,

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Softwarová bezpečnost v oblasti IoT

Internet of Things Security

2017

Vítězslav Grygar

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Vítězslav Grygar**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Softwarová bezpečnost v oblasti IoT
Internet of Things Security

Jazyk vypracování: čeština

Zásady pro vypracování:

Internet věcí (Internet of Things, IoT) je označení pro propojení vestavěných zařízení s Internetem. Propojení zařízení je zejména bezdrátové a mělo by přinést nové možnosti vzájemné interakce nejen mezi těmito systémy. V současné době se používají různé embedded moduly, jejichž bezpečnost nebyla důsledně testována. Cílem této práce je prozkoumat bezpečnostní otázky nejčastěji užívaných zařízení (NAS, síťové tiskárny, měřící přístroje, spotřební elektronika atd.) a navrhnout automatický framework pro testování jejich zranitelností. Implementace by měla zahrnovat nástroj, kterému se na vstup předloží systémová image zařízení a ten automaticky rozpozná používané binární soubory, knihovny, jejich verze a v databázi zranitelností (např. CVE, Metasploit) dohledá, zda-li obsahují potenciálně zranitelná místa.

Řešení bude obsahovat:
1. Seznámení s problematikou.
2. Popis bezpečnosti stávajících systémů využívaných v oblasti internetu věcí.
3. Podrobný popis analýz a penetračních testů.
4. Implementaci SW pro automatické rozbalení systémové image a vyhledání potenciálně zranitelných míst na základě analýzy konfiguračních skriptů, binárních souborů a knihoven.
5. Testování navrženého SW balíku na volně dostupných obrazech embedded systémů.
6. Vyhodnocení testů a závěrečné shrnutí.

Seznam doporučené odborné literatury:

[1] Gary A. Donahue. Network Warrior. O Reilly Media. 2011 ISB.N 1449387861
[2] Peter Kim. The Hacker Playbook: Practical Guide To Penetration Testing. CreateSpace Independent Publishing Platform. 2014. ISBN 1494932636
[2] David Kennedy et al. Metasploit: The Penetration Tester's Guide. No Starch Press; 1 edition. 2011. ISBN 159327288X

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Mgr. Ing. Michal Krumnikl, Ph.D.**

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 25. dubna 2017 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 25. dubna 2017 .....

Rád bych tímto způsobem poděkoval Mgr. Ing. Michalu Krumniklovi, Ph.D. za odborné vedení při psaní této práce a Bc. Tomáši Bauerovi za cenné rady související s optimalizací SQL dotazů.

Abstrakt

Cílem této práce je vytvořit modulární software umožňující podrobit vybrané systémové image analýze zranitelností. První kapitola popisuje termín Internet of Things a požadavky kladené na zařízení do této oblasti spadající. V další kapitole jsou zdůrazněna bezpečnostní rizika v této oblasti a shrnuty údálosti posledních let. Čtenář je dále seznámen s možnostmi detekce zranitelností. Ve čtvrté kapitole je představen framework Locasploit a je zde popsána implementace klíčových souborů a požadovaných modulů. Závěrečná část shrnuje výsledky analýzy dostupných vzorků firmwaru.

Klíčová slova: IoT, Internet Věcí, firmware, Linux, Locasploit, Python, bezpečnost, CVE

Abstract

The aim of this thesis is to create a modular sofware capable of performing vulnerability assessment on given system images. First chapter describes the term Internet of Things and requirements for IoT devices. Next chapter emphasizes security concerns in the area and summarizes recent events. The thesis then discusses possible vulnerability detection methods. In chapter 4 the Locasploit framework is introduced, its key source files and IoT-relevant modules are described. Last part is dedicated to practical analysis of publicly available samples and its results.

Key Words: IoT, Internet of Things, firmware, Linux, Locasploit, Python, security, CVE

Obsah

Seznam použitých zkratek a symbolů **10**

Seznam obrázků **12**

Seznam tabulek **13**

0 Úvod **15**

1 Internet věcí **16**
1.1 Interoperabilita . . . 16
1.2 Spotřeba . . . 18
1.3 Cena . . . 19
1.4 Spolehlivost . . . 20

2 Bezpečnost IoT **21**
2.1 ARP . . . 21
2.2 802.11 . . . 21
2.3 DHCP . . . 22
2.4 DNS . . . 23
2.5 Transportní protokoly . . . 23
2.6 Skenování portů . . . 23
2.7 Softwarové zranitelnosti . . . 25
2.8 Zranitelnost webových aplikací . . . 27
2.9 Uživatelské účty a hesla . . . 27
2.10 Bezpečnostní incidenty v IoT . . . 27

3 Detekce zranitelností **29**
3.1 Extrakce . . . 30
3.2 Analýza konfigurace . . . 30
3.3 Detekce softwaru . . . 32
3.4 Detekce zranitelností . . . 32
3.5 Vyhledání exploitů . . . 34

4 Locasploit **36**
4.1 Klíčové soubory . . . 37
4.2 Úložiště dat . . . 42
4.3 Implementace případů užití . . . 45
4.4 Práce s frameworkem . . . 56
4.5 Tvorba modulů . . . 59

5 Analýza vzorků **61**
5.1 Extrakce 61
5.2 Analýza 62

6 Zhodnocení **67**

Literatura **68**

Přílohy **71**

Seznam použitých zkratek a symbolů

AP	–	Access Point
ARM	–	Advanced RISC Machine
ARP	–	Address Resolution Protocol
ASLR	–	Address Space Layout Randomization
BSS	–	Block Started by Symbol
CSMA/CA	–	Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD	–	Carrier Sense Multiple Access with Collision Detection
CSRF	–	Cross-site Request Forgery
CTS	–	Clear to Send
CVE	–	Common Vulnerabilities and Exposures
CVSS	–	Common Vulnerability Scoring System
DB	–	Database
DEP/NX	–	Data Execution Prevention/Non-executable
DHCP	–	Dynamic Host Configuration Protocol
DNS	–	Domain Name System
DVRF	–	Damn Vulnerable Router Firmware
FTP	–	File Transfer Protocol
GNU	–	GNU's Not Unix
GOT	–	Global Offset Table
GPIO	–	General-purpose Input/Output
GPL	–	General Public License
HTTP	–	Hypertext Transfer Protocol
ID	–	Identifier
IEEE	–	Institute of Electrical and Electronics Engineers
IMAP	–	Internet Message Access Protocol
IO	–	Input/Output
IoT	–	Internet of Things
IP	–	Internet Protocol
ISO	–	International Organization for Standardization
LoRa	–	Long Range
MAC	–	Media Access Control
MIPS	–	Microprocessor without Interlocked Pipeline Stages
MitM	–	Man in the Middle
MSCHAP	–	Microsoft Challenge-Handshake Authentication Protocol
NAS	–	Network Attached Storage
NOP	–	No Operation

OSI	–	Open Systems Interconnection
PIE	–	Position-independent Executable
PPC	–	PowerPC
PSK	–	Pre-shared Key
QEMU	–	Quick Emulator
RAM	–	Random Access Memory
RBP	–	Register Base Pointer
RELRO	–	Relocation Read-only
ret2lib	–	Return to Library
ret2reg	–	Return to Register
RFC	–	Request for Comments
RIP	–	Register Instruction Pointer
RPM	–	Red Hat Package Manager
RTP	–	Real-time Transport Protocol
RTS	–	Request to Send
SMTP	–	Simple Mail Transfer Protocol
SQL	–	Structured Query Language
SSH	–	Secure Shell
TB	–	Temporary Base
TCP	–	Transmission Control Protocol
UDP	–	User Datagram Protocol
UID	–	User ID
USB	–	Universal Serial Bus
WEP	–	Wired Equivalent Privacy
WPA	–	Wi-Fi Protected Access
XML	–	Extensible Markup Language
XSS	–	Cross-site Scripting

Seznam obrázků

1 Vztahy mezi vrstvami modelů ISO/OSI a TCP/IP 17
2 ESP8266-01[34] 19
3 Raspberry Pi 3[35] 20
4 ARP Spoofing 21
5 TCP Handshake (vlevo), SYN Scan (uprostřed), SYN Flood (vpravo) 24
6 Přetečení zásobníku 25
7 Exploit-db 35
8 Vyhledání modulů 38
9 Detailní informace o modulu 39
10 Schéma databáze *analysis.db* 43
11 Schéma databáze *checksum.db* 44
12 Schéma databáze *dict.db* 44
13 Schéma databáze *vuln.db* 45
14 Navázání SSH spojení 49
15 Analýza — sekvenční diagram 51
16 Vzorová zpráva — základní informace 55
17 Vzorová zpráva — souborový systém 55
18 Vzorová zpráva — CVE shody 56
19 Počty vzorků v jednotlivých skupinách 61
20 Doba analýzy vzhledem k velikosti souboru 62
21 Počty zranitelností (NAS) 63
22 Počty exploitů (NAS) 63
23 Počty zranitelností (OpenWRT) 64
24 Počty exploitů (OpenWRT) 64
25 Počty zranitelností (Dragino) 65
26 Počty exploitů (Dragino) 65
27 Počty zranitelností (Debian-based) 66
28 Počty exploitů (Debian-based) 66

Seznam tabulek

1 Parametry Raspberry Pi 3 B 19
2 Umístění databází správců balíčků 32
3 Případ užití *Formáty verzí pro různé produkty* 34
4 Případ užití *Update vulnerability database* 36
5 Případ užití *Create SSH connection* 36
6 Případ užití *Analyze* 36
7 Případ užití *Generate report* 37
8 Případ užití *Generate diff report* 37
9 Použití modulů pro jednotlivé případy užití 45
10 Příkazy frameworku Locasploit 56

Seznam výpisů zdrojového kódu

1 Prolomení WEP zabezpečení (ChopChop) . . . 22
2 Prolomení WPA2/PSK zabezpečení . . . 22
3 MitM DNS Spoofing . . . 23
4 SYN scan (*nmap*) . . . 24
5 Detekce softwaru (*nmap*) . . . 24
6 Vzorový CVE záznam . . . 33
7 Spuštění frameworku Locasploit . . . 38
8 Přístup k TB (Modul) . . . 42
9 Přístup k TB (Framework) . . . 42
10 Modul *locasploit.update.vulnerabilities* . . . 46
11 Aktualizace seznamu exploitů (*locasploit.update.exploit*) . . . 47
12 SSH – autentizace pomocí hesla (*connection.ssh*) . . . 48
13 Extrakce dat (*iot.binwalk.extract*) . . . 51
14 Detekce distribuce (*linux.enumeration.distribution*) . . . 52
15 Detekce verze jádra (*linux.enumeration.kernel*) . . . 52
16 Detekce uživatelů (*linux.enumeration.users*) . . . 52
17 Detekce plánovaných příkazů (*linux.enumeration.cron*) . . . 53
18 Detekce balíčků v dpkg databázi (*packages.dpkg.installed*) . . . 53
19 Seznam příkazů pro aktualizaci databáze . . . 57
20 Seznam příkazů pro analýzu lokálního systému . . . 57
21 Seznam příkazů pro analýzu vzdáleného systému . . . 57
22 Seznam příkazů pro pro analýzu systémové image . . . 58
23 Seznam příkazů pro porovnání dvou systémových image . . . 58
24 Šablona Locasploit modulu (*basic.py*) . . . 59

0 Úvod

V dnešní době se na každém kroku setkáváme s fenoménem Internet of Things. I zdánlivě dokonalé předměty získávají internetovou konektivitu a umožňují tak jednotnou správu přes centrální prvek, smartphone nebo automatické zapojení do prostředí bez jakéhokoli zásahu uživatele. S tak velkou mírou integrace však přicházejí problémy. Jedním z nich je právě otázka zabezpečení zařízení.

Cílem práce je navrhnout software, který bude usnadňovat bezpečnostní analýzu systémových image založených na jádře Linux. První kapitola *Internet věcí* je věnována fenoménu Internet of Things. Jsou zde popsány oblasti, kde se s tímto konceptem setkáváme a jsou stručně popsány základní požadavky na zařízení do této oblasti spadající. Druhá kapitola *Bezpečnost IoT* shrnuje aktuální stav bezpečnosti světa IoT. Zde jsou demonstrovány známé techniky pro ovládnutí počítačové sítě, připojených zařízení a získání přístupu k citlivým datům. Ve třetí kapitole *Detekce zranitelností* je popsán jeden z možných způsobů analýzy systémové image, který lze aplikovat na značnou část aktuálně používaných zařízení. Ve čtvrté kapitole *Locasploit* je popsána implementace takové metody ve formě modulu frameworku Locasploit. Čtenář je seznámen s klíčovými soubory frameworku, využívaným datovým úložištěm a základními ovládacími příkazy. Locasploitem byly následně otestovány vzorky běžně dostupných firmwarů zařízení Internet of Things. Výsledky jsou popsány v páté kapitole *Analýza vzorků*.

1 Internet věcí

"If we had computers that knew everything there was to know about things — using data they gathered without any help from us — we would be able to track and count everything, and greatly reduce waste, loss and cost. We would know when things needed replacing, repairing or recalling, and whether they were fresh or past their best."

KEVIN ASHTON, *That 'Internet of Things' Thing*[1]

Základy konceptu Internet of Things[1][43][44][45] (IoT) byly položeny již v roce 1999 na Massachusetts Institute of Technology. Cílem je připojit každodenně používané objekty do počítačové sítě pro zajištění vzájemné interakce a zpřístupnění nových možností analýzou získaných dat s minimálním zapojením člověka.

Světem IoT jsme obklopeni už nyní. Na trhu jsou běžně k dostání chytré spotřebiče, tzv. wearables (přístroje, které nosíme na těle a jenž často umožňují sledovat životní funkce) i komplexní řešení pro automatizaci a zabezpečení domů. Díky jednodeskovým počítačům (Arduino, Raspberry Pi) může každý vyvíjet způsob, jak interagovat s celou škálou analogových i digitálních zařízení. V průmyslu nacházejí IoT zařízení své místo také, zejména ve formě senzorů v oblastech automatizace, zemědělství a dopravy (za zmínku stojí autonomní vozidla nebo projekty chytrých měst).

V následujícím textu budou definovány požadavky, které by měly být při návrhu IoT zařízení zohledněny.

1.1 Interoperabilita

Technologie pro vzájemnou interakci počítačů jsou zde již desítky let a souhrně se dají označit termínem Internet. Nová zařízení musí být se stávajícím řešením kompatibilní, v opačném případě by musela být aktuální infrastruktura navržena znovu. To ale není, zejména z ekonomického pohledu, reálné, a ve své podstatě to ani není žádoucí. Standardy (protokoly) používané pro komunikaci jsou založené na referenčním modelu ISO/OSI[2] (ISO 7498) , kde každá ze sedmi vrstev modelu má definovány požadavky, které musí pro zajištění správné funkcionality být splněny. Prostředí Internetu je také často popisováno srozumitelnějším čtyřvrstvým TCP/IP modelem[2] (obrázek 1). Samotné komunikační protokoly jsou definovány v RFC (Request for Comments) dokumentech. RFC jsou dostupné například na `https://tools.ietf.org/`. Implementací požadovaných protokolů je zajištěna interoperabilita zařízení bez závislostí na výrobci či architektuře. V následujícím textu jsou stručně popsány jednotlivé vrstvy TCP/IP modelu a požadavky, které jsou kladeny[2].

Obrázek 1: Vztahy mezi vrstvami modelů ISO/OSI a TCP/IP

1.1.1 Vrstva síťového rozhraní

Protokoly vrstvy síťového rozhraní umožňují jednotlivým zařízením přístup k fyzickému médiu. V současné době se nejvíce využívají bezdrátové technologie (například IEEE 802.11[42]) a Ethernet[3]. Přepínače, zařízení pracující na této úrovni, jsou zodpovědné za předávání rámců správným cílům (v lokální síti), případně přepínačům určeným směrovacími mechanismy. Probíhá zde kontrola chyb; poškozené rámce (vznikající kvůli kolizím nebo závadnému hardwaru) jsou zahazovány. Na kolizních doménách (v případě sdíleného přenosového média) využívá Ethernet mechanismus CSMA/CD, kdy se čeká na uvolnění média a při zasílání je kontrolováno, zda nepřichází data od jiné stanice — v takovém případě se data po náhodně zvoleném čase zasílají znovu. Metoda CSMA/CA používaná u bezdrátových sítí je odlišná. Zařízení čeká, dokud se médium neuvolní a následně si rámci RTS a CTS "vyžádá právo" médium využívat.

1.1.2 Síťová vrstva

Na úrovni síťové vrstvy je zajištěna konektivita mezi zařízeními. Protokoly této vrstvy musí být definovány na koncových zařízeních a směrovačích. Dnes jsou běžně využívány dva protokoly: IPv4 a IPv6.

Protokol IPv4[19] je definován v RFC 791. IPv4 adresa má 32 bitů a musí být pro dané zařízení unikátní. Vzhledem k relativně malé velikosti adresního prostoru byl navržen mechanismus NAT, díky kterému je možné tvořit sítě s lokálně platnými (a tedy znovupoužitelnými) adresami (10.0.0.0/8, 172.16.0.0/12 a 192.168.0.0/16) při zachování konektivity. Problém docházejících adres řeší také novější protokol IPv6.

IPv6[21] (RFC 2460) vzniklo jako následník protokolu IPv4. Hlavním důvodem bylo rozšíření adresního prostoru (adresa zde má velikost 128 bitů), což umožňuje přidělit adresu mnohem více zařízením i vytvářet víceúrovňové hierarchie. Je zde definován nový typ adresy *anycast*, používán k zaslání paketu na libovolné zařízení z definované skupiny. Samotný formát IP hlavičky byl výrazně zjednodušen, zároveň však byla vylepšena podpora pro rozšíření. V neposlední řadě byl definován standard IPSec určen pro unifikované zajištění autentizace, utajení a integrity pro všechny protokoly nad IPv6 provozované. Mnoho vylepšení (mezi nimi i právě anycast a IPSec) bylo zpětně doimplementováno do IPv4.

1.1.3 Transportní vrstva

Úkolem transportní vrstvy je přizpůsobit tok dat potřebám aplikace. Součástí hlavičky datagramu je zdrojový a cílový port, díky čemuž je umožněno využívat více služeb na jednom zařízení a vytvářet paralelní spojení.

Protokol TCP[20] (RFC 793) zajišťuje spolehlivý přenos dat a izoluje tvůrce aplikací od problémů ztrátovosti paketů, rozdílu zasílání a zpracovávání dat a přijímání paketů v nesprávném pořadí. Spojení je definovaným způsobem navazováno (tzv. TCP Three-way handshake) i ukončováno (pomocí přepínačů FIN, případně RST).

Protokol UDP[22] (RFC 768) slouží k nespolehlivému přenosu dat, čehož využívají ty protokoly aplikační úrovně, které buď řeší problémy přenosu vlastním způsobem (například TFTP) nebo je TCP funkcionalita nevhodná (jako v případě real-time aplikací, typickým představitelem je RTP[28] (RFC 1889) pro přenos zvukových a obrazových dat).

1.1.4 Aplikační vrstva

Do aplikační vrstvy spadají protokoly, které již přímo zajišťují požadovanou funkcionalitu. K nejpoužívanějším protokolům patří HTTP[23] (port tcp/80, RFC 1945 a další) pro přenos webových stránek, SMTP[24] (port tcp/25, RFC 5321) a IMAP[25] (port tcp/143, RFC 1176) pro zasílání elektronické pošty, DNS[26] (port tcp/53 a udp/53, RFC 1034) pro překlad doménových jmen na IP adresy a SSH[27] (port tcp/22, RFC 4251) pro vzdálenou správu zařízení.

1.2 Spotřeba

Zejména u bezdrátových zařízení je použití externího napájecího adaptéru nevhodné, a to hlavně z důvodu mobility. I když některé senzorové prvky[30] mají schopnost získávat veškerou energii z okolního prostředí, většina přístrojů je odkázána na napájení z baterie. Bezdrátová komunikace je přitom jedním z největších faktorů ovlivňujících spotřebu[31], proto bylo pro snížení spotřeby navrženo několik technologií[18].

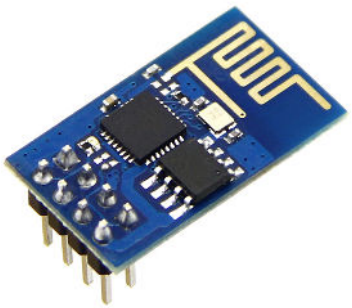
LoRA[32] (LOng RAnge) je bezdrátový komunikační systém s velmi dlouhým dosahem. Data lze touto technologií přenášet rychlostí až 50 kbps při dosahu v řádu desítek kilometrů. Standard IEEE 802.15.4[18] definuje nejnižší ISO/OSI vrstvy pro tzv. low-rate bezdrátové sítě. Povoluje

přenosovou rychlost až 250 kbps a maximální výkon 1mW. Signál má dosah v řádu desítek metrů.

Zařízení by obecně měla využívat technologie pro úsporu energie[29] (režim hlubokého spánku, radio duty cycling) v maximální možné míře.

1.3 Cena

Nízká cena zařízení je pro rozšíření Internetu věcí přirozeným požadavkem. V dnešní době je naštěstí trh tomuto fenoménu otevřen a cena nejlevnějších modulů se pohybuje v řádu jednotek dolarů. Typickým příkladem z této kategorie je populární modul ESP8266[8] (obrázek 2) od výrobce Espressif Systems známý od roku 2014. Jeho primárním cílem je sloužit jako spojení mezi existujícími mikrokontroléry a počítačovou sítí, může však sloužit i jako přístupový bod, monitorovat síťový provoz nebo jej využít k útočným účelům[7].



Obrázek 2: ESP8266-01[34]

I tak univerzální high-end zařízení, jakým je jednodeskový počítač Raspberry Pi 3 B (na obrázku 3) lze získat v ceně 36 dolarů[1]. Specifikace[9] (tabulka 1) takového zařízení je přitom srovnatelná s běžnými stolními počítači.

Procesor	Broadcom BCM2837 64b ARMv7 Quad Core, 1.2 GHz
RAM	1 GB
Wifi	ano (BCM43143)
Bluetooth	ano
GPIO	40 pinů
USB 2.0	4x
video výstup	HDMI

Tabulka 1: Parametry Raspberry Pi 3 B

[1] dle https://www.element14.com/community/community/raspberry-pi, údaj platný k 12. 4. 2017

Obrázek 3: Raspberry Pi 3[35]

1.4 Spolehlivost

Na rozdíl od běžných osobních počítačů a mobilních telefonů jsou na mnohá zařízení Internetu Věcí kladeny mnohem větší požadavky na spolehlivost nebo alespoň včasnou a správnou detekci problémů[33]. Typickým případem této problematiky je senzor určen k monitorování zdravotního stavu uživatele — zde může mít chybné chování (ať už vinou vadného hardwaru nebo softwarového vybavení) katastrofální důsledky. Na nesprávné chování přístroje by měl být uživatel včas upozorněn. Mnoho problémů týkajících se přenosu dat naštěstí řeší odpovídající protokoly vrstvy síťového přístupu (jedná se zejména o přenos poškozených rámců) a TCP, případně protokoly aplikační vrstvy (chybné pořadí paketů, chybějící data a duplicity).

Pro některé typy zařízení (zejména senzory) je spolehlivost definována spíše jako schopnost vykonávat svou funkci správně po dlouhou dobu, někdy i v řádu let. Pro taková zařízení je vhodné, aby obsahovala jen základní funkcionalitu — získání potřebných dat a jejich zaslání centrálnímu prvku — z důvodu úspory spotřeby energie a přenosového pásma.

2 Bezpečnost IoT

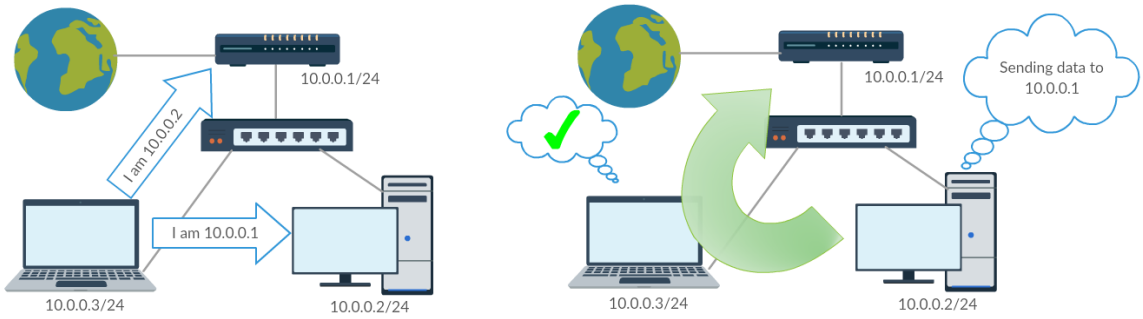
S fenoménem Internet of Things jsou ovšem spjata také značná bezpečnostní rizika. Již z definice je zřejmé, že zařízení musí být schopna komunikovat mezi sebou i s již existující infrastrukturou. Pro zajištění kompatibility je nutné využít existujících standardů — protokolů ISO/OSI vrstev. Jakékoli problémy v návrhu těchto protokolů se tedy projeví i u IoT zařízení. V této kapitole budou ve zkratce popsány problémy běžně využívaných technologií, jak mohou být útočníkem využity a jak se proti nim bránit.

2.1 ARP

ARP (Address Resolution Protocol) slouží ke zjišťování fyzických adres zařízení. Pracuje na spojové vrstvě a dotazy se šíří broadcastem, má tedy platnost pouze v lokální síti. Na této úrovni může útočník provést ARP Spoofing[39], kdy cílovým zařízením zašle svou MAC adresu spolu s falešnou IP adresou, díky čemuž je všechem provoz přeposílán přes něj a může tedy být analyzován a modifikován (obrázek 4). Provedení útoku je otázkou jediného příkazu:

ettercap -T -w /tmp/capturefile -M arp:remote /10.0.0.2/ /10.0.0.1/

Tomuto útoku se lze bránit použitím statických MAC adres, případně využitím funkcionality Dynamic ARP Inspection (kdy přepínač u příchozích rámců kontroluje, zda jednotlivé IP a MAC adresy patří k sobě).



Obrázek 4: ARP Spoofing

2.2 802.11

Většina IoT zařízení bude ke komunikaci pravděpodobně využívat bezdrátové připojení. I zde existují slabiny[39]. Při použití standardu 802.11 může být provoz při zvolení nedostatečného způsobu zabezpečení odposloucháván.

Pokud není provoz zabezpečen žádným způsobem, lze provoz odchytávat pouhým přepnutím síťové karty do tzv. monitor módu (neprobíhá asociace s AP).

Zastaralé zabezpečení WEP (Wired Equivalent Privacy) lze jednoduše prolomit sadou nástrojů aircrack-ng (výpis 1).

```
airmon-ng start wlan0 # monitor mode
airodump-ng mon0 # sniff all APs
airodump-ng -c <channel> --bssid <rbssid> -w <file> mon0 # sniff specific AP
aireplay-ng -4 -e <essid> -b <rssid> mon0
```

Výpis 1: Prolomení WEP zabezpečení (ChopChop)

Běžně používané WPA2/PSK zabezpečení lze prolomit slovníkovým útokem (výpis 2) pokud je zachycena autentizace klienta. Tu je však možné vynutit.

```
airmon-ng start wlan0 # monitor mode
airodump-ng mon0 # sniff all APs
airodump-ng -c <channel> --bssid <rbssid> -w <file> mon0 # sniff specific AP
aireplay-ng -0 2 -a <rbssid> -c <cbssid> mon0 # deauthentication
aircrack-ng -a2 -b <rbssid> -w <wordlist> <.cap_file> # crack file
```

Výpis 2: Prolomení WPA2/PSK zabezpečení

Je třeba pamatovat také na útoky na dostupnost. Přerušená komunikace je totiž v souvislosti s Internet of Things kritický problém, a to zejména v těchto modelových případech:

- Senzor nebude schopen včas informovat zbytek infrastruktury o nastálé události.
- Záznam s bezdrátové IP kamery nebude při ztrátě konektivity ukládan na server.

Kromě deautentizačního útoku zmíněného u prolamování WPA2 zabezpečení může útočník využít 2 jednoduché flood útoky:

- RTS Flood — zařízení bude neustále vyžadovat právo komunikovat s AP,
- CTS Flood — okolním zařízením bude zasílána informace, že kanál využívá někdo jiný.

Pokud není použito WPA2 Enterprise zabezpečení s MSCHAPv2, může útočník vytvořit falešný Access Point, přes který bude provoz směrován; opět hrozí únik informací a modifikace.

2.3 DHCP

DHCP (Dynamic Host Configuration Protocol) je určen k automatickému přiřazování IP adresy na základě adresy fyzické. Jsou však nastavovány i další parametry, mimo jiné také IP adresa defaultní brány, DNS servery nebo adresa NFS serveru pro případ nahrávání operačního systému ze sítě. Pokud tyto údaje cíli nezašle legitimní DHCP server, ale útočník[39], nastává nepříjemná situace:

- Zařízení s falešnou defaultní bránou zasílá provoz do cizích sítí přes útočníka (jako v případě ARP spoof útoku),

- falešné DNS servery mohou zajistit, aby při použití doménových jmen byla data zasílána jinam, než je očekáváno,
- při nahrání útočníkem upraveného systému je zařízení kompromitováno v plném rozsahu.

Útočník pak může doufat, že falešná konfigurace se k cíli dostane dříve, nebo může nejdříve na legitimním serveru zabrat všechny použitelné adresy pro sebe — této technice se říká DHCP Starvation. Řešením je statická konfigurace, případně technika DHCP Snooping, kdy jsou DHCP Offer zprávy z nedůvěryhodných portů přepínače zahazovány. Omezením počtu MAC adres lze zabránit DHCP Starvation útokům.

2.4 DNS

Zcela kritickým prvkem každé síťové infrastruktury je DNS server, který slouží k překladům IP adres na doménová jména a naopak. Po úspěšném MitM útoku nebo v případě ovládnutí DNS serveru může útočník zajistit, že vybrané servery budou podvrhovány[39], stejně jako v případě ovládnutí DHCP serveru.

MitM variantu lze jednoduše demonstrovat pomocí programu *ettercap* (výpis 3):

```
1 cat >> /etc/ettercap/etter.dns << EOF
2 vsb.cz A 10.0.0.3
3 *.vsb.cz A 10.0.0.3
4 EOF
5 ettercap -T -q -P dns_spoof -M arp // //
```

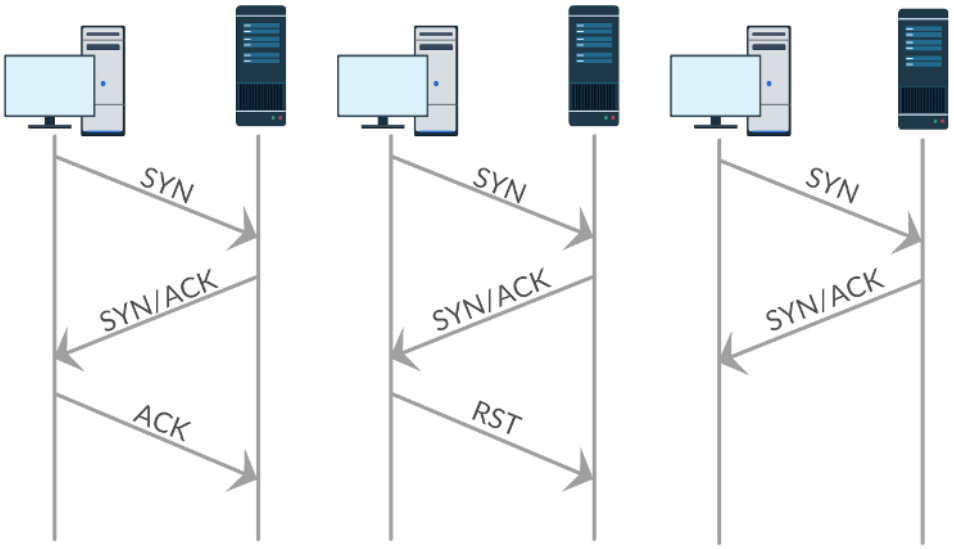
Výpis 3: MitM DNS Spoofing

2.5 Transportní protokoly

Uživatele často nemusí zajímat funkcionalita nižších ISO/OSI vrstev, požadovaná aplikace většinou spadá až do poslední (aplikační) vrstvy a ve většině případů používá TCP nebo UDP jako transportní protokol. Útočník opět získává možnost použít známé útoky z této oblasti, za zmínku stojí například SYN Flood, který zneužívá chování serveru při vytváření TCP spojení. Klient nejdříve zasílá datagram s příznakem SYN, server odpovídá příznaky SYN/ACK. Očekává se, že klient handshake dokončí (ACK). Server si přirozeně musí pamatovat částečně navázané spojení. Při velkém počtu požadavků, u kterých nebyl TCP handshake dokončen, může server přestat obsluhovat legitimní požadavky[39]. Tento útok je také výhodný v tom, že zdrojová (útočníkova) adresa může být podvržena. Princip útoku je znázorněn na obrázku 5.

2.6 Skenování portů

Aby byla zajištěna funkcionalita služby, musí na správných portech naslouchat odpovídající software. Není podstatné, zda se bude jednat o samostatný proces běžící v prostředí operačního



Obrázek 5: TCP Handshake (vlevo), SYN Scan (uprostřed), SYN Flood (vpravo)

systému nebo o funkcionalitu celistvého kódu. Otevřený port je detekovatelný známými skenovacími metodami. Ty nejznámější jsou obsaženy v nástroji *nmap*[40]. Zjištěním stavů portů je možné odhadnout účel systému a přítomné zranitelnosti.

```
1  $ nmap -sS -n example.com
2  Starting Nmap 7.40 ( https://nmap.org ) at 2017-03-27 14:19 CEST
3  Nmap scan report for example.com
4  Host is up (0.022s latency).
5  Not shown: 997 filtered ports
6  PORT    STATE SERVICE
7  22/tcp  open  ssh
8  80/tcp  open  http
9  443/tcp open  https
10
11 Nmap done: 1 IP address (1 host up) scanned in 5.13 seconds
```

Výpis 4: SYN scan (*nmap*)

Výpis 4 napovídá, že na testovaném zařízení běží SSH, HTTP a HTTPS. Další služby (pokud nějaké existují a běží na 1000 nejpoužívanějších portech) jsou chráněny firewallem. Použita byla velmi rychlá a účinná metoda SYN scan (obrázek 5). V následujícím příkladu je stejným nástrojem provedena detekce konkrétních programů zajišťujících dostupné služby, jejich verzí a také operačního systému.

```
1  $ nmap -sSV -O -p 22,80,443,55555 example.com
2  Starting Nmap 7.40 ( https://nmap.org ) at 2017-03-27 14:32 CEST
3  WARNING: RST from example.com port 22 -- is this port really open?
4  Nmap scan report for example.com
```

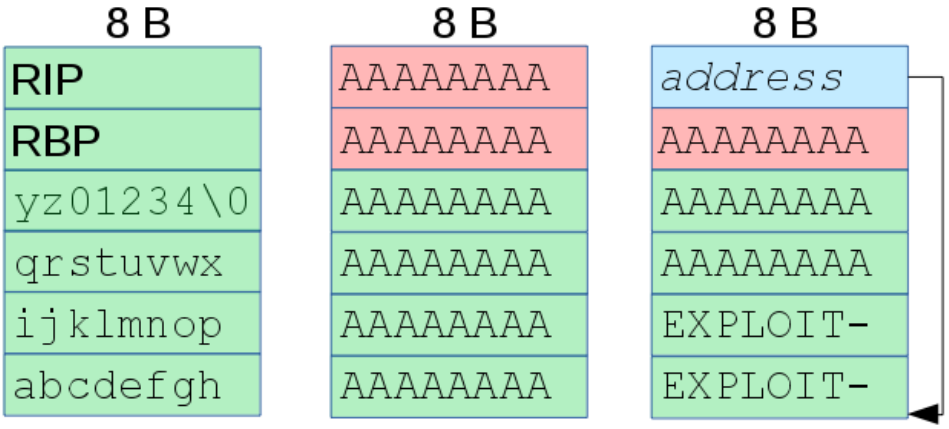
```
5  Host is up (0.014s latency).
6  PORT     STATE SERVICE VERSION
7  22/tcp   open  ssh     OpenSSH 6.7p1 Raspbian 5+deb8u3 (protocol 2.0)
8  80/tcp   open  http    nginx 1.6.2
9  443/tcp  open  ssl/http nginx 1.6.2
10 55555/tcp closed unknown
11 Aggressive OS guesses: Linux 3.11 - 4.1 (95%), Linux 4.4 (95%), Linux 3.2 - 3.8 (91%), QNAP
       NAS Firmware 3.8.3 (Linux 3.X) (91%), Linux 2.6.32 (91%), Android 5.0.1 (90%), Linux
       3.10 - 3.12 (90%), Wyse ThinOS (90%), Linux 3.18 (90%), Linux 2.6.18 - 2.6.22 (89%)
12 No exact OS matches for host (test conditions non-ideal).
13 Network Distance: 12 hops
14 Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
15
16 OS and Service detection performed. Please report any incorrect results at https://nmap.org/
       submit/ .
17 Nmap done: 1 IP address (1 host up) scanned in 18.92 seconds
```

Výpis 5: Detekce softwaru (*nmap*)

Pozn: Na testovaném zařízení skutečně běží Raspbian 8 (tedy Linux 4.4) se službami OpenSSH_6.7p1 Raspbian-5+deb8u3 a nginx 1.6.2. Nebyla ovšem odhalena služba OpenVPN 2.3.4, protože port tcp/1194 nespadá do defaultně testovaného rozsahu[38].

2.7 Softwarové zranitelnosti

U softwaru (a zejména u programů, jež se účastní síťového provozu) může i sebemenší zranitelnost umožnit spuštění nedůvěryhodného kódu. Riziko zde představuje zejména problematika správy paměti. Zde bude stručně popsán princip nejznámější zranitelnosti z této kategorie — Stack Overflow[41].



Obrázek 6: Přetečení zásobníku

Levé schéma na obrázku 6 popisuje běžný stav paměti po uložení 32B řetězce na zásobník. RIP a RBP jsou hodnoty registrů, které jsou platné pro nadřazenou funkci — po ukončení právě běžící funkce budou tyto hodnoty opět použity jako aktuální hodnoty registrů. Schéma uprostřed zobrazuje stav paměti při neúmyslném přetečení zásobníku. Řídící hodnoty jsou přepsány a proto dojde po ukončení funkce k pádu procesu s důvodem Chyba segmentace (Segmentation Fault). Třetí příklad je jiný. Pokud je obsah řetězce vytvořen "na míru", namísto hodnoty RIP je zapsána adresa začátku řetězce, kde je uložen škodlivý kód. Po skončení funkce je zde přesměrován tok programu.

Adresu začátku řetězce je v tomto případě nutné znát dopředu, zásobník musí být navíc spustitelný. Z těchto důvodů je provedení takových útoků na moderních systémech problematické. Navíc bylo vyvinuto množství ochranných mechanismů:

- RELRO — sekce obsahující ukazatele na další části procesu (GOT, tabulka destruktorů apod.) jsou umístěny před sekce pro data, u kterých hrozí přetečení (.data, .bss); GOT může být nastaveno jako read-only,
- Stack Protector — při volání funkce jsou po hodnotách registrů na zásobník vloženy speciální hodnoty, u kterých lze detekovat změnu ještě před samotným použitím falešné hodnoty RIP,
- DEP/NX — u důležitých struktur paměti je zablokována možnost spouštět kód,
- PIE (Position-independent executable) — kód může být umístěn na jakékoli adrese při zachování funkcionality,
- ASLR (Address Space Layout Randomization) — oblasti dat pro proces jsou nedeterministicky rozházené po paměti,
- Fortify — kritické funkce jsou nahrazeny vylepšenými variantami, které při porušení paměti zastaví program.

Přirozeně bylo vyvinuto mnoho technik, které umožňují některé z výše zmíněných ochran překonat. Mezi základní techniky patří:

- ret2lib — na zásobník jsou umístěny pouze argumenty, škodlivý kód je složen z postupného volání funkcí s předem známou adresou,
- ret2reg — pokud je škodlivý kód obsažen v bufferu, jehož adresa je uložena v registru, lze vhodnou instrukcí přejít na danou adresu,
- ROP chain — rozšíření ret2lib; kód je složen ze sekvencí instrukcí zakončených instrukcemi skoků nebo RET,
- NOP Sled — dlouhá sekvence instrukcí bez funkcionality zjednodušují hádání správné adresy.

Nelze se spoléhat ani na jazyky s automatickou správou paměti — příkazy jsou v konečném důsledku převáděny na jednotlivé instrukce procesoru. Vzhledem k tomu, že paměťové buňky nemají nekonečnou velikost jako v případě Turingových strojů, je stále možné využít chyb compilerů a interpreterů k získání přístupu k jiným částem paměti, než bylo určeno.

2.8 Zranitelnost webových aplikací

V mnoha situacích není třeba získat přístup, ale pouze informace. Za tímto účelem může být cílem zneužití chyb v návrhu samotné aplikace. V praxi se to nejvýrazněji projevuje v oblasti bezpečnosti webových aplikací. Ač útočník nezíská nad zařízením plnou kontrolu (i když i to je v některých případech možné), může narušit komunikaci mezi zařízeními nebo mezi zranitelným zařízením a klientem, případně získat data uložená v databázích. Mezi nejznámější zranitelnosti z této kategorie patří:

XSS[11] (Cross-site scripting) — Uživatelský vstup je prohlížečem zpracován jako HTML/Javascriptový kód, což nejčastěji vede k získání identifikátoru sezení (Session Hijacking), může však dojít až k úplné kompromitaci cíle.

CSRF[10] — oběť nevědomky provede akci zvolenou útočníkem, která je úspěšná kvůli chybně ošetřené autorizaci.

SQL Injection[10] — nesprávné ošetření uživatelských vstupů vede k úpravě SQL dotazu, což může vést k úniku informací, obcházení autentizačních mechanizmů i vzdálenému spuštění kódu.

2.9 Uživatelské účty a hesla

Na zařízení v továrním nastavení se lze typicky přihlásit pomocí předem známých přihlašovacích údajů. Předpokládá se, že uživatel si konfiguraci upraví a přihlašovací údaje vhodně změní (hesla odolné proti brute-force a slovníkovým útokům, případně autentizace pomocí klíčů). Jak ale události posledních let[6] ukázaly, situace má do ideálu daleko.

2.10 Bezpečnostní incidenty v IoT

V roce 2013 byl odhalen červ Linux.Darlloz. Využíval PHP zranitelnost CVE-2012-1823 k přístupu do systému, následně prováděl eskalaci práv slovníkovým útokem a vytvářel zadní vrátka. Bylo navrženo několik verzí pro nejpoužívanější platformy (x86, ARM, PPC, MIPS, MIPSEL). Darlloz se také pokoušel deaktivovat červa Linux.Aidra[12].

Červ Aidra, odhalen v roce 2013, se zaměřoval na IoT zařízení s běžícím Telnetem a defaultním či žádným heslem. Pokoušel se ochránit zařízení před ostatními druhy malwaru a vykonával příkazy zasílané přes IRC. Červ byl uchováván pouze v RAM, byl tedy ostraněn pouhým restartem zařízení.

Qbot byl poprvé odhalen v roce 2009. Je určen pro operační systémy Windows. Využívá slabin webových stránek, po stažení se injektuje do procesu explorer.exe a z něj infikuje další

spuštěné procesy. Následně získává přihlašovací údaje z uložených konfigurací, síťového provozu a pomocí brute-force útoků. Některé varianty využívaly tzv. server-based polymorfismu, tedy měnily svůj kód podle instrukcí serveru.

BASHLITE botnet obsahuje přes 1 milion ovládaných zařízení, převážně IP kamery a DVR jednotky. Botnet lokalizoval Telnet servery a následně se snažil hrubou silou získat přihlašovací údaje nebo využíval externí scannery k vyhledávání zařízení náchylných na známé problémy. Zdrojový kód je zveřejněn.

Nejznámějším bezpečnostním incidentem, který se za poslední měsíce v oblast Internet of Things udál, je však bezesporu vznik Mirai botnetu. Kód použitého malwaru je dnes volně dostupný například na `https://github.com/0x27/linux.mirai`. Malware se připojoval k cílovým zařízením pomocí SSH nebo Telnetu a známých/defaultních přihlašovacích údajů. Zařízení pak plnilo příkazy řídícího serveru. Cílený útok Mirai botnetu ve spolupráci s BASHLITE botnetem na server KrebsOnSecurity.com v září 2016 pak dosáhl intenzity až 650 Gbps[13], několik dní poté se francouzský provider DVH stal cílem útoku s takřka dvojnásobnou silou (1.1-1.5 Tbps). V říjnu téhož roku byla zasažena infrastruktura Managed DNS, kde se předpokládá, že intenzita přesáhla 1.2 Tbps.

Z výše uvedených reálných případů a vektorů útoků je zřejmé, že potenciální útočník tedy může na IoT zařízení používat stejné nebo velmi podobné metody, které jsou již dlouho známé a pro které je dostupná jejich implementace. Internet of Things zařízení navíc zvýšeným způsobem ohrožují naše soukromí, riziko představují zejména kamery a wearables zařízení.

3 Detekce zranitelností

Cílem této práce je vytvořit software, který by pomáhal analyzovat bezpečnost firmwaru určeného pro zařízení Internetu věcí. Ověření, zda je libovolný software bezpečný či nikoli, je však algoritmicky nerozhodnutelné[14], což plyne z Riceovy věty[15]:

Věta 1 *"Každá netriviální vstupně-výstupní vlastnost programů je nerozhodnutelná."*

Důkaz. Představme si algoritmus, který ze vstupu (strojového kódu testovaného programu) určí, zda je zranitelný či nikoli. Je zřejmé, že vlastnost "zranitelnost" je vstupně-výstupní — bezpečnost programu závisí na jeho kódu. Tato vlastnost je také netriviální — existuje bezpečný program (například takový, který ihned skončí) i program zranitelný. ∎

Pravdě se tedy lze pouze přiblížit — například technikami statické či dynamické analýzy.

Statická analýza[4] spočívá v kontrole zdrojového kódu bez nutnosti provádění testovaných instrukcí. Základní funkcí nástroje pro statickou analýzu je kontrola syntaxe, dále pak vyhledávání nebezpečných funkcí (klasickým příkladem je *gets()* v C/C++). Pokročilejší metodou může být například tzv. Taint analýza, kde jsou všechny uživatelské vstupy označené jako nebezpečné () a u všech funkcí se následně kontroluje, zda není taková hodnota použita nevhodným způsobem. Takto lze například odhalit i zranitelnosti typu SQL Injection.

Dynamická analýza[17] je prováděná nad aktuálně běžícím kódem. Kontroluje se zejména správná práce s pamětí. Tato metoda v žádném případě nezaručuje odhalení všech problémů (ve většině případů je nereálné otestovat všechny stavy programu), poukazuje však na ty, které se při běžném používání vyskytnou.

Žádnou z uvedených možností však nelze na řešený problém vhodně aplikovat. Dynamická analýza už z definice vyžaduje, aby byl testovaný software spuštěn. Cílem práce je provádět analýzu firmwarů (souborů v binární podobě úzce spjatých s hardwarem, pro které byly určeny), zařízení, na kterém by byl firmware spustitelný, tedy nemusí být k dispozici. Kód by bylo možné emulovat (např. pomocí QEMU), nicméně konfigurace emulátoru může být pro každý vzorek rozdílná a nemusí existovat způsob, jak ji pro neznámý vzorek detekovat. IoT zařízení také často využívají periferie, které by v emulátoru dostupné nebyly (uvažme například Raspberry Pi a širokou škálu možností díky vyvedení GPIO pinů).

Binární podoba firmwaru také značně komplikuje nasazení metod statické analýzy. Spustitelný kód je již v podobě strojového kódu a tedy obtížně čitelný. Strojový kód není jednotný — existuje nespočetné množství výrobců IoT zařízení a od toho se odvíjí i různorodost zařízení a architektur použitých komponent.

Tato práce se tedy zaměřuje na ta zařízení, které ve svém firmwaru obsahují unifikovaný (jistým způsobem standardizovaný) software pro řešení základních požadavků na funkcionalitu (práce se soubory, procesy, periferiemi, pamětí atd.) — tedy operační systém. Půjde zde pouze o systém linuxového typu, neboť mnoho populárních firmwarů právě více či méně modifikovaný

Linux využívá. Linux a běžně používané balíčky bývají open-source, lze tedy očekávat i jakousi komunitní kontrolu kódu (peer review process).

Analýza linuxového systému má navíc jednu velkou výhodu — uplatňuje se zde zásada "vše je soubor"[16]. Pokud se dostaneme ke vhodným souborům, získáme úplný přehled o možnostech daného zařízení.

3.1 Extrakce

Nejprve je nutné převést analyzovaný soubor do podoby, ve které je možné k souborům přistupovat. K tomu bude využita Python knihovna binwalk[5], jejíž funkcionalitu nejlépe demonstruje stejnojmenný nástroj. Binwalk pracuje na principu hledání signatur známých typů souborů a dokáže extrahovat běžně používané typy archivů. Je vyvíjen skupinou /dev/ttyS0[1].

3.2 Analýza konfigurace

Ve velkém množství extrahovaných dat musíme najít onu linuxovou část. Nabízí se jednoduchý způsob - vyhledání adresářů typických pro UNIX systémy. Jedná se například o adresáře bin, etc, usr, lib. Ty jsou umístěny v kořenovém adresáři. Po extrakci se může stát, že takových kandidátů na kořenový adresář bude nalezeno několik. Implementace bude v takovém případě zkoumat všechny nalezené "systémy".

Již v této fázi je k dispozici mnoho informací o systému, které pro nás mohou být z bezpečnostního hlediska zajímavé.

3.2.1 Seznam uživatelů

Seznam lokálně definovaných uživatelů se nachází v souboru */etc/passwd*. Zde zjistíme, které účty jsou uzamčené nebo mají zakázán příkazový řádek. Účty určené pro uživatele mají obvykle UID 1000 nebo více, superuživatel vystupuje pod UID 0. Nápadné jsou zejména účty s duplicitní hodnotou UID, kdy se (zejména při UID=0) může jednat o jednoduchý backdoor.

3.2.2 Informace o operačním systému

Informace o použité distribuci lze najít v souboru */etc/os-release*, případně v dalších souborech */etc/*-release*, podobná informace se někdy může nacházet i v souboru */etc/issue*, který slouží pro zobrazení informací pro ještě nepřihlášeného uživatele. Útočník s touto informací může odhadnout, které služby na serveru poběží a které exploity by bylo vhodné vyzkoušet.

3.2.3 Informace o jádře

Zjištění verze jádra je pro útočníka naprosto klíčové — detekce zranitelné verze jádra s dostupným exploitem vede často k okamžitému ovládnutí cíle v plném rozsahu (není třeba žádná

[1] http://www.devttys0.com/

následná eskalace práv). Detekce verze jádra je v případě běžícího systému jednoduchá, poslouží nám některý z příkazů *dmesg | grep Linux*, *uname -r* nebo *cat /proc/version*. V našem případě to však nebude použitelné — příkazy na extrahovaném systému spouštět nelze a soubor */proc/version* je vytvářen až za běhu. Můžeme však zkontrolovat soubory */boot/vmlinuz-** a očekávat, že používaný kernel bude ten nejnovější.

3.2.4 Příkazy spouštěné při určité události

Kontrolou souborů */etc/crontab*, */etc/cron.*/** a */var/spool/cron/** získáme seznam příkazů, které jsou prováděny v pravidelných intervalech. Dále mohou být zajímavé příkazy, které se spouští při běžně se vyskytující události:

- spuštění systému — */etc/rc.local*, */etc/inid.d/**,
- změna runlevelu — */etc/rc*.d/**, */etc/systemd/system*, */etc/systemd/user*,
- spuštění shellu — */etc/bashrc*, *$HOME/.bashrc*,
- spuštění grafického prostředí / přihlášení — *$HOME/.xsession*, */etc/X11/Xsession.d/*.

3.2.5 Konfigurace firewallu

Získání konfigurace firewallu zjednoduší útočníkovi fázi enumerace — vidí, které porty jsou otevřené světu (a může tedy odhadnout službu, která zde poběží) a zároveň zjistí, které porty jsou chráněny a je tedy zbytečné cokoliv zkoušet. Pro ochranu bude použit pravděpodobně firewalld nebo iptables, uložené konfigurace bývají v */etc/firewalld/*, */etc/iptables/rules.v** a */etc/sysconfig/ip*tables*.

3.2.6 Konfigurace služeb

Jak je z výše uvedených příkladů zřejmé, téměř veškerá konfigurace se nachází ve složce */etc*. Zde je pro útočníka zajímavá zejména konfigurace systémových služeb a služeb, které jsou na firewallu povoleny.

3.2.7 Informace o síťových rozhraních

V souvislosti s nastavením firewallu bude pro útočníka jistě zajímavá i konfigurace síťových rozhraní. Umístění se pro různé distribuce liší, za pokus stojí */etc/network/interfaces*, */etc/dhcpcd.conf* nebo */etc/sysconfig/network-scripts*.

3.2.8 Informace o připojovaných svazcích

V souboru */etc/fstab* lze nalézt seznam zařízení, které jsou při startu systému připojeny. Útočníka mohou zajímat například síťové svazky, ke kterým se může dostat i ovládnutím jiného zařízení.

3.2.9 Certifikáty a privátní klíče

V neposlední řadě získává útočník přístup k certifikátům, veřejným klíčům a hlavně klíčům privátním. Ty lze nejčastěji nalézt v adresářích */etc/ssl*, */root/.ssh/* a */home/*/.ssh/*. Díky nim je možné získat přístup k dalším důvěryhodným zařízením nebo vytvářet zařízení falešná.

3.3 Detekce softwaru

Pro analýzu jsou ale nejdůležitější seznamy nainstalovaných balíčků a jejich verzí. Ty lze zjistit jednoduše, pokud je na systému tzv. package manager (správce balíčků). Tyto nástroje se starají o správu balíčků a zajišťování konzistence v případě provádění změn. Seznam balíčků, jejich verze a další údaje tedy musí být v systému někde uloženy (tabulka 2]:

Správce balíčků	Umístění (dle manuálových stránek a na testovaných vzorcích)
dpkg	/var/lib/dpkg/status
ipkg	/usr/lib/ipkg/status
opkg	/usr/lib/opkg/status, /usr/lib/opkg/info/*.control
rpm	/var/lib/rpm/Packages (Berkeley DB)

Tabulka 2: Umístění databází správců balíčků

Verze balíčků má specifický formát, výsledný program vychází ze syntaxe deb-version[37], která ve velké míře koresponduje se záznamy CVE XML Vulnerability Feeds (popsáno dále) a vyhovuje jí také většina záznamů ve správcích ipkg a opkg v analyzovaných vzorcích.

Verze je zapsána ve tvaru [epoch:]upstream-version[-debian-revision], kde

- epoch — kladné celé číslo; může být vynecháno (ekvivalentní s hodnotou 0),
- upstream-version — nejpodstatnější část; měla by začínat číslem, obsahuje písmena, čísla a znaky . + - : ~,
- debian-revision — podrobnější specifikace vzhledem k upstram-version (v CVE záznamech se téměř nevyskytuje).

3.4 Detekce zranitelností

Chyby, které jsou v programech každodenně nalézány, jsou popisovány prostřednictvím CVE záznamů. CVE záznam s unikátním identifikátorem obsahuje popis problému, CVSS skóre (hodnocení vážnosti problému) a reference na další relevantní materiály. Na adrese `https://nvd.nist.gov/vuln/data-feeds` je možné si stáhnout databáze CVE záznamů (označované jako XML Vulnerability Feeds) pro jednotlivé roky včetně informací, které produkty v jakých verzích jsou zranitelností afektovány.

3.4.1 Specifikace výrobce

Výpis 6 ilustruje strukturu CVE záznamu z XML Vulnerability Feeds. Je zde přítomen atribut *vendor*, který umožňuje rozlišovat různé produkty se stejným názvem. Jako příklad může sloužit produkt *cups* od vendorů *easy_software_products* (CVE-2007-3387), *cups* (CVE-2005-4873) a *apple* (CVE-2010-0302). V databázích správců balíčku se atribut Vendor ovšem nevyskytuje. V implementaci budou nahlášeny všechny shody názvů produktů a verzí, zhodnocení relevance výsledků pak bude ponecháno na uživateli.

```
1  <entry type="CVE" name="CVE-2014-0160" seq="2014-0160" published="2014-04-07"
       modified="2017-01-06" severity="Medium" CVSS_version="2.0" CVSS_score="5.0"
       CVSS_base_score="5.0" CVSS_impact_subscore="2.9" CVSS_exploit_subscore="10.0"
       CVSS_vector="(AV:N/AC:L/Au:N/C:P/I:N/A:N)">
2    <desc>
3      <descript source="cve">The (1) TLS and (2) DTLS implementations in OpenSSL 1.0.1
           before 1.0.1g do not properly handle Heartbeat Extension packets, which allows remote
           attackers to obtain sensitive information from process memory via crafted packets that
           trigger a buffer over-read, as demonstrated by reading private keys, related to
           d1_both.c and t1_lib.c, aka the Heartbleed bug.</descript>
4    </desc>
  . . .
5    <vuln_soft>
6      <prod name="openssl" vendor="openssl">
7        <vers num="1.0.1d"/>
8        <vers num="1.0.1e"/>
  . . .
9      </prod>
10   </vuln_soft>
11 </entry>
```

Výpis 6: Vzorový CVE záznam

3.4.2 Rozdílný formát verzí

Další problém spočívá ve faktu, že formát verze není u všech produktů shodný. Tabulka 3 obsahuje několik ukázkových případů. Nezbývá než předpokládat, že verze balíčků u takto problematických produktů bude definována podobným zápisem a bude tedy možno skutečnou a zranitelnou verzi porovnávat.

3.4.3 Rozdílná přesnost verzí

Z rozdílným zápisem verzí souvisí i jejich přesnost — u některých produtků se i při použití stejného způsobu zápisu zachází při určení verze do větší hloubky. V tabulce 3 je to ilustrováno produkty roundcube_webmail (tři části), webmin (dvě části) a internet_explorer (jedna část).

CVE ID	Produkt	Verze
CVE-2015-2172	dokuwiki	2014-09-29b
CVE-2015-2180	roundcube_webmail	1.1.0
CVE-2007-5066	webmin	1.360
CVE-2007-5348	windows-nt	vista
CVE-2007-5348	internet_explorer	6
CVE-2007-5348	report_viewer	2008
CVE-2007-5348	windows	2003_server

Tabulka 3: Případ užití *Formáty verzí pro různé produkty*

Vhodným řešením je volbu této "přesnosti" ponechat na uživateli. Takto může být při nalezení příliš malého/velkého množství shod analýza provedena znovu s vhodnějšími parametry. Úplné ignorování verze pak umožní odhalit i situace se zcela nekompatibilními formáty verzí.

3.4.4 Přítomnost hodnoty epoch

Případná hodnota epoch u verze instalovaného balíčku není jednotně uváděna v CVE záznamech. V těchto případech nelze jednoznačně určit, zda je tato hodnota ignorována nebo se stala součástí hodnoty upstream-version. Způsob vyhodnocení těchto situací bude opět ponechán na uživateli.

3.4.5 Jednoznačnost názvů produktů

Porovnání zranitelných produktů a produktů přítomných na testovaném systému není jednoznačné. Samotný název se totiž může lišit — známým představitelem této anomálie je Apache HTTP Server, jenž se v Debian systémech skrývá pod označením apache2, ale na distribucích vycházejících z Red Hat Enterprise Linux nese označení httpd. Rozdílné názvy pro stejný objekt lze také nalézt v samotných CVE záznamech — 72[1] CVE identifikátorů se vztahuje k balíčku kernel, jenž na některých distribucích popisuje jádro systému. Mnohem více shod (4758[1]) však získáme hledáním produktu linux_kernel.

3.5 Vyhledání exploitů

Pokud známe čísla CVE, jimiž je systém afektován, můžeme zjistit, zda již byly vytvořeny exploity, které na dané zranitelnosti cílí. To je důležité jak pro útočníka (jak se nejrychleji do systému dostat), tak pro programátora/správce systému (jak akutní je daný problém). Exploit-db (https://www.exploit-db.com) je databáze veřejně známých exploitů spravována společností Offensive Security. Na stránce `http://cve.mitre.org/data/refs/refmap/source-EXPLOIT-DB.html` (obrázek 7) lze nalézt tabulku vztahů exploit–CVE.

[1] údaje platné k 14. 4. 2017

Obrázek 7: Exploit-db

4 Locasploit

V minulé kapitole byl po teoretické stránce popsán proces analýzy firmwaru, jejímž výstupem by mohla být technická zpráva o úrovni bezpečnostního rizika. V softwaru umožňujícím provádět takovou analýzu tedy musí být implementovány nejméně tři případy užití — *Update vulnerability database*, *Analyze* a *Generate report*. Vzhledem k univerzálnímu přístupu k lokalizaci zranitelností by však bylo škoda omezit se jen na tyto případy užití — vždyť stejným způsobem je možné testovat i běžící systém, a to aktuálně používaný i vzdálený. K případům užití bude tedy ještě přidáno *Create SSH connection* a fáze extrakce a hledání souborového systému v Analýze se bude provádět pouze při testování systémové image. Rovněž může být výhodné mít možnost generovat zprávu porovnávající dva různé vzorky — bude tak možné například ověřit, že v novější verzi produktu byla konkrétní chyba opravena. Vzniká tedy další případ užití, *Generate diff report*. Tabulky 4, 5, 6, 7 a 8 popisují jednotlivé případy užití.

I Update vulnerability database
1. Uživatel zadá požadavek na aktualizaci
2. Systém aktualizuje databázi zranitelností
3. Systém aktualizuje databázi exploitů

Tabulka 4: Případ užití *Update vulnerability database*

II Create SSH connection
1. Uživatel zadá požadavek na vytvoření spojení
2. Systém vytvoří spojení na vzdálené zařízení

Tabulka 5: Případ užití *Create SSH connection*

III Analyze
1. Uživatel požádá o analýzu daného vzorku
2. Systém zkontroluje zvolenou metodu
3. Systém získá základní informace o systému
4. Systém získá seznam instalovaných balíčků
5. Systém porovná seznam balíčků se seznamem zranitelností
6. Systém vyhledá exploity pro konkrétní zranitelnosti
2a) Je zvolena analýza image
2a1. Systém provede extrakci systémové image
2a2. Systém lokalizuje kořenový souborový systém

Tabulka 6: Případ užití *Analyze*

Jednotlivé případy užití je možné používat nezávisle na ostatních, i některé kroky (typicky sběr informací o systému) mohou být využity ve zcela odlišných situacích. Je tedy logické, aby byla implementace rozdělena do jednotlivých modulů (komponent) a uživatel by tyto komponenty využíval prostřednictvím unifikovaného rozhraní. Rozhodl jsem se proto požadovanou

IV Generate report
1. Uživatel požádá o generování zprávy pro daný vzorek 2. Systém vygeneruje zprávu

Tabulka 7: Případ užití *Generate report*

V Generate diff report
1. Uživatel požádá o generování zprávy pro 2 dané vzorky 2. Systém vygeneruje zprávu

Tabulka 8: Případ užití *Generate diff report*

funkcionalitu zahrnout do frameworku Locasploit, který vyvíjím od ledna 2016. Locasploit je psán v jazyce Python3 a jeho zdrojový kód je dostupný pod licencí GNU/GPLv2 na adrese `https://github.com/lightfaith/locasploit`. Jeho cílem je zejména:

- možnost vytváření modulů,
- multiplatformita,
- možnost analýzy konfigurace systémů,
- možnost analýzy síťového nastavení a provozu,
- podpora práce se slovníky pro kryptoanalytické problémy,
- unifikovaný přístup pro práci s lokálním i vzdáleným zařízením,
- čitelnost zdrojového kódu.

V této kapitole budou nejprve uvedeny klíčové soubory frameworku, jejichž funkcionalitu může uživatel při vytváření vlastních modulů využít. V druhé části bude popsána databázová struktura a principy ukládání dočasných i perzistentních dat. Třetí část se již bude věnovat samotnému zdrojovému kódu, jenž implementuje požadovanou funkcionalitu. Na závěr budou představeny nejdůležitější uživatelské příkazy a bude popsána šablona nového modulu.

4.1 Klíčové soubory

Soubory, které budou v této části popsány, tvoří samotné jádro celého frameworku. Na rozdíl od modulů nelze tyto soubory modifikovat za běhu programu — jakákoli změna se projeví až po opětovném spuštění.

4.1.1 locasploit.py

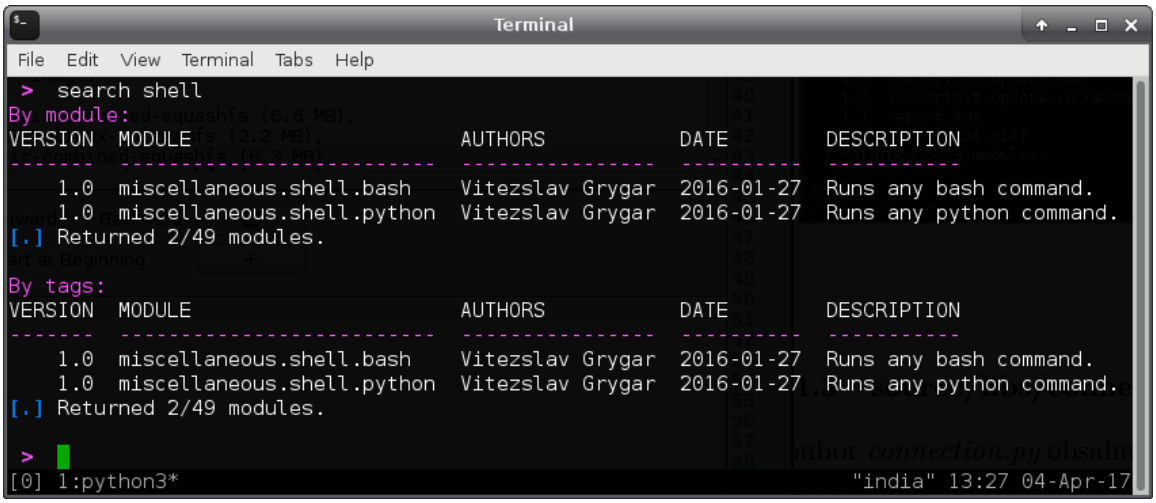
Soubor *locasploit.py* je skript spouštěn uživatelem (podle výpisu 7). Načítají se zde dostupné moduly a probíhá spouštění příkazů od uživatele nebo ze seznamu příkazů.

```
1  ./locasploit.py
2  ./locasploit.py -c soubor_se_seznamem_prikazu
```

Výpis 7: Spuštění frameworku Locasploit

4.1.2 source/libs/commands.py

Soubor *commands.py* obsahuje funkci *execute_command()*, ve které jsou zpracovávány příkazy zadané uživatelem. Dále jsou zde umístěny metody pro zobrazení modulů vyhovujících hledanému kritériu (obrácek 8), detailních informací o modulech (obrácek 9) a nápovědy ve standardizované podobě.



Obrázek 8: Vyhledání modulů

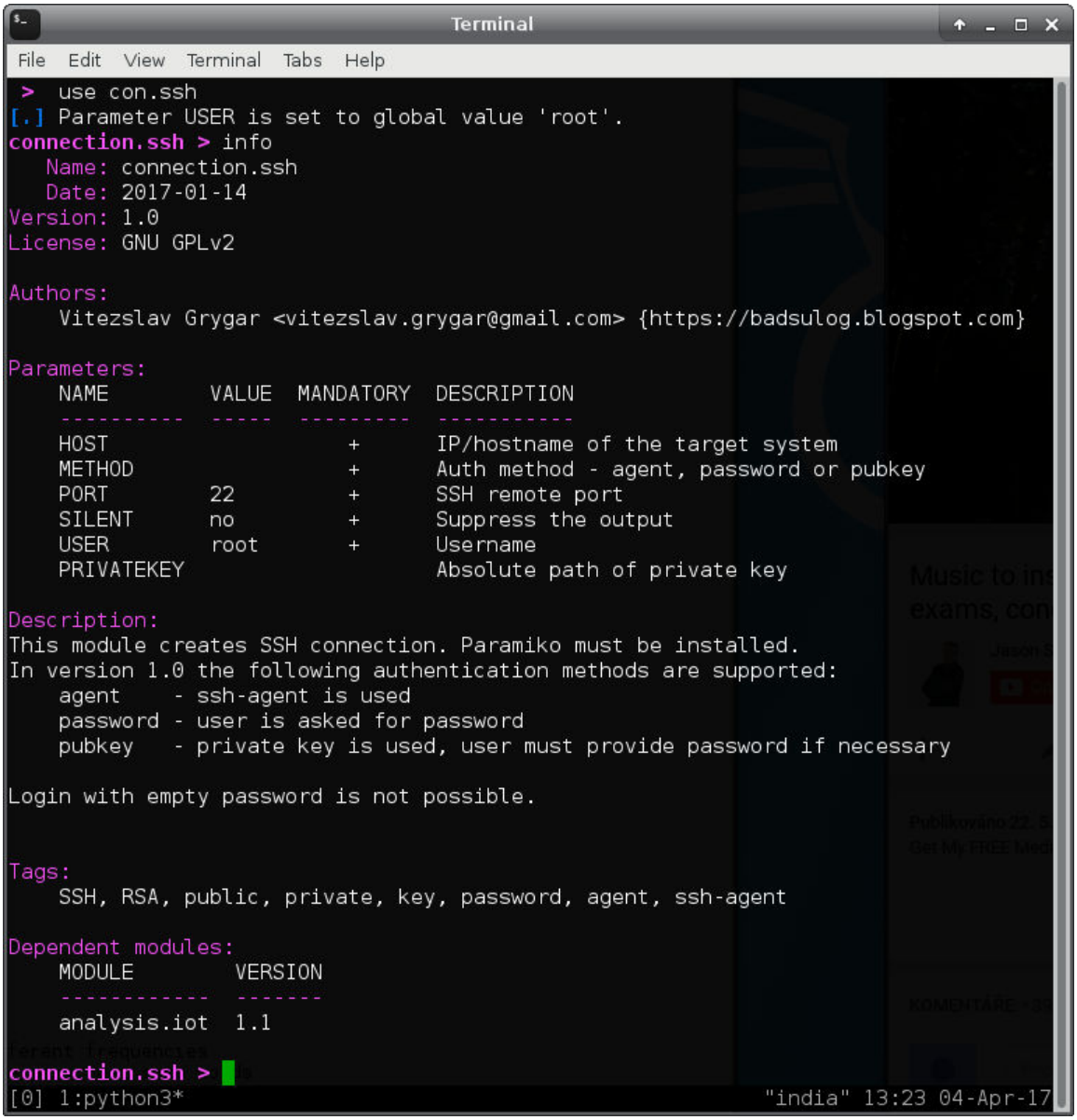
4.1.3 source/libs/connection.py

Soubor *connection.py* obsahuje třídu Connection, která slouží k definici vzdáleného zdroje (např. SSH session). Vytvořené objekty typu Connection jsou pro tvůrce modulů dostupné v seznamu *lib.connections*, uživatel si je může zobrazit příkazem *connections*.

4.1.4 source/libs/db.py

Soubor *db.py* obsahuje třídu DB, ve které jsou definovány základní funkce pro práci s databází (položení jednoho či více dotazů, získání seznamů tabulek, commit a uzavření databáze). Dále jsou zde definovány třídy pro jednotlivé používané databáze se specifickými metodami.

- *analysis.db* - databáze určená k uchovávání informací o analyzovaných systémech
- *checksum.db* - databáze definující vztah mezi souborem s konkrétním obsahem a aplikací (pro účely detekce)



Obrázek 9: Detailní informace o modulu

- *dict.db* - databáze slovníků a slov pro účely analýzy textu nebo slovníkových útoků
- *vuln.db* - databáze obsahující známé zranitelnosti (CVE) a odpovídající aplikace

4.1.5 source/libs/define.py

V souboru *define.py* jsou definovány konstanty pro testování chyb (*IO_ERROR*, *DB_ERROR*), výsledků testů použitelnosti modulů (*CHECK_**) a dalších. Také zde po spuštění frameworku dochází k definování základních globálních parametrů (např. aktuální uživatelské jméno, proměnná PATH apod.) a deklarace struktur nezbytných pro kooperaci modulů a práci s databázemi

(*tb*, *db*, *dicts*, *scheduler*, *connections*). Tento soubor je importován do souboru *include.py* jako 'lib', zmíněné struktury budou tedy přístupné jako *lib.<nazev>*.

4.1.6 source/libs/include.py

Soubor *include.py* obsahuje definice podpůrných funkcí (např. *exit_program()*, *natural_sort()*). Zde jsou také importovány důležité soubory ze složky *source/libs/* - po importování *include.py* je pak funkcionalita z těchto souborů přístupná i dále.

4.1.7 source/libs/io.py

Soubor *io.py* obsahuje metody pro práci se soubory a adresáři. Většina metod zde uvedených má společné dva parametry: *system* a *path*. Path je běžná cesta k souboru, system definuje počáteční bod, od kterého se cesta bude odvíjet. Jedním ze základních cílů tohoto frameworku je totiž maximální jednoduchost vytváření modulů, bylo tedy nutné vytvořit unifikovaný přístup pro různé případy.

Tuto problematiku lze dobře ilustrovat modulem *linux.enumeration.users*, jehož cílem je otevřít soubor */etc/passwd* a získat informace o uživatelích. Nelze však přistupovat přímo k souboru */etc/passwd*, soubor se totiž může nacházet:

- na lokálním systému,
- v podadresáři lokálního systému:
  - chroot,
  - rozbalený image (firmware),
  - systém na jiném diskovém oddílu,
- na vzdáleném systému:
  - dostupném přes SSH,
  - dostupném přes FTP,
  - ...

Za normální situace by pro každý z těchto případů musel existovat specifický modul. Problém elegantně řeší unifikovaná funkce *io.read(activeroot, '/etc/passwd')*, kde activeroot popisuje cílový systém (zvolený uživatelem nebo nastaven v rodičovském modulu). Běžné hodnoty activeroot jsou:

- */* - aktuální systém,
- */tmp/folder* - podadresář lokálního systému,
- *ssh://root@jiny_system:22/* - vzdálený systém (dostupné systémy a hodnoty pro activeroot lze vypsat příkazem *connections*)

4.1.8 source/libs/log.py

Metody z *log.py* ulehčují tvůrcům modulů výpisy na obrazovku, zejména standardizované (barevně odlišené) výpisy pro stavy *ok*, *info*, *warning* a *error* (vedoucí v patrnosti to, zda jsou volány z jiného vlákna či nikoli).

4.1.9 source/libs/scheduler.py

Soubor *scheduler.py* obsahuje dvě třídy - *Job* a *Scheduler*. Job je určen pro specifikaci modulu, který má běžet na pozadí. Scheduler pak slouží k vytváření těchto objektů a práci s nimi. Může také uchovávat reference na běžná vlákna, kterým může být např. při požadavku o ukončení programu tato informace zaslána. Modul určen pro spuštění na pozadí musí v metodě *run()* vracet referenci na objekt odvozen z *threading.Thread*.

4.1.10 source/libs/search.py

Soubor *search.py* obsahuje třídy a metody umožňující vyhledávání mezi moduly na základě zvolených podřetězců. Uživatel může použít tradiční operátory v nejznámějších způsobech zápisu:

- and, &&, &,
- or, ||, |,
- not, !,
- kulaté závorky pro změnu priority.

Pokud nejsou použity výše zmíněné operátory, jsou výsledky zobrazeny zvlášť podle výskytu (prohledává se název modulu, tagy, názvy parametrů, autoři, závislosti a verze modulu). Je podporován také zkrácený zápis názvu modulů — 'linux.enumeration.kernel' lze například zapsat jako 'li.e.k' nebo 'lin', pokud je zkrácený zápis jednoznačný.

4.1.11 Další soubory v adresáři source/libs

- *source/libs/author.py* - definuje třídu *Author*, díky které je možné specifikovat autory jednotlivých modulů,
- *source/libs/parameters.py* - definuje třídu *Parametr* pro specifikování parametru modulu (hodnota, povinnost, popis),
- *source/libs/statistics.py* - obsahuje některé základní statistické funkce.

4.2 Úložiště dat

Data generovaná frameworkem mohou být ukládána dvěma způsoby — dočasně pro uchování mezivýsledků a předávání informací mezi jednotlivými moduly, a dlouhodobě (persistentně), kdy bude zajištěn přístup k těmto datům i po opětovném spuštění frameworku či restartu systému. Pro perzistentní uchování dat je k dispozici několik SQLite databází. SQLite bylo zvoleno pro možnost ukládat data do souboru bez nutnosti využívat separátní server[36].

4.2.1 Temporary Base

Pro dočasné ukládání dat je k dispozici tzv. Temporary Base (*tb*). Princip je velice jednoduchý — jedná se o slovník, do kterého je v modulech možné ukládat jakákoli data pod definovaným klíčem.

Použití v modulu je naznačeno ve výpisu 8.

```
1  tb['klicek'] = 'hodnotka'
2  tb['policko'] = [0, 'slovicko']
3  tb['slovnicek'] = {'A': 'B', 'C': 'D'}
```

Výpis 8: Přístup k TB (Modul)

Použití ve frameworku:
Pro vkládání dat do TB jsou uživateli k dispozici tři moduly:

- *tb.insert* - uložení řetězce VALUE pod daným klíčem KEY
- *tb.move* - přesun dat z klíče KEY1 pod klíč KEY2
- *tb.readfile* - uložení obsahu souboru INPUTFILE na systému ACTIVEROOT pod klíčem KEY

Data pod daným klíčem (případně i dalšími podklíči) lze zobrazit pomocí příkazu *tb*, podle výpisu 9:

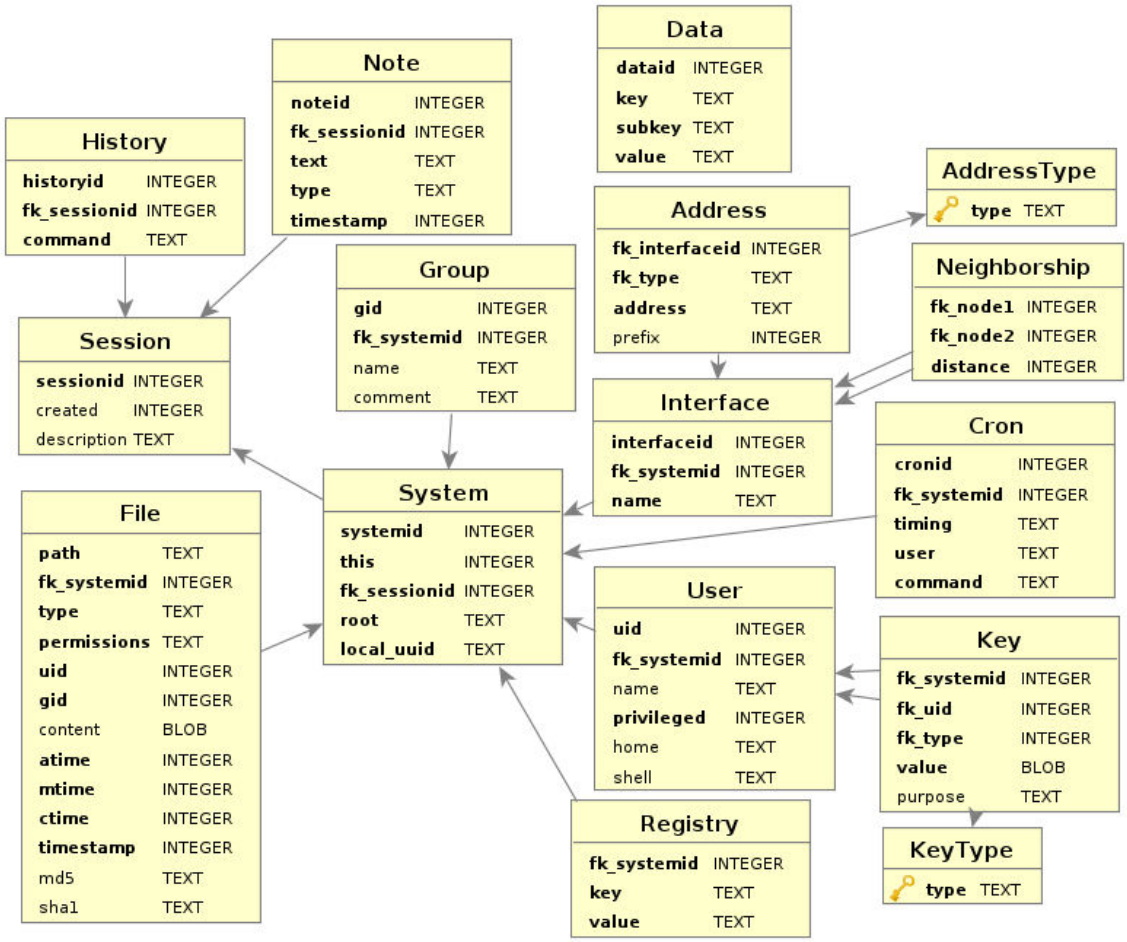
```
1  tb                      # zobrazi vse
2  tb klicek               # zobrazi hodnotu ulozenou pod klicem klicek
3  tb klicek 0             # zobrazi prvni znak retezce tb[klicek] ('h')
4  tb policko 0            # zobrazi prvni prvek seznamu policko
5  tb policko 1 1          # zobrazi druhy znak druheho prvku seznamu policko ('l')
6  tb slovnicek keys()     # zobrazi klice slovniku slovnicek ('A', 'C')
7  tb slovnicek values()   # zobrazi hodnoty slovniku slovnicek ('B', 'D')
```

Výpis 9: Přístup k TB (Framework)

4.2.2 Analysis.db

Analysis.db (schéma na obrázku 10) je databáze určená k uchování dat o systémech. Záznamy v tabulce Session označují sourhn systémů, které náleží k jednomu testu — jeden systém lze tedy testovat vícekrát a výsledky jsou od sebe odlišeny. V tabulce System jsou pak definována jednotlivá zařízení. Za zmínku stojí ještě tabulka Note, ze které je možné vyčíst události, které při analýze systému nastaly, a tabulka Data, kde lze ukládat aditivní informace ke kterékoli tabulce. Řešení spočívá v definování dvou klíčů, kde první ukazuje na primární klíč relevantního záznamu a druhý definuje popisovanou vlastnost. Příslušnost k tabulce musí být sémanticky odvozena z druhého klíče.

Tato databáze je určena pro implementaci dalších funkcionalit (v tuto chvíli je v plánu komplexní analýza sítě bez zásahu uživatele) a v principu není pro analýzu firmware potřebná. Proto jsou zatím implementovány jen nezbytné funkce vztahující se k tabulkám Session, Note, System, File, User, Cron a Data.

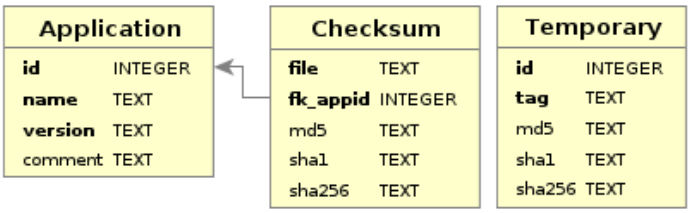


Obrázek 10: Schéma databáze *analysis.db*

4.2.3 Checksum.db

Checksum.db (schéma na obrázku 11) je jednoduchá databáze umožňující určení typu souboru na základě jeho kontrolního součtu. Tabulky Application a Checksum jsou určeny k uchovávání předem známých hodnot. Do tabulky Temporary jsou nahrána data k otestování a samotné porovnávání tak lze provést jediným dotazem. Tento přístup redukuje počet dotazů a umožňuje využít vysokého výkonu databáze a je použit i u databází Dict a Vuln.

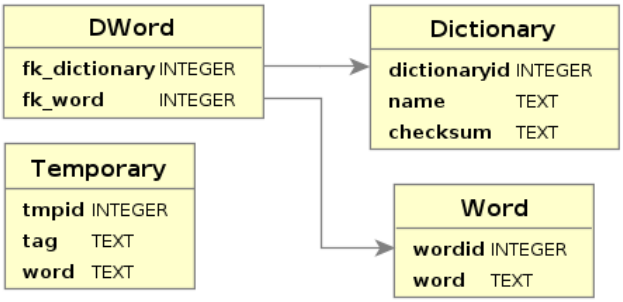
Analýza systému je prováděna na základě verzí detekovaných balíčků, jak bylo popsáno v minulé kapitole, v tuto chvíli tedy tato databáze není využívaná.



Obrázek 11: Schéma databáze *checksum.db*

4.2.4 Dict.db

Dict.db (schéma na obrázku 12) také není potřebná pro analýzu systémů, její smysl spočívá v uchovávání slovníků pro analýzu textů a slovníkové útoky. Tabulka DWord popisuje vazbu mezi slovy (Word) a slovníky (Dictionary), tabulka Temporary slouží k porovnávání záznamů.

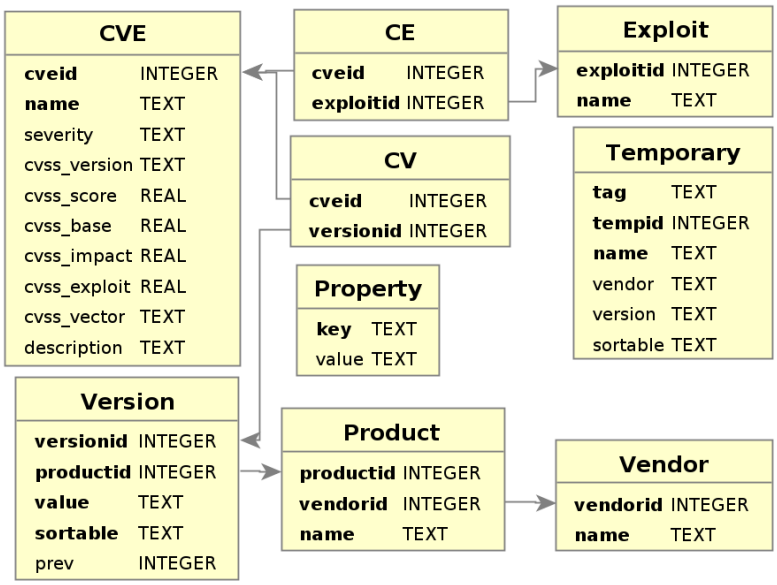


Obrázek 12: Schéma databáze *dict.db*

4.2.5 Vuln.db

Databáze *vuln.db* (schéma na obrázku 13) je naopak pro systémovou analýzu naprosto nezbytná. Při aktualizaci CVE záznamů jsou naplněny tabulky CVE a trojice Vendor-Product-Version, vazba mezi nimi je realizována prostřednictvím tabulky CV. Identifikační čísla exploitů jsou následně uloženy do tabulky Exploit a svázány (CE) s tabulkou CVE. Temporary tabulka opět

slouží k porovnávání, do tabulky property jsou ukládány pomocné údaje, jmenovitě datum poslední aktualizace a kontrolní součty souborů s CVE záznamy pro jednotlivé roky. Jak bylo zmíněno v předchozí kapitole, tabulka Vendor slouží pouze k odlišení stejnojmenných produktů a při hledání shod se nevyužívá.



Obrázek 13: Schéma databáze *vuln.db*

4.3 Implementace případů užití

Samotné případy užití jsou již implementovány jako víceméně nezávislé moduly. Na následujících řádcích bude popsán princip jednotlivých případů užití a nejpodstatějších částí zdrojových kódů. Tabulka 9 obsahuje výčet modulů, které jsou pro jednotlivé případy užití použity.

Případ užití	Použité moduly
I Update vulnerability database	locasploit.update.vulnerabilities
	locasploit.update.cve
	locasploit.update.exploits
	locasploit.cleanup
II Create SSH connection	connection.ssh
III Analyze	analysis.iot
	iot.binwalk.extract
	linux.enumeration.*
	packages.[dio]pkg.installed
IV Generate report	report.iot
V Generate diff report	report.iot.diff

Tabulka 9: Použití modulů pro jednotlivé případy užití

4.3.1 Aktualizace databází

Pro zajištění aktuálnosti databáze s CVE záznamy a seznamy exploitů je uživateli k dispozici modul *locasploit.update.vulnerabilities* (výpis 10). Zde (řádky 94–105) je nejdříve provedena kontrola data poslední aktualizace. Pokud byla poslední aktualizace provedena před více než osmi dny, je nutné stáhnout všechny zdroje od roku 2002. V opačném případě budou všechny změny popsány v souboru Modified. Seznam let určených ke stažení a následnému rozboru je předán ke zpracování modulu *locasploit.update.cve* (řádky 109–111). Modul *'locasploit.update.exploit'* (řádky 114–118) je následně naplánován tak, aby se spustil po zpracování CVE záznamů. Pokud oba moduly dokončí svou činnost, je spuštěn modul *locasploit.cleanup* (řádky 121–125).

```
92  m = lib.modules['locasploit.update.cve']
93  m.parameters['BACKGROUND'].value = 'yes' if self.background else 'no'
94  last_update = lib.db['vuln'].get_property('last_update')
    ...
98  if last_update != DB_ERROR and (datetime.now() - datetime.strptime(last_update, '%Y-%
        m-%d')).days < 8:
99      if not self.silent:
100         log.info('Entries have been updated less than 8 days ago, checking Modified feed only
                ...')
101     m.parameters['YEARS'].value = 'Modified'
102 else:
103     if not self.silent:
104         log.info('Entries have been updated more than 8 days ago, checking all feeds for
                change...')
105     m.parameters['YEARS'].value = ' '.join(map(str, range(2002, datetime.now().year+1)))
    ...
109 job = m.run()
110 # get job id so we can wait for it
111 lucid = None if job is None else [lib.scheduler.add(m.name, time.time(), job)]
112
113 # update exploits
114 m = lib.modules['locasploit.update.exploit']
    ...
117 job = m.run()
118 lue = None if job is None else [lib.scheduler.add(m.name, time.time(), job, timeout=None,
        waitfor=lucid)]
119
120 # cleanup
121 m = lib.modules['locasploit.cleanup']
    ...
124 job = m.run()
125 lib.scheduler.add(m.name, time.time(), job, timeout=None, waitfor=lucid+lue)
```

Výpis 10: Modul *locasploit.update.vulnerabilities*

Zdrojový kód modulu *locasploit.update.cve* je k nahlédnutí v příloze A. Funkce *download_years()* na řádku 124 zajistí stažení záznamů vyžádaných let. Pokud se data liší od poslední aktualizace (liší se hash, řádky 149–151), je každý soubor parsován (řádek 177–217). V případě, že byla stažena pouze modifikovaná data za posledních 8 dní, jsou ještě dodatečně získány hashe souborů pro roky, u nichž ke změně došlo (řádky 219–229).

Vazby CVE–exploit jsou získávány ze stránky `http://cve.mitre.org/data/refs/refmap/source-EXPLOIT-DB.html`, jak je patrné z výpisu 11. Data jsou dočasně uložena do lokálního textového souboru, samotný proces parsování je pak proveden pomocí třídy dědící z třídy HTMLParser (řádek 98–131).

```
98  class HTMLP(HTMLParser):
99      def __init__(self):
100         super().__init__()
101         self.intable = False
102         self.tmpkey = ''
103         self.tmpvalue = []
104         self.result = {}
    . . .
109     def handle_starttag(self, tag, attrs):
110         if tag == 'table':
111             self.intable = True
112             self.tmpkey = ''
113
114     def handle_endtag(self, tag):
115         if self.intable and tag == 'table':
116             self.add_previous_to_result()
117             self.intable = False
118             self.tmpkey = ''
119
120     def add_previous_to_result(self):
121         if self.tmpkey != '':
122             self.result[self.tmpkey] = self.tmpvalue
123
124     def handle_data(self, data):
125         if self.intable and len(data.strip())>0:
126             if data.startswith('EXPLOIT-DB:'):
127                 self.add_previous_to_result()
128                 self.tmpkey = data
129                 self.tmpvalue = []
130             elif data.startswith('CVE-') and len(self.tmpkey)>0:
131                 self.tmpvalue.append(data)
    . . .
136 localfile = './vulnerabilities/exploit.html'
137 try:
```

```
138         urlretrieve ('http://cve.mitre.org/data/refs/refmap/source-EXPLOIT-DB.html', localfile)
    . . .
151     parser = HTMLP()
152     parser.feed(data)
    . . .
155     db['vuln'].add_exploits(parser.result)
156     if not self.silent:
157         log.ok('Exploits updated.')
```

Výpis 11: Aktualizace seznamu exploitů (*locasploit.update.exploit*)

Jako poslední je volán modul *locasploit.cleanup*, jehož jediným úkolem je smazání stažených a již nepotřebných souborů.

4.3.2 Vytvoření SSH spojení

Modul *connection.ssh* využívá Paramiko pro vytváření spojení. Podporovány jsou tři metody spojení:

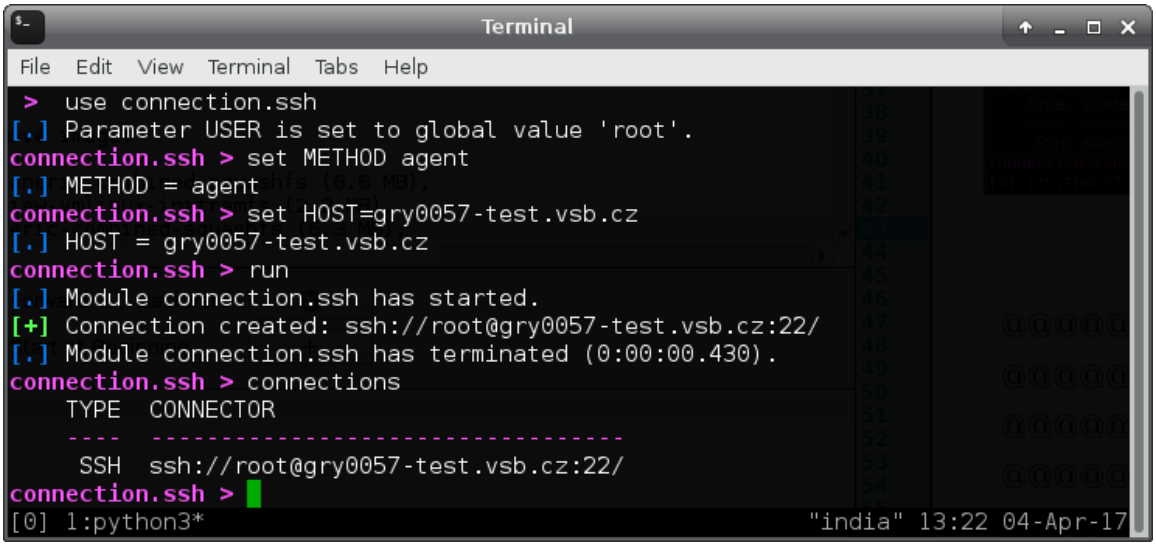
- agent — předpokládá přítomnost klíče v agentovi, nevyžaduje další parametry,
- password — klasické připojení se zadáváním hesla,
- pubkey — přihlášení pomocí konkrétního privátního klíče (volitelně chráněného heslem).

Ve výpisu 12 je uvedena část zdrojového kódu umožňující autentizaci pomocí hesla, další metody jsou obdobné. Navázání spojení je ilustrováno na obrázku 14.

```
118     if method == 'password':
119         import getpass
120         password = getpass.getpass('Password for user %s: ' % (user))
121         try:
122             client.connect(host, port=port, username=user, password=password)
123         except paramiko.ssh_exception.NoValidConnectionsError:
124             log.err('Cannot connect to the host \'%s\'.' % (host))
125             client = None
126         except Exception as e:
127             log.err('Connection with password failed: %s.' % (str(e)))
128             client = None
    . . .
153     if client is not None:
154         c = Connection('ssh://%s@%s:%s/' % (user, host, port), client, 'SSH')
155         lib.connections.append(c)
156         if not silent:
157             log.ok('Connection created: %s' % (c.description))
158         # test sftp
```

```
159     try:
160         sftp = client.open_sftp()
161         sftp.close()
162     except:
163         log.err('SFTP connection cannot be established.')
```

Výpis 12: SSH – autentizace pomocí hesla (*connection.ssh*)



Obrázek 14: Navázání SSH spojení

4.3.3 Analýza

O samotnou analýzu se stará modul *analysis.iot*. Vzhledem k rozsáhlé funkcionalitě a problémům zmíněným ve třetí kapitola má tento modul několik parametrů.

TAG je parametr umožňující analyzovat více systémů nezávisle na sobě. Pokud chceme například srovnat dva firmwary různých verzí, bude modul *analysis.iot* zavolán dvakrát s různými hodnotami TAG a poté bude použit modul *report.iot.diff*.

METHOD udává typ systému, který bude analyzován. Platné hodnoty jsou 'local', 'image' a 'ssh'. Pro analýzu přes SSH je nutné mít k dispozici existující SSH spojení — o to se stará modul *connection.ssh*. Mód 'image' je primárně určen pro analýzu binárního souboru. Volba 'local' je určena pro testování aktuálně běžícího systému, případně systému na jiném oddílu nebo v podsložce (chroot).

EXTRACT je logický příznak, kterým lze deaktivovat samotný proces extrakce. To může být výhodné například v případě, že data již byla extrahována, ale měla by být znovu analyzována, nebo při analýze jiného lokálního cíle. V módech 'local' a 'ssh' je ignorován.

TARGET popisuje objekt, který bude analyzován. Zde se zadává cesta k binárnímu souboru (metoda 'image'), tzv. Connection string pro SSH nebo cestu k systému pro metodu 'local'.

TMPDIR udává umístění, do kterého bude cílový binární soubor extrahován. Pro metody 'ssh' a 'local' není potřeba.

Jak už bylo zmíněno ve třetí kapitole, v CVE záznamech se vyskytují názvy produktů, které jsou rozdílné, ale ve skutečnosti odpovídají stejným balíčkům (například *kernel* a *linux_kernel*). Příznak ALIASES určuje, zda se mají při analýze použít i tyto známé alternativní názvy.

Příznak EPOCH je na tom podobně — některé balíčky uvádějí hodnotu epoch jako součást verze a je obtížné jednoznačně určit, zda verze v jednotlivých CVE záznamech odpovídají verzi, kde je epocha součástí číslování, nebo je ignorována.

Parametr ACCURACY umožňuje zpřesnit nalezené výsledky tak, že bere v potaz jen část detekované hodnoty verze balíčku. Může nabývat několika hodnot:

- none — verze je při analýze zcela ignorována (bere se v potaz jen název produktu),
- major — bere se v potaz pouze část před první tečkou (případně pomlčkou),
- minor — bere se v potaz pouze část před druhou tečkou (případně pomlčkou),
- build — bere se v potaz pouze část před třetí tečkou (případně pomlčkou),
- full — je použita původní hodnota.

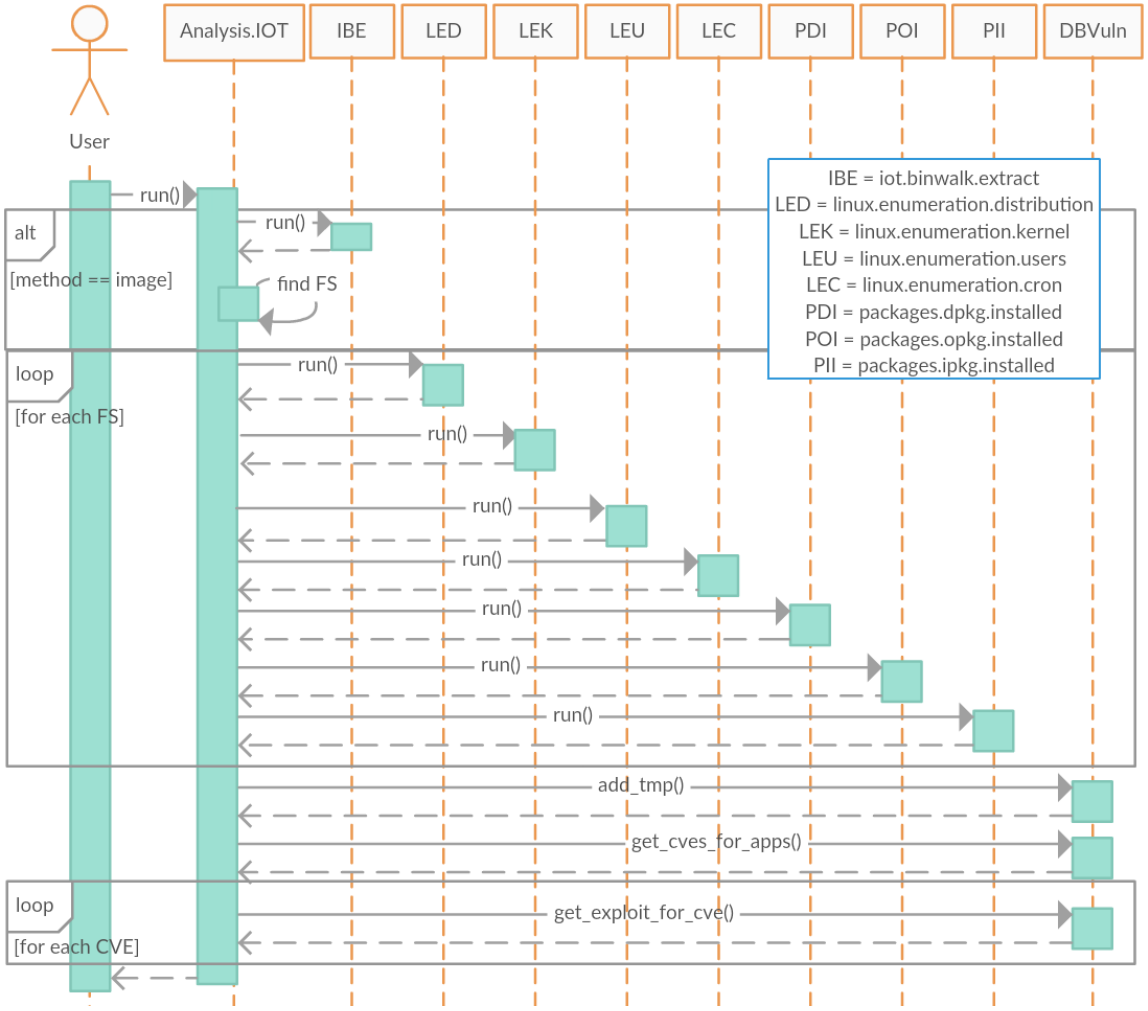
Informace získané analýzou se ukládají do Temporary Base do struktur pod klíčem ve tvaru '<tag>_<category>', kde 'category' nabývá hodnot:

- accuracy — použitá hodnota parametru ACCURACY,
- alias_packages — seznam alternativních názvů pro nalezené balíčky,
- exploits — slovník, kde klíčem je CVE-ID a hodnotou seznam identifikátorů známých exploitů,
- fake_packages — seznam prvků, které samy o sobě nejsou balíčky, ale mají s velkou pravděpodobností CVE záznam (typicky kernel v Debian systémech)
- general — základní informace (datum, cíl, kontrolní součty binárního souboru, parametr ALIASES)
- filesystems — seznam slovníků popisujících jednotlivé aspekty detekovaných souborových systémů:
  - name — relativní umístění souborového systému,
  - cron — cron záznamy ve tvaru (perioda, uživatel, příkaz),
  - os — informace o systému ve tvaru (klíč, hodnota),
  - packages — informace o nalezených balíčkách ve tvaru (produkt, vendor, verze),

– cves — informace o CVE záznamech pro zobrazení ve výsledné zprávě,

– system — další informace o systému (verze jádra, uživatelé s UID 1000 a více, uživatelé s UID 0, detekované správce balíčků).

Zdrojový kód modulu je k dispozici v příloze B. Pro jeho pochopení je však vhodnější diagram na obrázku 15. V první fázi, pokud je zvolena analýza systémové image, probíhá extrakce dat z



Obrázek 15: Analýza — sekvenční diagram

binárního souboru (řádky 198–212). Je spuštěn modul *iot.binwalk.extract*, který spouští funkci *binwalk.scan* (výpis 13).

```
106  for module in binwalk.scan(path, **{'signature' : True, 'quiet' : True, 'extract' : True, '
         directory' : self.parameters['TMPDIR'].value, 'matryoshka' : True, 'rm': clean}):
107      pass
```

Výpis 13: Extrakce dat (*iot.binwalk.extract*)

Po extrakci je nutné najít kořen souborového systému. To je realizováno na řádcích 230–234 prostým hledáním adresáře *etc* ve všech podadresářích výsledku extrakce. Zároveň je kontrolováno, zda se poblíž vyskytuje dostatečné množství (v tuto chvíli je požadováno alespoň 30 %) dalších klíčových adresářů (*usr*, *lib*, *sbin*, *tmp* apod.), aby se omezilo množství tzv. false-positive výsledků.

Následuje analýza konfigurace nalezených systémů. V tuto chvíli jsou používány moduly *distribution*, *kernel*, *users* a *cron* z kategorie *linux.enumeration*. Diagram na obrázku 15 je ve skutečnosti z důvodu přehlednosti zjednodušen — reálně jsou výsledky jednotlivých modulů uloženy do databáze Analysis a modul *analysis.iot* tyto informace následně odtud získá, jak je patrné na řádcích 280–313. Výpisy 14, 15, 16 a 17 obsahují klíčové části jednotlivých *linux.enumeration* modulů.

```
62  # get /etc/issue
63  issue = io.read_file(activeroot, '/etc/issue', verbose=False)
    ...
73  # get /etc/*-release
74  releases = [x for x in io.list_dir(activeroot, '/etc') if x.endswith('-release')]
75  for x in releases:
76      path = os.path.join('/etc/', x)
77      release = io.read_file(activeroot, path)
```

Výpis 14: Detekce distribuce (*linux.enumeration.distribution*)

```
80  # /boot/vmlinuz-*
81  boots = sorted([x[8:] for x in io.list_dir(activeroot, '/boot') if x.startswith('vmlinuz-')],
        reverse=True)
82  # let's use the highest one
83  if len(boots)>0:
    ...
87      known = boots[0]
88
89  # get /proc/version
90  if io.can_read(activeroot, '/proc/version'):
91      proc_version = io.read_file(activeroot, '/proc/version')
92      if proc_version != IO_ERROR:
    ...
97          known = proc_version
```

Výpis 15: Detekce verze jádra (*linux.enumeration.kernel*)

```
67  users = io.read_file(activeroot, '/etc/passwd')
```

Výpis 16: Detekce uživatelů (*linux.enumeration.users*)

```
65  # solve files in /etc
66  for etcfile in ['/etc/crontab'] + [os.path.join('/etc/cron.d/', x) for x in io.list_dir(active
        root, '/etc/cron.d/')]:
67      if not io.can_read(activeroot, etcfile):
68          continue
69      tmp = io.read_file(activeroot, etcfile)
70      if tmp != IO_ERROR:
71          lines += tmp.splitlines()
72
73  # solve user crons
74  for user in io.list_dir(activeroot, '/var/spool/cron/crontabs'):
75      path = os.path.join('/var/spool/cron/crontabs/', user)
76      if not io.can_read(activeroot, path):
77          continue
78      tmp = io.read_file(activeroot, path)
79      if tmp != IO_ERROR:
80          for line in tmp.splitlines():
81              data = re.split('[ \t]+', line)
82              lines.append(' '.join(data[:5] + [user] + data[5:]))
83
84  # scrap comments, variable definitions and add run-parts files
85  ignore = ('#', 'SHELL=', 'PATH=', 'MAILTO=', 'DEFAULT=', 'NICETIGER=', 'HOME=', '
        LOGNAME=')
86  lines = [x for x in lines if len(x.strip())>0 and not x.startswith(ignore)]
87  for line in lines:
88      if 'run-parts --report ' in line: # another folder
89          how = ' '.join(x for x in re.split('[ \t]+', line)[:(2 if line.startswith('@') else 6)
                ])
90          folder = line.partition('run-parts --report ')[2].split(' ')[0]
91          for f in io.list_dir(activeroot, folder):
92              lines.append('%s %s' % (how, f))
```

Výpis 17: Detekce plánovaných příkazů (*linux.enumeration.cron*)

Na řádcích 321–328 modulu *analysis.iot* je uveden postup detekce správce balíčků — u podporovaných modulů je jejich předpokládaná úspěšnost kontrolována metodou *check()*. V případě chybějící databáze je tedy zřejmé, že daný správce balíčků není na systému přítomen. Samotná funkcionalita modulů pro podporované správce balíčků dpkg, ipkg a opkg je téměř identická, zde tedy bude pro představu uveden pouze zdrojový kód modulu *packages.dpkg.installed* (výpis 18). Pokud je detekován balíček 'kernel' (jak je běžné u ipkg a opkg) a modul *linux.enumeration.kernel* nebyl úspěšný, je verze takového balíčku použita jako skutečná verze jádra. Naopak, pokud je známá verze kernelu, ale odpovídající balíček neexistuje, je vytvořen balíček fiktivní (a budou se pro něj hledat relevantní CVE záznamy).

```
72  content = io.read_file(activeroot, '/var/lib/dpkg/status')
    ...
76  # grep correct lines
77  info = list(zip(*[iter([x for x in content.splitlines() if x.startswith(('Package', 'Status', '
        Version'))])]*3))
78  # add appropriate lines into TB
79  for entry in info:
80      try:
81          pkg = [x.partition(': ')[2] for x in entry if x.startswith('Package')][0]
82          version = [x.partition(': ')[2] for x in entry if x.startswith('Version')][0]
83          status = [x.partition(': ')[2] for x in entry if x.startswith('Status')][0]
84      except: # weird order, skip
85          continue
86      if 'installed' in status:
87          results.append((pkg, None, version))
```

Výpis 18: Detekce balíčků v dpkg databázi (*packages.dpkg.installed*)

Funkce *get_alias_packages()* je v modulu *analysis.iot* definována na řádku 418 a volána na řádku 361. Doplňuje stávající seznam detekovaných balíčků o známé alternativní názvy. Funkce *get_accurate_version()* (řádek 431) se stará o normalizaci verze balíčku vzhledem k parametrům ACCURACY a EPOCH. Její volání se vyskytuje na řádcích 370 (pro získání požadovaných výsledků) a 383 (pro zobrazení hledaných hodnot ve finální zprávě). Na řádku 373 jsou informace o nalezených balíčkách vloženy do tabulky Temporary databáze Vuln. Na řádku 381 je pak zavolána metoda vracející nalezené shody. Na řádcích 398–401 probíhá kontrola existence exploitů pro nalezené problémy.

Nyní jsou všechna potřebná data uložená v TB a mohou být zpracována moduly *report.iot* či *report.iot.diff*.

4.3.4 Generování zpráv

Příloha C obsahuje zdrojový kód modulu *report.iot*. Jeho účel je jednoduchý — zpracovat údaje uložené v Temporary Base a převést je pomocí nástroje reportlab do PDF formátu. Zhodnocení relevance jednotlivých záznamů je ponecháno na uživateli, díky problémům uvedeným ve třetí kapitole nemusí být totiž správně zohledněna hodnota epoch, úroveň záplatování konkrétního produktu a podobně.

Jednotlivé CVE záznamy jsou seřazeny podle těchto kritérií:

1. CVE s exploity nejdříve,
2. CVE s vysokým skóre (podle kategorií high, medium, low) nejdříve,
3. CVE s vyšším ID (”novější”) nejdříve.

Modul *iot.report.diff* funguje obdobně, obsahuje však informace a tabulku zranitelných balíčků pro oba testované vzorky. CVE záznamy jsou pak navíc ještě rozděleny podle toho, zda se vyskytují pouze u vzorku prvního (nového), druhého (staršího) nebo jsou platné pro oba testované systémy. Části zprávy jsou ilustrovány na obrázcích 16, 17 a 18.

VULNERABILITY ANALYSIS REPORT

Date:	06. 02. 2017
Target:	openwrt-10.03.1-x86-generic-combined-squashfs.img
Location:	/media/root/Entertainment/locasploit_files/firmwares
MD5:	f7d9f5684214a281ffc8ad4751c354f0
SHA1:	1124ee2a26f2575a85989f33922b5ba700c6b1fa
SHA256:	eea8d82ca17823f18c6a1d2f6fd6b5beb2f3e3faa90fb0bf5d64d637aa8bbd27
Aliases enabled:	NO
Vulnerable:	YES (*build* accuracy)
Known exploits:	0

Obrázek 16: Vzorová zpráva — základní informace

File System #0

Root: squashfs-root

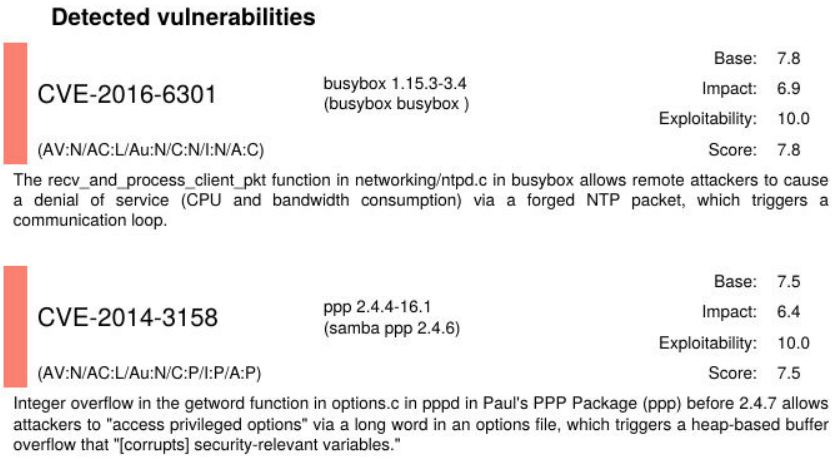
System info

Kernel:	2.6.32.27-1
Users:	nobody
Privileged users:	root
Package managers:	opkg

Vulnerable packages

Package	Version	Vulnerabilities		
		HIGH	MEDIUM	LOW
busybox	1.15.3-3.4	2	1	
dnsmasq	2.55-6.1		7	
firewall	2-34.8		2	
grub	0.97-3			1
iptables	1.4.6-3.1	1		
kernel	2.6.32.27-1	1	1	
lua	5.1.4-7		1	
luci	0.10.0-1	1	1	
ppp	2.4.4-16.1	2	2	1
Total:		7	15	2
		24		

Obrázek 17: Vzorová zpráva — souborový systém



Obrázek 18: Vzorová zpráva — CVE shody

4.4 Práce s frameworkem

Po spuštění frameworku (podle výpisu 7) je možné příkazem *help* vypsat všechny dostupné příkazy. V tabulce 10 jsou popsány ty nejdůležitější.

Příkaz	Funkce
!*command*	spuštění systémového příkazu
ls	výpis dostupných modulů (abecedně)
ls date	výpis dostupných modulů (od nejnovějších)
search abc	vyhledání modulu podle klíčového slova
use analysis.iot	selekce modulu
info	zobrazení informací o modulu
set PARAMETER VALUE	nastavení parametru modulu
setg PARAMETER VALUE	nastavení globálního parametru
getg	zobrazení globálních parametrů
check	kontrola úspěšnosti modulu
run	spuštění modulu
tb	zobrazení Temporary Base
jobs	výpis modulů běžících na pozadí
connections	výpis navázaných spojení
exit	ukončení programu

Tabulka 10: Příkazy frameworku Locasploit

Význam globálních parametrů spočívá v tom, že je daná hodnota dostupná i při změně modulu. Tyto hodnoty jsou rovněž automaticky nastaveny při použití příkazu *use*. Na hodnoty uložené v Temporary Base a v globálních parametrech se lze také odkazovat zápisem $KEY.

4.4.1 Vzorové soubory s příkazy

Jak bylo uvedeno ve výpisu 7, seznam příkazů je možné předpřipravit ve zvláštním souboru a ten následně předat locasploitu ke zpracování. Aktualizace databáze (výpis 19) je velmi jednoduchá, protože není potřeba nastavovat žádné parametry.

```
1 use locasploit.update.vulnerabilities
2 run
```

Výpis 19: Seznam příkazů pro aktualizaci databáze

Výpis 20 obsahuje příkazy umožňující analýzu lokálního systému. Povšimněme si, že parametr TAG na řádku 4 je definován jako globální a není nutné jej tedy znovu specifikovat pro modul *report.iot*.

```
1  use analysis.iot
2  set TARGET /
3  set METHOD local
4  setg TAG local
5  set ACCURACY build
6  set EPOCH yes
7  run
8  use report.iot
9  set OUTPUTFILE /tmp/report.pdf
10 run
```

Výpis 20: Seznam příkazů pro analýzu lokálního systému

Výpis 21 obsahuje příkazy umožňující analýzu systému dostupného přes SSH. Na řádku 1 je použit modul *connection.constr*, který slouží pouze jako wrapper pro modul *connection.ssh*. Díky němu je možné všechny parametry definovat na jediném řádku a navíc se díky globální definici tato hodnota dá znovu použít při specifikaci cíle (řádek 7).

```
1  use connection.constr
2  setg CONSTR ssh://root@localhost:22/
3  set METHOD agent
4  run
5  use analysis.iot
6  set METHOD ssh
7  set TARGET $CONSTR
8  setg TAG ssh
9  set EPOCH yes
10 run
11 use report.iot
12 set OUTPUTFILE /tmp/report.pdf
13 run
```

Výpis 21: Seznam příkazů pro analýzu vzdáleného systému

Příkazy z výpisu 22 jsou určeny k analýze systémové image. Parametr TARGET je zde definován jako globální, proto mohou být po vygenerování zprávy spuštěny systémové příkazy pro ověření velikosti a kontrolního součtu souboru (řádky 13–14).

```
1  use analysis.iot
2  set TMPDIR /tmp
3  setg TARGET /binary/openwrt-7.06-x86-squashfs.image
4  set EXTRACT yes
5  set METHOD image
6  setg TAG iot
7  set ACCURACY build
8  set EPOCH yes
9  run
10 use report.iot
11 set OUTPUTFILE /tmp/report.pdf
12 run
13 !ls -lh $TARGET
14 !md5sum $TARGET
```

Výpis 22: Seznam příkazů pro pro analýzu systémové image

Podobné příkazy jsou použity při srovnání dvou systémových image (výpis 23). Modul *analysis.iot* je zde spouštěn celkem dvakrát (řádky 11 a 14). Modul *report.iot.diff* přebírá globální parametry TAG1 a TAG2.

```
1  setg TAG1 diff1
2  setg TAG2 diff2
3  use analysis.iot
4  set TMPDIR /tmp
5  set TARGET /binary/dragino2-yun-common-v2.0.7-rootfs-squashfs.bin
6  set EXTRACT yes
7  set METHOD image
8  set TAG $TAG1
9  set ACCURACY build
10 set EPOCH yes
11 run
12 set TARGET /binary/dragino-yun--v4.1.1-squashfs-sysupgrade.bin
13 set TAG $TAG2
14 run
15 use report.iot.diff
16 set OUTPUTFILE /tmp/report.pdf
17 run
```

Výpis 23: Seznam příkazů pro porovnání dvou systémových image

4.5 Tvorba modulů

Samotné moduly jsou umístěny v adresáři *source/modules*. Při vytváření nového modulu je možné vycházet z již existujících modulů nebo z předpřipravených šablon (v podadresáři *templates*). Zde bude ve stručnosti popsána šablona *basic.py* (výpis 24), detailní popis obsahuje *basic_commented.py*.

```
1  #!/usr/bin/env python3
2  from source.modules._generic_module import *
3  class Module(GenericModule):
4      def __init__(self):
5          self.authors = [
6              Author(name='', email='', web=''),
7          ]
8          self.name = 'template'
9          self.short_description = 'Serves as a module template.'
10         self.references = ['']
11         self.date = '2999-12-31'
12         self.license = 'GNU GPLv2'
13         self.version = '0.0'
14         self.tags = ['template']
15         self.description = """This module is a template for new modules."""
16         self.dependencies = {
17             #'linux.enumeration.distribution': '1.0',
18         }
19         self.changelog = """"""
20
21         self.reset_parameters()
22
23     def reset_parameters(self):
24         self.parameters = {
25             'SILENT': Parameter(value='no', mandatory=True, description='Suppress the
                   output'),
26             'ACTIVEROOT': Parameter(mandatory=True, description='System to work with'),
27         }
28
29     def check(self, silent=None):
30         # CHECK_SUCCESS - module will do exactly what it is designed for
31         # CHECK_PROBABLY - it will probably work
32         # CHECK_NOT_SUPPORTED - nothing can be checked, but it may work
33         # CHECK_UNLIKELY - module can be executed, but it will probably fail
34         # CHECK_FAILURE - module cannot be executed
35
36         if silent is None:
37             silent = positive(self.parameters['SILENT'].value)
```

```
38            result = CHECK_NOT_SUPPORTED
39            return result
40
41        def run(self):
42            silent = positive(self.parameters['SILENT'].value)
43            # Define your code here
44            log.ok('Template module says: "Hello World!"')
45            return None
46
47    lib.module_objects.append(Module())
```

Výpis 24: Šablona Locasploit modulu (*basic.py*)

Na řádcích 5–19 jsou definovány základní informace o daném modulu. Atribut *self.name* by měl odpovídat názvu souboru s tečkami namísto podtržítek pro oddělení kategorií. Verze (*self.version*) by měla být aktualizována při každé změně zdrojového kódu, která by mohla obnovit funkčnost ostatních modulů. Ve slovníku *self.dependencies* musí být zmíněny všechny moduly, které jsou z aktuálního modulu přímo volány. Mezi parametry modulu (řádky 24–27) by se měl vyskytovat parametr SILENT, na jehož hodnotě by měly být závislé výpisy uživateli v metodách *check()* a *run()*.

Funkce *check()* je volána na žádost uživatele a automaticky před samotným spuštěním modulu. Zde by tedy mělo být ověřeno, zda jsou parametry nastaveny správně, uživatel má právo přistupovat k souborům, jsou instalovány správné python knihovny a podobně. Pokud je navrácena hodnota *CHECK_FAILURE*, ke spuštění nedojde.

Funkce *run()* obsahuje kód, který bude prováděn v aktuálním vlákně. Navratová hodnota by měla být None, případně reference na instanci třídy odvozené z *threading.Thread*, pokud je modul určen k běhu na pozadí (jak je ilustrováno v šabloně *thread.py*v příloze D).

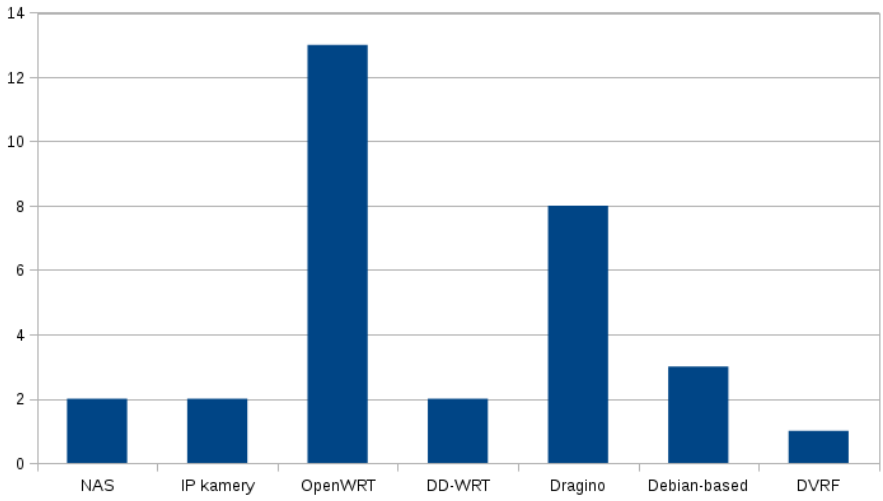
Tvůrce modulů by měl dodržovat několik pravidel:

- V metodě *check()* by se měly vyskytovat pouze chybové výpisy a to za předpokladu, že parametr SILENT má hodnotu False.
- V metodě *run()* by měly být chybové výpisy zobrazeny vždy, informace o úspěšné operaci a informativní výpisy by měly být závislé na parametru SILENT.
- Boolean parametry by měly být testovány metodami *positive()* a *negative()*, a to kvůli podpoře rozličných formátů zápisu (např. 'yes', 'true', '+').
- K souborům by mělo být přistupováno pomocí metod definovaných v *io.py*.
- Zobrazování informací uživateli by mělo být realizováno funkcemi definovanými v *log.py*.
- Při delegování funkcionality do jiného vlákna by měla být definována metoda *stop()*, která by měla zajistit korektní ukončení modulu.

5 Analýza vzorků

Analýze byly podrobeny volně dostupné systémové image založené na operačním systému Linux. Výsledné vygenerované zprávy pro úspěšné analýzy je možné nalézt v přiložených souborech. Množinu analyzovaných vzorků lze rozdělit do těchto skupin:

- NAS,
- IP kamery,
- OpenWRT,
- DD-WRT,
- Dragino,
- Debian-based systémy,
- DVRF.



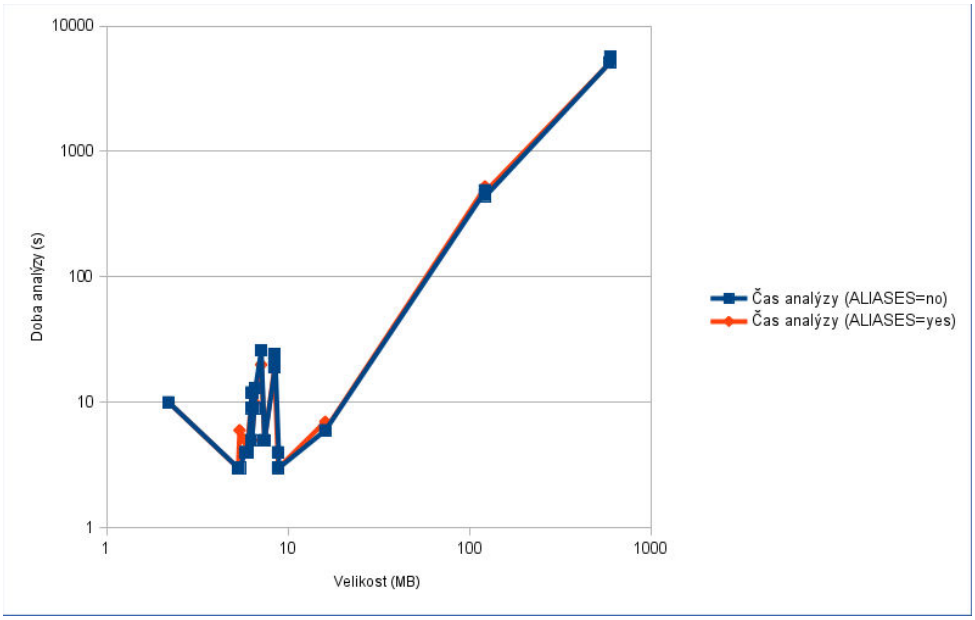
Obrázek 19: Počty vzorků v jednotlivých skupinách

Graf na obrázku 19 znázorňuje velikosti jednotlivých skupin. Velký rozdíl v počtech vzorků v jednotlivých skupinách vznikl díky snaze porovnat rozdíly mezi analýzou stejné image v různých verzích, analýza skupin s velkým počtem vzorků tedy dosahovala vysoké úspěšnosti.

5.1 Extrakce

Fáze extrakce a vyhledání systémových adresářů proběhly úspešné u všech testovaných vzorků. Samotná extrakce pak výrazně ovlivnila čas kompletní analýzy.

Z grafu na obrázku 20 je patrné, že velikost image je hlavním faktorem ovlivňujícím dobu analýzy. Dochází zde k drobným odchylkám, ty lze přičíst úrovni komprimace a počtu souborů.



Obrázek 20: Doba analýzy vzhledem k velikosti souboru

5.2 Analýza

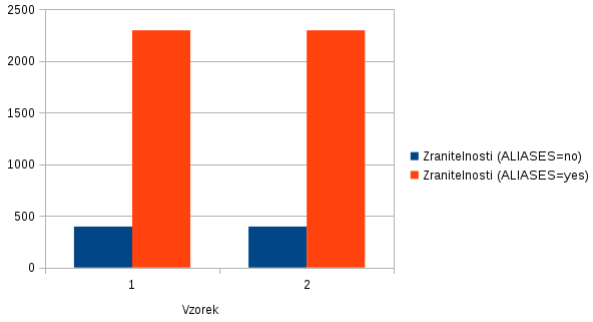
Systémové image byly analyzovány s parametry ACCURACY=build (neboť databáze zranitelností využívá převážně tuto přesnost) a EPOCH=no (hodnota epoch u balíčků nebyla ve většině případů zohledněna v CVE záznamech). Byly testovány obě hodnoty parametrů ALIASES, neboť právě tento parametr zásadně ovlivňoval množství shod (přičemž jeho vliv na celkovou dobu testování byl zanedbatelný, jak vyplývá z grafu 20). Detekovanými (a také Locasploitem podporovanými) správci balíčků jsou dpkg, ipkg a opkg.

5.2.1 NAS

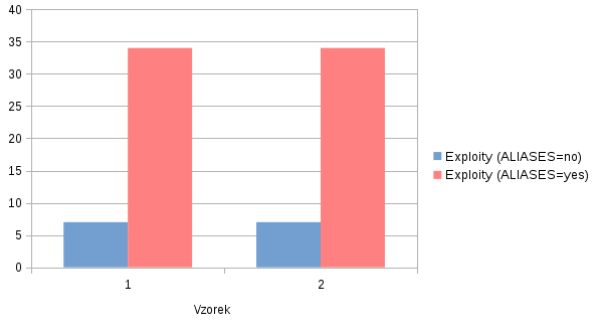
NAS (Network Attached Storage) je zařízení sloužící jako úložiště připojené k síti, může však obsahovat i jinou funkcionalitu (nejčaštěji webový server). Byly otestovány tyto systémové image:

1. seagate_nas_1400319.img,
2. seagate_nas-update-1500322-2bay.img.

Podle analýzy se v těchto produktech vyskytuje největší množství zranitelností a tedy i exploitů (obrázky 21 a 22). To je logické — od NAS stanic se očekává podpora velkého množství funkcí bez nutnosti manuální modifikace softwarového vybavení.



Obrázek 21: Počty zranitelností (NAS)



Obrázek 22: Počty exploitů (NAS)

5.2.2 IP kamery

IP kamery jsou kamery využívající IP protokolu ke streamování či ukládání záznamu na jiné zařízení v síti. Firmware je jen zřídka k dispozici jako open-source, proto byly otestovány pouze tyto systémové image:

1. MJPEG indoor PT camera-11.22.2.51 — lr_cmos_11_22_2_51.bin,
2. MJPEG indoor PT camera-11.37.2.65-20150603 —lr_cmos_11_37_2_65.bin,
3. Smartware OV9712 1.0.8.19 (není open-source).

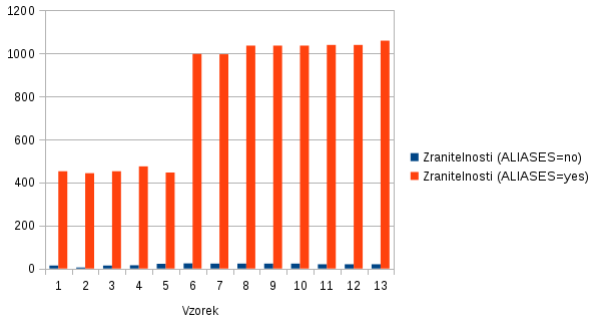
U obou MJPEG vzorků analýza selhala z důvodu prázdných adresářů */etc*, */var* a */usr*. Při analýze třetího vzorku nebyl detekován správce balíčků.

5.2.3 OpenWRT

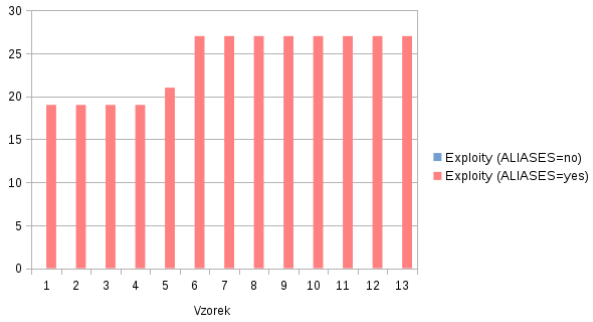
V kategorii OpenWRT jsou zařazeny oficiální image systému OpenWRT, což je velmi populární operační systém určen pro embedded zařízení. Z důvodu velké úspěšnosti bylo v této kategorii otestováno více vzorků různých verzí, aby bylo možné určit, zda dochází ke zlepšení. Test byl proveden na těchto vzorcích:

1. openwrt-15.05.1-x86-generic-combined-squashfs,
2. openwrt-15.05.1-realview-vmlinux-initramfs,
3. openwrt-15.05-x86-generic-combined-squashfs,
4. openwrt-14.07-x86-generic-combined-squashfs,
5. openwrt-12.09-x86-generic-combined-squashfs,
6. openwrt-10.03.1-x86-generic-combined-squashfs,
7. openwrt-10.03-x86-squashfs,

8. openwrt-8.09.2-x86-squashfs),
9. openwrt-8.09.1-x86-squashfs,
10. openwrt-8.09-x86-squashfs,
11. openwrt-7.09-x86-squashfs,
12. openwrt-7.07-x86-squashfs,
13. openwrt-7.06-x86-squashfs.



Obrázek 23: Počty zranitelností (OpenWRT)



Obrázek 24: Počty exploitů (OpenWRT)

Pro všechny vzorky tohoto rozšířeného systému byly nalezeny zranitelnosti, při použití parametrů ALIASES i odpovídající exploity. Z grafů (obrázky 23 a 24) je patrné, že novější verze obsahují vzhledem k jejich předchůdcům až na pár drobných odchylek méně zranitelností. Odchylky vznikají zejména při změně verze jádra a také při nasazení nové funkcionality.

5.2.4 DD-WRT

DD-WRT je systém určen primárně pro bezdrátové směrovače a přístupové body, je často používán jako náhrada oficiálních firmwarů. Byly otestovány tyto systémové image:

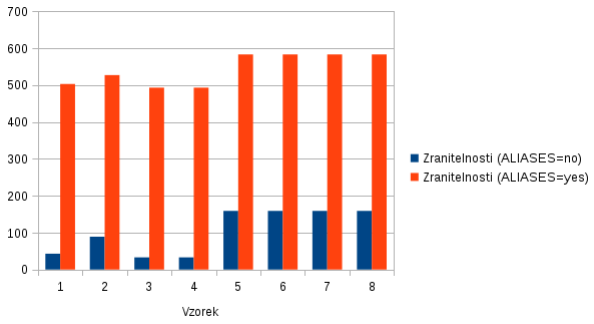
1. dd-wrt_public_serial.image,
2. dd-wrt.v24-21061_NEWD-2_K2.6_mini_wrt160nv3.bin.

Locasploit odhadl přítomnost správce balíčků ipkg, nicméně následná analýza byla neúspěšná, neboť podstatné soubory a adresáře jsou přítomny ve formě symbolických odkazů ukazujících do adresáře */tmp*. Lze tedy předpokládat, že analýza systému za běhu zařízení by poskytla požadované výsledky.

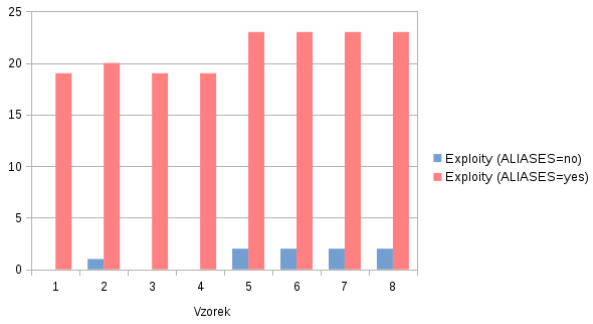
5.2.5 Dragino

Dragino je upravená varianta OpenWRT používaná pro zařízení pracující s technologiemi VoIP, WiFi a Lora. Byly otestovány tyto systémové image:

1. dragino2-IoT–v3.4.0-squashfs-sysupgrade.bin,
2. dragino-yun–v4.1.1-squashfs-sysupgrade.bin,
3. dragino2-fxs–v3.2-squashfs-sysupgrade.bin,
4. dragino2-fxs–v3.3.0-squashfs-sysupgrade.bin,
5. dragino2-yun-common-v2.0.6-rootfs-squashfs.bin,
6. dragino2-yun-common-v2.0.6-squashfs-sysupgrade.bin,
7. dragino2-yun-common-v2.0.7-rootfs-squashfs.bin,
8. dragino2-yun-common-v2.0.7-squashfs-sysupgrade.bin,



Obrázek 25: Počty zranitelností (Dragino)



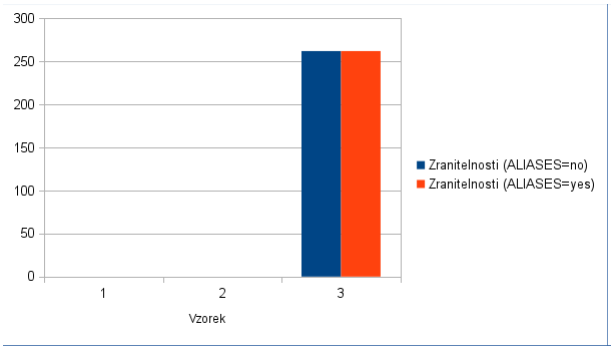
Obrázek 26: Počty exploitů (Dragino)

Analýza ukázala, že i tyto systémové image pravděpodobně obsahují zranitelnosti se známými exploity (obrázky 25 a 26).

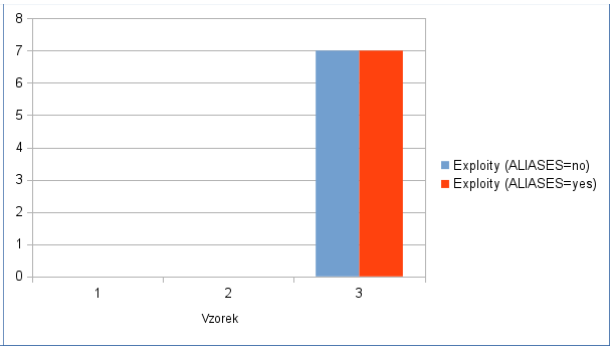
5.2.6 Debian-based systémy

Do kategorie Debian-based systémů jsou zařazeny populární IoT systémy vycházející ze systému Debian, jmenovitě:

1. ubuntu-core-16-pi2.img,
2. ubuntu-core-16-pi3.img,
3. Raspbian Jessie Lite.



Obrázek 27: Počty zranitelností (Debian-based)



Obrázek 28: Počty exploitů (Debian-based)

Výsledky analýzy jsou ilustrovány na obrázcích 27 a 28. U Ubuntu Core nedokázal Locasploit nalézt konfigurační soubory a správce balíčků. U Raspbianu sice došlo během extrakce k překročení lokálního 100GB limitu, nicméně i přesto byl souborový systém součástí rozbalené části a analýza byla úspěšně dokončena. Parametr ALIASES zde nalezené shody neovlivnil, také proto, že nebyla zjištěna verze jádra.

5.2.7 DVRF

DVRF (Damn Vulnerable Router Firmware) je záměrně zranitelný systém určen pro výuku a testování penetračních technik. Bohužel zde nebyly nalezeny požadované konfigurační soubory a podobně jako v případě DD-WRT jsou zde přítomny symbolické odkazy do */tmp*.

5.2.8 Shrnutí

V této kapitole byly analyzováno 32 vzorků firmwaru. U firmwarů typu NAS, OpenWRT, Dragino a Raspbian (75 % vzorků) odhalil Locasploit přítomnost většího množství zranitelností a odpovídajících exploitů. U systémů DD-WRT a DVRF (9.375 % vzorků) analýza selhala z důvodu neúspěšné detekce konfiguračních souborů, nicméně lze předpokládat, že analýza emulovaného systému či reálného zařízení by požadované výsledky poskytla. V případě firmwaru určeného pro IP kamery a systému Ubuntu Core (15.625 % vzorků) analýza selhala z důvodů chybějících klíčových souborů a správců balíčků.

Seznam testovaných souborů včetně kontrolních součtu lze nalézt v přiložených souborech.

6 Zhodnocení

Cílem této práce je popsat stávající situaci bezpečnosti zařízení Internet of Things a navrhrnout aplikaci schopnou na základě informací získaných z předané systémové image označit potenciálně zranitelná místa.

První kapitola je věnována pojmu Internet of Things, jsou zde definovány oblasti, kde se s tímto fenoménem setkáváme a jsou stručně popsány základní požadavky na zařízení do této oblasti spadající.

Bezpečnost zařízení Internet of Things je zcela klíčová. Z důvodu interoperability jsou však používány stávající technologie a protokoly i s jejich problémy. Ve druhé kapitole jsou proto předvedeny nejznámější útočné techniky, pomocí kterých je možné získat neoprávněný přístup k zařízením, počítačovým sítím a datům. V závěru kapitoly jsou zmíněny nejzávažnější IoT bezpečnostní incidenty posledních let.

Zhodnocení zranitelnosti softwaru nelze jednoduše zajistit. V praxi se používají techniky statické a dynamické analýzy, tyto metody však nejsou na danou úlohu jednoduše aplikovatelné. Stav zabezpečení lze však srovnat se známými problémy popsanými ve formě CVE záznamů a při zjištění verze softwaru lze k relevantním záznamům přistupovat. Tento přístup je možné použít pro testování systémů založených na známém operačním systému. Vzhledem k širokému zastoupení systémů na bázi Linuxu ve světě Internet of Things je tento způsob následně implementován.

V následující části práce je představen Locasploit, open-source framework psán v jazyce Python3, který dokáže jednoduše provést dříve popsanou analýzu zabezpečení předaného firmwaru, lokálního systému nebo systému dostupného přes SSH. Práce popisuje klíčové soubory, které Locasploit tvoří, a struktury pro uchovávání dočasných a perzistentních dat. Dále je zde vysvětlen zdrojový kód identifikovaných případů užití. Čtvrtá část pohlíží na systém z pohledu uživatele; jsou zde vysvětleny základní příkazy a způsob práce s frameworkem. Pátá část je určena pro případné zájemce o vývoj dalších modulů a popisuje šablonu modulů a základní pravidla vývoje.

V poslední části je čtenář seznámen s výsledky testování implementace na převážně volně dostupných vzorcích firmwaru. Výsledné zprávy jednotlivých testů jsou poskytnuty v přiložených souborech. Výsledky ukazují, že Locasploit exceluje při testech populárního systému OpenWRT a na firmwaru určeného pro high-end zařízení (jmenovitě NAS a Raspberry Pi).

Literatura

[1] *That 'Internet of Things' Thing* [online]. 2009, **2009**(50) [cit. 2017-04-23]. Dostupné z: http://www.rfidjournal.com/articles/view?4986

[2] *Síťový model TCP/IP* [online]. 1992, **1992**(31) [cit. 2017-04-23]. Dostupné z: http://www.earchiv.cz/a92/a231c110.php3

[3] *Ethernet II vs. IEEE 802.3* [online]. 1999, **1999**(31) [cit. 2017-04-23]. Dostupné z: http://www.earchiv.cz/anovinky/ai2058.php3

[4] NGUYEN, Radek. *Statická analýza kódu* [online]. Praha, 2012 [cit. 2017-04-23]. Dostupné z: https://dip.felk.cvut.cz/browse/pdfcache/nguyehuy_2011dipl.pdf. Diplomová práce. České vysoké učení technické. Vedoucí práce Ing. Radek Mařík, CSc.

[5] HUNTLEY, Samuel. *Embedded Device Security: Pwn the device*. US: CreateSpace, 2015.

[6] SCOTT, James a Drew SPANIEL. *Rise of the Machines: The Dyn Attack Was Just a Practice Run* [online]. Institute for Critical Infrastructure Technology, 2016 [cit. 2017-04-23]. Dostupné z: http://icitech.org/wp-content/uploads/2016/12/ICIT-Brief-Rise-of-the-Machines.pdf

[7] WILLIAMS, Elliot. *Inject Packets with an ESP8266*. In: *HACKADAY* [online]. 2016 [cit. 2017-04-23]. Dostupné z: http://hackaday.com/2016/01/14/inject-packets-with-an-esp8266/

[8] KOLBAN, Neil. Kolban's Book on ESP8266 [online]. 2016 [cit. 2017-04-23]. Dostupné z: https://leanpub.com/ESP8266_ESP32

[9] Raspberry Pi 3 Model B [online]. 2016 [cit. 2017-04-24]. Dostupné z: http://docs-europe.electrocomponents.com/webdocs/14ba/0900766b814ba5fd.pdf

[10] OWASP, Breakers Community. *OWASP Testing Guide*. 3rd ed. [s.l.]: OWASP Foundation, 2009, 349 s. Dostupné z: https://www.owasp.org/images/5/56/OWASP_Testing_Guide_v3.pdf

[11] KÜMMEL, Roman. *XSS: Cross-Site Scripting v praxi : o reálných zranitelnostech ve virtuálním světě* [online]. Zlín: Tigris, 2011 [cit. 2017-04-24]. ISBN 978-80-86062-34-1.

[12] HAYASHI, Kaoru. Linux Worm Targeting Hidden Devices. In: *Symantec Official Blog* [online]. 2013 [cit. 2017-04-24]. Dostupné z: https://www.symantec.com/connect/blogs/linux-worm-targeting-hidden-devices

[13] MOFFITT, Tyler. Source Code for Mirai IoT Malware Released. In: *Webroot Threat Blog* [online]. 2016 [cit. 2017-04-24]. Dostupné z: https://www.webroot.com/blog/2016/10/10/source-code-mirai-iot-malware-released/

[14] MCGRAW, Gary. *Software security: building security in*. Upper Saddle River, NJ: Addison-Wesley, c2006. ISBN 03-213-5670-5.

[15] HOPCROFT, John E., Rajeev. MOTWANI a Jeffrey D. ULLMAN. *Introduction to automata theory, languages, and computation*. 3rd ed. Boston: Pearson/Addison Wesley, c2007. ISBN 03-214-5536-3.

[16] GRANNEMAN, Scott. *Linux phrasebook*. Indianapolis, Ind.: Sams, c2006. ISBN 06-723-2838-0.

[17] SEWARD, Julian, Nicholas NETHERCOTE a JOSEF WEIDENDORFER.. [ET AL.]. *Valgrind 3.3: Advanced Debugging and Profiling for GNU/Linux Applications*. Bristol: Network theory, 2008. ISBN 09-546-1205-1.

[18] PEREZ, Ugaitz Amozarrain. *Low Power WiFi: A study on power consumption for Internet of Things* [online]. Barcelona, 2015 [cit. 2017-04-24]. Dostupné z: http://upcommons.upc.edu/bitstream/handle/2099.1/25583/104901.pdf. Diplomová práce. BarcelonaTech. Vedoucí práce Jose Maria Barceló Ordinas.

[19] THE INTERNET ENGINEERING TASK FORCE (IETF). RFC 791: *INTERNET PROTOCOL*. 1981. Dostupné z: http://www.ietf.org/rfc/rfc791

[20] THE INTERNET ENGINEERING TASK FORCE (IETF). RFC 793: *TRANSMISSION CONTROL PROTOCOL*. 1981. Dostupné z: http://www.ietf.org/rfc/rfc793

[21] THE INTERNET ENGINEERING TASK FORCE (IETF). RFC 2460: *Internet Protocol, Version 6 (IPv6) Specification*. 1998. Dostupné z: http://www.ietf.org/rfc/rfc2460

[22] THE INTERNET ENGINEERING TASK FORCE (IETF). RFC 768: *User Datagram Protocol*. 1980. Dostupné z: http://www.ietf.org/rfc/rfc768

[23] THE INTERNET ENGINEERING TASK FORCE (IETF). RFC 1945: *Hypertext Transfer Protocol – HTTP/1.0*. 1996. Dostupné z: http://www.ietf.org/rfc/rfc1945

[24] THE INTERNET ENGINEERING TASK FORCE (IETF). RFC 5321: *Simple Mail Transfer Protocol*. 2008. Dostupné z: https://tools.ietf.org/html/rfc5321

[25] THE INTERNET ENGINEERING TASK FORCE (IETF). RFC 1176: *INTERACTIVE MAIL ACCESS PROTOCOL - VERSION 2*. 1990. Dostupné z: http://www.ietf.org/rfc/rfc1176

[26] THE INTERNET ENGINEERING TASK FORCE (IETF). RFC 1034: *DOMAIN NAMES - CONCEPTS AND FACILITIES*. 1987. Dostupné z: http://www.ietf.org/rfc/rfc1034

[27] THE INTERNET ENGINEERING TASK FORCE (IETF). RFC 4251: *The Secure Shell (SSH) Protocol Architecture*. 2006. Dostupné z: http://www.ietf.org/rfc/rfc4251

[28] THE INTERNET ENGINEERING TASK FORCE (IETF). RFC 1889: *RTP: A Transport Protocol for Real-Time Applications*. 1996. Dostupné z: http://www.ietf.org/rfc/rfc1889

[29] Oliver Hahm, Emmanuel Baccelli, Hauke Petersen, Nicolas Tsiftes. *Operating Systems for Low-End Devices in the Internet of Things: a Survey*. IEEE Internet of Things Journal, IEEE, 2016, 3 (5), pp.720-734. <10.1109/JIOT.2015.2505901>. <hal-01245551>

[30] LAMPI, Mikko. Internet of Things Ambient Energy Harvesting. In: *Aalto University* [online]. Espoo, 2011 [cit. 2017-04-24]. Dostupné z: https://wiki.aalto.fi/download/attachments/59704179/lampi-iot-ambient-energy-harvesting.pdf

[31] REZAEI, Zahra a Shima MOBININEJAD. Energy Saving in Wireless Sensor Networks. *International Journal of Computer Science & Engineering Survey* [online]. 2012, (3), 15 [cit. 2017-04-24]. Dostupné z: http://airccse.org/journal/ijcses/papers/0212ijcses03.pdf

[32] Augustin, A.; Yi, J.; Clausen, T.; Townsley, W.M. A Study of LoRa: Long Range & Low Power Networks for the Internet of Things. *Sensors* 2016, 16, 1466.

[33] KEMPF, James, Jari ARKKO, Neda BEHESHTI a Kiran YEDAVALLI. Thoughts on Reliability in the Internet of Things. In: *Internet Architecture Board* [online]. 2011 [cit. 2017-04-24]. Dostupné z: https://www.iab.org/wp-content/IAB-uploads/2011/03/Kempf.pdf

[34] Modul ESP-01 ESP8266. In: *Hamshop* [online]. 2015 [cit. 2017-04-24]. Dostupné z: https://www.hamshop.cz/data/product/272_417.jpg

[35] Raspberry Pi 3 - Model B. In: *ModMyPi* [online]. [cit. 2017-04-24]. Dostupné z: https://www.modmypi.com/image/cache/data/rpi-products/raspberry-pi-3-model-b/DSC_0303-420x318.jpg

[36] Sqlite3 — DB-API 2.0 interface for SQLite databases. In: *Python 3.6.1 documentation* [online]. [cit. 2017-04-24]. Dostupné z: https://docs.python.org/2/library/sqlite3.html

[37] Deb-version(5). Linux man-pages online [online]. 2017 [cit. 2017-04-24]. Dostupné z: https://linux.die.net/man/5/deb-version

[38] Nmap(1). Linux man-pages online [online]. 2016 [cit. 2017-04-24]. Dostupné z: https://linux.die.net/man/1/nmap

[39] MCCLURE, Stuart, Joel Scambray a George Kurtz. *Hacking exposed 6: network security secrets & solutions*. 10th anniversary ed. New York: McGraw-Hill, 2009, 687 s. ISBN 0071613749.

[40] LYON, Gordon Fyodor. *Nmap network scanning: official Nmap project guide to network discovery and security scanning*. Sunnyvale, CA: Insecure.Com, LLC, 2008, 434 s. ISBN 09-799-5871-7.

[41] LEVY, Elias. *Smashing The Stack For Fun And Profit*. Phrack [online]. 1996, roč. 7, č. 49 [cit. 2017-04-24]. Dostupné z: http://phrack.org/issues/49/14.html

[42] LI, Tianji, BSc., MSc. *Improving Performance for CSMA/CA Based Wireless Networks* [online]. Dublin, 2007 [cit. 2017-04-24]. Dostupné z: http://www.hamilton.ie/publications/Thesis_tianji.pdf. Disertační práce. National University of Ireland. Vedoucí práce Douglas Leith.

[43] An Introduction to the Internet of Things (IoT): Part 1. of "The IoT Series". In: *Cisco* [online]. San Francisco: Lopez Research, 2013 [cit. 2017-04-25]. Dostupné z: http://www.cisco.com/c/dam/en_us/solutions/trends/iot/introduction_to_IoT_november.pdf

[44] EVANS, Dave. *The Internet of Things: How the Next Evolution of the Internet Is Changing Everything*. In: Cisco [online]. 2011 [cit. 2017-04-25]. Dostupné z: http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[45] The Internet of Things: An Overview: Understanding the Issues and Challenges of a More Connected World. In: *Internet Society* [online]. 2015 [cit. 2017-04-25]. Dostupné z: https://www.internetsociety.org/sites/default/files/ISOC-IoT-Overview-20151014_0.pdf

Seznam příloh

Příloha A: Zdrojový kód *locasploit.update.cve*
Příloha B: Zdrojový kód *analysis.iot*
Příloha C: Zdrojový kód *report.iot*
Příloha D: Zdrojový kód *template/thread.py*
Příloha E: Příloha na CD

A Zdrojový kód *locasploit.update.cve*

```
124 def download_years(self, years):
125     from urllib.request import urlretrieve
126     from urllib.error import HTTPError
127     import gzip
128 
129     years_to_update = {} # year: sha1
130     for year in years:
131         if self.terminate:
132             break
133         # get cves
134         localfile = './vulnerabilities/nvdcve-%s.xml' % (year)
135         try:
136             urlretrieve('https://nvd.nist.gov/download/nvdcve-%s.xml.gz' % (year), localfile+
                '.gz')
137         except HTTPError:
138             log.warn('Cannot get data for %s.' % (year))
139         # extract
140         try:
141             with gzip.open(localfile+'.gz', 'rb') as fg:
142                 io.write_file('/', localfile, fg.read())
143             io.delete('/', localfile+'.gz')
144             if year == 'Modified':
145                 years_to_update[year] = ''
146                 continue
147 
148             # mark for update if hash is different
149             sha1 = io.sha1('/', localfile)
150             if sha1 != lib.db['vuln'].get_property('%s_sha1' % (year)):
151                 years_to_update[year] = sha1
152         except FileNotFoundError:
153             log.warn('GZ extraction failed for year %s' % (year))
154     return years_to_update
155 
156 
157 def run(self):
158     from datetime import datetime
159     import xml.etree.ElementTree as etree
160 
161     # clear db
162     if self.clear:
163         lib.db['vuln'].delete_cves_apps()
164 
```

```
165         p = '{http://nvd.nist.gov/feeds/cve/1.2}'
166         if not self.silent:
167             log.info('Downloading CVE files...')
168 
169         io.mkdir('/', './vulnerabilities')
170         years_to_update = self.download_years(self.years)
171         modified_years_to_update = set()
172 
173         for year in sorted(years_to_update.keys()):
174             if self.terminate:
175                 break
176             if not self.silent:
177                 log.info('Parsing %s data...' % (year))
178             # parse the files
179             xmlfile = './vulnerabilities/nvdcve-%s.xml' % (year)
180             try:
181                 tree = etree.parse(xmlfile)
182             except FileNotFoundError:
183                 log.err('Cannot open %s' % (xmlfile))
184                 continue
185             root = tree.getroot()
186 
187             actuples = []
188             cvetuples = []
189             cves = [x for x in root if 'type' in x.attrib.keys() and x.attrib['type']=='CVE' and
                        not ('reject' in x.attrib.keys() and x.attrib['reject']=='1')]
190             for cve in cves:
191                 # should not stop?
192                 if self.terminate:
193                     break
194                 # insert into db
195                 cveid = cve.attrib['name']
196                 if year == 'Modified':
197                     cveyear = cve.attrib['seq'][:4]
198                     modified_years_to_update.add(cveyear if cveyear>'2002' else '2002')
199 
200                 description = cve.find('%sdesc' % p).find('%sdescript' % p).text
201                 cvetuples.append((cve.attrib, description))
202 
203                 vs = cve.find('%svuln_soft' % p)
204 
205                 if vs is None:
206                     products = []
207                 else:
```

```
208                     products = vs.findall('%sprod' % p)
209                 for product in products:
210                     for version in product.findall('%svers' % p):
211                         # prepare for insertion
212                         if 'prev' not in version.attrib:
213                             version.attrib['prev'] = 0
214                         actuples.append((cveid, product.attrib['name'], product.attrib['vendor'],
                                version.attrib['num'], version.attrib['prev']))
215             # push into db
216             lib.db['vuln'].add_cves(cvetuples)
217             lib.db['vuln'].add_apps_for_cves(actuples)
218
219         # from 'Modified' year? Update checksums for altered years
220         if self.terminate:
221             return
222         if 'Modified' in self.years:
223             if not self.silent:
224                 log.info('Updating checksums for modified years...')
225             updated_years = self.download_years(modified_years_to_update)
226         else:
227             updated_years = years_to_update
228         for year, sha1 in updated_years.items():
229             lib.db['vuln'].add_property('%s_sha1' % (year), sha1)
230
231         lib.db['vuln'].add_property('last_update', datetime.now().strftime('%Y-%m-%d'))
232         if not self.silent:
233             log.ok('CVEs updated.')
```

Výpis 25: Modul *locasploit.update.cve*

B Zdrojový kód *analysis.iot*

```
165 def run(self):
166     silent = positive(self.parameters['SILENT'].value)
167     activeroot = self.parameters['ACTIVEROOT'].value
168     method = self.parameters['METHOD'].value
169     target = self.parameters['TARGET'].value
170     tmpdir = self.parameters['TMPDIR'].value
171     accuracy = self.parameters['ACCURACY'].value
172     tag = self.parameters['TAG'].value
173     extract = positive(self.parameters['EXTRACT'].value)
174     use_epoch = positive(self.parameters['EPOCH'].value)
175     use_aliases = positive(self.parameters['ALIASES'].value)
176
177     # 0. Preparation
178     tb[tag+'_accuracy'] = accuracy
179     tb[tag+'_general'] = []
180     tb[tag+'_general'].append(('Date', time.strftime("%d. %m. %Y")))
181     if method == 'ssh':
182         tb[tag+'_general'].append(('Target', target))
183     elif method == 'image':
184         path, filename = os.path.split(target)
185         tb[tag+'_general'].append(('Target', filename))
186         tb[tag+'_general'].append(('Location', path))
187
188     tb[tag+'_filesystems'] = []
189     exploits = {}
190     tb[tag+'_fake_packages'] = [] # like kernel for Debian systems (version is detected, but it
            is not a package)
191     tb[tag+'_alias_packages'] = [] # kernel is defined as linux_kernel in most CVEs
192
193     aliases_lines = io.read_file('/', './source/support/package_aliases.csv')
194     aliases_lines = [] if aliases_lines == IO_ERROR else aliases_lines.splitlines()
195     package_aliases = [tuple(x.split(';')) for x in aliases_lines if x[0] not in ['#'] and len(
            x.strip()) > 0]
196
197     # 1. Extraction
198     if method == 'image':
199         if not silent:
200             log.info('Gathering file stats ... ')
201         tb[tag+'_general'].append(('MD5', io.md5(activeroot, target)))
202         tb[tag+'_general'].append(('SHA1', io.sha1(activeroot, target)))
203         tb[tag+'_general'].append(('SHA256', io.sha256(activeroot, target)))
204
```

```
205             if extract:
206                 if not silent:
207                     log.info('Extracting firmware...')
208                 ibe = lib.modules['iot.binwalk.extract']
209                 ibe.parameters['ACTIVEROOT'].value = activeroot
210                 ibe.parameters['BINFILE'].value = target
211                 ibe.parameters['TMPDIR'].value = tmpdir
212                 ibe.run()
213 
214             # get extracted dir (last-modified dir with matching name)
215             tmpdirs = [x for x in io.list_dir(activeroot, tmpdir, sortby=IOSORT_MTIME) if x.
                    startswith('_%s' % (os.path.basename(target))) and x.endswith('.extracted')]
216             if len(tmpdirs) > 0:
217                 tmpdir = os.path.join(tmpdir, tmpdirs[-1])
218             else:
219                 log.err('Cannot access extract folder.')
220 
221             if not silent:
222                 log.info('', end='')
223                 log.attachline('==========================', log.Color.BLUE)
224             if io.can_read(activeroot, tmpdir):
225                 if not silent:
226                     log.info('Analyzing data in \'%s\'...' % (tmpdir))
227             else:
228                 log.err('Cannot access %s' % (tmpdir))
229 
230             # 2. Root location
231             log.info('Looking for directory trees..')
232             found = [x[:-len('/etc')] for x in io.find(activeroot, tmpdir, 'etc') if io.
                    get_system_type_from_active_root(x[:-len('/etc')], verbose=True, dontprint=
                    tmpdir) == 'linux']
233             if len(found) > 0 and not silent:
234                 log.ok('Found %d linux directory trees.' % len(found))
235 
236         if method == 'local':
237             found = [target]
238 
239         if method == 'ssh':
240             found = [target]
241 
242         tb[tag+'_general'].append(('Aliases enabled', 'YES' if use_aliases else 'NO'))
243 
244         fscount = -1
245         # for each found filesystem
```

```
246         for f in found:
247             fscount+=1
248             if not silent:
249                 log.info('', end='')
250                 log.attachline('------------------------', log.Color.BLUE)
251                 log.info('Analyzing %s:' % (f))
252             data = {} # FS-specific, to be stored in TB
253 
254             oses = []
255             kernels = []
256             pms = []
257             users = []
258             pusers = []
259             crons = []
260             startups = []
261 
262             if method == 'image':
263                 data['name'] = f[len(tmpdir):]
264             elif method == 'ssh':
265                 data['name'] = f[len(target):]
266                 if not data['name'].startswith('/'):
267                     data['name'] = '/'+data['name']
268             elif method == 'local':
269                 data['name'] = f[len(target):]
270                 if not data['name'].startswith('/'):
271                     data['name'] = '/'+data['name']
272             else: # in case of new method
273                 data['name'] = 'UNDEFINED DUE TO WEIRD METHOD'
274 
275             # 3. SYSTEM INFO GATHERING
276             if not silent:
277                 log.info('Dumping system info...')
278             data['system'] = []
279 
280             led = lib.modules['linux.enumeration.distribution']
281             led.parameters['ACTIVEROOT'].value = f
282             led.parameters['SILENT'].value = 'yes'
283             led.run()
284             issue = db['analysis'].get_data_system('ISSUE', f)
285             if len(issue)>0:
286                 oses.append(('Issue', issue[0][3]))
287             releases = db['analysis'].get_data_system('RELEASE', f, like=True)
288             for x in releases:
289                 oses.append((x[1], x[3]))
```

```
290
291         lek = lib.modules['linux.enumeration.kernel']
292         lek.parameters['ACTIVEROOT'].value = f
293         lek.parameters['SILENT'].value = 'yes'
294         lek.run()
295         kernel = db['analysis'].get_data_system('KERNEL', f)
296         if len(kernel) > 0:
297             kernels.append(kernel[0][3])
298
299         leu = lib.modules['linux.enumeration.users']
300         leu.parameters['ACTIVEROOT'].value = f
301         # leu.parameters['SILENT'].value = 'yes'
302         leu.run()
303         users += [x[2] for x in db['analysis'].get_users(f) if x[0] >= 1000]
304         pusers += [(x[2] if x[2] == x[2].strip() else '%s' % (x[2])) for x in db['analysis'].
                get_users(f) if x[0] == 0]
305
306         if not silent:
307             log.info('Getting cron data...')
308         lec = lib.modules['linux.enumeration.cron']
309         lec.parameters['ACTIVEROOT'].value = f
310         lec.run()
311         crons += db['analysis'].get_cron(f)
312
313         data['cron'] = crons
314
315
316         # 4. Package enumeration
317         tb[tag+':%d_tmp_packages' % (fscount)] = [] # array for detected packages
318         tmp_packages = [] # cause multiple package managers overwrite tmp data in tb
319         if not silent:
320             log.info('Enumerating package managers...')
321         for p in self.packathors:
322             pxi = lib.modules['packages.%s.installed' % (p)]
323             pxi.parameters['ACTIVEROOT'].value = f
324             pxi.parameters['TAG'].value = tag+':%d_tmp_packages' % (fscount)
325             pxi.parameters['SILENT'].value = 'yes'
326             if pxi.check() == CHECK_FAILURE:
327                 continue
328             pxi.run()
329             if len(tb[tag+':%d_tmp_packages' % (fscount)]) == 0:
330                 continue
331             pms.append(p)
332             if not silent:
```

```
333                 log.ok('Detected \'%s\' package manager' % (p))
334             tmp_packages += tb[tag+':%d_tmp_packages' % (fscount)]
335             # add also known aliases for packages (e.g. kernel = linux_kernel)
336             # 'kernel' package is present when dealing with opkg or ipkg, so ...
337             if p in ['opkg', 'ipkg']:
338                 if len(kernels) == 0 and tag+':%d_tmp_packages' % (fscount) in tb:
339                     kernels += [ps[2] for ps in [x for x in tb[tag+':%d_tmp_packages' % (
                            fscount)] if x[0] == 'kernel']]
340             # add kernel as "package" for other package managers
341             else:
342                 if len(kernels)>0:
343                     tmp_packages.append(('kernel', None, kernels[0]))
344                     tb[tag+'_fake_packages'].append('kernel')
345 
346         if len(kernels) == 0:
347             kernels.append('UNKNOWN')
348 
349         # prepare data gathered so far (it's here because pms and kernel changed)
350         data['os'] = oses
351         data['system'].append(('Kernel', set(kernels)))
352         if len(users)>0:
353             data['system'].append(('Users', users))
354         if len(pusers)>0:
355             data['system'].append(('Privileged users', pusers))
356         data['system'].append(('Package managers', pms))
357 
358         if not silent:
359             log.info('Enumerating packages...')
360         if use_aliases:
361             alias_names, alias_packages = self.get_alias_packages(tmp_packages,
                        package_aliases)
362         else:
363             alias_names = []
364             alias_packages = []
365         tb[tag+'_alias_packages'] += alias_names
366         data['packages'] = tmp_packages + alias_packages
367         del tb[tag+':%d_tmp_packages' % (fscount)]
368         packages = []
369         if 'packages' in data:
370             packages = [(tag+':%d' % (fscount), x[0], x[1], self.get_accurate_version(
                        accuracy, x[2], use_epoch)) for x in data['packages']]
371 
372         if len(packages) > 0:
373             db['vuln'].add_tmp(packages)
```

```
374                 if not silent:
375                     log.ok('Found %d packages.' % (db['vuln'].count_tmp(tag+':%d' % (fscount))))
376
377             # 5. CVE detection
378             if not silent:
379                 log.info('Detecting CVEs...')
380
381             cves = db['vuln'].get_cves_for_apps(tag+':%d' % (fscount), accuracy!='none')
382             # accuratize the returned version for report
383             cves = [list(x[:2]) + [self.get_accurate_version(accuracy, x[2], use_epoch)] + list(x
                    [3:]) for x in cves]
384
385             # create dictionary of vulnerable packages (because we want original version to be
                    shown, too)
386             vulnerable = {k:v for k in [(x[0], x[1]) for x in cves] for v in [x[2] for x in data['
                    packages'] if x[0] == k[1] and(x[1] == k[0] or x[1] is None)]}
387             cves = [list(x)+[vulnerable[(x[0], x[1])]] for x in cves]
388             data['cves'] = cves
389             if not silent:
390                 if len(cves)>0:
391                     log.ok('Found %d CVEs.' % (len(cves)))
392                 else:
393                     log.info('No CVEs found.')
394
395             # 6. Exploit detection
396             if not silent:
397                 log.info('Detecting exploits ... ')
398             for cve in set([x[4] for x in cves]):
399                 exlist = db['vuln'].get_exploits_for_cve(cve)
400                 if len(exlist)>0:
401                     exploits[cve] = exlist
402
403             # nothing? don't report this filesystem
404             if len(data['cves'])+len(data['packages'])+len(oses+users+pusers+crons+startups) ==
                    0 and 'UNKNOWN' in kernels:
405                 continue
406             tb[tag+'_filesystems'].append(data)
407
408         if not silent:
409             log.attachline('------------------------', log.Color.BLUE)
410         if len(exploits)>0:
411             if not silent:
412                 log.ok('%d exploits found.' % (len(set([x for _,v in exploits.items() for x in v]))
                        ))
```

```
413             tb[tag+'_exploits'] = exploits
414         return None
415
416
417
418     def get_alias_packages(self, packages, known):
419         alias_matches = []
420         result = []
421         for k in known:
422             for p in packages:
423                 if p[0] in k:
424                     aliases = [(x, p[1], p[2]) for x in k if x != p[0]]
425                     alias_matches += [x[0] for x in aliases]
426                     result += aliases
427                     break
428         return alias_matches, result
429
430
431     def get_accurate_version(self, accuracy, version, use_epoch):
432         # deal with epoch
433         if use_epoch:
434             version = version.replace(':', '.')
435         else:
436             if ':' in version:
437                 version = version.partition(':')[2]
438
439         if accuracy == 'none':
440             return ''
441         if accuracy in ['major', 'minor', 'build']:
442             majorparts = version.partition('.')
443             if accuracy in ['major', 'minor', 'build'] and majorparts[0].isdigit():
444                 version = majorparts[0].partition('-')[0]
445             minorparts = majorparts[2].partition('.')
446             if accuracy in ['minor', 'build'] and minorparts[0] != '':
447                 version = '.'.join([majorparts[0], minorparts[0].partition('-')[0]])
448             buildparts = minorparts[2].partition('.')
449             if accuracy == 'build' and buildparts[0] != '':
450                 version = '.'.join([majorparts[0], minorparts[0], buildparts[0].partition('-')[0]])
451         return version
452
453     on
454         tb[tag+'_accuracy'] = accuracy
455         tb[tag+'_general'] = []
456         tb[tag+'_general'].append(('Date', time.strftime("%d. %m. %Y")))
```

```
457         if method == 'ssh':
458             tb[tag+'_general'].append(('Target', target))
459         elif method == 'image':
460             path, filename = os.path.split(target)
461             tb[tag+'_general'].append(('Target', filename))
462             tb[tag+'_general'].append(('Location', path))
463 
464         tb[tag+'_filesystems'] = []
465         exploits = {}
466         tb[tag+'_fake_packages'] = [] # like kernel for Debian systems (version is detected, but it
                is not a package)
467         tb[tag+'_alias_packages'] = [] # kernel is defined as linux_kernel in most CVEs
468 
469         aliases_lines = io.read_file('/', './source/support/package_aliases.csv')
470         aliases_lines = [] if aliases_lines == IO_ERROR else aliases_lines.splitlines()
471         package_aliases = [tuple(x.split(';')) for x in aliases_lines if x[0] not in ['#'] and len(
              x.strip()) > 0]
472 
473         # 1. Extraction
474         if method == 'image':
475             if not silent:
476                 log.info('Gathering file stats ...')
477             tb[tag+'_general'].append(('MD5', io.md5(activeroot, target)))
478             tb[tag+'_general'].append(('SHA1', io.sha1(activeroot, target)))
479             tb[tag+'_general'].append(('SHA256', io.sha256(activeroot, target)))
480 
481             if extract:
482                 if not silent:
483                     log.info('Extracting firmware...')
484                 ibe = lib.modules['iot.binwalk.extract']
485                 ibe.parameters['ACTIVEROOT'].value = activeroot
486                 ibe.parameters['BINFILE'].value = target
487                 ibe.parameters['TMPDIR'].value = tmpdir
488                 ibe.run()
489 
490             # get extracted dir (last-modified dir with matching name)
491             tmpdirs = [x for x in io.list_dir(activeroot, tmpdir, sortby=IOSORT_MTIME) if x.
                  startswith('_%s' % (os.path.basename(target))) and x.endswith('.extracted')]
492             if len(tmpdirs) > 0:
493                 tmpdir = os.path.join(tmpdir, tmpdirs[-1])
494             else:
495                 log.err('Cannot access extract folder.')
496 
497             if not silent:
```

```
498             log.info('', end='')
499             log.attachline('=========================', log.Color.BLUE)
500         if io.can_read(activeroot, tmpdir):
501             if not silent:
502                 log.info('Analyzing data in \'%s\'... ' % (tmpdir))
503         else:
504             log.err('Cannot access %s' % (tmpdir))
505
506         # 2. Root location
507         log.info('Looking for directory trees .. ')
508         found = [x[:-len('/etc')] for x in io.find(activeroot, tmpdir, 'etc') if io.
                get_system_type_from_active_root(x[:-len('/etc')], verbose=True, dontprint=
                tmpdir) == 'linux']
509         if len(found) > 0 and not silent:
510             log.ok('Found %d linux directory trees.' % len(found))
511
512     if method == 'local':
513         found = [target]
514
515     if method == 'ssh':
516         found = [target]
517
518     tb[tag+'_general'].append(('Aliases enabled', 'YES' if use_aliases else 'NO'))
519
520     fscount = -1
521     # for each found filesystem
522     for f in found:
523         fscount+=1
524         if not silent:
525             log.info('', end='')
526             log.attachline('------------------------', log.Color.BLUE)
527             log.info('Analyzing %s:' % (f))
528         data = {} # FS-specific, to be stored in TB
529
530         oses = []
531         kernels = []
532         pms = []
533         users = []
534         pusers = []
535         crons = []
536         startups = []
537
538         if method == 'image':
539             data['name'] = f[len(tmpdir):]
```

```
540             elif method == 'ssh':
541                 data['name'] = f[len(target):]
542                 if not data['name'].startswith('/'):
543                     data['name'] = '/'+data['name']
544             elif method == 'local':
545                 data['name'] = f[len(target):]
546                 if not data['name'].startswith('/'):
547                     data['name'] = '/'+data['name']
548             else: # in case of new method
549                 data['name'] = 'UNDEFINED DUE TO WEIRD METHOD'
550 
551             # 3. SYSTEM INFO GATHERING
552             if not silent:
553                 log.info('Dumping system info...')
554             data['system'] = []
555 
556             led = lib.modules['linux.enumeration.distribution']
557             led.parameters['ACTIVEROOT'].value = f
558             led.parameters['SILENT'].value = 'yes'
559             led.run()
560             issue = db['analysis'].get_data_system('ISSUE', f)
561             if len(issue)>0:
562                 oses.append(('Issue', issue[0][3]))
563             releases = db['analysis'].get_data_system('RELEASE', f, like=True)
564             for x in releases:
565                 oses.append((x[1], x[3]))
566 
567             lek = lib.modules['linux.enumeration.kernel']
568             lek.parameters['ACTIVEROOT'].value = f
569             lek.parameters['SILENT'].value = 'yes'
570             lek.run()
571             kernel = db['analysis'].get_data_system('KERNEL', f)
572             if len(kernel) > 0:
573                 kernels.append(kernel[0][3])
574 
575             leu = lib.modules['linux.enumeration.users']
576             leu.parameters['ACTIVEROOT'].value = f
577             # leu.parameters['SILENT'].value = 'yes'
578             leu.run()
579             users += [x[2] for x in db['analysis'].get_users(f) if x[0] >= 1000]
580             pusers += [(x[2] if x[2] == x[2].strip() else '%s' % (x[2])) for x in db['analysis'].
                    get_users(f) if x[0] == 0]
581 
582             if not silent:
```

```
583             log.info('Getting cron data...')
584         lec = lib.modules['linux.enumeration.cron']
585         lec.parameters['ACTIVEROOT'].value = f
586         lec.run()
587         crons += db['analysis'].get_cron(f)
588 
589         data['cron'] = crons
590 
591 
592         # 4. Package enumeration
593         tb[tag+':%d_tmp_packages' % (fscount)] = [] # array for detected packages
594         tmp_packages = [] # cause multiple package managers overwrite tmp data in tb
595         if not silent:
596             log.info('Enumerating package managers...')
597         for p in self.packathors:
598             pxi = lib.modules['packages.%s.installed' % (p)]
599             pxi.parameters['ACTIVEROOT'].value = f
600             pxi.parameters['TAG'].value = tag+':%d_tmp_packages' % (fscount)
601             pxi.parameters['SILENT'].value = 'yes'
602             if pxi.check() == CHECK_FAILURE:
603                 continue
604             pxi.run()
605             if len(tb[tag+':%d_tmp_packages' % (fscount)]) == 0:
606                 continue
607             pms.append(p)
608             if not silent:
609                 log.ok('Detected \'%s\' package manager' % (p))
610             tmp_packages += tb[tag+':%d_tmp_packages' % (fscount)]
611             # add also known aliases for packages (e.g. kernel = linux_kernel)
612             # 'kernel' package is present when dealing with opkg or ipkg, so ...
613             if p in ['opkg', 'ipkg']:
614                 if len(kernels) == 0 and tag+':%d_tmp_packages' % (fscount) in tb:
615                     kernels += [ps[2] for ps in [x for x in tb[tag+':%d_tmp_packages' % (
                            fscount)] if x[0] == 'kernel']]
616             # add kernel as "package" for other package managers
617             else:
618                 if len(kernels)>0:
619                     tmp_packages.append(('kernel', None, kernels[0]))
620                     tb[tag+'_fake_packages'].append('kernel')
621 
622         if len(kernels) == 0:
623             kernels.append('UNKNOWN')
624 
625         # prepare data gathered so far (it's here because pms and kernel changed)
```

```
626             data['os'] = oses
627             data['system'].append(('Kernel', set(kernels)))
628             if len(users)>0:
629                 data['system'].append(('Users', users))
630             if len(pusers)>0:
631                 data['system'].append(('Privileged users', pusers))
632             data['system'].append(('Package managers', pms))
633
634             if not silent:
635                 log.info('Enumerating packages...')
636             if use_aliases:
637                 alias_names, alias_packages = self.get_alias_packages(tmp_packages,
                        package_aliases)
638             else:
639                 alias_names = []
640                 alias_packages = []
641             tb[tag+'_alias_packages'] += alias_names
642             data['packages'] = tmp_packages + alias_packages
643             del tb[tag+':%d_tmp_packages' % (fscount)]
644             packages = []
645             if 'packages' in data:
646                 packages = [(tag+':%d' % (fscount), x[0], x[1], self.get_accurate_version(
                        accuracy, x[2], use_epoch)) for x in data['packages']]
647
648             if len(packages) > 0:
649                 db['vuln'].add_tmp(packages)
650                 if not silent:
651                     log.ok('Found %d packages.' % (db['vuln'].count_tmp(tag+':%d' % (fscount))))
652
653             # 5. CVE detection
654             if not silent:
655                 log.info('Detecting CVEs...')
656
657             cves = db['vuln'].get_cves_for_apps(tag+':%d' % (fscount), accuracy!='none')
658             # accuratize the returned version for report
659             cves = [list(x[:2]) + [self.get_accurate_version(accuracy, x[2], use_epoch)] + list(x
                    [3:]) for x in cves]
660
661             # create dictionary of vulnerable packages (because we want original version to be
                    shown, too)
662             vulnerable = {k:v for k in [(x[0], x[1]) for x in cves] for v in [x[2] for x in data['
                    packages'] if x[0] == k[1] and(x[1] == k[0] or x[1] is None)]}
663             cves = [list(x)+[vulnerable[(x[0], x[1])]] for x in cves]
664             data['cves'] = cves
```

```
665             if not silent:
666                 if len(cves)>0:
667                     log.ok('Found %d CVEs.' % (len(cves)))
668                 else:
669                     log.info('No CVEs found.')
670 
671             # 6. Exploit detection
672             if not silent:
673                 log.info('Detecting exploits ... ')
674             for cve in set([x[4] for x in cves]):
675                 exlist = db['vuln'].get_exploits_for_cve(cve)
676                 if len(exlist)>0:
677                     exploits[cve] = exlist
678 
679             # nothing? don't report this filesystem
680             if len(data['cves'])+len(data['packages'])+len(oses+users+pusers+crons+startups) ==
                    0 and 'UNKNOWN' in kernels:
681                 continue
682             tb[tag+'_filesystems'].append(data)
683 
684         if not silent:
685             log.attachline('----------------------------', log.Color.BLUE)
686         if len(exploits)>0:
687             if not silent:
688                 log.ok('%d exploits found.' % (len(set([x for _,v in exploits.items() for x in v]))
                        ))
689             tb[tag+'_exploits'] = exploits
690         return None
691 
692 
693 
694 def get_alias_packages(self, packages, known):
695     alias_matches = []
696     result = []
697     for k in known:
698         for p in packages:
699             if p[0] in k:
700                 aliases = [(x, p[1], p[2]) for x in k if x != p[0]]
701                 alias_matches += [x[0] for x in aliases]
702                 result += aliases
703                 break
704     return alias_matches, result
705 
706 
```

```
707 def get_accurate_version(self, accuracy, version, use_epoch):
708     # deal with epoch
709     if use_epoch:
710         version = version.replace(':', '.')
711     else:
712         if ':' in version:
713             version = version.partition(':')[2]
714 
715     if accuracy == 'none':
716         return ''
717     if accuracy in ['major', 'minor', 'build']:
718         majorparts = version.partition('.')
719         if accuracy in ['major', 'minor', 'build'] and majorparts[0].isdigit():
720             version = majorparts[0].partition('-')[0]
721         minorparts = majorparts[2].partition('.')
722         if accuracy in ['minor', 'build'] and minorparts[0] != '':
723             version = '.'.join([majorparts[0], minorparts[0].partition('-')[0]])
724         buildparts = minorparts[2].partition('.')
725         if accuracy == 'build' and buildparts[0] != '':
726             version = '.'.join([majorparts[0], minorparts[0], buildparts[0].partition('-')[0]])
727     return version
```

Výpis 26: Zdrojový kód (*analysis.iot*)

C Zdrojový kód *report.iot*

```
69  def run(self):
70      outputfile = self.parameters['OUTPUTFILE'].value
71      tag = self.parameters['TAG'].value
72      silent = positive(self.parameters['SILENT'].value)
73
74      #from reportlab.pdfgen import canvas
75      from reportlab.lib import colors
76      from reportlab.lib.units import cm
77      from reportlab.lib.pagesizes import A4
78      from reportlab.platypus import KeepTogether, SimpleDocTemplate, Table, TableStyle,
            Paragraph, PageBreak
79      from reportlab.lib.styles import getSampleStyleSheet
80      styles = getSampleStyleSheet()
81
82      doc = SimpleDocTemplate(outputfile, pagesize=A4)
83      entries = []
84
85      #print(styles.list())
86      # PREPARE STYLES
87      titlestyle = styles['Title']
88      heading1style = styles['Heading1']
89      headingstyle = styles['Heading2']
90      textstyle = styles['BodyText']
91
92      # HEADING
93      entries.append(Paragraph('<para align=center spaceAfter=20>VULNERABILITY
            ANALYSIS REPORT</para>', titlestyle))
94
95
96      cves_lists = [x.get('cves') for x in tb[tag+'_filesystems']] if tag+'_filesystems' in tb
            else []
97      exploits = {} if tag+'_exploits' not in tb else tb[tag+'_exploits']
98      exploit_count = len(set([x for k,v in exploits.items() for x in v]))
99
100     # ADD GENERAL INFO
101     if tag+'_general' in tb:
102         #entries.append(Paragraph('<para spaceBefore=30>General info<para>', headingstyle)
                )
103         data = [[x[0]+':', ', '.join(x[1]) if type(x[1]) in [list, tuple, set] else x[1]] for x
                in tb[tag+'_general']]
104         if len(cves_lists) > 0:
```

```
105             data.append(['Vulnerable:', Paragraph('<para><font color=%s><b>%s</b></
                    font> (<i>%s</i>accuracy)</para>' % (('red', 'YES', tb[tag+'_accuracy']) if
                    max([0]+[len(x) for x in cves_lists]) else ('green', 'NO', tb[tag+'_accuracy'])),
                    textstyle)])
106             data.append(['Known exploits:', Paragraph('<para><font color=%s><b>%s</b
                    ></font></para>' % (('red', exploit_count) if exploit_count>0 else ('green',
                    exploit_count)), textstyle)])
107         entries.append(Table(data, None, None, hAlign='LEFT'))
108
109     # for each fs
110     fscount = -1
111     for fs in tb[tag+'_filesystems'] if tag+'_filesystems' in tb else []:
112         fscount += 1
113         entries.append(Paragraph('<para spaceBefore=20>File System #%d<para>' % (
                fscount), heading1style))
114         data = [['Root:', fs['name']]]
115         entries.append(Table(data, None, None, hAlign='LEFT'))
116
117         # ADD SYSTEM INFO
118         if 'system' in fs:
119             entries.append(Paragraph('<para spaceBefore=20>System info<para>',
                    headingstyle))
120             data = [[x[0]+':', ', '.join(x[1]) if type(x[1]) in [list, tuple, set] else x[1]]
                    for xin fs['system']+fs['os']]
121             t = Table(data, None, None, hAlign='LEFT')
122             t.setStyle(TableStyle([
123                 ('VALIGN', (0, 0), (-1, -1), 'TOP'),
124             ]))
125             entries.append(t)
126
127
128         # ADD CRON ENTRIES
129         data = []
130         tospan = []
131         if 'cron' in fs:
132             for linec in range(len(fs['cron'])):
133                 line = fs['cron'][linec]
134                 if len(re.split('[ \t]+', line[0]))==1:
135                     l = [line[0], '', '', '', '', line[1], Paragraph('<para>%s</para>' % (
                            line[2]),textstyle)]
136                     tospan.append(linec)
137                 else:
138                     l = re.split('[ \t]+', line[0])[:5]+[line[1], Paragraph('<para>%s</para
                            >' %(line[2]), textstyle)]
```

```
139                 data.append(l)
140             if len(data)>0:
141                 entries.append(Paragraph('<para>Cron entries:<para>', headingstyle))
142                 t=Table(data, (None, None, None, None, None, None, 11*cm), None, hAlign='
                        LEFT')
143                 t.setStyle(TableStyle([('SPAN', (0, x), (4, x)) for x in tospan]+
144                                       [('%sPADDING' % x, (0, 0), (-1, -1), 0) for x in ['TOP'
                                            , 'BOTTOM']]+
145                                       [('%sPADDING' % x, (0, 0), (-1, -1), 2) for x in ['LEFT
                                            ', 'RIGHT']]+
146                                       [
147                                           ('VALIGN', (0, 0), (-1, -1), 'TOP'),
148                                           #('GRID', (0, 0), (-1, -1), 0.5, colors.grey),
149                                       ]
150                                      ))
151                 entries.append(t)
152
153
154         # ADD CVE SUMMARY
155         entries.append(Paragraph('<para spaceBefore=30>Vulnerable packages</para>',
                headingstyle))
156         data = [['Package', 'Version', ' Vulnerabilities ', '', ''], ['', '', Paragraph('<para
                textColor=red align=center><b>HIGH</b></para>', textstyle), Paragraph('<
                para textColor=orange align=center><b>MEDIUM</b></para>', textstyle),
                Paragraph('<para textColor=yellowgreen align=center><b>LOW</b></para>',
                textstyle)]]
157         vulnerable = {}
158         totals = [0, 0, 0]
159         if 'cves' in fs:
160             for x in fs['cves']:
161                 key = (x[1], x[14])
162                 if key not in vulnerable:
163                     vulnerable[key] = [0, 0, 0]
164                 vulnerable[key][(0 if x[5] == 'High' else (1 if x[5] == 'Medium' else 2))] +=
                        1
165             notnull = lambda x: x if x > 0 else ''
166         for x in sorted(vulnerable.keys(), key=lambda x: x[0]):
167             name = self.get_name_with_origin(x[0])
168             data.append((name, self.limit(x[1], 20), notnull(vulnerable[x][0]), notnull(
                    vulnerable[x][1]), notnull(vulnerable[x][2])))
169             for i in range(0, 3):
170                 totals[i] += vulnerable[x][i]
171         data.append(['Total:', '', totals[0], totals[1], totals[2]])
172         data.append(['', '', sum(totals), '', ''])
```

```
173
174             tvulnerable = Table(data, (6*cm, 4*cm, 2*cm, 2*cm, 2*cm), None, hAlign='LEFT')
175             tvulnerable.setStyle(TableStyle([
176                 ('SPAN', (0, 0), (0, 1)), # product
177                 ('SPAN', (1, 0), (1, 1)), # version
178                 ('SPAN', (-3, 0), (-1, 0)), # vulnerabilities
179                 ('SPAN', (0, -2), (1, -1)), # total
180                 ('SPAN', (-3, -1), (-1, -1)), # grand total
181                 ('GRID', (0, 0), (-1, -1), 0.5, colors.grey),
182                 ('ALIGN', (0, 0), (-1, -1), 'CENTER'),
183                 ('VALIGN', (0, 0), (-1, -1), 'MIDDLE'),
184             ]))
185
186             entries.append(tvulnerable)
187
188             # ADD SPECIFIC CVEs
189             if 'cves' in fs and len(fs['cves'])>0:
190                 entries.append(Paragraph('<para spaceBefore=30>Detected vulnerabilities<para>'
                        , headingstyle))
191                 # exploitable first
192                 for c in sorted(fs['cves'], key=lambda x: x[4] not in exploits):
193                     if len(c) < 14:
194                         continue
195                     if c[6] == '2.0': # CVSS 2.0
196                         description = c[12].replace('<', '&lt;').replace('>', '&gt;')
197                         para_exploits = '' if c[4] not in exploits.keys() else Paragraph('<para
                                align=justify>Exploits: '+', '.join(exploits[c[4]])+'</para>',
                                textstyle)
198                         data = [['', c[4], '%s %s\n(%s %s %s)\n' % (self.get_name_with_origin(
                                c[1]), self.limit(c[14], 20), c[0], c[1], c[2]), 'Base:', c[8]], ['', '', '
                                ', 'Impact:', c[9]], ['', '', '', 'Exploitability:', c[10]], ['', c
                                [11], '', 'Score:', c[7]], [Paragraph('<para align=justify>'+
                                description+'</para>', textstyle), '', '', '', ''], [para_exploits, '',
                                '', '', '']]
199                     else:
200                         data = []
201                     t = Table(data, colWidths=(0.5*cm, 6*cm, 7*cm, 2.5*cm, 1.5*cm))
202                     color = colors.yellow # low severity
203                     if c[5] == 'Medium':
204                         color = colors.orange
205                     elif c[5] == 'High':
206                         color = colors.salmon
207                     t.setStyle(TableStyle([
208                         #('GRID', (0, -2), (-1, -2), 0.5, colors.grey),
```

```
209                         ('VALIGN', (0, 0), (-1, -1), 'MIDDLE'),
210                         #('BACKGROUND', (0, 0), (-1, 3), color),
211                         ('BACKGROUND', (0, 0), (0, 3), color),
212                         ('SPAN', (0, -1), (-1, -1)), # exploit
213                         ('SPAN', (0, -2), (-1, -2)), # description
214                         ('SPAN', (1, 0), (1, 2)), # cve
215                         ('SPAN', (2, 0), (2, -3)), # package
216                         ('ALIGN', (3, 0), (3, 4), 'RIGHT'),
217                         ('FONTSIZE', (0, 0), (-1, -1), 10),
218                         ('FONTSIZE', (1, 0), (1, 0), 15), # cve
219                     ]))
220                     entries.append(KeepTogether(t))
221             entries.append(PageBreak()) # for every FS
222         doc.build(entries)
223         if not silent:
224             log.ok('Report generated.')
225 
226         return None
227 
228     def get_name_with_origin(self, name):
229         tag = self.parameters['TAG'].value
230         if name in tb[tag+'_fake_packages']:
231             name += ' (detected)'
232         elif name in tb[tag+'_alias_packages']:
233             name += ' (alias)'
234         return name
235 
236 
237     def limit(self, string, maxlen):
238         if len(string)>maxlen-3:
239             return string[:maxlen-3]+'...'
240         return string
```

Výpis 27: Modul *report.iot*

D Zdrojový kód *template/thread.py*

```
#!/usr/bin/env python3
"""
This file serves as template for modules meant to run in the background.
"""
from source.modules._generic_module import *

class Module(GenericModule):
    def __init__(self):
        super().__init__()
        self.authors = [
            Author(name='', email='', web=''),
        ]
        self.name = ''
        self.short_description = ''
        self.references = [
            '',
        ]
        self.date = '2999-12-31'
        self.license = 'GNU GPLv2'
        self.version = '0.0'
        self.tags = [
            '',
        ]
        self.description = """
"""
        self.dependencies = {
        }
        self.changelog = """
"""
        self.reset_parameters()

    def reset_parameters(self):
        self.parameters = {
            'SILENT': Parameter(value='yes', mandatory=True, description='Suppress the output'),
            'ACTIVEROOT': Parameter(mandatory=True, description='System to work with'),
            'BACKGROUND' : Parameter(value='yes', mandatory=True, description='yes = run in background, no = wait for it...'),
            'TIMEOUT' : Parameter(value='60', mandatory=True, description='Number of seconds to run'),
        }

```

```
40      def check(self, silent=None):
41          if silent is None:
42              silent = positive(self.parameters['SILENT'].value)
43          result = CHECK_PROBABLY
44          # incorrect BACKGROUND value?
45          if not positive(self.parameters['BACKGROUND'].value) and not negative(self.
                parameters['BACKGROUND'].value):
46              if not silent:
47                  log.err('Bad %s value: %s.', 'BACKGROUND', self.parameters['
                        BACKGROUND'].value)
48              result = CHECK_FAILURE
49          # incorrect TIMEOUT value?
50          if not self.parameters['TIMEOUT'].value.isdigit() or int(self.parameters['TIMEOUT'].
                value) < 0:
51              if not silent:
52                  log.err('Bad timeout value: %d', int(self.parameters['TIMEOUT'].value))
53              result = CHECK_FAILURE
54          return result
55
56      def run(self):
57          silent = positive(self.parameters['SILENT'].value)
58          # Don't put functionality here,
59          # this will execute immediately even if module is executed as waitfor!
60          t = T(silent, int(self.parameters['TIMEOUT'].value))
61          if positive(self.parameters['BACKGROUND'].value):
62              return t
63          t.start()
64          t.join()
65          return None
66
67  class T(threading.Thread):
68      def __init__(self, silent, timeout):
69          threading.Thread.__init__(self)
70          self.silent = silent
71          self.timeout = timeout
72          self.terminate = False
73      def stop(self):
74          self.terminate = True
75      def run(self):
76          # stop executing if self.terminate
77          pass
78  lib.module_objects.append(Module())
```

Výpis 28: Šablona *thread.py*

E Příloha na CD

locasploit/locasploit.py	hlavní soubor frameworku
locasploit/source/libs/	klíčové soubory frameworku
locasploit/source/modules/	moduly frameworku
locasploit/*.db	databáze frameworku
locasploit/install*	instalační skripty
locasploit/cveupdate	seznam příkazů pro aktualizaci databáze
locasploit/iot	vzorový seznam příkazů pro analýzu firmwaru
locasploit/iotdiff	vzorový seznam příkazů pro analýzu dvou vzorků
locasploit/iotssh	vzorový seznam příkazů pro analýzu vzdáleného systému
locasploit/iotlocal	vzorový seznam příkazů pro analýzu lokálního systému
firmwares/	testovací vzorky
reports/	výsledné zprávy
samples.txt	seznam testovaných souborů a kontrolních součtů