přijetí paketu je v tabulce vyhledán záznam se stejným klíčem. Pokud je nalezen, jsou aktualizovány neklíčové atributy (hodnota záznamu). Jinak je v tabulce založen nový záznam, což znamená, že se v provozu objevil nový tok.

Jako každý zdroj, ani paměť toků není nevyčerpatelná a má určitou velikost. Agregované záznamy proto musí být včas exportovány na kolektor, aby nedošlo k jejímu zaplnění.

Záznam je exportován ve chvíli, kdy je tok s daným identifikátorem označen jako ukončený. K tomu dojde při splnění některé z následujících podmínek [4] [10]:

Detekce ukončení toku je nejpřirozenějším signálem pro export toku, je však možná pouze u spojované komunikace. Typicky se využívá transportní protokol TCP, u kterého je konec spojení indikován příjmem paketu s příznakem FIN. Při využití jiných protokolů bez nástroje umožňujícího aktivní ukončení komunikace je nutné nastavit odlišné podmínky.

Aktivní časový limit vyvolá po jeho uplynutí násilné ukončení stále probíhajícího toku. Limit slouží k zajištění exportu toků, které trvají příliš dlouhou dobu. Např. stahování velkého souboru z cloudového úložiště. Kdyby limit neexistoval, k analýze by se informace o toku dostaly až po jeho ukončení, což může trvat v řádu hodin i několika dnů. Bylo by tak nereálné včas reagovat na aktivitu takto komunikujících účastníků. Na exportérech od společnosti Cisco je ve výchozím nastavení aktivní časový limit 30 minut.

Neaktivní časový limit zajišťuje ukončení toku, od kterého po stanovenou dobu nepřišly další pakety. Limit má velký význam při nespojované komunikaci (např. protokol UDP), kdy žádný z účastníku aktivně neukončuje komunikaci. V případě výpadku jedné strany komunikace díky neaktivnímu limitu nezůstávají v paměti nadále nepoužívané toky. Exportér Cisco tok uzavírá po 15 sekundách od posledního paketu.

Provozní důvody se většinou týkají neobvyklých situací jako je přeplnění paměti nebo vypláchnutí celé *flow cache* z důvodu synchronizace času napříč síťovými prvky. Exportér také může spouštět proces, který spravuje paměť a udržuje její část volnou pro nově příchozí toky.

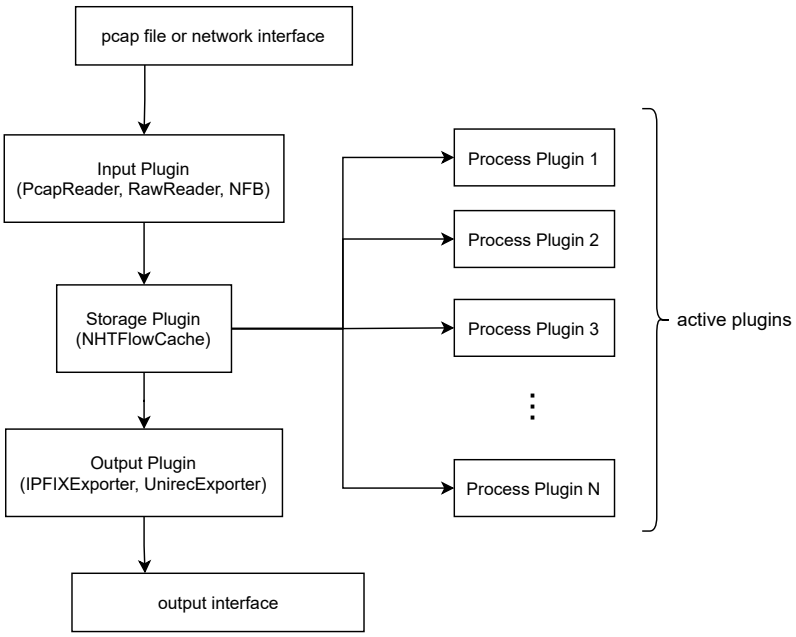
Atribut	Identifikátor IPFIX
Zdrojová IP adresa	sourceIPv4(6)Address
Cílová IP adresa	destinationIPv4(6)Address
Zdrojový port	sourceTransportPort
Cílový port	destinationTransportPort
Číslo IP protokolu	protocolIdentifier
Časová značka prvního paketu	*flowStartMilliseconds*
Časová značka posledního paketu	*flowEndMilliseconds*
Počet paketů	*packetDeltaCount*
Počet bytů	*octetDeltaCount*

Tabulka 2.1: Základní prvky, které tvoří záznam toku v paměti. Agregované vlastnosti jsou kurzívou [4].

Po ukončení, exportu a odebrání záznamu z paměti budou příchozí pakety s totožným klíčem označeny za nový tok – vznikne nový záznam. Tato klasifikace může být správná: předchozí tok byl řádně ukončen a po nějakém časovém intervalu začali účastníci znovu komunikovat se stejnými identifikátory (IP, porty). Mohlo ale dojít k jiné podmínce označení toku za ukončený a jedná se proto o původní tok. Situaci musí rozpoznat kolektor a v druhém případě záznamy sloučit do jednoho.

2.4 IPFIX probe

IPFIX probe[8] je řešení flow exporteru od sdružení CESNET[9]. V základu je to softwarový nástroj s otevřenými zdrojovými kódy v jazyce C++, který nabízí funkcionalitu vytváření záznamů typu Biflow ze vstupních paketů. Další činnost je rozšiřována pomocí zásuvných modulů (tzv. plugin), v jejichž podobě je také distribuována implementace standardních funkcí.



Obrázek 2.3: Schéma architektury softwaru IPFIXprobe

Na obrázku 2.3 se nachází architektura IPFIX probe. Komponenty i prováděné procesy odpovídají teoretickému popisu v 2.3. Ze schematu je patrné, že všechny součásti systému jsou implementované pomocí zásuvných modulů. Z pohledu objektového návrhu se jedná o vrstvu abstrakce, díky které je možné rozšiřovat funkcionalitu nebo software přizpůsobit pro ad-hoc začlenění do jiného systému (úprava vstupů a výstupů).

Po přijetí paketu ze vstupního rozhraní, případně PCAP souboru, jsou extrahovány klíčové vlastnosti toku (viz 2.1), paket je následně zařazen do flow cache. Záznam typu Biflow je buď aktualizován, nebo vytvořen nový pro neznámé vlastnosti toku. Při tom jsou volány obslužné rutiny aktivovaných procesních pluginů. Úkolem procesního pluginu je extrakce a agregace sledovaných vlastností síťových protokolů. Obslužné rutiny jsou různé pro aktualizaci a vytvoření nového toku v paměti. Speciální obslužná rutina je pro každý plugin zavolána ve chvíli exportu záznamu (tok byl označen za ukončený – viz 2.3). Výstupní zásuvný modul udržuje spojení s kolektorem a odesílá exportované toky prostřednictvím výstupního rozhraní.

Zásuvné moduly ovlivňují způsob příjmu vstupního provozu (např. PCAP soubory, soket operačního systému), formát exportu dat (IPFIX a/nebo UniRec[10]) a informace exportované z toku. Standardní zásuvné moduly nabízí parsery a agregátory aplikačních protokolů,

[8] https://github.com/CESNET/ipfixprobe
[9] CESNET, zájmové sdružení právnických osob, https://www.cesnet.cz
[10] záznam pro detekční systém NEMEA, https://nemea.liberouter.org

např. DNS, HTTP, SIP, NTP, SMTP. Samozřejmostí je možnost programování uživatelských zásuvných modulů, proces tvorby je zjednodušen vytvořením kostry pluginu pomocí skriptu (`create_plugin.sh`).

Software není závislý na hostujícím operačním systému, ani hardwarové platformě. To umožňuje univerzální využití v různorodých prostředích, např. anotování dat na pracovní stanici nebo integrace IPFIX probe do aktivního síťového prvku. Statistika může být vytvořena „offline“ z dříve zachyceného provozu v PCAP souboru nebo lze zpracovávat provoz na vstupním rozhraní v reálném čase. Ve výchozím režimu se používá soket operačního systému, který předává nezpracované pakety (tzv. *raw*). Hostující prostředí tak může být stanice nebo server s běžnou síťovou kartou. Alternativou nezbytnou pro monitorování vysokorychlostních linek je využití specializovaných rozšiřujících karet z rodiny NFB (*Network FPGA Boards*). S hardwarovou kartou software komunikuje prostřednictvím rozhraní NDP (*Network Data Plane*), implementace v podobě vstupního pluginu je standardně dostupná. Spojení akcelerační karty se softwarem IPFIX probe v jednom zařízení tvoří vysokorychlostní sondu pro monitorování síťového provozu s rychlostmi podle specifikace karty až 200 Gb/s. V současné době již probíhá vývoj technologie, který cílí na rychlost linky 400 Gb/s.

NFB nebo také COMBO akcelerační karty jsou určeny zejména pro zpracování síťových dat. Jejich jádrem je FPGA čip, což je technologie programovatelných hradlových polí pro implementaci hardwarových komponent. S hostitelským počítačem komunikuje prostřednictvím rozhraní PCI-Express, při přenosech se pro vysoký výkon využívá principu přímého přístupu do paměti (DMA – *Direct Memory Access*). DMA umožňuje vstupně/výstupním zařízením přímý zápis a čtení paměti RAM bez aktivní účasti procesoru, který se v té chvíli může věnovat rozpoznávání a agregaci síťových toků. Karta přijímá monitorovaný provoz skrze porty SFP (QSFP28) a dále ho zpracovává hardwarovými komponentami v FPGA. Činnost FPGA se řídí nahraným firmwarem (syntetizované komponenty z popisu v jazyce VHDL[11]). Základní funkcionalita firmware v NFB kartě sestává z označení příchozího paketu časovou značkou a analýzy hlavičky za účelem rozpoznání obsažených protokolů [17, 18].

Exportér se konfiguruje pomocí argumentů příkazového řádku – program nemá uživatelské rozhraní. Částečně se tak děje při kompilaci (následná dostupnost zásuvných modulů) a částečně při spouštění exportéru – command line programu s názvem `ipfixprobe`. Parametry programu zajišťují zejména aktivaci pluginů a předání pro ně nezbytných hodnot. Následuje příklad spuštění `ipfixprobe` s konfigurací pro čtení z raw soketu, export na IPFIX kolektor a aktivovaným procesním zásuvným modulem pro zpracování HTTP protokolu.

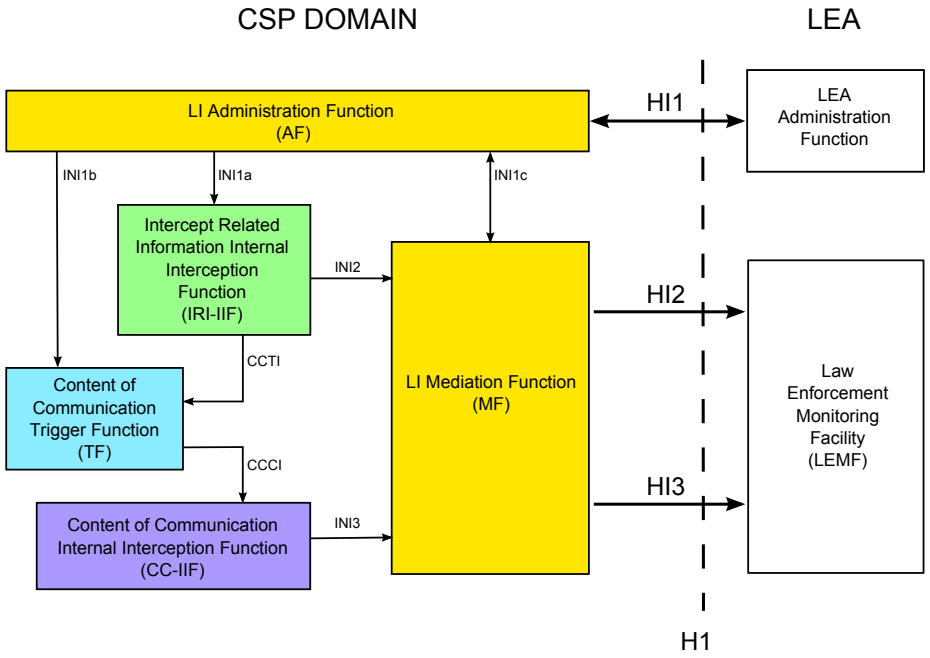
```
ipfixprobe  -i 'raw;ifc=wlp2s0' -p http
            -o 'ipfix;u;host=collector.example.com;port=4739'
```

[11] speciální programovací jazyk určený k implementaci hardwarových komponent v FPGA

Kapitola 3

Systém pro zákonné odposlechy

Druhým ze způsobů monitorování síťového provozu, kterým se tato práce zabývá jsou tzv. zákonné odposlechy. Jedná se o záchyt a následnou analýzu komunikace podezřelých osob využívajících veřejných komunikačních prostředků jako jsou telefonní sítě nebo internet. K realizaci odposlechů oprávněnými orgány slouží systém pro zákonné odposlechy (*Lawful Interception System*, LIS), který vychází z evropských norem ETSI [3]. V rámci výzkumného projektu[1] byla na FIT VUT v Brně navržena architektura systému popsaná v [13].



Obrázek 3.1: Architektura systému pro zákonné odposlechy podle norem ETSI

Struktura navrženého systému pro zákonné odposlechy je zachycena na obrázku 3.1. Systém je rozdělen mezi dva subjekty. Prvním je poskytovatel služeb (*Communications Service Provider* – CSP, např. poskytovatel internetového připojení – ISP). V jeho prostředí je nasazeno zařízení, které je schopno veškerou komunikaci sledovaného cíle zachytávat a předávat

[1] Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace, https://www.fit.vut.cz/research/project/517/.cs

ji oprávněným orgánům. Druhým subjektem jsou oprávněné orgány (*lawful enforcement agency* – LEA) se sběrným zařízením, které umožňuje zachycený provoz ukládat a později podrobněji analyzovat [8].

Pozornost je věnována zejména části na straně poskytovatele služeb. Zde je funkce nejčastěji realizována specializovanými hardwarovými zařízeními. Typicky je to síťová sonda, která provoz zpracovává a mediační zařízení (nazváno podle mediační funkce, kterou implementuje) zajišťující krátkodobé uložení a předání dat oprávněným orgánům. Ke komunikaci s LEA slouží tři protokoly: HI1, HI2 a HI3, které jsou ale v rámci popisované architektury velmi zjednodušeny. HI1 slouží k přijímání požadavků na správu odposlechů. Předpokládá se, že zprávy budou zasílány externími službami – formou listovní nebo elektronické pošty. Za konfiguraci odposlechu v systému je odpovědný pověřený pracovník u CSP. Skrze rozhraní HI2 jsou odesílána metadata týkající se síťových spojení sledovaných subjektů. Např. zahájení a ukončení komunikace (hovoru), připojení a odpojení od sítě, změna identity (IP adresy), apod. Tyto informace jsou předávány formou textových souborů, kde jeden soubor odpovídá jednomu odposlechu. Řádky souboru dokumentují události, ke kterým v průběhu záchytu došlo. Pro přenos zachycené zájmové komunikace je vyhrazena linka HI3. Provoz je na straně CSP nejprve ukládán do PCAP souborů, které jsou následně odeslány do LEA.

Architektura systému pro záchyt zájmového provozu je rozdělena do několika funkčních bloků, které zodpovídají za tématické skupiny procesů prováděných při realizaci odposlechu. Vstupní požadavky k odposlechu jsou přijaty administrační funkcí (AF). Zde nejprve proběhne kontrola správnosti údajů. Je zkontrolován správný rozsah časového intervalu, během kterého se bude odposlech realizovat, a identifikátor odposlouchávané osoby (tzv. *Network Identifier* – NID). V AF je také poprvé registrován identifikátor odposlechu, tzv. *Lawful Interception Identifier* – LIID. Ten musí být v systému unikátní.

V případě, že jsou všechny údaje správné, odposlech je zařazen do fronty čekajících odposlechů. AF je následně zodpovědná za korektní inicializaci a ukončení odposlechu, tj. konfiguraci ostatních částí systému (skrze rozhraní INI1a, INI1b, INI1c) tak, aby bylo zajištěno, že budou zachycena všechna zájmová data v povoleném intervalu pro odposlech a zároveň nebudou zaznamenána žádná data mimo platnost odposlechu.

Funkce dynamické identity (IRI-IIF) udržuje informace o síťové identitě odposlouchávaných uživatelů (např. jejich IP adresy). V průběhu času se identita může měnit a úkolem bloku IRI-IFF je tuto událost detekovat. Po detekci změny jsou informace o nové identitě distribuovány prostřednictvím rozhraní CCTI trigerovací funkci, která zajistí korektní konfiguraci záchytu. Z událostí jsou dále vytvářeny zprávy sledující začátky a konce spojení, např. uživateli byla přiřazena IP adresa protokolem DHCP nebo platnost adresy vypršela. Tyto zprávy s metadaty jsou odesílány skrze rozhraní INI2 do mediační funkce.

Blok CC-IIF, který by mohl být označen jako funkce odposlechu obsahu komunikace sleduje síťový provoz a kopíruje veškerý obsah komunikace vztahující se k některému dříve konfigurovanému identifikátoru sledovaného cíle. Vstupní požadavky na konfiguraci nových sledovaných identifikátorů, započetí, či ukončení odposlechu jsou přijímány od trigerovací funkce na rozhraní CCCI. Zachycený provoz je odesílán rozhraním INI3. Implementace musí zajistit hned několik procesů, které jsou kritické z hlediska bezeztrátového záchytu při plné vytíženosti linky. Odposlouchávané identifikátory musí být v provozu rozpoznány a zbytkový provoz zahozen. Pro následnou rekonstrukci je také nezbytné kopírované pakety označit časovými značkami. Architektura proto připouští hardwarově akcelerovanou variantu CC-IIF.

Za správu zachycených dat aktuálně probíhajících odposlechů je odpovědná mediační funkce (MF). Ta zároveň přijímá a zpracovává metainformace od IRI-IIF a obsah zachy-

cené komunikace od bloku CC-IIF. Data zkombinovaná z obou kanálů jsou dále předávána oprávněným orgánům. Děje se tak periodicky nebo v reakci na události, např. ukončení odposlechu, vyžádání ze strany LEA.

3.1 Sonda FlexProbe

Flexibilní sonda FlexProbe je zařízení určené k realizaci zákonných odposlechů na lince o rychlosti až 10 Gb/s. Díky efektivnímu využití hardwarové akcelerace umožňuje filtraci síťového provozu nejen na síťové vrstvě (L3), ale také na úrovni aplikačních protokolů (L7). Ilustrační schéma sondy se nachází na obrázku 3.2.

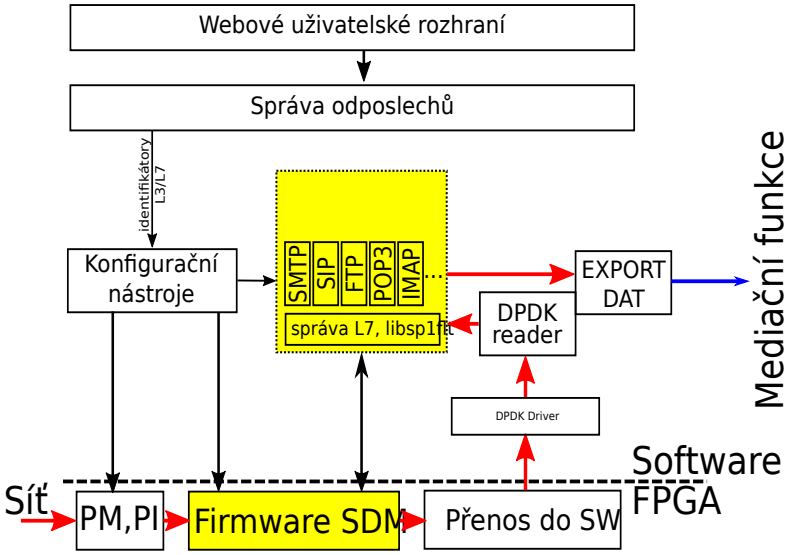
Na rozdíl od exportéru IPFIX probe, kterému se věnovala kapitola 2.4 je sonda FlexProbe specializované hardwarové zařízení připravené k okamžitému nasazení do produkční sítě. Při vývoji se proto spíše než na univerzálnost klade důraz na optimalizaci firmware pro konkrétní hardwarovou architekturu a zajištění konfigurovatelnosti pomocí jednoúčelových softwarových nástrojů. Sonda FlexProbe je již několikáté zařízení zabývající se záchytem síťového provozu, které bylo navrženo v rámci výzkumných projektů na FIT VUT v Brně. První prototyp byl vytvořen v rámci projektu SEC6NET[2], který umožňoval záchyt provozu na základě identifikátoru síťové vrstvy (IPv4, IPv6). Součástí projektu SProbe[3] bylo rozšíření sondy o funkcionalitu záchytu podle identifikátoru aplikační (L7) vrstvy. Vznikl také prototyp zařízení odposlouchávající provoz na lince o rychlosti 1 Gb/s, na který přímo navazuje sonda FlexProbe. Projekt FlexProbe [9] má za cíl kromě zvýšení rychlosti linky zajistit flexibilitu sondy. Flexibilita spočívá v použití pro několik možných scénářů, funkcionalita zákonných odposlechů je rozšiřována o tvorbu statistik provozu – monitorování síťových toků (viz 2.2).

Jádrem sondy je SoC FPGA Intel Arria 10, který v sobě kombinuje ARM procesor a programovatelné hradlové pole (FPGA). Do FPGA jsou zapojeny čtyři 1/10 Gbps SFP+ porty určené k připojení odposlouchávané linky. Řízení rozhraní PCI-Express je zajištěno síťovým procesorem Intel XL710, pomocí kterého bude platforma připojena k hostitelskému serveru. Konfigurace sondy je řízena ze strany hostitelského počítače.

Sonda implementuje systém pro zákonné odposlechy (LIS). Návrh LIS v 3 předpokládá pouze odposlech na základě identifikátorů síťové vrstvy, proto se architektura částečně liší. Záchyt na základě síťového (L3) a aplikačního (L7) identifikátoru se zpracovává odděleně. V systému jsou tedy dva bloky typu CC-IIF a dva exportní kanály. Rozhraní INI3 je nahrazeno rozhraními L3 Stream a L7 Stream, INI2 nahrazuje SLIS Control (*SProbe Lawful Interception System*). Na rozhraních se používají stejnojmenné protokoly. Těmito kanály je zajištěno předání odposlechnutého provozu a kontrolních metadat mediační funkci. Mediační funkce je implementována jako software specializovaný k přijímání odposlechnutého provozu a statistik ze sond FlexProbe. MF může být integrována do sondy nebo fungovat jako samostatné zařízení, které data přijímá ze sítě. V případě vestavěné varianty je software nainstalován v hostitelském serveru a se sondou interaguje přes PCI-Express.

Konfigurace sondy probíhá primárně přes webové uživatelské rozhraní. Administrace spočívá zejména v definici pravidel pro filtrování síťové komunikace. Odposlechy, kterým tato pravidla přísluší pak mohou být přes rozhraní spouštěny, pozastaveny a ukončeny. Pro vzdálený přístup se používá protokol SSH nebo REST API.

[2] https://www.fit.vut.cz/research/publication/11560/.cs
[3] https://www.vut.cz/vav/projekty/detail/26509



Obrázek 3.2: Ilustrační schema sondy. Komponenty jsou rozděleny na hardwarové (FPGA) a softwarové. Žlutě jsou označeny komponenty zodpovědné za filtraci podle L7 identifikátorů.

Primárním cílem sondy je filtrace a zachycení zájmového provozu. Zájmový provoz, respektive zájmový datový tok je definován pomocí pravidel. Na síťové (L3) nebo transportní (L4) vrstvě se tato pravidla skládají z cílové a zdrojové IP adresy, cílového a zdrojového TCP/UDP portu a protokolu. Podporované jsou protokoly IPv4 i IPv6. Pro filtraci na aplikační L7 vrstvě jsou využívány regulární výrazy obsahující aplikační identifikátory (např. e-mailová adresa).

Analýza aplikačních protokolů zahrnuje podporu protokolů SMTP, POP3, IMAP, FTP a SIP. Zájmový provoz s protokolem SMTP může být identifikován pomocí odesílatele nebo příjemce emailu. U POP3 a IMAP sezení se zachytávají jednotlivé zprávy nebo celé sezení (na základě IP adresy nebo uživatelského jména). Z FTP komunikace se odposlouchává řídicí i datový kanál a podobně je tomu u protokolu SIP, kde jsou data přenášena transportním protokolem UDP a filtrována na základě SIP ID.

Záchyt a filtrace paketů je na vysokorychlostní lince výpočetně náročná. Sonda proto využívá pro urychlení zpracování paketů hardwarovou zřetězenou linku implementovanou v technologii FPGA. Zřetězená linka obsahuje komponenty pro rozpoznání aplikačního protokolu, vyhledávání regulárních výrazů, analýzu hlaviček protokolů a další. Identifikátory rozpoznaných síťových toků (zdrojová/cílová adresa, porty) jsou přidávány do komponenty Filter, kde probíhá jejich klasifikace. Kapacita těchto komponent je omezená a pevně definovaná při sestavení firmwaru. Pro předzpracování provozu před komponentou Filter se předpokládá použití dvou paralelních zřetězených linek. Vstupní provoz je proto rozdělen na tzv. upstream a downstream (např. pomocí hardwarového zařízení TAP).

Kapitola 4

Testování síťových sond

Hlavním tématem této práce je testování sond popsaných v předchozích kapitolách. Sonda je zařízení složené z mnoha softwarových i hardwarových modulů, jejichž funkčnost musí být ověřena samostatně i jako součást celého systému. K tomuto účelu vznikají sady testů, které ověří korektní činnost, dodržení očekávaného rozhraní a dosažení požadovaného výkonu. Obzvláště pro zařízení, kde se kombinují hardwarové a softwarové komponenty platí, že každá část systému vyžaduje jiný typ testů.

Existují dvě základní metody pro ověření správného chování systému: statická a dynamická analýza. Při statické analýze se zkoumají vlastnosti programu bez jeho provádění. Zástupcem statické analýzy je formální verifikace, která pomocí formálních metod dokazuje, že systém odpovídá specifikaci. Při dynamické analýze je naopak hlavním zdrojem dat běh programu a zkoumá se jeho průběh i výstupy.

Při ověřování funkce hardwarových komponent se často využívá simulace. Simulace je druh dynamické analýzy, při které je ověřováno dodržení specifikace monitorováním vstupů a výstupů v simulačním prostředí. U komplexních systémů se provádí funkční verifikace, která ověřuje funkční parametry pomocí pokročilých metod simulace. K provedení funkční verifikace je nutné vytvořit referenční model podle specifikace systému. Verifikovaná komponenta, tzv. *device under verification* (DUV) je spuštěna se stejnými vstupy jako referenční model. Dodržení specifikace je dáno srovnáním výstupů DUV a referenčního modelu [12].

Jiným způsobem dynamické analýzy je testování, při kterém se určuje korektnost systému na základě výsledků spuštění. Hardwarové i softwarové komponenty mohou být testovány společně, podmínkou je zajištění prostředí, ve kterém je systém spustitelný. Testovaný systém nebo podsystém je nazýván *system under test* (SUT). Ověření spočívá ve srovnání výstupů SUT s očekávanými hodnotami nebo je kontrolován vnitřní stav SUT.

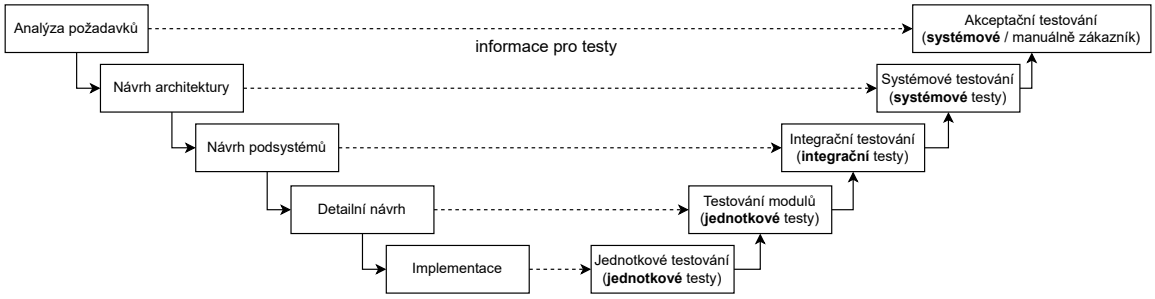
Test se obvykle skládá ze 4 částí, jejichž názvy se většinou nepřekládají: *setup*, *exercise*, *verify* a *teardown*[1]. Ve fázi *setup* se připravuje prostředí pro běh testu. Např. jsou zkonstruovány potřebné objekty, nastaveny globální proměnné, vloženy testovací data do databáze a podobně. Následně je ve fázi *exercise* předáno řízení SUT společně s testovacími vstupy. Výsledky spuštění SUT jsou kontrolovány ve fázi *verify*. Zde musí být rozhodnuto, zda výstupy odpovídají očekávaným a jestli byl test úspěšný. Ve fázi *teardown* je prostředí běhu vyčištěno tak, aby byl stav stejný, jako před spuštěním testu.

[1] http://xunitpatterns.com/Four%20Phase%20Test.html

4.1 Druhy testů

Testy je možné rozdělit podle několika různých kritérií. Základní dělení vychází požadavků, které se souběžně zaměřují na funkci a výkon systému. Testy se proto nazývají funkční a výkonnostní. Jemnější rozdělení funkčních testů využívá jako kritérium velikost podsystému, jehož činnost je testována (SUT). V této kapitole budou postupně popsány testy jednotkové, integrační a systémové. Jiná varianta dělení vyplývá ze způsobu provedení, např. pomocí simulace versus s použitím fyzického zařízení. Kvalitní testovací sada by měla obsahovat kombinaci všech zmíněných druhů testů. I když by se mohlo zdát, že pokrytí požadavků sadou hierarchicky nejvyšších testů docílí požadovanou funkčnost, problém nastane v případě zanesení chyby do implementace. Nižší úrovně testů přináší jednodušší lokalizaci chyby a jistotu, že se při vývoji neobjevila znovu.

Zastoupení jednotlivých druhů testů je docíleno částečně přirozeně. Implementace začíná u modulů a postupuje k jejich integraci do funkčního systému. V těchto fázích již vznikají testy k okamžitému ověření korektního chování jednotek, modulů i celého systému. Úrovně testů úzce souvisí s fázemi návrhu systému a v ideálním případě by proto měla být testovací sada navržena již před samotnou implementací. Spojení mezi návrhem a testováním výsledného produktu ilustruje tzv. V-model (obrázek 4.1). Vývojové aktivity se promítají do etap testování a vzniklý návrh jednotlivých úrovní poskytuje informace pro testy ověřující správné chování.



Obrázek 4.1: V-model vycházející z certifikace ISTQB [1]

Testy nejmenších celků se nazývají jednotkové. Jednotkou je myšlena část systému, která může být izolována od zbytku a současně je nedělitelná. Pokud je jednotka izolována, není závislá na externích systémech, např. na jiných jednotkách systému, souborovém systému, konfiguraci zařízení a podobně. Výstup je v tom případě plně závislý na vstupech jednotky. Nedělitelná je nejmenší část systému, která samostatně zajišťuje určitou funkci. Co přesně bude považováno za jednotku musí být stanoveno při návrhu testů. V případě softwaru to jsou typicky funkce nebo třídní metody ve zdrojovém kódu. Někdy se ale jako jednotkové testy označují i testy celých modulů.

O úroveň vyšší jsou testy integrační. Při integračním testování se moduly testují ve skupinách, ve kterých spolupracují. Ověřuje se způsobilost jednotek k následné integraci do systému. Testy se proto zaměřují na odhalení chyb v rozhraní modulů a jejich vzájemné komunikaci (vstupy a výstupy). Moduly jsou testovány jak v rámci podsystému, který zajišťuje určitou funkci, tak mezi jednotlivými podsystémy.

Systémové testy ověřují chování systému jako celku. Test často chápe SUT jako tzv. *black-box*: nereflektuje implementační specifika a ke své činnosti využívá pouze uživateli dostupné vstupy a výstupy. Systémové testy slouží k přímému ověření požadavků, které

jsou specifikovány na počátku návrhu (např. jako v kapitole 5). Podmínkou provedení systémových testů je, že systém musí být zkompletován. Cílem je realizovat testy v prostředí, které se co nejvíce podobá produkčnímu.

Systémový test odhaluje chyby implementace nepřímo na základě výstupu, který nebyl očekáván. Pro následnou lokalizaci chybného podsystému je nutná znalost implementace. Při neúspěšném systémovém testu se tester často navrací k nižším testům, pomocí kterých chybu lokalizuje. Test, který byl vytvořen pro konkrétní chybu a zaručuje, že tato chyba nebude znovu zanesena do implementace, se nazývá regresní.

V případě hardwarového zařízení se mohou nazývat jednotkovými testy také testy komponent, které jsou implementovány v FPGA. V systému totiž hardwarová komponenta vystupuje jako nezávislá jednotka zodpovědná za jednu funkci. Komponenty implementované v FPGA jsou často seskupeny do zřetězené linky, která postupně zpracovává data (např. síťový provoz). Integrační test ověřuje správné začlenění hardwarových komponent do linky na základě zpracování testovacích dat. Rozšířený scénář integračních testů zkoumá spolupráci hardware a software. Softwarovými nástroji bývá hardware konfigurován, opačným směrem jsou do softwaru předávána data. Mezi integrační testy mohou být zařazeny také testy grafického uživatelského rozhraní. Zde je ověřena korektní komunikace tzv. *frontendu* (přímé rozhraní s uživatelem) s *backendem*, který je zodpovědný za nastavení ostatních softwarových i hardwarových komponent. Právě konfigurace podsystémů administračním modulem (*backendem*) může být tématem jiného integračního testu, případně je toto chování pokryto v rámci testů GUI. Systémové testy již využívají výhradně uživatelsky dostupné rozhraní pro konfiguraci (GUI nebo veřejné API) a primární vstupy a výstupy zařízení. Pro příklad monitorovací a exportní síťové rozhraní.

Pro výše popsané funkční testy obecně platí, že je kladen velký důraz na ověření výsledků a výstup SUT musí být ekvivalentní s očekávaným výstupem. Specifickou kategorií jsou testy výkonnostní. Průběh výkonnostních testů je podřízen účelu dosažení horních limitů zpracování dat systémem a ověření výsledků proto může být zjednodušeno, případně jsou předpokládány ztráty. Jeden ze způsobů výkonnostního testování spočívá ve zvyšování zátěže, dokud systém reaguje. Při takovém testu se neověřují výsledky, pouze se hledá maximální únosná hodnota zatížení. V jiném případě se kontrolují pouze metadata: pro příklad zpracování síťového provozu počet zpracovaných paketů a jejich velikost v bytech. Je nutné zohlednit, že i při správných hodnotách metadat mohlo dojít k chybě a obsah dat může být poškozen. Pro očekávanou zátěž v produkčním prostředí by proto měl být výstup SUT podroben detailnější analýze. Například po skončení testu, aby analýza neovlivnila výkon.

I výkonnostní testy lze dále rozdělit na zátěžové a tzv. stress testy. Zátěžové testují výkon v mezích, ve kterých by zařízení mělo bezchybně pracovat. Takový výkon byl specifikován jako požadavek. Stress testy naopak hledají skutečný maximální výkon a zařízení cíleně přetěžují. Pro bezproblémové fungování v produkčním prostředí by mělo platit, že maximální výkon bude násobně převyšovat očekávanou zátěž. Zajímavá může být také kombinace stress testu s funkčním, pomocí níž lze otestovat, jestli i v případě velkého zatížení systém stále funguje. Stress test nejprve zahltí zařízení, což povede k zahození části vstupních dat. Algoritmus zotavení systému po určité době vyčistí paměť komponent. Jejich opětovná funkčnost je vyzkoušena funkčním testem.

Kapitola 5

Specifikace požadavků

Požadavky na fungování a podporovanou funkcionalitu sond pro monitorování síťového provozu pochází jak z obecně navržených principů (viz kapitola 2), tak od uživatelů, případně zákazníků při vývoji specializované sondy. V základu by bylo možné požadavky rozdělit do dvou skupin: funkční a výkonnostní. Zatímco požadavky na funkci z velké části definuje architektura a implementované protokoly (např. v 2.3 a 3), požadavky na výkon souvisí především s cílovým prostředím, kde bude sonda nasazena a jsou omezeny možnostmi hardwarové platformy.

Specifikace funkce sondy musí obsahovat popis jejího výstupu (výsledku) a rozhraní, kde je možné výstup odečíst. Například: počet bytů přenesených v rámci jednoho síťového toku je součástí záznamu toku exportovaného na rozhraní s připojeným kolektorem. Požadavky se mohou týkat hlavní nebo obslužné činnosti sondy. Hlavní činností se myslí agregace síťových toků nebo záchyt zájmové komunikace podle typu sondy. Obslužné funkce se zabývají administrací sondy (např. prostřednictvím GUI), udržováním spojení mezi exportérem a kolektorem, případně předáváním kontrolních zpráv s metadaty.

Výkon sondy bývá udán rychlostí zpracování provozu. Na tuto rychlost je však možné pohlížet ve dvou místech. Jedním z nich je vstupní rozhraní sondy. Podle prostředí umístění (firemní síť, poskytovatel internetového připojení) mají monitorované linky kapacitu od jednoho až po stovky gigabitů za sekundu. Většinou je požadováno zpracování vstupního provozu na plné rychlosti vstupní linky. Komponenty, které se zabývají zpracováním vstupního provozu bývají z tohoto důvodu hardwarově akcelerované. Druhý výkonnostně důležitý bod je exportní rozhraní. Zde je výkon zatížen režií softwarového zpracování paketů, síťové rozhraní navíc bývá typicky spravováno operačním systémem. Z těchto důvodů se může jednat o úzké místo systému. Na druhou stranu je v místě exportu vstupní provoz značně redukován: agregován na záznamy toků při monitorovaní síťových toků nebo odfiltrován nezájmový provoz u sondy pro zákonné odposlechy. Požadavek na rychlost exportu tak může být udán jako procento z kapacity vstupní linky nebo počet ukončených toků za vteřinu.

Důležitou vlastností sondy je zotavení se z nestandardních stavů. Tyto stavy vznikají překročením výkonnostních limitů, následným zaplněním paměti, zahazováním paketů atd. Požadavky na zotavení určují, jak má sonda na nestandardní stav reagovat, např. zahozením nezbytné části provozu a notifikací uživatele. Nestandardní stav nastává také při ztrátě spojení mezi sondou a kolektorem. Je specifikováno, jak se bude spojení obnovovat.

Splnění požadavků musí být ověřeno, nejlépe průběžně při vývoji. Používají se k tomu testy, které svou činností zkoušejí, tzv. pokrývají, vlastnosti specifikované v požadavcích.

Podle pokrytých požadavků jsou i testy označovány jako funkční a výkonnostní. O návrhu a implementaci testů síťových sond budou pojednávat další kapitoly této práce.

5.1 Požadavky na sondu IPFIX probe

Exportér IPFIX probe je univerzální nástroj, jehož funkce není přímo upravována dle požadavků konkrétního uživatele, ale je definována standardním procesem monitorování síťových toků. Základní funkcionalitou je plná podpora agregace síťových toků na vrstvách TCP/IP a implementace protokolu IPFIX pro export záznamů. S tím souvisí požadavky na procesy popsané v kapitole 2.3 jako je správná analýza paketu za účelem vytvoření hash síťového toku, nalezení toku ve flow cache a detekce ukončení toku. Požadavky směřují k zajištění přesných statistik o síťovém provozu. Sonda musí poskytovat pro stejný vstupní provoz vždy stejné statistiky.

Pro měření přesných statistik je klíčové zajistit bezztrátový příjem paketů. Je potřeba zabezpečit, aby ze síťového rozhraní byly do software přenášeny pakety libovolné velikosti na plné rychlosti linky. Vstupní modul také musí zajistit bezeztrátovost, neboť jakákoliv ztráta v tomto místě by znamenala jisté zkreslení statistik. Důležitým aspektem je také to, že je pro software komplikované takovou ztrátu detekovat a jednalo by se tak o nekontrolované zahazování paketů.

1. Příchozí pakety musí být ze zvoleného vstupního rozhraní předány programu IPFIX probe. Měla by být podporována rozhraní raw soket, NDP (rozhraní NFB karty) nebo PCAP soubor.
   - 1.1 Musí být zajištěno zpracování všech typů rámců podporovaných síťovým rozhraním (většinou omezeno pouze maximální velikostí paketu).
   - 1.2 Paket musí být předán k softwarovému zpracování beze změny – při přenosu nesmí dojít k porušení informací obsažených v paketu.
   - 1.3 Je nutné zpracovat všechny pakety vstupního provozu beze ztrát, žádný paket se nesmí při přenosu zahodit.
   - 1.4 Pakety musí být opatřeny časovou značkou.

Aby mohly být síťové toky agregovány, musí dojít k extrakci sledovaných vlastností. Korektní analýza paketu zaručí správné přiřazení do síťového toku a dává předpoklad ke správnému výpočtu statistik daného toku. Minimální množina vlastností identifikujících síťový tok se skládá z atributů vrstvy síťového rozhraní (MAC adresa zdroje a příjemce), síťové vrstvy (IP adresy) a transportní vrstvy (zdrojový a cílový port, transportní protokol). Pro extrakci hodnot síťové vrstvy je nutné paket analyzovat do nejhlubšího zapouzdření a identifikovat skutečné parametry odesilatele a příjemce. Tímto je zajištěno jednoznačné přiřazení paketu v případě použití tunelů pro síťové protokoly (IP in IP).

2. Z paketu musí být získány správné hodnoty atributů: MAC adresa zdroje a příjemce, IP adresa zdroje a příjemce, zdrojový a cílový port, transportní protokol.
   - 2.1 Hodnoty atributů síťové vrstvy (IP adresy) je nutné extrahovat z nejhlouběji zapouzdřené hlavičky protokolu (IP in IP).

Aby bylo možné agregovat statistiky k síťovým tokům, je potřeba rychle vyhledat k přijatému paketu odpovídající tok v tabulce síťových toků. K vyhledání se používají hash

funkce. Využití hash funkcí pro rychlé vyhledávání s sebou ale může přinést řadu problémů. Hash funkce mapují větší prostor hodnot (všechny kombinace klíčových atributů) do menšího, musí mapovat množinu hodnot atributů na stejnou hodnotu hash funkce. Kvůli metodě tvorby Biflow záznamů zároveň platí, že na stejnou hodnotu hash musí být mapován také opačný směr komunikace. Zároveň by ale mělo platit, že odlišná kombinace klíčových atributů by měla být mapována na jinou hodnotu hash funkce. Tuto vlastnost nicméně nelze zajistit vzhledem k mapování do menšího rozsahu hodnot. Mapovat by se mělo do dostatečně velkého prostoru, aby byla pravděpodobnost překryvu hash přijatelně malá. Pokud by požadované vlastnosti mapovací funkce nebyly splněny, docházelo by k agregaci různých toků v jednom záznamu nebo k vytváření duplicitních záznamů pro jeden tok.

3. Z klíčových položek paketu je nutné vytvořit kvalitní hash toku.

    3.1 Hash musí být invariantní ke směru komunikace (lze zaměnit zdrojové a cílové položky).

    3.2 Obor hodnot mapovací funkce musí být dostatečně velký, aby v běžném síťovém provozu nedocházelo k záměně toků.

Kritickou komponentou exportéru je paměť toků (*flow cache*). Až do chvíle exportu musí udržovat jako hlavní paměť validní záznam s průběžnými statistikami toku. Zároveň je formou uložení a druhem použité paměti nejvíce ovlivněn výkon exporteru. Paměť toků musí být schopna za každých okolností vložit síťový tok. Na vyhledávací dotaz musí poskytnout záznam o síťovém toku, nebo informovat, že záznam nebyl nalezen. Pokud není záznam v tabulce toků nalezen, musí se v tabulce pro hledaný tok vytvořit záznam nový.

V průběhu činnosti je možné dosáhnout situace, kdy bude paměť při vkládání nového toku plná. V tu chvíli je nutné označit některý v tabulce uložený záznam za ukončený a nahradit jej nově vzniklým. Odstraněný záznam je nutné před přepsáním exportovat na kolektor. Díky mechanismu nuceného exportu nejsou zahozeny informace o nově příchozím síťovém toku a zároveň, pokud není zahlcena exportní linka, nedochází ke ztrátě již agregovaných statistik. V běžném síťovém provozu k těmto situacím zpravidla nedochází a export záznamů probíhá na základě detekce ukončení toku, případně při překročení aktivních a neaktivních časových limitů. Nicméně problém může nastat například při kybernetickém útoku typu *Denial of Service* (DoS/DDoS), kdy po sítí v rychlém časovém sledu přicházejí nové síťové toky. Záznamy z paměti toků jsou rychle odstraňovány a může dojít k přetížení exportního rozhraní, při kterém dochází ke ztrátám exportovaných dat. Sonda musí být schopna se z těchto stavů zotavit, čehož je dosaženo za pomoci neaktivního časového limitu, který po útoku postupně vyřadí neplatné záznamy a vyčistí tak paměť toků.

4. Flow cache musí fungovat jako asociativní tabulka s unikátním klíčem, kterým je hash síťového toku.

    4.1 Podle hash musí být v tabulce správně vyhledán záznam v tabulce toků.

    4.2 Síťový tok je označen k exportu vždy při splnění jedné z podmínek: detekce ukončení toku, vypršení aktivního nebo neaktivního časového limitu, zaplnění paměti toků, vyžádání procesním zásuvným modulem.

    4.3 Pokud je paměť zaplněná, musí být při vkládání nového toku některý uložený záznam exportován a nahrazen novým.

    4.4 Exportovaný záznam musí být z flow cache vymazán nebo být invalidován. Na vyhledání identického hash dále paměť toků odpovídá jako nenalezeno.

Pokud paměť toků funguje správně, mohou být agregovány vlastnosti toků a tvořeny statistiky síťového provozu. Agregaci základních položek provádí jádro softwaru IPFIX probe, statistiky pokročilých vlastností zejména na aplikační vrstvě sbírají procesní zásuvné moduly. Mezi základní vlastnosti se řadí například počet bytů, počet paketů, časová značka prvního a posledního paketu atd. Hodnoty jsou agregovány pomocí agregačních funkcí, které musí být správně implementovány. Výsledky statistik by jinak byly nepoužitelné, nebo by mohl být negativně ovlivněn mechanismus exportu záznamů. Například při špatné aktualizaci časové značky posledního viděného paketu by byl tok exportován předčasně kvůli neaktivnímu časovému limitu.

5. Při přijetí nového paketu musí být v záznamu správně aktualizovány všechny agregační položky podle určených agregačních funkcí. Používá se stávající a nová hodnota, extrahovaná z příchozího paketu. Pro agregační funkce platí:
   - 5.1 Suma – do položky je ukládán správný součet hodnot agregovaného atributu ze všech paketů toku (počet paketů, bytů).
   - 5.2 Informace z prvního paketu – agregovaná hodnota odpovídá hodnotě atributu z prvního přijatého paketu náležícího do síťového toku (TCP příznaky, směr komunikace). Časová značka musí odpovídat času přijetí prvního paketu.
   - 5.3 Informace z posledního paketu – v položce se vždy musí nacházet hodnota z posledního paketu: při každém příchozím paketu, který náleží do toku je hodnota přepsána novou. V případě časové značky hodnota odpovídá času přijetí posledního paketu.
6. Software musí správně obsluhovat procesní zásuvné moduly.
   - 6.1 Při aktivaci musí být zásuvnému modulu umožněno rozšířit vnitřní reprezentaci síťového toku o nestandardní položky. Při použití IPFIX exportního protokolu plugin rozšiřuje také šablonu.
   - 6.2 Zásuvný modul musí být notifikován o přidání nového toku do flow cache.
   - 6.3 Zásuvný modul musí být notifikován o přijetí nového paketu.
   - 6.4 Při notifikaci zásuvný modul vždy získá záznam ke zpracovávanému toku. Současně musí být zásuvný modul schopen do záznamu zapisovat.
   - 6.5 Data jsou zpracována pouze aktivovanými zásuvnými moduly podle argumentů příkazové řádky.

Exportované záznamy toků musí dodržovat specifikaci exportního protokolu, kterým je IPFIX nebo UniRec. V případě nedodržení struktury, datových typů či pořadí bytů by záznamy nemohly být přijaty kolektorem a statistické informace by byly zahozeny. Protokol IPFIX definuje šablony, které popisují položky exportovaného záznamu. Tyto šablony odesílá sonda na kolektor ihned po navázání spojení [5].

Spojení na kolektor je udržováno aktivní pro zajištění efektivity kontinuálního exportu. V případě přerušení spojení s kolektorem je znovu navázána komunikace při odeslání následující dávky agregovaných záznamů. Z důvodu minimalizace ztrát se při neúspěšném připojení proces opakuje. Pokud se spojení nepodaří obnovit do stanoveného časového limitu, dojde ke kontrolované ztrátě záznamu, která je zanesena v logu.

7. U exportovaných záznamů je nutné dodržet protokol IPFIX nebo UniRec. Záznamy musí obsahovat všechny klíčové i agregované hodnoty uložené v tabulce síťových toků.

8. Výstupní zásuvný modul musí udržovat aktivní spojení s kolektorem. Pokud je spojení přerušeno, musí se nejpozději při následujícím odesílání dat spojení znovu navázat. V případě neúspěchu a zahození dat musí být záznam o ztrátě logován.

Každý model sondy má podle použitého hardware jiné výkonnostní parametry. Hlavní metrikou výkonu je maximální propustnost, která je měřena pro různé podmínky. Například při zachování nulové ztráty paketů nebo s určitou mírou ztrátovosti. Propustnost nesmí výrazně klesat v souvislosti se změnou konfigurace, např. s aktivací některého ze zásuvných modulů. To by znamenalo neefektivní softwarové zpracování vstupních paketů, které by vytvářelo úzké místo systému.

9. Při aktivaci libovolné kombinace zásuvných modulů nesmí klesnout výkon zpracování o více než 20 % oproti stavu bez aktivovaných zásuvných modulů.

5.2 Požadavky na sondu FlexProbe

Projekt flexibilní sondy vychází z požadavků Policie České republiky (PČR). PČR je oprávněna na základě povolení udělené soudem provést zákonný odposlech tak, aby bylo možné dohledat pachatele trestné činnosti. Požadavek na sondu je realizovat korektní odposlech v souladu se zákonem a zajistit kvalitní výstup, který bude možné použít jako důkazní materiál. Zákonný odposlech je jasně definovaný dlouhodobý proces. Následuje jeho popis.

Povolení zákonných odposlechů ze strany soudu je úzce spojeno s konkrétním identifikátorem pachatele trestné činnosti a s konkrétním časovým obdobím. Realizace zákonných odposlechů se skládá z několika kroků: nejdříve je sonda nasazena do cílového prostředí a je ověřeno, že je sonda schopna přijímat relevantní data. Monitorovaná linka musí obsahovat komunikaci zájmových osob. Následně je sonda nastavena tak, aby mohla realizovat záchyt zájmového provozu podle zadaného povolení. S ohledem na dodržení povolení je důležité, aby sonda zachytila pouze data schválených osob ve vymezeném časovém období. Po ukončení odposlechu je ze sondy konfigurace smazána. Zachycená data se průběžně ukládají na mediační zařízení, které musí umožnit přístup k uloženému zájmovému provozu jak v průběhu, tak i po ukončení zákonného odposlechu.

1. Sonda musí provést zákonný odposlech přesně podle povolení soudu.
   - 1.1 Příchozí paket je analyzován a je označen za zájmový, pokud nese identifikátor síťové (L3) nebo aplikační (L7) vrstvy některého z aktivních odposlechů.
   - 1.2 Podpora tunelů pro síťové protokoly (IP in IP): paket je analyzován až do nejhlubšího zapouzdření a je rozpoznán L3 nebo L7 identifikátor.
   - 1.3 Všechny pakety, které jsou v průběhu analýzy označeny jako zájmové se exportují na mediační zařízení.
   - 1.4 Pakety, které se v provozu vyskytly před začátkem nebo po konci vymezeného časového období nejsou zachyceny.

Identifikátor, ke kterému se váže odposlech odpovídá některému z podporovaných aplikačních protokolů (viz kapitola 3.1) nebo IP adrese (množině adres). Případy užití sondy je proto možné rozdělit do dvou skupin. (i) Záchyt provozu podle L3 identifikátorů a (ii) záchyt provozu podle aplikačních identifikátorů. Pro testování je vhodné využít tohoto rozdělení, neboť každý scénář je zaměřen na odlišné parametry sondy a využívá různé testovací případy.

2. Záchyt provozu lze provést podle L3 identifikátorů (identifikátor síťové vrstvy).

   2.1 Musí být podporována IP adresa ve verzi 4 nebo 6. Množina aktivních odposlechů může být složená různě z IPv4 a IPv6 adres.

   2.2 Jako identifikátor musí být možné specifikovat službu – port a IP adresu.

3. Záchyt provozu lze provést podle L7 identifikátorů (identifikátor aplikační vrstvy).

   3.1 Na aplikační vrstvě jsou rozpoznány na základě přenášeného obsahu (regulárních výrazů) protokoly SMTP, POP3, IMAP, FTP a SIP.

   3.2 Ve množině L7 odposlechů se vyskytují různé aplikační protokoly. Všechny zájmové toky svázané se zadaným odposlechem musí být exportovány.

4. Množina aktivních odposlechů se skládá z kombinace L3 a L7 identifikátorů. Exportovány musí být toky jak na síťové, tak na aplikační vrstvě, které odpovídají zadaným zákonným odposlechům.

Ke každému exportovanému paketu je přiřazena časová značka udávající dobu příchodu paketu z připojené linky. Časová značka musí odpovídat internímu času sondy v okamžiku přijetí paketu ze sítě. Tím je zajištěno zejména to, že bude možné zachycené pakety seřadit a korektně rekonstruovat komunikaci. Společně s paketem je exportována informace o náležitosti k odposlechu – identifikátor LIID (viz 3), podle kterého je následně zachycený provoz tříděn na mediačním zařízení a ukládán do oddělených souborů.

5. Exportované pakety mají přiřazenou časovou značku, je tak možné rekonstruovat původní komunikaci ve správném pořadí.

6. Exportované pakety mají přiřazený unikátní LIID identifikátor v rámci skupiny aktivních odposlechů. Pakety se stejným LIID patří ke stejnému odposlechu, proto musí být každý síťový tok označen tímto identifikátorem.

Mediační zařízení (mediační funkce) se pokládá za bezpečné úložiště pro export zachycených dat, ale také pro pravidelnou zálohu provozních logů sondy. S bezpečností souvisí šifrování disku a notifikace reagující na události, například zaplnění disku. Šifrováno je také úložiště logů a databáze aktivních odposlechů na sondě. Citlivé informace tvořené osobními identifikátory podezřelých osob jsou tak chráněny i v případě vyjmutí perzistentní paměti.

Záchyt zájmového provozu musí být sonda schopna provést na plně saturované lince (1 Gbps nebo 10 Gbps podle varianty), bez ohledu na skutečnost, zda se jedná o filtraci na síťové (L3) nebo aplikační vrstvě (L7). Tím je myšleno, že se žádný tok s výskytem odposlouchávaného identifikátoru nesmí ztratit. V takovém případě by důkaz trestné činnosti nebyl platný. Sonda musí splnit tento požadavek pokud množství zájmového provozu tvoří maximálně 5 procent celkové kapacity linky. Zachycené pakety prochází všemi subsystémy sondy. Proto je nutné, aby byl na plné rychlosti linky síťový provoz nejen filtrován, ale současně byl zajištěn bezztrátový export dat na mediační funkci, kde je dlouhodobě ukládán na disk. Pokud by ke ztrátě dat došlo, musí být o tom zapsán záznam do logu.

7. Sonda musí být schopna zachytávat zájmové toky na plně saturované lince 1/10 Gbps. Pokud zájmový provoz tvoří maximálně 5 procent kapacity linky, nesmí být žádné pakety ztraceny.

Mediační zařízení udržuje stabilní spojení se sondou a může tak reagovat na výpadky napájení. To je realizováno jednak odesíláním udržovacích (keep alive) paketů směrem k mediační funkci, jednak periodickým ukládáním prázdných souborů, které signalizují funkční záchyt. Po výpadku spojení mezi sondou a mediační funkcí dochází k zotavení. Sonda využívá omezené buffery k zajištění bezeztrátového záchytu, který je, jak již bylo zmíněno, klíčový. Pokud výpadek nepřesáhne určitou mez, pakety jsou přechodně ukládány do bufferu a s určitým časovým zpožděním odesílány na mediační zařízení. S každým novým spojením je nutné synchronizovat množinu aktivních LIID. Aby nedocházelo ke ztrátě, před dokončením synchronizace jsou příchozí pakety ukládány jako nezařazené.

8. Je udržováno stabilní spojení mezi sondou a mediační funkcí. Při výpadku spojení se sonda musí pokusit o obnovení v 5s intervalech.

9. Po obnově připojení sonda odesílá pakety přechodně uložené ve frontě.

10. Sonda periodicky odesílá udržovací (keep alive) pakety na mediační zařízení za účelem ověření funkce spojení. Pokud pakety přestanou přicházet, vznikne záznam v logu na mediačním zařízení.

Aby bylo dosaženo naprosto přesného odposlechu síťové komunikace, využívá sonda hardwarové zpracování pomocí zřetězené linky. Komponenty zřetězené linky se konfigurují pomocí tzv. generických parametrů. Ty udávají kapacitu jednotlivých komponent a tím i maximální počet identifikátorů, podle kterých bude souběžně prováděn záchyt. Generické parametry lze měnit při sestavení firmware sondy, což umožňuje rychle reagovat na změnu požadavků PČR. Kapacitu lze také navýšit v případě, že by byl při testování zjištěn nedostatečný výkon filtrace na aplikační vrstvě.

Sonda musí být schopna provádět záchyt pro množinu identifikátorů (více aktivních odposlechů najednou). Pro identifikátory na síťové úrovni je počet zachytávaných IP adres stanoven na 32, což je dáno velikostí hardwarově implementovaného L3 filtru. K filtraci na aplikační úrovni se využívají komponenty Pattern Match a tabulka síťových toků (filtr podle IP adres a portů). Pattern Match rozeznává zájmové toky na základě regulárních výrazů, kterých může být současně definováno až 16 a každý odposlech na aplikační vrstvě může nastavit více výrazů. Pattern Match dále vkládá pravidla pro rozpoznané síťové toky do tabulky toků. Po naplnění tabulky (velikost odpovídá 4097 položkám) se nejprve odmazávají stará pravidla.

11. Sonda musí pracovat až do dosažení limitů hardwarových komponent.

    11.1 Filtrace vstupního síťového provozu podle až 32 IP adres. Exportovány jsou všechny síťové toky identifikovány těmito adresami.

    11.2 Podpora odposlechů s jedním nebo více L7 identifikátory. Součet regulárních výrazů vytvořených na základě identifikátorů není větší než 16. Všechny zájmové toky obsahující hledané identifikátory jsou exportovány.

    11.3 Na vstupní lince opakovaně přicházejí toky s neznámými L4 identifikátory (IP + port). Dokud toků není přibližně 4000, jsou zájmové pakety exportovány bez výjimky. Poté se z tabulky odmazávají nejdéle neaktivní toky.

Odposlouchávaná data jsou na transportní vrstvě přenášena protokoly TCP a UDP. Pakety mohou obsahovat různá zapouzdření hlaviček, data mohou být tunelovaná prostřednictvím IPv6 nebo jinak zapouzdřena (IP in IP). Sonda musí z těchto paketů správně extrahovat informace potřebné k identifikaci provozu.

Hlavní a jediné rozhraní, se kterým se počítá při práci v produkčním prostředí, je webové GUI. Rozhraní tak musí být úplné z hlediska nastavení funkce sondy a předání informací uživateli. Z GUI musejí být dostupné všechny provozní logy, stav síťových rozhraní a jejich statistiky (počet přenesených paketů/bytů). Dále musí být dostupná konfigurace sondy: nastavení IP adresy pro management, času, IP adresy mediační funkce při přenosu přes síť. Skrze GUI musí být možné sondu aktualizovat, změnit heslovou frázi pro šifrovanou část disku nebo spravovat uživatele a jejich přístup do systému. Nejdůležitější součástí uživatelského rozhraní je správa odposlechů. Zde jsou poskytovány informace o stavu odposlechu evidovaného v databázi a je možné nové odposlechy konfigurovat. Požadavky na editaci nebo zavedení nového odposlechu jsou předány administrační funkci (viz 3). Její odpověď (přidání odposlechu / odmítnutí z důvodu nesprávnosti údajů) je sdělena uživateli. GUI musí poskytovat vizuální notifikace, které upozorní na ztrátu spojení mezi sondou a MF, zaplnění disku na MF, neočekávané ukončení provozně důležitého procesu a podobně.

12. GUI je úplným rozhraním pro konfiguraci sondy/mediační funkce a k získávání informací o jejich stavu.

    12.1 Z GUI lze vyčíst provozní logy, stav a statistiku síťových rozhraní (počet přijatých, odeslaných a zahozených paketů/bytů).

    12.2 Musí být umožněna konfigurace sondy: IP adresa sondy, připojení k MF, nastavení času, správa uživatelů, změna heslové fráze pro šifrovaný disk, aktualizace systému.

    12.3 Musí být umožněna konfigurace MF: IP adresa MF, cesta k úložišti zachycené komunikace, dělení souborů po překročení velikosti.

    12.4 Správa odposlechů: vytvoření/editace odposlechu, zpětná vazba o chybách a stavu jednotlivých odposlechů (čekající, aktivní, ukončený).

    12.5 V GUI se zobrazují notifikace reagující na neočekávané události a varování o naplnění disku MF.

Statistiky síťového provozu jsou realizovány pomocí softwaru IPFIX probe. V tomto směru proto pro sondu platí požadavky definované v kapitole 5.1. Současně musí být agregované výstupy statistik integrovány do FlexProbe GUI. Výpis statistik je v GUI možné filtrovat dle položek v agregovaných síťových tocích, řadit nebo seskupit podle klíčového atributu.

13. Sonda provádí sběr statistik v podobě monitorování síťových toků (*flow export*).

    13.1 Statistiky jsou o provozu shromažďovány souběžně s prováděním odposlechů.

    13.2 GUI obsahuje pohled na statistiky monitorování síťových toků. Data je možné filtrovat, řadit nebo seskupit (agregovat) podle klíče.

Kapitola 6

Návrh systému testů

Následující text se zabývá návrhem systému testování sond pro monitorování síťového provozu. Hlavním výstupem je sada testů, která ověří požadavky uvedené v kapitole 5 u obou vyvíjených sond: IPFIX probe a FlexProbe. Druhým úkolem je navrhnout způsob, jakým budou testy automatizovaně spouštěny. K dosažení cílů je nutné zabývat se také specifiky testovacích dat a způsoby jejich získání, prostředím, ve kterém mohou být testy izolovaně spouštěny i nástroji pro automatizaci. Ačkoli je návrh určen pro dvě různá zařízení s odlišnými případy užití, je možné nalézt společné vlastnosti a při návrhu zachovat do jisté míry společný postup. Například nasazení sond v analyzované síti způsobem zapojení exportéru a kolektoru je prakticky totožné. Fyzické testovací prostředí proto může být použito jak pro testování sondy monitorující síťové toky, tak se sondou určenou k realizaci zákonných odposlechů. Ve funkčnosti sondy FlexProbe navíc dochází k překryvu záchytu provozu a sběru statistik. Zařízení proto musí být otestováno na korektní chování obou funkcí.

Pod testováním hardwarových sond je potřeba vidět komplexní systém testů různých úrovní, které pokryjí funkci jednotlivých modulů i celého zařízení. Na rozdíl od testování libovolného softwarového řešení nelze vždy na testy nahlížet jako na součást implementace, tedy část zdrojových kódů, které volají a instanciují jednotky systému za účelem ověření správné činnosti. Podsystémy sondy jsou často implementovány v různých programovacích jazycích, některé jsou pro vyšší efektivitu implementovány v hardware. Velká část testů proto na sondu nahlíží jako na tzv. *black box*: používají běžné rozhraní ke konfiguraci sondy, řídí vstupní provoz a ověřují výstup ze zařízení.

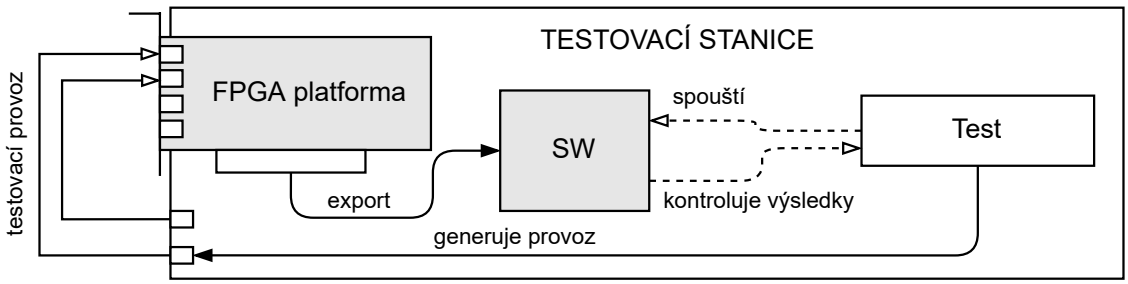
V kapitole jsou nejprve představeny dva navržené způsoby testování sondy. Liší se tím, že první přístup spouští testy na zařízení sondy a může tak vyčítat interní data a ovlivňovat chování. Druhý způsob používá zařízení sondy jako úplný *black box* a více se přibližuje produkčnímu nasazení. Různé způsoby testování jsou vhodné pro různé skupiny testů, které jsou popsány v následující podkapitole. Testovací sady seskupují testy podle jejich druhu, ale také podle funkce nebo scénáře použití sondy. Dále je navržen testovací framework, který nabízí nástroje k programování testů pro síťové sondy. Představeno je také fyzické prostředí, ve kterém budou sondy testovány. Následuje popis implementovaných testů pro sondy FlexProbe a IPFIX probe s důrazem na pokrytí požadavků definovaných v kapitole 5. Posledním tématem návrhu je automatizace testů. Jakým způsobem se řeší automatizované spouštění testovacích sad je popsáno z pohledu software Jenkins, který je pro automatizaci využit.

6.1 Způsoby testování

Pro testování síťových sond platí, že různé testovací sady vyžadují různé způsoby provedení testů. Část testů lze spustit při sestavení softwaru nebo provést pomocí simulace. Zbylé testy ale ke svému běhu vyžadují akcelerační FPGA platformu s funkčním firmware. Integrační testy jsou spuštěny nad částí systému a musí mít proto přístup k interním komunikačním rozhraním v sondě. Nejvyšší systémové testy naopak nahlížejí na sondu jako na black-box a chyby odhalují nepřímo na základě analýzy výstupů. Z toho vyplývá, že lze navrhnout několik scénářů, ve kterých budou testy uskutečněny a jim uzpůsobit testovací prostředí.

Dva odlišné způsoby testování lze odvodit z chronologických fází, kterými prochází vyvíjený systém. První fáze se týká sestavení software a firmware. Bezprostředně po sestavení je vhodné spustit jednotkové testy a provést verifikaci hardware za pomocí simulace. Úspěšný průchod testy by měl být základní prerekvizitou k vydání stabilní verze systému. Testy v této fázi nekladou nároky na testovací prostředí a obecně platí, že mohou být provedeny na stroji, kde došlo k sestavení a syntéze. Výsledkem sestavení je systémový obraz, který je nahrán do sondy nebo sada balíčků instalovatelných do operačního systému. Po zprovoznění nově sestaveného systému přichází druhá fáze, ve které dynamické testy interagují s fyzickým zařízením sondy. Zde přichází v úvahu dva způsoby zapojení testovacího prostředí. (i) Testovací stanice a sonda je jedno zařízení nebo (ii) je testovací stanice plně oddělená od sondy.

Způsoby testování fyzické sondy ve fázi po sestavení systému jsou ukázány na obrázcích 6.1 a 6.2. Zapojení a nárokům na testovací prostředí se podrobně věnuje kapitola 6.5.

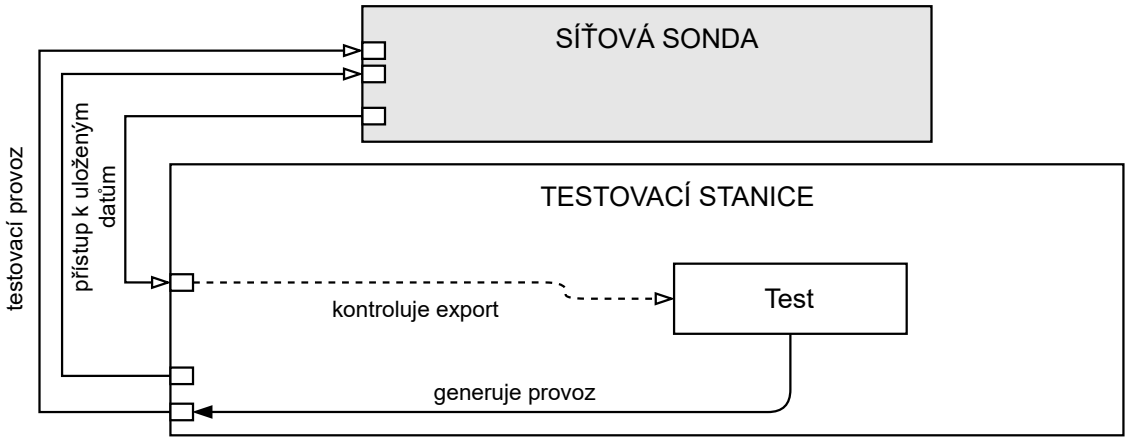


Obrázek 6.1: (i) způsob testování, kdy je sonda součástí testovací stanice. Šedou barvou je označeno SUT.

6.2 Testovací sady

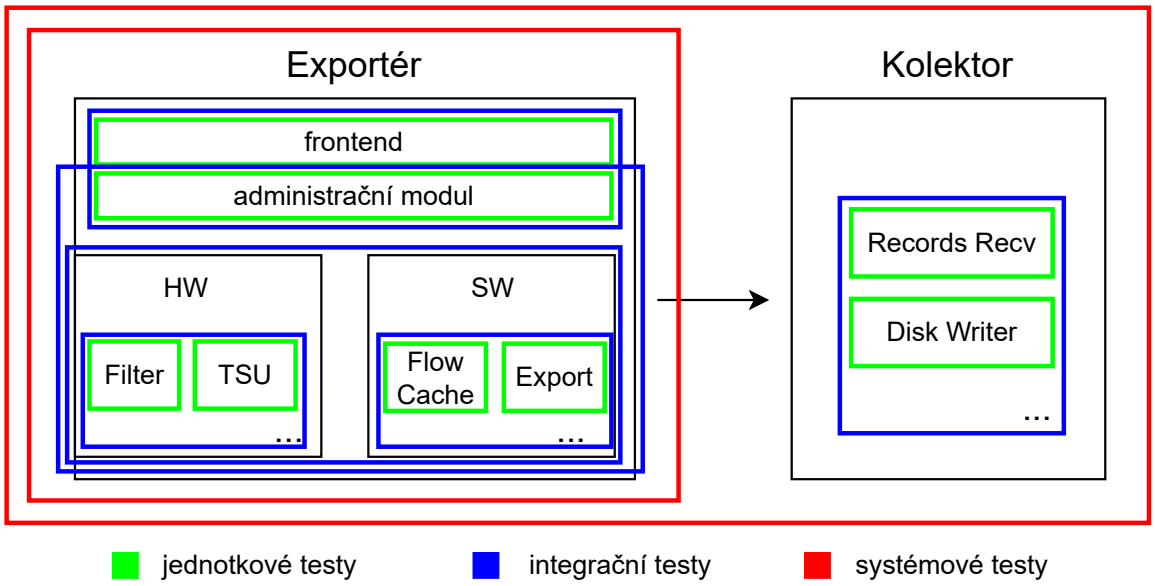
Přestože práce cílí na vytvoření ucelené testovací sady pokrývající všechny aspekty sondy, reálně bude testovací sada rozdělena. Menší testovací sady pokrývají různé aspekty specifikace požadavků, která je uvedena v kapitole 5. Zařazení testů do testovacích sad je dáno jejich druhem. Problematika klasifikace testů je popsána v kapitole 4.1. Navrženy jsou sady jednotkových testů, integračních, systémových, výkonnostních, ale také sady ověřující schopnost zotavení a funkci servisních činností.

Na obrázku 6.3 je vidět návrh systému testů pro modelovou sondu, která odpovídá oběma zařízením. Jak sondě pro zákonné odposlechy, tak sondě IPFIX probe. Obsah obdélníků, kterými jsou testy vyznačeny odpovídá ověřovaným částem systému – SUT. V tomto příkladu je považován exportér za samostatně funkční systém, proto je pro něj navržen systémový test. Způsobilost k takovému testu záleží na implementaci – musí existovat možnost



Obrázek 6.2: (ii) způsob testování, při kterém je sonda samostatné zařízení a výsledky jsou ověřeny nepřímo na základě uložených dat.

odklonu kolektoru a připojení testu přímo na exportní kanál. V případě FlexProbe i IPFIX probe je toto možné, nicméně je nutné dodržet zapojení, při kterém je testovací stanice a sonda jedno zařízení (viz obrázek 6.1 v předchozí kapitole).



Obrázek 6.3: Druhy funkčních testů v kontextu síťové sondy

Popis testovacích sad se nejprve bude zabývat sondou FlexProbe, pro kterou jsou navrženy jednotkové testy, testy hardwarových komponent, testy záchytu na síťové (L3) a aplikační (L7) úrovni i výkonnostní testy. Testovací sady pokrývají také grafické uživatelské rozhraní a tvorbu statistik síťového provozu v podobě monitorování síťových toků. Rozdělení testů vyplynulo z potřeby ověření funkce systému na různých úrovních, které přispívá k dobré lokalizaci chyb a přehledu o stavu implementace podsystémů. Zároveň jsou dodrženy scénáře použití, které určuje druh prováděného zákonného odposlechu.

První skupinou jsou testy jednotkové, které se zabývají ověřením správné implementace softwarových komponent. Jejich výhodou je verifikace ještě před sestavením celého systému a nejpřesnější lokalizace chyb. Vytvořit testy pro jednotky všech softwarových komponent by bylo náročné, proto byly pro realizaci vybrány pouze programy kritické pro záchyt komunikace. Jedná se o systém zajišťující export podle identifikátorů aplikačních protokolů (PaSt – *Packet Stack*) a software pro přijímání zachycených dat (*Dumper*). Jednotkové testy software jsou přímo přiloženy ke zdrojovým souborům, v obou případech v programovacím jazyce C++. Bezproblémový průchod jednotkovými testy je prerekvizitou k sestavení vydání systému sondy. Provedení testů je plánováno ihned po kompilaci, s čímž může pomoci nástroj pro automatizaci (viz kapitola 6.6).

Další testy jsou již prováděny nad zařízením sondy a je proto důležité specifikovat testovací případ tak, aby ověřoval danou vlastnost systému. Ačkoli jsou testovací data zpracována celým systémem sondy, můžeme se zaměřit na jednu skupinu vzorků provozu, pomocí kterého dostatečně otestujeme chování daného podsystému. Pro hardwarovou zřetězenou linku je navržena sada testů komponent, bez kterých by nebylo možné provést korektní záchyt. Hardwarové komponenty v testech vystupují jako jednotky zodpovědné za jednotlivé fáze zpracování provozu. Ověření se týká vstupního rozhraní, komponenty pro přidání časové značky (TSU), filtru síťových toků, analyzátoru hlaviček protokolů (HFE) a rozhraní mezi hardwarem a softwarem (dispatcher). Kromě základní funkcionality se testují také neobvyklé nebo méně zastoupené příklady provozu z důvodu ověření jejich podpory. Příkladem jsou pakety o velké velikosti (tzv. *jumbo*), různě zapouzdřené síťové protokoly (IP in IP) nebo tok s náhodně variabilní délkou paketů. Testy tak přímo pokrývají některé ze specifikovaných požadavků (5.2): požadavek č. 1.2 a 5.

Aby byly splněny hlavní požadavky (ze skupin 1, 2 a 3), které definují záchyt na síťové a aplikační vrstvě, byla navržena sada testů generující vzorky provozu s L3 i L7 protokoly. Vzhledem ke skutečnosti, že integrační i systémové testy mohou využívat stejná testovací data, jsou testy navrženy tak, aby plnily obě funkce. Záleží na parametrech spuštění těchto testů, pomocí kterých je možné ovlivnit, jak bude sonda konfigurována a jak bude vyzvedáván zachycený provoz. Integrační test nastavuje odposlech přímo v softwarových i hardwarových komponentách a zachycená data přijímá z exportního rozhraní. V takovém případě je testována součinnost hardwarové zřetězené linky a softwaru pro záchyt podle L3 nebo L7 identifikátorů. Ostatní podsystémy sondy se při integračních testech nevyužívají. Druhou možností je ovládat sondu prostřednictvím REST API nebo GUI, konfiguraci odposlechu přenechat administrační funkci a zachycený provoz přečíst z uložených PCAP souborů na disku. V druhém případě vystupuje test jako systémový. Testy fungují na principu generování testovacího provozu, který obsahuje jak pakety zájmové, tak ostatní provoz. Jinými slovy komunikaci, ve které se vyskytl odposlouchávaný identifikátor a ostatní provoz (šum). Na konci testu se ověřuje, že byly exportovány pouze pakety zájmového provozu.

V reálném provozu je důležitou vlastností dlouhodobá stabilita zařízení. Po síti mohou přicházet porušené pakety, síťové útoky nebo velké množství zájmové komunikace, která vytěžuje prostředky sondy. Testy generující provoz v delším časovém intervalu si berou za cíl ověřit, že je sonda schopna pracovat i v situaci, která se přibližuje nasazení do reálné sítě. Pozornost je věnována korektnímu uvolňování paměti, opakovanému zachycení toků se zájmovým identifikátorem a zároveň ověření, že nejsou zachytávána data, která neobsahují odposlouchávaný identifikátor ani při zaplnění paměti komponent. Testy zotavení jsou navrženy k cílenému zahlcení paměti filtru síťových toků, při kterém se zkoumá reakce systému. Zahlcení je provedeno neustálým generováním nových toků. Očekávaná reakce sondy

je pravidelné odmazání již neaktivních toků (30 minut bez příchodu paketu), při přeplnění paměti filtru se jedná o eliminaci aktivních toků s úsilím o minimalizaci ztrát.

Kromě stability sondy a správné funkce je klíčovým parametrem sondy také propustnost. Pro zjištění výkonových parametrů musí test odesílat velké množství provozu, aby byla saturována odposlechová linka. Srovnáním počtu zachycených paketů s referenčním se následně určí, zda sonda zpracovala veškerý síťový provoz a výsledek zpracování je správný. Nejjednodušším způsobem získání testovacího provozu je jeho vygenerování a uložení do PCAP souborů. Výkonnostní test načítá syntetický provoz z disku a reprodukuje jej s požadovanou rychlostí. Testování probíhá odděleně pro záchyt na L3 a L7 vrstvě. Pro vysokorychlostní odesílání paketů na vrstvě L3 se používá nástroj TRex [2]. Provoz je generován softwarově (TRex server využívá Linux DPDK) a není tedy potřeba využívat specializovaný hardware. Výstupní metrika L3 výkonových testů je počet zahozených paketů. Testuje se závislost ztráty paketů na velikosti paketů v tocích a také na množství zájmového provozu. Pro splnění požadavku č. 7 se očekává záchyt beze ztrát do 5 % zájmové komunikace.

Výsledkem výkonnostních testů záchytu dle identifikátorů aplikačních protokolů je maximální rychlost zpracování vstupního provozu. Připravené jsou PCAP soubory se zájmovou komunikací přenášenou protokoly SMTP, FTP, IMAP, POP3 a SIP. Testují se pakety o velikosti 1, 10, 100 kilobytů s poměrem zájmového provozu 1 % a 10 %. Pakety jsou odesílány programem Tcpreplay, který v případě využití knihovny Netmap[1] umožňuje přehrávání PCAP souborů v rychlostech v řádu gigabitů za sekundu. Hledání maximální rychlosti zpracování pro daný vzorek provozu probíhá binárním půlením intervalu. Pokud při záchytu došlo ke ztrátě, rychlost je o polovinu aktuálního intervalu snížena. V případě, že byly všechny pakety úspěšně zachyceny, je rychlost o polovinu intervalu zvýšena. Testování probíhá, dokud je interval větší než nastavená mez.

Pro software IPFIX probe jsou navrženy dvě testovací sady, které obsahují jednotkové a systémové testy. Jednotkové testy jsou napsány v jazyce C++ a používají framework GoogleTest[2]. Svou činností pokrývají pomocné funkce, jako je přečtení argumentů příkazové řádky, nástroje pro operace s řetězci, ale také třídu pro komunikaci s pamětí toků. Hlavní testovací sada obsahuje systémové testy, které ověřují primární funkci IPFIX probe – agregaci síťových toků pomocí procesních zásuvných modulů. IPFIX probe je software, který umožňuje načítání vstupního provozu z PCAP souboru a jeho výstup je taktéž možno uložit do souboru. Toho je využito ve funkčních testech, které tak mohou být spuštěny na libovolném počítači. Průběh testu spočívá ve spuštění zkompilovaného programu `ipfixprobe` s argumenty, které zajistí načtení testovacího PCAP souboru (např. pro provoz http, smtp atd.) a uložení výstupu ve formátu UniRec. Výstupní soubor s agregovanými toky je následně srovnán s referenčním výstupem. Obsahy obou souborů musí být shodné. Referenční výstup je při tvorbě testu vytvořen ručně pomocí analýzy testovacího PCAP souboru.

Při testování je nutné ověřit správnou integraci IPFIX probe do sondy FlexProbe, kde software zajišťuje statistickou funkci. Pro tyto účely je navržena sada statistických testů pro sondu FlexProbe. Princip testování je velmi podobný funkčním testům IPFIX probe: také se využívají anotované PCAP soubory a ověřuje se výstup uložený na disku. Rozdíl je v načítání dat, které zde probíhá jako u ostatních funkčních testů FlexProbe zasíláním provozu na fyzické vstupní rozhraní. V operačním systému sondy FlexProbe běží jak IPFIX exportér, tak IPFIX kolektor. Data jsou uložena na disk kolektorem do formátu JSON, kde mohou být jednoduše srovnány s anotací. Zapojení testovacího prostředí pro statistické testy odpovídá obrázku 6.1.

[1] http://info.iet.unipi.it/~luigi/netmap/
[2] https://github.com/google/googletest

6.3 Testovací případy a pokrytí požadavků

Návrh se nyní dostává na nižší úroveň testovacích sad – zaměří se na samotné testy. Analýzou činnosti testu, zejména ve fázi ověření, se rozlišuje, které požadavky z kapitoly 5 test pokrývá. Pokrytí požadavků je podstatné při interpretaci celkového vyhodnocení testování, kdy stanovujeme, z kolika procent je sonda otestována. Požadavek je pokryt buď přímo nebo nepřímo. V případě nepřímého pokrytí vynucuje splnění požadavku test, který je implementován za účelem ověření odlišné funkcionality.

Nejprve jsou rozebrány požadavky na sondu IPFIX probe, přičemž se postupuje vzestupně podle číselného označení požadavků. Některé požadavky ověřujeme v rámci integrace do sondy FlexProbe. Vzhledem k tomu, že je software IPFIX probe použit beze změn hlavní funkcionality, požadavky se považují za obecně pokryté.

První skupina požadavků se zabývá příchodem paketu a předáním k softwarovému zpracování. Požadavky jsou ověřovány nepřímo za pomoci funkčních testů. Funkční test nechává zpracovat PCAP soubor softwarem IPFIX probe a výstup srovnává s manuální anotací. V případě, že by byl některý paket změněn (požadavek 1.2), zahozen (1.3) nebo by neobsahoval časovou značku (1.4), projevila by se chyba na výstupu. Funkční testy ověřují pouze načítání provozu ze souboru a netestují limity zpracování rámců síťovou kartou (požadavek 1.1). Napojení na hardwarovou platformu FlexProbe, jakožto další podporované rozhraní softwarem IPFIX probe, se nepřímo testuje v rámci statistických testů FlexProbe. Taktéž jsou v této konfiguraci přímo ověřeny limity zpracování paketů (1.1) testem hardware.

Požadavky z kategorie 2 a 3 opět nepřímo ověřují funkční testy. Požadavky se týkají extrakce klíčových atributů z paketu a vytvoření hash síťového toku. Bez korektní činnosti těchto systémů by se toky špatně agregovaly nebo by jejich záznamy úplně chyběly, což by se projevilo při srovnání s anotací. Požadavek na extrakci atributů IP vrstvy z nejhlouběji zapouzdřené hlavičky (2.1) je konkrétně ověřen testem `ip-in-ip.sh`. Invarianci hash vůči směru komunikace pokrývají testy standardních zásuvných modulů, neboť anotované PCAP soubory obsahují oboustrannou komunikaci. Vetší množství toků typu *biflow* lze očekávat na agregovaném výstupu testu `basicplus.sh`.

Funkce paměti toků (*flow cache*) je pokryta přímo jednotkovými testy. Testy postupně ověřují požadavky 4.1 až 4.4 na instanci třídy `NHTFlowCache`, která se používá jako standardní paměť toků. V případě změny zásuvného modulu implementující paměť toků je v testu možné objekt nahradit nově implementovaným. Chování by se nahrazením nemělo změnit.

Hlavní disciplínou několikrát zmiňovaných funkčních testů, které mají podobu systémových testů, je ověření požadavků ze skupiny s číslem 5. Pro každý standardní procesní zásuvný modul existuje jeden funkční test. Zásuvné moduly se testují vždy odděleně – ostatní moduly nejsou aktivovány. Požadavky 5.1 až 5.3 pokrývá test `basic`, při kterém jsou vypnuty všechny rozšiřující funkce a testuje se pouze agregace základních hodnot. Velká skupina zásuvných modulů přidává funkci extrakce metainformací z různých aplikačních protokolů. Testy takových modulů používají PCAP soubory obsahující komunikaci pomocí konkrétního aplikačního protokolu a anotace s přesnými hodnotami metadat. Chybná agregace je odhalována na základě rozdílu výstupních záznamů ze sondy a anotace. Z rozdílu lze v některých případech určit, zda se jedná o chybu zabudované agregace, rozšiřujícího zásuvného modulu nebo jiné funkce exportéru. Pokud jsou například správně agregované základní vlastnosti toku (počet paketů, čas prvního paketu apod.), ale přidaná pole s metadaty protokolu *http* nesou špatné hodnoty, je vysoce pravděpodobné, že nastala chyba

agregace v zásuvném modulu `http`. Pokud záznam toku úplně chybí, je možné, že jeho pakety vůbec nedorazily do software nebo nebyly korektně extrahovány klíčové hodnoty.

Správa procesních zásuvných modulů definovaná požadavky s číslem 6 je pokryta jednotkovými a funkčními testy. Požadavek na možnost rozšíření záznamu toku (6.1) ověřuje test `flowifc.cpp`. Komponenta `flowifc` má na starost komunikaci s pamětí toků a definuje třídu záznamu síťového toku, který je rozšířitelný. Notifikace zásuvného modulu (6.2 až 6.4) je pokryta jednotkovým testem `plugin-management.cpp`. Jelikož žádný produkční zásuvný modul neposkytuje podrobné informace o volání svých metod, byla vytvořena zástupná třída (tzv. *mock* [15]), která implementuje rozhraní zásuvného modulu. Test přidává pakety do instance paměti a objekt testovacího zásuvného modulu sleduje, jestli jsou volány všechny očekávané metody.

Dodržení exportních protokolů IPFIX a UniRec (požadavek 7) je kontrolováno nepřímo na základě výstupu funkčních testů IPFIX probe a statistických testů FlexProbe. Funkční testy využívají přímé uložení UniRec záznamů do souboru, které je provedeno exporterem. Statistické testy FlexProbe využívají IPFIX kolektor, který přijímá data od exportéru ve formátu IPFIX.

Požadavky na sondu FlexProbe ověřujeme podobným způsobem. Zpracování paketů na nízké úrovni, jehož detailům se specifikace příliš nevěnuje, je ověřováno hardwarovými integračními testy nebo nepřímo systémovými.

Testy sondy FlexProbe primárně využívají navržený testovací framework (6.4), a proto mohou být použity jako integrační nebo systémové podle aktuální konfigurace. Základní způsob nepoužívá mediační funkci (kolektor) z důvodu ověření exportu bez ovlivňujících faktorů. Nejjednodušším testem je test přidání odposlechu, který konfiguruje jeden odposlech s L7 identifikátorem. Zájmový identifikátor se následně nachází v každém testovacím paketu. V intervalu 1 minuty jsou pakety posílány do sondy a ověřuje se jejich export. Pro úplné pokrytí požadavku č. 1.1 je nutné vyzkoušet, zda sonda exportuje pouze toky se zájmovým identifikátorem a ostatní zahazuje. Toto chování je ověřeno v testu záchytu dle IP adresy a také v každém testu zaměřeném na konkrétní aplikační protokol. Ověření probíhá způsobem, kdy test do sondy odesílá skupinu paketů, které nemají být exportovány. Následně se jejich zahození explicitně kontroluje. Exportován nesmí být takový paket, který neobsahuje zájmový identifikátor a zároveň jeho síťový tok v minulosti nebyl označen za zájmový, nebo byl jeho tok ukončen.

Pokud se test nakonfiguruje jako systémový, připojuje se k mediační funkci a ověřuje správnou činnost exportního kanálu (požadavek 1.3). Mediační funkce také zajišťuje zahození paketů podle časové značky tak, aby zachycená data přesně odpovídala povolenému odposlechu. Proto musí být požadavek 1.4 pokryt systémovým testem.

Sada tzv. autotestů ověřuje záchyty podle všech podporovaných identifikátorů na úrovních L3 i L7 (požadavky ze skupin 2 a 3). Do sady jsou přidávány regresní testy reagující na objevené chyby. Testy tak obsahují reprezentativní vzorek síťového provozu, který je sonda schopna zpracovat. Při testech aplikačních protokolů se testuje správná kooperace hardwaru, který provoz předzpracovává a softwaru provádějící podrobnou analýzu. Důraz je proto kladen na případy, při kterých dochází k falešně pozitivním detekcím (tzv. *false alarm*). Falešná detekce nastává v situaci, kdy paket obsahuje odposlouchávaný identifikátor, ale v nekorespondujícím aplikačním protokolu. Hardware paket propustí a software ho musí vyřadit.

Přiřazení časové značky je realizováno čistě v programu FPGA. Požadavek č. 5 proto pokrývá test hardwarové komponenty TSU. Testuje se jak samotné přidání značky, tak správná inkrementace a přiložení do exportovaných dat. Test funguje na principu generování

unikátních paketů, které zasílá do sondy v jasně daném pořadí. Exportované pakety musí po seřazení dle časové značky tvořit identickou posloupnost, jaká byla zaslána.

Pro ověření unikátnosti LIID a správné zařazení paketů (požadavek 6) není vytvořen test. Vlastnost je nicméně nepřímo ověřena systémovými testy, které stahují exportované soubory ze složek pojmenovaných dle LIID. Pokud by mechanismus nefungoval, test by exportované pakety nenalezl. Současně by byla odhalena záměna LIID díky kontrole zachycených dat po jednotlivých odposleších s unikátním LIID.

Požadavek s číslem 7 se zabývá výkonem sondy FlexProbe. Princip výkonnostních testů je podrobně popsán v kapitole 6.2. Metriky vyhodnocující výkon záchytu podle L3 i L7 identifikátorů (rychlost, procento zájmového provozu) poskytují dostatečné informace k ověření požadavku na zpracování provozu na plné rychlosti linky do 5 % zájmového provozu.

Funkci spojení mezi sondou a mediační funkcí popisují požadavky 8 a 9. Splnění požadavků ověřuje test `reconnect.py`. Test se zaměřuje zejména na opětovné odeslání exportovaných paketů po obnovení připojení. Provedení testu je možné výhradně se zapojením hardwarové platformy FlexProbe přímo do testovací stanice (viz schema 6.1). Test implementuje jednoduchý kolektor, který může v průběhu deaktivovat a simulovat tak odpojení od mediační funkce.

Limity sondy FlexProbe jsou ověřeny pro 32 IP adres (11.1) a maximální počet pravidel v hardwarovém filtru (11.3). Kapacita a funkčnost hardwarového filtru je testována zátěžovými testy. K cílenému přeplnění filtru existují dva testy, které se liší rychlostí, jakou filtr zaplní. Princip obou testů spočívá v iterativním generování zájmových síťových toků, které se v provozu objevují poprvé. To znamená, že pro každý příchozí paket musí být zaneseno nové pravidlo do filtru. Druhá fáze iterace ověřuje, která pravidla ve filtru zůstala. Pravidlo ve filtru přetrvalo, pokud je exportován paket z korespondujícího síťového toku. Kapacitu filtru rychleji zaplňuje test `avalanche.py`, který exponenciálně navyšuje počet nově generovaných toků v každé iteraci (časový interval okolo 1 sekundy). Korektní chování sondy se projevuje tak, že jsou v iteracích s počtem toků nižším než je kapacita filtru exportovány všechny toky. Následně jsou vyřazovány nejstarší viděné toky. Zároveň sonda udržuje velké zaplnění filtru a nedochází k nadměrnému vyřazování.

Druhým zátěžovým testem je `periodic.py`, jehož cílem je ověřit, že toky s častějším zasíláním paketů budou ve filtru zůstávat déle. Aktuálně software upřednostňuje udržení toků, u kterých je v posledních 30 minutách potvrzená aktivita. Zároveň by neměly být vyřazeny nové toky s neznámou aktivitou. Na rozdíl od testu rychlého zaplnění je filtr přeplňován pozvolně, což se přibližuje situaci, která může nastat v běžném provozu. Čas zaslání prvního paketu se zájmovým identifikátorem odpovídá indexu toku a stejně tak periodě, se kterou jsou zasílány ověřovací pakety. Paket prvního toku je odeslán v první sekundě testu a jeho pakety budou zasílány s periodou 1s, vysílání druhého toku začne v čase 2s a pakety budou zasílány s periodou 2s atd.

Testy grafického uživatelského rozhraní, které je popsáno požadavky ze skupiny 12 jsou integrovány přímo do testovacího frameworku 6.4. Jedna z implementací třídy pro konfiguraci sondy přímo interaguje s GUI. Uvedení sondy do požadovaného stavu je následně ověřeno provedením testu. Pokrytí GUI není úplné, protože se některé činnosti u funkčních testů neprovádějí (typicky konfigurace z požadavku 12.2).

6.4 Testovací framework

Jedním z nejdůležitějších aspektů návrhu systému testů je použití testovacího frameworku. Testovací framework je soubor nástrojů, který zjednodušuje tvorbu testů a řídí jejich spouš-

tění. Nabídnout by měl snadnou implementaci běžných fází testu jako je *set-up*, *exercise*, *verify*, *teardown* a způsob, jakým testy seskupovat do testovacích sad. S implementací testu souvisejí funkce, pomocí kterých se provádí ověření správné činnosti SUT, tzv. *assert*. K testování síťových sond je vhodné framework rozšířit o specifické funkce, díky kterým bude programátor řídit testování hardwarového zařízení síťové sondy.

Průběh jednoho testu je složen z několika fází. Nejprve je sonda konfigurována pro správný export dat – připojení na kolektor. V případě spuštění záchytu komunikace jsou do sondy nahrány zájmové identifikátory. Následně test generuje nebo načítá předem uložený provoz, který musí být odeslán na síťové rozhraní sondy. Po uplynutí časového intervalu, který odpovídá zpracování příchozího provozu, jsou exportovaná data vyzvednuta a je ověřena jejich správnost. Zkontrolována mohou být také metadata, která sonda poskytuje. Například čítače zpracovaných, zahozených a exportovaných paketů/toků.

Testovací framework by měl nabízet minimálně tři funkcionality plynoucí z průběhu testu, kterými jsou konfigurace sondy, zasílání provozu a vyzvedávání exportovaných dat. Každá z těchto funkcí musí být připravena pro sondu FlexProbe i IPFIX probe. Implementací provádějící funkci přitom může být více. Už ze dvou způsobů zapojení (6.1 a 6.2) vyplývá, že vyzvedávání dat je nutné řešit odlišně. V jednom případě test přímo přijímá data jako kolektor, v druhém případě stahuje data z disku. Je proto vhodné, aby byl testovací framework sestaven z abstraktních tříd (rozhraní), které reprezentují komponenty určené k provedení funkce. Implementace rozhraní je následně zaměnitelná a test je proto možné nakonfigurovat na různé způsoby testování. Díky abstraktním třídám také může být testovací framework jednoduše rozšířen, např. implementováním konfiguračního rozhraní jiné síťové sondy nebo napojením na hardwarový generátor provozu.

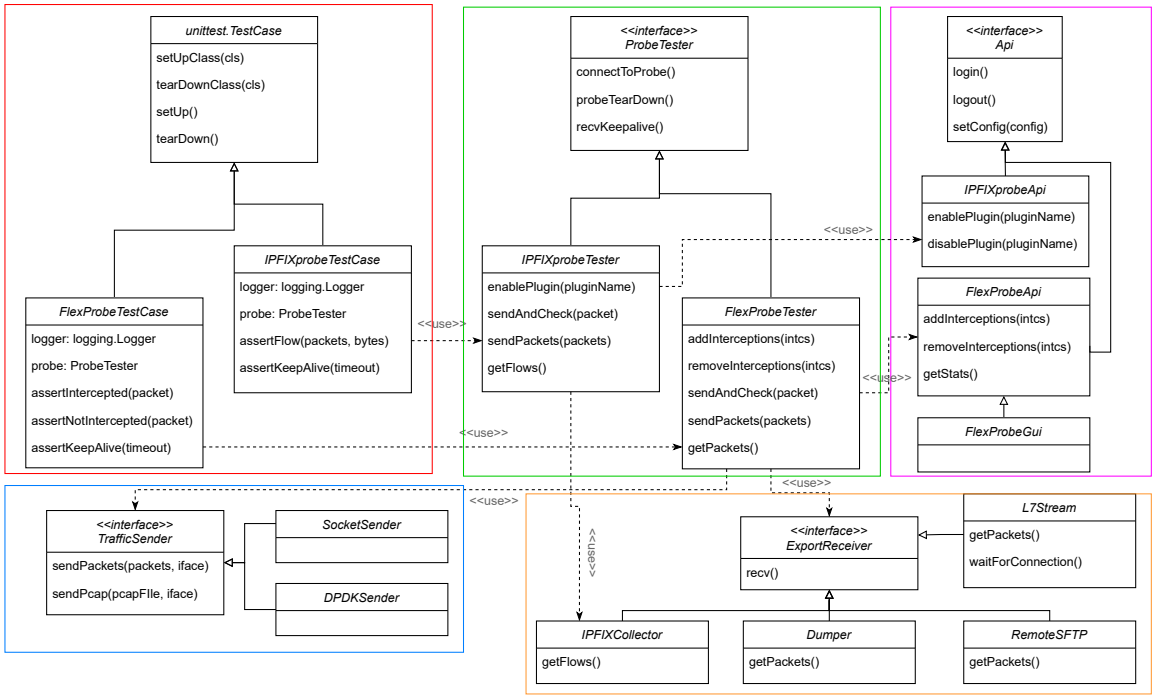
Navržený testovací framework operuje ve druhé fázi vývoje (viz sekce 6.1), kdy je systém již sestaven a testuje se hardwarové zařízení sondy. Je proto možné, aby vznikal zcela odděleně od provedené implementace. Volba programovacího jazyka pro framework závisí na dostupnosti knihoven, které budou použity pro odesílání a přijímání síťového provozu a vzdálenou konfiguraci sondy. Pro svou jednoduchost a produktivitu se jako standard k tomuto typu testování využívá jazyk Python, pro který jsou volně dostupné potřebné knihovny. Standardní modul Socket[3] umožňuje používání soketů na úrovni operačního systému. Knihovna Scapy[4] nabízí jednoduchý způsob sestavení paketu a může být proto použita pro generování testovacího provozu. Pro ovládání softwaru TRex [2], který slouží k vysokorychlostnímu generování a odesílání síťového provozu, je v jazyce Python implementován klient. Dalším důvodem k použití jazyka Python je, že navržené testy vycházejí z prvotní verze testů sondy FlexProbe, kde se k řízení testů Python využíval.

Pro jazyk Python existují knihovny implementující základní funkcionalitu testovacího frameworku. Jedním z nich je unittest [14], který je dokonce součástí standardní knihovny a je tedy distribuován s interpretem jazyka Python. Unittest byl vybrán jako základ pro navrhovaný framework určený k testování síťových sond. V základu se jedná o jednoduchou knihovnu pro tvorbu jednotkových testů. Třídy reprezentující testovací případy v unittest tvoří testovací sady, které mohou být následně automatizovaně spouštěny v kaskádě. Vyhodnocení běhu testu probíhá pomocí metod typu *assert*, jejichž chování je konfigurovatelné. Typicky při nesplnění ověření test selže, druhou možností je ale očekávaný neúspěch, při kterém se chování obrátí. Různými způsoby také může testovací sada reagovat na neúspěšný test: bude pokračovat nebo se testování okamžitě ukončí. Unittest dále spravuje argumenty

[3] https://docs.python.org/3/library/socket.html
[4] https://scapy.net

příkazové řádky a umožňuje pomocí nich vybrat testy ke spuštění, spojit testovací sady, nastavit styl výpisu chyb a podobně.

Na obrázku 6.4 se nachází návrh testovacího frameworku v podobě diagramu tříd. Třídy jsou rozděleny do barevných tématických skupin podle funkce, kterou zajišťují. Kromě pomocných tříd pro testovací případ a řízení testu se tak na obrázku nachází třídy zajišťující konfiguraci sondy, odesílání a přijímání testovacího provozu. Jejich implementaci může využít program testu k definovaným interakcím se sondou.



Obrázek 6.4: Testovací framework pro sondy FlexProbe a IPFIX probe

Třídy implementující rozhraní `ProbeTester`, které jsou vyznačeny zelenou barvou zajišťují celý proces testování. Pomocí `IPFIXprobeTester` nebo `FlexProbeTester` se přistupuje k metodám zbylých tříd. `ProbeTester` je zodpovědný za konfiguraci sondy, posílání testovacího provozu a přijetí/vyzvednutí exportovaných dat. Algoritmus musí reflektovat specifika hardwarových sond. Jednou z nich je nutnost vyčkat na aplikaci pravidel do FPGA komponent. V opačném případě by nebylo garantováno správné zpracování provozu. Objekt `ProbeTester` tak musí ověřit, že je sonda připravena přijímat data. Ověření se řeší například pomocí testovacího paketu (tzv. *probing packet*), který je periodicky odesílán, dokud se nedetekuje podle výstupu aplikování zadané konfigurace. Celý průběh testu vypadá následovně: v metodě testovacího případu je vygenerován testovací provoz a konfigurace sondy, vygenerovaná data jsou předána instanci s rozhraním `ProbeTester`, která si přebírá řízení. `ProbeTester` konfiguruje sondu a vyčká na propagaci konfigurace. Jakmile je konfigurace nastavena, posílá testovací provoz. Po určité době (ručně zvolený časový limit) vyzvedne exportovaná data ze sondy a vrací je zpět do metody testu. Zde po převzetí řízení proběhne ověření výstupů.

Červená skupina tříd se přímo napojuje na unittest. Dvojice tříd `FlexProbeTestCase` a `IPFIXprobeTestCase` dědí třídu `unittest.TestCase`, což je základní abstraktní třída pro tvorbu testovací sady. Stejně jako v celém testovacím frameworku je navržena varianta

třídy pro FlexProbe a IPFIX probe. Při psaní testu programátor zdědí jednu z těchto tříd a využije implementované části. Podtřída, která již obsahuje implementaci jednoho nebo více testů, získá nástroj pro formátování výpisu (`logger`) a referenci na objekt typu `ProbeTester`. Třídy testovacích případů také obsahují speciální metody typu *assert* pro jednoduché ověření, např. že byl paket zachycen. Složitější způsoby testování (např. časová souslednost, nestandardní IPFIX pole, apod.) musí programátor implementovat sám s daty poskytnutými od instance `ProbeTester`.

Fialová skupina tříd slouží ke komunikaci s veřejným rozhraním sondy. Funguje jako klient REST API nebo implementuje jiný způsob, kterým lze na sondě nastavit parametry, např. příkazy odeslané prostřednictvím SSH. Primárně jde o konfiguraci sondy, třída však může být použita také k vyzvednutí provozních hodnot a logů. Takto získané informace, pro příklad čítače síťových rozhraní nebo zahozených paketů, mohou být použity v ověřovací fázi testu. Speciálním případem klienta je třída `FlexProbeGui`, která byla vytvořena za účelem provedení GUI testů nebo nejvyšších systémových testů. Třída implementuje všechny metody `FlexProbeApi` a může být proto použita rovnocenným způsobem. Veškerá konfigurace však probíhá prostřednictvím grafického uživatelského rozhraní, což je ekvivalentní s produkčním použitím sondy FlexProbe. Pro interakce s GUI je využita knihovna Selenium[5], která provádí skutečné akce v instanci webového prohlížeče.

Následující skupinu tříd využívá `ProbeTester` k odesílání testovacího provozu na monitorovací rozhraní sondy. Na obrázku jsou třídy označeny modře. Oddělení funkcionality do abstraktní třídy bylo nutné k zajištění variability pro různé druhy testů. Obzvláště pro výkonnostní testy totiž není vhodné používat na odesílání paketů sokety operačního systému, které nedosahují požadované rychlosti v desítkách až stovkách gigabitů za sekundu a příliš zatěžují paměť RAM. Pro vysokorychlostní odesílání dat se proto používá systém DPDK, který implementuje třída `DPDKSender`. Pro použití DPDK musí být síťová rozhraní připojená (tzv. *binded*) pomocí jiných ovladačů než při běžném použití s OS. Přepojení by měl objekt `TrafficSender` zajistit při konstrukci.

Poslední (oranžovou) skupinou je sada tříd pro příjem nebo čtení exportovaných dat. Zde záleží zejména na způsobu zapojení síťové sondy a tedy, jestli exportovaná data mohou být přímo přijata testem nebo jsou nejprve uložena na disk sondy, ze kterého musí být následně stažena. Vzhledem k více variantám je opět implementace provedena modulárně s využitím rozhraní `ExportReceiver`. Pro sondu FlexProbe se možnosti větví na `L7Stream`, který přijímá zachycený provoz přímo z exportní linky sondy a `Dumper`. Dumper je název produkční aplikace, která plní funkci mediačního zařízení (viz 3.1).

6.5 Testovací prostředí

K provedení testů, které využívají hardwarovou sondu je vhodné zapojit stálé testovací prostředí, tzv. *testbed*. Nejdůležitějšími vlastnostmi testovacího prostředí jsou dostupnost (prostřednictvím internetu nebo lokální sítě) a stabilita (zálohované napájení, automatický start, stabilní síť). Předpokládá se totiž vzdálená obsluha a v horším případě nemusí být prostředí pro testera fyzicky dostupné.

Základní komponenty testovacího prostředí jsou síťová sonda a testovací stanice. Pokud není v sondě integrován kolektor, musí se pro běh systémových testů k zařízením přidat. Testovací stanice řídí běh testování a je tedy bodem, na který se vzdáleně připojuje uživatel (tester) nebo systém pro automatizaci. Sonda je s testovací stanicí propojena linkou, po

[5] https://www.selenium.dev/documentation/

které se posílá testovací provoz a linkou pro konfiguraci a stažení exportovaných dat. Pokud je sonda součástí testovací stanice, linka s testovacím provozem propojuje stanici samu se sebou.

Hardware testovací stanice musí umožňovat vysokorychlostní zasílání síťového provozu, které je realizováno pomocí specializovaných síťových karet. Vysokorychlostní síťové karty využívají systém DPDK, díky kterému je možné obejít operační systém a dosáhnout rychlosti odesílání v desítkách až stovkách gigabitů za sekundu. Karty jsou připojovány pomocí PCI-Express a jejich plný potenciál je využit při výkonnostních testech, kdy generovaným provozem plně saturují testovací linku. Sonda v případě integrace do testovací stanice také využívá slot PCI-Express. Testovací stanice musí být dále připojena do sítě pro vzdálený přístup (internet) a do sítě, ze které je dostupná sonda pro konfiguraci v případě, že je realizována jako samostatné zařízení. Na základní desce testovací stanice by se tedy měly nacházet alespoň 2 sloty PCI-Express a 2 ethernetové porty, u kterých dostačuje rychlost 1 Gb/s.

Softwarové vybavení testovací stanice musí umožňovat spuštění testů (instalované potřebné knihovny pro Python), ale také by mělo umožňovat překlad software sondy (překladač jazyka C++ a potřebné knihovny, např. Qt). Je totiž pravděpodobné, že bude testovací stanice využita pro sestavení programu ze zdrojových kódů určených k otestování. K automatizaci testů (viz kapitola 6.6) musí být testovací stanice nainstalována jako tzv. agent. To znamená, že na pozadí OS běží program komunikující s hlavním serverem automatizačního systému, pomocí kterého pak mohou být vzdáleně spouštěny libovolné procesy. Vzdálený přístup probíhá pomocí protokolu SSH. Agent systému Jenkins, který je používán v tomto projektu, vyžaduje funkční instalaci interpreta programovacího jazyka Java (např. volně dostupné OpenJDK[6]).

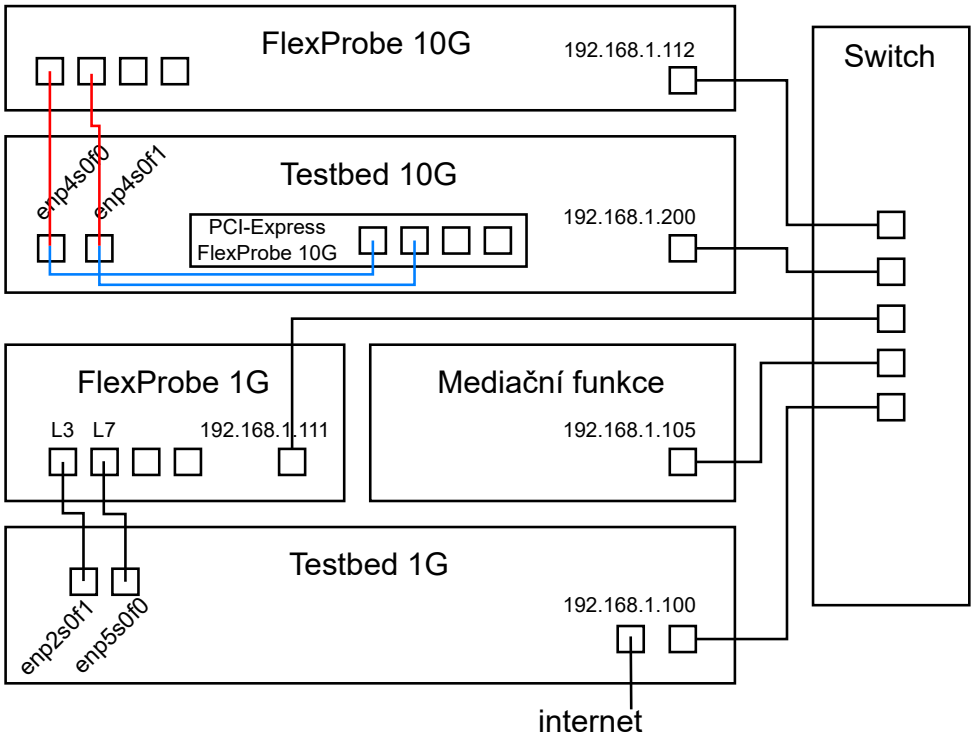
Ke spouštění testů, které generují testovací provoz a využívají raw soket, ale také pro připojení rozhraní k ovladačům DPDK je potřeba příkazy spustit jako administrátor. V prostředí operačního systému Linux existuje pro vykonání programu s administrátorskými právy nástroj `sudo`. Oprávnění k použití `sudo` by měly mít všechny uživatelské účty určené k testování sondy. V případě účtu, který používá automatizační agent je vhodné nastavit operační systém tak, aby při použití `sudo` nevyžadoval zadání hesla, což je výchozí chování. Toho lze dosáhnout přidáním konstanty `NOPASSWD` k danému uživatelskému účtu v souboru `/etc/sudoers`.

Pro účely testování sondy FlexProbe bylo postaveno testovací prostředí v laboratoři L311 FIT VUT v Brně. Schéma prostředí se nachází na obrázku 6.5. Jsou zde zapojeny obě varianty sondy FlexProbe – s rychlostí zpracování provozu 1 gigabit a 10 gigabitů za sekundu. Exportér IPFIX probe je testován jako součást sondy FlexProbe 10G. Testy ovládají dvě testovací stanice z důvodu nedostatečného počtu PCI-Express pro testování obou sond jednou stanicí. Část pro gigabitovou sondu se skládá z testovací stanice, sondy a mediačního zařízení, která je v případě starší verze oddělená. V druhé testovací stanici je zapojená sonda FlexProbe 10G a realizuje zapojení z obrázku 6.1. Pokud je potřeba provést systémové testy s produkčním zapojením sondy (6.2), musí se ručně přepojit linka testovacího provozu do zařízení FlexProbe.

Všechna zařízení v testovacím prostředí jsou zapojena do jedné lokální sítě, která je využita také pro konfiguraci sond a vyzvedávání/export zachyceného provozu. Testovací prostředí disponuje vzdáleně ovladatelnou zásuvkou, pomocí které lze odpojit od energie jednotlivá zařízení. Vzdálené ovládání zásuvky se ukázalo jako velmi užitečné zejména pro

[6] https://openjdk.java.net

gigabitovou sondu FlexProbe, která je realizována ve formě vestavěného systému. Při jejím zátěžovém testování v některých situacích docházelo k úplnému zaplnění paměti, při kterém sondu nebylo možné softwarově restartovat.



Obrázek 6.5: Testovací prostředí

6.6 Automatizace

Ve chvíli, kdy jsou napsány testy a vytvořeny testovací sady, je potřeba začít se zabývat způsoby, kterými budou testy spouštěny. Jednou z možností je testy spouštět manuálně. To je typické pro vývojovou fázi, v níž vývojář zkouší svoji implementaci, ke které vytvořil podle návrhu testy. Při nárůstu počtu testů a rozdělení do různých testovacích sad začne být manuální spouštění neefektivní, navíc může negativně zapůsobit lidský faktor. Některé testy mohou být omylem vynechány, výsledky špatně interpretovány a podobně. Automatizované spouštění testů a testovacích sad docílí opakovatelnost, deterministické vyhodnocení kvality systému a ušetří čas testerovi i vývojářům [7].

Jednotlivé testy byly vytvořeny tak, aby je bylo možné používat automatizovaně. Použitý testovací framework (Unittest nebo GoogleTest) umožňuje postupné spuštění testů z testovací sady. Správně implementovaný test sám nakonfiguruje sondu ve fázi *setup* (viz kapitola 4). Přesto není automatizace dostatečná: testy se nezabývají stavem sondy (nainstalovanou verzí softwaru, nahraným firmware do FPGA), stále je nutná interakce s člověkem, a pokud je testovacích sad více, musí je tester spustit jednu po druhé.

S úplnou automatizací navržených testů pomůže software, který umožňuje automatizaci procesů, jako je sestavení, testování a nasazení počítačových systémů. Automatizační systémy nabízejí možnost tvorby skriptů, pomocí kterých může být provedeno otestování

celé síťové sondy v reakci na jednu událost. Události vznikají na základě času, vývojářské aktivity (např. vydání nové verze balíčku) nebo uživatelské interakce prostřednictvím GUI. Reakcí na událost je provedení automatizovaného scénáře, kde v případě testování každý krok odpovídá spuštění jedné testovací sady. Z pohledu operačního systému je automatizační systém program, který je spuštěn na pozadí a čeká na definované události.

Testy musí být na integraci do automatizovaného scénáře náležitě připraveny. U testů navržených v této práci se nejvíce problémů objevilo ve fázi ověřování výstupů sondy (*verify*). Zátěžové testy, které zkoušejí funkci sondy v přetíženém stavu, není snadné vyhodnotit. Z tohoto důvodu byla činnost ověřena posouzením vývojáře nebo testera na základě výstupů testu. Algoritmy vyčištění paměti a zotavení sondy navíc fungují částečně nedeterministicky. Při automatizaci testů musely být zvoleny takové podmínky, aby detekovaly nežádoucí chování, ale zároveň rozpoznaly korektní funkci ovlivněnou nedeterminismem. Podmínka například kontroluje, zda jsou při zahazování provozu na vstupu filtry zaplněny alespoň z 80 %. Tedy již není možné jednoduše přidávat nová filtrační pravidla a pakety jsou zahazovány. Dále se jedná o kontrolu, jestli jsou prioritně zpracovávány pakety těch toků, u kterých nevypršel aktivní časový limit.

Automatizační scénář probíhá v několika fázích. Nejprve je prověřena dostupnost sondy v síti. Následně je sonda přivedena do stavu, ve kterém má být testována: je aktualizován firmware FPGA i softwarové balíčky. Sonda je restartována z důvodu aplikace nových verzí i zajištění spuštění testů s nezaplněnou pamětí a ve funkčním stavu. Dále jsou spuštěny jednotlivé testovací sady. Sady se řadí podle úrovně testů od jednotkových po systémové. Jednotkové testy, které jsou součástí zdrojových kódů a tvoří prerekvizitu k sestavení systému jsou prováděny při sestavení software. Instalovaný software je tak prověřen jednotkovými testy a do automatizovaného scénáře tyto testy nejsou integrované. Ostatní sady testů jsou postupně provedeny v následujícím pořadí: testy hardwarových komponent, testy na záchyt provozu dle identifikátorů L3 a L7 vrstvy, testy statistik, zátěžové testy, testy zotavení a systémové testy pokrývající funkci sondy i s kolektorem. Pokud během testování některý z testů skončí neúspěšně, zbytek automatizovaného scénáře se neprovádí. Předpokládá se, že sonda nebude fungovat pro testy na vyšší úrovni, pokud nejsou splněny požadavky na jednotlivé podsystémy. Na závěr automatizovaného scénáře je vyčištěno testovací prostředí. Zejména se jedná o smazání exportovaných souborů, které je provedeno i v případě, že některý test neuspěl. V poslední fázi se také archivují výsledky testů, které by mohly být užitečné pro pozdější analýzu.

6.6.1 Jenkins

Jako vhodný nástroj k automatizaci testů byl vybrán software Jenkins [6]. Hlavní motivací k výběru tohoto systému byla nutnost kompatibility se stávajícím systémem automatizovaného překladu, který je používán při vývoji sondy FlexProbe. Jenkins je původně vyvinutý pro kontinuální integraci (CI), což je praktika, která zajišťuje, že sestavení a následovné testování programu se provádí při každé změně ve zdrojovém kódu [7]. Svým návrhem je ale Jenkins použitelný pro jakoukoli činnost, kterou je nutné automatizovat. Umožňuje rozšířitelnost pomocí zásuvných modulů a skriptování ve speciálním jazyce Groovy. Díky těmto vlastnostem může být systém přizpůsoben k realizaci libovolného testu, v případě této práce k testování hardwarových sond ve vzdáleně ovládaném testovacím prostředí (viz kapitola 6.5). K řízení testování jsou navrženy dva automatizované scénáře. První seskupuje testovací sady zaměřené na funkci sondy. Druhý testovací scénář slouží ke spuštění výkonnostních testů.

Jenkins je primárně ovládán prostřednictvím webového grafického uživatelského rozhraní. GUI sjednocuje způsob, jakým nahlížet na výsledky testů, prohlížet historii jejich spuštění, sledovat kvalitu vyvíjeného systému ve větším časovém úseku atd. Jedním z největších zjednodušení testování (po samotné automatizaci) je právě náhled do historie, pro který server udržuje všechny výstupy spuštěných programů, strom vytvořených souborů a přesnou konfiguraci s jakou byl testovací scénář spuštěn. Užitečná je také archivace souborů, pomocí které může vývojový tým vyřešit evidenci a úložiště vydaných verzí programů (obdoba balíčkového repozitáře).

Spuštění automatizovaného scénáře se v Jenkins nazývá sestavení – *build*. V reakci na nahrání nové verze do repozitáře (*commit*) se vždy spustí test a ověří se korektnost vytvořené implementace. Dochází tak ke kontinuální integraci jednotlivých funkcí do systému (CI). Podpora různých systémů správy verzí je instalována v podobě zásuvných modulů, například systém Git nebo Subversion (SVN). Program je nejprve sestaven, čímž je zajištěna kontrola přeložitelnosti zdrojového kódu. Následně je nově sestavený program podroben testům. Postupně jsou provedeny jednotkové testy softwaru popsané v kapitole 6.2. Na projektu FlexProbe je scénář doplněn o vytvoření balíku pro operační systém Linux, který je v Jenkins archivován. Balíky sestavené v Jenkins se následně používají k aktualizaci sondy prostřednictvím balíčkového repozitáře, případně je může převzít jiný testovací scénář.

Architektura systému Jenkins se skládá z hlavního serveru, který se nazývá *master*, a připojených agentů. Zatímco *master* udržuje všechna data a konfiguraci, agenti jsou pověřeni vykonáváním automatizovaných scénářů. Všechny výstupy, které vzniknou na agentech jsou uloženy na hlavním uzlu včetně archivovaných souborů. Nedostupnost agenta tak nezpůsobí omezení služeb a nezáleží, který agent práci vykoná. V případě testovacího prostředí pro síťové sondy (kapitola 6.5) jsou agenti specializovaní. Testovací scénář je přímo přiřazen agentovi testovací stanice a nemůže ho vykonat jiný agent. Agent má v Jenkins určitý počet slotů, který udává maximální počet souběžně vykonávaných scénářů. V případě testovacích stanic je počet slotů úměrný počtu připojených sond. Hardwarová sonda nemůže být zároveň testována více testy. Exportovaná data by obsahovala zpracovaný provoz ze všech testů, což by se velmi špatně ověřovalo. Systém Jenkins automaticky zařídí, aby spuštěný scénář vyčkal na volný slot.

Primárně se automatizované scénáře implementují v jazyce Groovy. Groovy [16] je programovací jazyk vytvořený společností Apache. Je interpretován pomocí Java Virtual Machine, a může tak využívat funkce a knihovny implementované v jazyce Java. Groovy je plně integrován do systému Jenkins a lze z něj volat funkce z instalovaných zásuvných modulů, vytvářet vlastní funkce a knihovny, které je následně možné importovat do skriptu scénáře. Aktuálně je pro tvorbu automatizačního scénáře doporučeno uspořádat zdrojový kód do zřetězené linky, která je definována deklarativně (*Jenkins Declarative Pipeline*). Deklarativní způsob programování je syntaktická nadstavba Groovy od vývojářů Jenkins, která si bere za cíl zjednodušit a zpřehlednit skripty automatizovaných scénářů.

Testovací scénáře pro sondu FlexProbe využívají zřetězenou linku k postupnému spouštění testovacích sad navržených v kapitole 6.2. Zřetězená linka scénáře je sestavena z několika etap (*stage*), kde každá etapa odpovídá spuštění jedné testovací sady. Kromě testovacích etap se ve scénáři nachází také obslužné etapy, ve kterých se konfiguruje testovací prostředí a aktualizuje systém sondy. Scénáře se konfigurují pomocí parametrů, které nesou hodnoty testovacího prostředí a příznaky, zda mají být jednotlivé etapy provedeny. Parametry je možné zadat při ručním spuštění scénáře v GUI, nebo mohou být hodnoty pro daný scénář předvyplněny. Druhá možnost je vhodná pro plně automatizované spuštění, např. na

základě vydání nového systému nebo v určitém čase. Díky volbě vynechání některých etap můžeme vytvořit varianty scénáře pro krátký a dlouhý běh.

V příloze A se nachází zkrácený úryvek skriptu, kterým se řídí testovací scénář zaměřený na funkci sondy. Každá zřetězená linka je zabalená do bloku `pipeline`. Při specializovaném scénáři může být uveden agent, který má scénář vykonat. Před samotnou deklarací etap jsou definovány parametry scénáře. V ukázce je jedním z parametrů cesta ke konfiguračnímu souboru, ve kterém jsou údaje o prostředí testování. Soubor obsahuje například IP adresu sondy, název rozhraní pro zasílání testovacího provozu, přihlašovací údaje sondy atd. Druhým parametrem se volí způsob aktualizace sondy. Možnosti jsou tři: aktualizace se neprovede, aktualizační balík nahraje uživatel prostřednictvím webového GUI, nebo je stažen nejnovější balík, který je archivovaný v Jenkins.

Dále je ukázána implementace 3 různých etap. První je zodpovědná za přípravu konfigurace, kterou využívají testy, v druhé je aktualizována a restartována sonda a ve třetí se provádí sada funkčních testů. I při deklarativním zápisu se vetšinou není možné vyhnout použití Groovy, a to při práci s proměnnými nebo při definici funkcí. Ve zdrojovém kódu deklarativní zřetězené linky je uzavřeno do bloků `skript`. Jak je na příkladu vidět, veškerá konfigurace sondy i samotné testování je prováděno pomocí připravených skriptů v jazyce Python. Využívá se tak potenciálu testovacího frameworku (6.4) a Jenkins vystupuje pouze jako spouštěč. U všech spuštěných programů prostřednictvím bloku `sh` je následně kontrolován návratový kód. Ve výchozím chování je etapa označena za neúspěšnou v případě, že je návratová hodnota větší než 0. Pokud některá z etap neuspěje, všechny následující etapy zřetězené linky jsou přeskočeny.

Druhý testovací scénář se zabývá výkonem sondy. Pro lepší přehlednost a možnost srovnání výkonu mezi různými vydáními systému je využit zásuvný modul *Plot*. Ten dokáže ve webovém GUI vykreslovat grafy u testovacích scénářů. Při každém spuštění je zaznamenána hodnota do několika různých grafů (rychlost zpracování jednotlivých aplikačních protokolů, maximální množství zájmového provozu). Tester tak může kdykoli prohlížet grafy a zhodnotit, jestli při vývoji nedochází k poklesu výkonu.

Návrhem dvou testovacích scénářů je pokryto spuštění všech funkčních i výkonnostních testů pro sondu FlexProbe. Testy statistik integrované do automatizovaného scénáře ověřují funkci sondy IPFIX probe. Přehled o výsledcích scénářů s možností naplánování provedení je dostupný prostřednictvím webového rozhraní Jenkins. Scénáře obsahují parametry, kterými je možné přizpůsobit běh testu pro danou situaci.

Kapitola 7

Výsledky

Na základě návrhu (kapitola 6) byly implementovány dva automatizované scénáře, které jsou prováděny v systému Jenkins. První se zabývá výhradně funkcí sondy FlexProbe a druhý měří její výkon. Zjištění výkonnostních parametrů je podmíněno úspěchem funkčních testů, které musí být provedeny nejdříve. Dalším argumentem pro oddělení testů je menší frekvence spouštění výkonnostních testů, které jsou spouštěny jenom v případě vydávání balíčků pro Policii ČR.

Od začátku implementace byl kladen důraz na parametrizaci automatizovaných scénářů. Díky tomu není scénář rezervován pouze pro jednu fyzickou sondu a jeden konkrétní průběh. Testovací prostředí je plně parametrizováno pomocí konfiguračního souboru uloženého na testovací stanici. V konfiguračním souboru jsou uloženy hodnoty, které je možné použít napříč celým testovacím scénářem. Mezi konfigurovatelné parametry patří například IP adresa sondy, přihlašovací údaje a síťová rozhraní, na která se zasílá testovací provoz. Průběh automatizovaného scénáře je ovlivněn parametry pro zapnutí a vypnutí jednotlivých testovacích sad. Scénáře ctí rozdělení testů do testovacích sad, které je popsané v kapitole 6.2. Samostatné testy není možné přeskočit s výjimkou testů ověřujících chování sondy v dlouhodobém provozu. Provedení dlouhodobých testů lze vypnout při potřebě primární zkoušky, která rychle vyzkouší funkci všech podsystémů.

Po čas práce na projektu FlexProbe nebylo potřeba spouštět scénáře automaticky (v reakci na událost). Sestavení produkčního vydání systému sondy bylo vždy oznámeno dopředu a spuštění scénářů tak vždy proběhlo ručně. Při ručním spuštění scénáře uživatel ve webovém rozhraní Jenkins nastaví parametry, které se skládají z cesty ke konfiguračnímu souboru testovacího prostředí, množiny testovacích sad určených ke spuštění a výběru, zda chce sondu aktualizovat. Manuální spuštění umožňuje aktualizaci sondy balíčkem, který je přímo nahrán do Jenkins prostřednictvím GUI, což je vhodný prostředek k ověření chování změněného balíčku před přidáním do repozitáře, kde se již očekává korektní funkce.

Systém je připraven i na automatické spouštění scénářů, pro které je v softwaru Jenkins vytvořeno několik instancí automatizovaného scénáře (tzv. *Job*). Každá instance odpovídá jednomu specifickému průběhu scénáře. Instance, která vynechává testy dlouhodobého provozu je nastavena na spouštění s krátkou periodou: každý den v určitou hodinu. Kompletní funkční otestování proběhne na konci každého měsíce společně s výkonnostními testy. Zdrojový kód deklarativní zřetězené linky (viz kapitola 6.6.1) pro množinu instancí scénáře zůstává stejný, v jednotlivých instancích se liší pouze předvyplněné hodnoty parametrů. Aktualizaci sondy při automatickém spuštění řeší scénář sám stažením nejnovějších balíčků ze soukromého repozitáře a nahráním na sondu pomocí REST API.

7.1 Doba běhu testů

Doba běhu automatizovaného scénáře se nejvíce odvíjí od zvolených parametrů, které určují, jaké testovací sady mají být spuštěny. U funkčních testů se doba běhu příliš nemění. Jediné nedeterministické chování ovlivňující celkovou dobu běhu spočívá v čekání na provedení konfigurace sondy, která může trvat různě dlouho. Konstantní čas běhu nelze očekávat ani v případě výkonnostních testů. Zde se doba odvíjí od maximální rychlosti zpracování provozu.

Testy ve scénáři, který pokrývá funkční požadavky jsou rozděleny na 3 testovací sady: hardwarové testy, testy záchytu dle identifikátoru L7 a systémové testy s mediační funkcí. Doba běhu hardwarových testů je v průměru 5 minut a 15 vteřin, u L7 testů to je přibližně 12 minut. Systémové testy, které ověřují záchyt podle L3 i L7 identifikátorů mají dobu běhu přibližně 25 minut. Delší doba běhu je dána jednak větším množstvím testovacích případů – testují se náhodné kombinace L3 identifikátorů, ale také delším intervalem pro vyčkání na připojení k mediační funkci. V čase, který odpovídá přibližně 45 minutám lze provést všechny funkční testovací sady a ověřit tak spolupráci podsystémů i funkčnost celku.

Pro pokročilé otestování se zapínají testy, které ověřují chování v dlouhodobém provozu a v zatíženém stavu. Test systému pro identifikaci síťových toků je ve výchozím režimu nakonfigurován tak, aby během své činnosti vygeneroval 100000 síťových toků. Postupné odeslání a ověření takového počtu toků je provedeno průměrně za 1 hodinu a 12 minut. Dalším testem s dlouhým během je test zaplnění hardwarového filtru (`periodic.py`, viz kapitola 6.3). Opět je doba běhu ovlivněna počtem generovaných síťových toků. V automatizovaném scénáři je tento počet nastaven na 11000. Běh testu trvá 3 hodiny a 4 minuty, což odpovídá času vygenerování posledního síťového toku (s indexem 10999). Každý tok je v periodickém testu poprvé odeslán až po uběhnutí intervalu, který je dán jeho pořadovým číslem. Spuštěním dlouhodobých testů v rámci automatizovaného scénáře se prodlouží jeho běh na přibližně 5 hodin s tím, že již není třeba spouštět jiné funkční testy.

7.2 Pokrytí specifikace požadavků

V této kapitole se nachází vyhodnocení pokrytí požadavků z kapitoly 5 implementovanou testovací sadou. Testy byly podrobně analyzovány a bylo rozlišeno, které systémy vystupují jako SUT nebo mají vliv na ověřovaný výstup síťové sondy. Pro každý požadavek bylo určeno, zda existuje test pro přímé ověření nebo je pokryt nepřímo chováním některého z testů. Požadavek je většinou ověřen více testy. Nejprve jsou přímo ověřeny požadavky na chování modulů, následně se systém testuje jako celek. Některé požadavky se skládají z více požadavků. Testy pak ověřují dílčí požadavky a celý požadavek je splněn za podmínky, že žádný z provedených testů neselhal.

Tabulky 7.1 a 7.2 zobrazují vyhodnocení pokrytí jednotlivých požadavků pro sondy IPFIX probe a FlexProbe. Stav pokrytí nabývá jednoho ze 3 stavů: požadavek je pokryt přímo, nepřímo nebo není vůbec ověřen. Pro stoprocentní otestování sondy by měly být dostupné testy pro všechny definované požadavky, což v současné chvíli neplatí.

Z vyhodnocení lze vyvodit, že je funkce exporteru IPFIX probe dobře pokryta. Testováno je 24 požadavků z celkových 28. Velká část podsystémů je pokryta nepřímo systémovými testy primárně určenými k ověření funkce zásuvných modulů. Nabízí se proto rozšíření sady jednotkových testů, které by přispělo k jednodušší lokalizaci případné chyby.

Testovací sada pro sondu FlexProbe reflektuje specifikované požadavky a z většiny je pokrývá. Z celkových 31 požadavků je pokryto 24, přičemž pouze 3 požadavky jsou po-

1.	nepřímo		4.	přímo		6.1	přímo
1.1	nepřímo		4.1	přímo		6.2	přímo
1.2	nepřímo		4.2	přímo		6.3	přímo
1.3	nepřímo		4.3	přímo		6.4	přímo
1.4	nepřímo		4.4	přímo		6.5	nepřímo
2.	nepřímo		5.	přímo		7.	nepřímo
2.1	přímo		5.1	přímo		8.	neověřeno
3.	neověřeno		5.2	přímo		9.	neověřeno
3.1	nepřímo		5.3	přímo			
3.2	neověřeno		6.	nepřímo			

Tabulka 7.1: Pokrytí specifikace požadavků sondy IPFIX probe

1.	přímo		4.	neověřeno		12.	neověřeno
1.1	přímo		5.	přímo		12.1	neověřeno
1.2	přímo		6.	nepřímo		12.2	neověřeno
1.3	přímo		7.	přímo		12.3	nepřímo
1.4	přímo		8.	přímo		12.4	nepřímo
2.	přímo		9.	přímo		12.5	neověřeno
2.1	přímo		10.	přímo		13.	neověřeno
2.2	přímo		11.	přímo		13.1	přímo
3.	přímo		11.1	přímo		13.2	neověřeno
3.1	přímo		11.2	přímo			
3.2	přímo		11.3	přímo			

Tabulka 7.2: Pokrytí specifikace požadavků sondy FlexProbe

kryty nepřímo. Přímé pokrytí požadavků je dáno tím, že byla testovací sada navržena pro ověření definované specifikace. Protože se jedná o komplexní zařízení, vytvoření testů pro stoprocentní pokrytí požadavků bylo nad rámec této práce. Nejvíce neověřených požadavků se zabývá pokročilou funkcí GUI. V testovací sadě se nekombinují L3 a L7 testy, které při vývoji vznikaly odděleně. Teprve implementací testovacího frameworku (6.4) bylo chování testů sjednoceno a je zde proto prostor pro vylepšení.

Na základě spuštění testů bylo vyhodnoceno, které ze specifikovaných požadavků nejsou v současné chvíli splněny. V případě IPFIX probe selhává test ověřující extrakci atributů z nejhlouběji zapouzdřené hlavičky IP protokolu (požadavek č. 2.1). Ukázalo se, že software exportuje hodnoty z vnější hlavičky. Chování tak není stejné jako při záchytu provozu sondou FlexProbe. Pomocí výkonnostních testů se podařilo odhalit neefektivitu implementace záchytu komunikace, která je realizována sondou FlexProbe. Nemůže být proto splněn požadavek č. 7. Podrobné výsledky měření výkonu jsou popsány v následující kapitole. Chování systémů s neúspěšnými testy bude diskutováno s vývojářským týmem za účelem určení místa vzniku chyby. Případně může být změněna specifikace na základě konzultace se zákazníkem (PČR).

7.3 Výkonové parametry

V tabulkách 7.3 a 7.4 se nachází vyhodnocení výkonu prototypu sondy FlexProbe 10G. Měření probíhalo na fyzickém zařízení sondy v testovacím prostředí na FIT VUT v Brně. Pro testování byl použit syntetický síťový provoz, ve kterém se nachází 1 % nebo 10 % zájmové komunikace, která musí být sondou zachycena. Při každém testu je konfigurován jeden odposlouchávaný identifikátor aplikačního protokolu a všechen zájmový provoz v testovacích datech je tvořen tímto protokolem. Změřená hodnota odpovídá nejvyšší rychlosti odesílání testovacího provozu, při které byly všechny pakety zachyceny.

Při verifikaci výkonnostních testů bylo zjištěno, že kvůli limitům zpracování PCAP souborů je maximální rychlost odesílání provozu testovací stanicí 5 Gb/s. Výkon sondy FlexProbe 10G však u žádného z aplikačních protokolů nedosahoval této rychlosti a měření proto nebylo ovlivněno.

1 % zájmového provozu	1kB (Mb/s)	10kB (Mb/s)	100kB (Mb/s)
SMTP	1374	1872	1260
POP3 (pouze email)	1270	2148	1328
POP3 (celé sezení)	1314	1692	78
IMAP (pouze email)	1718	2057	982
IMAP (celé sezení)	1508	2017	636
FTP (pouze soubor)	1824	1206	1484
FTP (celé sezení)	1546	1328	1445

Tabulka 7.3: Výsledky měření výkonu sondy FlexProbe 10G pro záchyt dle aplikačních identifikátorů. Zájmový provoz tvoří 1 % celkového provozu.

10 % zájmového provozu	1kB (Mb/s)	10kB (Mb/s)	100kB (Mb/s)
SMTP	1406	2112	472
POP3 (pouze email)	782	1552	480
POP3 (celé sezení)	1176	2346	524
IMAP (pouze email)	1690	2530	404
IMAP (celé sezení)	2188	2524	518
FTP (pouze soubor)	2362	1456	1244
FTP (celé sezení)	1524	1470	1170

Tabulka 7.4: Výsledky měření výkonu sondy FlexProbe 10G pro záchyt dle aplikačních identifikátorů. Zájmový provoz tvoří 10 % celkového provozu.

Z výše uvedených dat vyplývá, že sonda FlexProbe nesplňuje požadavek 7 z kapitoly 5, který vyžaduje zpracování vstupního provozu na plné rychlosti linky (10 Gb/s). Mezi výsledky zpracování provozu s 1 % a 10 % zájmových paketů není příliš velký rozdíl. Z toho lze usoudit, že implementace není efektivní kvůli chybě, která postihuje zpracování veškerého vstupního provozu. Výsledky měření již byly předány vývojovému týmu FlexProbe, který se bude zabývat optimalizací implementace systému sondy FlexProbe 10G.

7.4 Odhalování chyb a akceptační testování

Po celou doby vypracovávání diplomové práce probíhala intenzivní spolupráce s týmem FlexProbe. Všechny odhalené chyby byly nahlášeny a proběhla nebo postupně probíhá jejich náprava. Na začátku spolupráce bylo reportováno několik ručně odhalených chyb GUI, které do té doby nebylo při vývoji příliš využíváno. V počáteční fázi byly analyzovány existující testy a bylo ověřováno jejich chování. Při analýze se GUI ukázalo jako vhodný nástroj pro kontrolu konfigurace sondy a zjištění stavu záchytu síťového provozu. Jako příklad chyby je možné uvést špatné přizpůsobování osy y v grafech příchozího a odchozího provozu. Po vyřešení chyby byly grafy velmi užitečné zejména při kontrole průběhu výkonnostních testů.

V současné chvíli jsou udržovány dvě varianty sondy FlexProbe: 1G a 10G. Před začátkem práce byly veškeré dostupné testy vytvořeny pro gigabitovou variantu sondy, jejíž vývoj probíhal po nemalý časový úsek. Postupně byly testy upraveny pro rychlejší variantu sondy. Řadu existujících testů bylo nutné modifikovat, neboť jsou v případě 10G varianty sonda a mediační funkce realizovány jako jedno zařízení. Pro zajištění budoucí podpory testy i automatizované scénáře podporují obě varianty sondy FlexProbe.

Po vydání produkční aktualizace systému v létě roku 2021 se stal prototyp FlexProbe 1G plně funkční pro realizaci zákonných odposlechů. V reakci na to bylo předáno fyzické zařízení sondy Policii ČR pro akceptační testování. Dle očekávání odhalil produkční způsob použití jiné druhy chyb. Zásadní se ukázalo vyčerpání paměti RAM při dlouhodobém záchytu. Ačkoli se funkce záchytu v časovém horizontu několika hodin testovala, chyba nebyla odhalena. K vyčerpání paměti totiž došlo až po několika týdnech aktivního záchytu. Ladění chyby spočívalo v nasazení prostředků pro monitorování použité paměti. Monitorována byla celkově použitá paměť i alokace jednotlivými procesy. Z podstaty chyby, která se objevuje po několika týdnech provozu, byla replikace obtížná a částečně zdržovala další vývoj. Výsledkem analýzy je vyloučení chyby implementace software FlexProbe, ve kterém potenciálně mohlo docházet k únikům paměti (*memory leak*). Volná paměť RAM se nejspíše ztrácí přímo v jádru operačního systému. Podle dostupných dat ztráta nesouvisí se záchytem, ale exportem již zpracovaných dat. Vzhledem k vývoji varianty 10G a předpokládaným nahrazením zastaralé varianty bylo důležité zjištění, že chybu nezpůsobuje vyvíjený software. FlexProbe 10G je v poslední fázi vývoje a brzké době se očekává zahájení akceptačního testování ze strany Policie ČR.

Implementovaným systémem testů byly ověřeny celkem 3 produkční aktualizace sondy FlexProbe 1G. Funkční testy vždy procházely bez problémů a všechny pokryté aspekty sondy byly úspěšně ověřeny. Pozornost byla věnována také zátěžovým testům, které ověřily, že se sonda v přetíženém stavu chová dle očekávání a dále dokáže pracovat.

Kapitola 8

Závěr

Cílem diplomové práce bylo implementovat systém testů pro sondy určené k monitorování síťového provozu a aplikovat výsledek v podobě automatizovaných testů na sondy IPFIX probe a FlexProbe. Cíl byl splněn implementací testovacích sad, které převzaly původní sady a rozšířily jejich testy tak, aby pokrývaly požadované vlastnosti sond. Testy jsou automatizované a pro jejich spouštění je využit software Jenkins.

V rámci řešení práce byla nastudována problematika monitorování síťového provozu a princip fungování síťových sond. Teoretický rozbor se důkladně věnuje možnostem sledování síťových prvků i síťového provozu. Největší prostor získalo monitorování síťových toků a záchyt zájmového provozu, jelikož tyto techniky realizují sondy IPFIX probe a FlexProbe. Z teorie o testování jsou vybrány části, které jsou relevantní pro testování hardwarových zařízení. Text představuje druhy testů a jejich návaznost na návrh systému. Pro návrh testů jsou nejdůležitější požadavky, které mají testy ověřovat. Specifikace požadavků byla vypracována pro sondu IPFIX probe i FlexProbe. V případě FlexProbe specifikace vyplývá z požadavků Policie ČR, která je koncovým uživatelem sondy. Požadavky na sondu IPFIX probe z velké části pochází z teoretického rozboru, a tedy předpokladu, jak by sonda pro monitorování síťových toků měla fungovat.

Na základě specifikace požadavků bylo navrženo několik sad testů, které jsou seskupeny podle druhu. V návrhu jsou tak sady jednotkových, integračních a systémových testů. Za účelem automatizace a sjednocení testů byl navržen testovací framework, ve kterém jsou testovací sady implementovány. Většina testů je prováděna přímo nad hardwarovým zařízením síťové sondy. Návrh se proto zabývá zapojením testovacího prostředí a jeho specifiky. Použity jsou dva možné způsoby zapojení. V prvním je sonda součástí testovací stanice a ve druhém vystupuje jako oddělené zařízení. Pro automatizované spouštění testovacích sad jsou navrženy dva automatizované scénáře – funkční a výkonnostní. Testy ve funkčním scénáři se řadí od nejjednodušších po nejsložitější. Spuštění v seřazené kaskádě zaručuje, že je činnost jednotlivých podsystémů ověřena dříve, než se začne testovat systém jako celek. Pro implementaci automatizovaných scénářů byl vybrán systém Jenkins, který obsahuje všechny potřebné funkcionality.

Hlavním výsledkem práce je vyhodnocení pokrytí specifikovaných požadavků u obou sond. Vyhodnocení zohledňuje skutečnost, jestli byl požadavek pokryt přímo, nepřímo nebo nebyl ověřen. U dvou požadavků se ukázalo, že ověření selhává. Oba případy byly analyzovány a na základě provedených testů byla diskutována změna implementace, která by problémy vyřešila. Dále byl změřen výkon sondy FlexProbe 10G. Naměřené výsledky ukazují nedostatečnou propustnost implementace, která bude muset být optimalizována. V průběhu práce byly nalezené chyby konzultovány s týmem FlexProbe a mnoho z nich již bylo vyře-

šeno. U sondy FlexProbe probíhá akceptační testování ze strany Policie ČR, které přispívá k výsledné kvalitě systému.

Dalším rozšířením této práce by mohlo být ověření požadavků, které nejsou pokryté současnými testy. Jedná se zejména o testy GUI sondy FlexProbe, kde by mohla být sada rozšířena o testy kontrolující zpětnou vazbu, čitelnost provozních informací a zobrazení statistik. Výkonnostní testy v současné chvíli neověřují plnou rychlost linky 10 Gb/s. Ačkoli síťové karty testovací stanice plnou rychlost podporují, problém je v načítání provozu z PCAP souborů, které je omezeno frekvencí jednoho jádra procesoru. Do budoucna by bylo vhodné implementovat generování provozu přímo výkonnostním testem. Vysokorychlostní generování umožňuje software TRex, který se již v jiných testech využívá. Testování bude rozvíjeno v uvedených směrech během další spolupráce na projektu FlexProbe.

Literatura

[1] Black, R. *Advanced Software Testing Vol. 2: Guide to the ISTQB Advanced Certification as an Advanced Test Manager*. 1st Edition. Rocky Nook, 2008. ISBN 978-1-933952-36-9.

[2] Cisco Systems. *TRex traffic generator*. 2021. Dostupné z: https://trex-tgn.cisco.com.

[3] *ETSI TR 102 528: Lawful Interception (LI); Interception domain Architecture for IP networks. version 1.1.1*. Standard. European Telecommunications Standards Institute, říjen 2006.

[4] Hofstede, R., Čeleda, P., Trammell, B., Drago, I., Sadre, R. et al. Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE Communications Surveys Tutorials*. 2014, sv. 16, č. 4, s. 2037–2064. DOI: 10.1109/COMST.2014.2321898.

[5] Huták, L. *Nová generace IPFIX kolektoru*. Vysoké učení technické v Brně. Fakulta informačních technologií, 2018.

[6] Kawaguchi, K. *Jenkins*. 2022. Dostupné z: https://www.jenkins.io.

[7] Kolpakova, A. *Využití automatizovaných regresních testů v systému kontinuální integrace webové aplikace*. Vysoká škola ekonomická v Praze, 2017.

[8] Kořenek, J., Korček, P. a Kaštil, J. *Sondy pro monitorování provozu*. 2011. 26 s. Dostupné z: https://www.fit.vut.cz/research/publication/9817.

[9] Kořenek, J. a Korček, P. *Flexibilní sonda pro realizaci zákonných odposlechů* [online]. Brno: Vysoké učení technické v Brně, Fakulta informačních technologií, červenec 2019 [cit. 2022-01-12]. Dostupné z: https://www.fit.vut.cz/research/project/1304/.cs.

[10] Kretchmar, J. M. *Administrace a diagnostika sítí : pomocí OpenSource utilit a nástrojů*. 1. vyd. Brno: Computer Press, 2004. ISBN 80-251-0345-5.

[11] Matoušek, P. *Síťové aplikace a jejich architektura*. 1. vyd. Brno: VUTIUM, 2014. ISBN 978-80-214-3766-1.

[12] Podivínský, J. *Use of verification for testing fault-tolerance in FPGA-based system*. Brno, CZ, 2021. Disertační práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/phd-thesis/885/.

[13] Polčák, L., Kramoliš, P., Kajan, M. a Martínek, T. *Architektura systému pro zákonné odposlechy*. 2011. 25 s. Dostupné z: https://www.fit.vut.cz/research/publication/9829.

[14] Python Software Foundation. *Unit testing framework*. 2022. Dostupné z: https://docs.python.org/3/library/unittest.html.

[15] Taeksu, K., Chanjin, P. a Chisu, W. Mock Object Models for Test Driven Development. In: *Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06)*. IEEE, 2006, s. 221–228. ISBN 076952656X.

[16] The Apache Software Foundation. *Groovy Language*. 2022. Dostupné z: https://groovy-lang.org/index.html.

[17] Tran, D. *Ovladač netdev pro akcelerační karty COMBO*. Vysoké učení technické v Brně. Fakulta informačních technologií, 2017.

[18] Špringl, P. *Architektura programového vybavení monitorovací sondy na bázi toků = Software Architecture for Flow Based Monitoring Probe*. Brno: Vysoké učení technické, Fakulta informačních technologií, 2009.

Příloha A

Zdrojový kód automatizovaného scénáře

```
pipeline {
  agent { label 'testbed' }

  parameters {
    string(
      defaultValue: '/var/jenkins/env-sprobe-1g.properties',
      description: 'Probe config file (with eg. probe ip address)',
      name: 'ENV_PROPS_FILE',
      trim: true)
    choice(
      choices: ['No update' , 'Upload image', 'Copy from buildroot job'],
      description: 'How to update probe image',
      name: 'UPDATE_PROBE')
    ...
  }

  stages {
    stage('Global prepare') {
      steps {
        FlexProbeEnvsFromFile(env.ENV_PROPS_FILE)
        script {
          def conf = readYaml file: 'env-sprobe-1g.yaml'
          conf.probe.host = env.PROBE_IP
          conf.measurement.replay_interface = env.L7_TRAFFIC_INTERFACE
          conf['configuration'] = ['mediation_function_ip': env.SELF_IP]
          writeYaml file: 'jenkins.yaml', data: conf, overwrite: true
        }
      }
    }
    stage('Probe image update') {
      when { expression { params.UPDATE_PROBE != 'No update' } }
      steps {
```

```
        script {
          if (params.UPDATE_PROBE == 'Copy from buildroot job') {
            copyArtifacts filter: '*.tgz'
          }
        }
        sh './jenkins/scripts/software_update.py -ip $PROBE_IP -p $SSL_PORT
        -u $SSL_USER -pw $SSL_PASSWORD -t $PROBE_TYPE ./update.tgz'
        sh 'rm update.tgz'
      }
    }
    stage('Run autotest') {
      steps {
        dir('l7tester') {
          sh 'sudo ./autotesting.py'
        }
      }
    }
    ...
  }
}
```