VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

KONSTRUKCE SKRYTÝCH KANÁLŮ IPV6 A JEJICH VLIV V INFORMAČNÍM SYSTÉMU

CONSTRUCTION OF IPV6 COVERT CHANNELS AND THEIR IMPACT IN THE INFORMATION SYSTEM

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Ivan Lohunkov

AUTHOR

VEDOUCÍ PRÁCE — Ing. Viet Anh Phan

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Ivan Lohunkov ***ID:*** 241069

Ročník: 3 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Konstrukce skrytých kanálů IPv6 a jejich vliv v informačním systému

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte z literatury problematiku vývoje v programovacím jazyce python a fungování vybraných komunikačních protokolů: zejména IPv6, ICMPv6. Seznamte se také s aktuálně známými bezpečnostními problémy těchto protokolů a problematikou komunikace skrytými či postranními kanály. V rámci bakalářské práce vytvořte laboratorní scénáře, aplikaci v jazyce Python pro využití různých skrytých kanálů v IPv6 a také demonstrace jeho použití. Na základě dosažených výsledků analyzujte vlivy skrytých kanálů IPv6 na informační systémy a jejich schopnost vyhnout se detekci současných IDS (Intrusion Detection System) softwarů.

DOPORUČENÁ LITERATURA:

[1] J. Wang, L. Zhang, Z. Li, Y. Guo, L. Cheng and W. Du, "CC-Guard: An IPv6 Covert Channel Detection Method Based on Field Matching," 2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys), Hainan, China, 2022, pp. 1416-1421, doi: 10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00219.

[2] JEŘÁBEK, J. Pokročilé komunikační techniky. Skriptum FEKT Vysoké učení technické v Brně, 2020. s. 1-180.

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 28.5.2024

Vedoucí práce: Ing. Viet Anh Phan

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Tato práce se věnuje problematice skrytých kanálů pomocí protokolů IPv6 a ICMPv6. K přenosu paketů mezi zařízeními potřebujeme aplikaci napsanou v jazyce Python a také běžící software IDS na straně příjemce. Teoretická část zpracovává problematiku protokolů IPv6, ICMPv6 a možnost přenosu skrytého kanálu v nich. Zabývá se podrobnostmi jejich vytváření a možnými implementacemi. Praktická část rozebírá scénář, který popisuje, jak bude probíhat komunikace mezi odesílatelem paketů a jejich příjemcem. Popisuje také, jak vytvořit skrytý kanál a jak jej správně odeslat. Popsána také struktura aplikace, a to jak pro odesílání skrytého kanálu, tak pro jeho dešifrování. Závěrečná kapitola popisuje programy pro vytvoření a dešifrování skrytých kanálů, jejich komunikační chování a jejich silné a slabé strany. Podle analýzy teto práce lze konstatovat, že skryté kanály v protokolu IPv6/ICMPv6 ještě nejsou tak dobře známy, že téměř neexistují. To platí zejména pro rozšiřující hlavičky, které se v běžné komunikaci používají jen málo. Právě oni mohou být zdrojem, pro přenos skrytých informací. Systém Suricata IDS také dokázal propustit většinu paketů k příjemci, aniž by příjemce upozornil, že jde o podezřelý paket.

KLÍČOVÁ SLOVA

ICMPv6, IDS Suricata, IPv6, Kali Linux, Python, rozšiřující hlavičky, Scapy, skryté kanály, Wireshark.

ABSTRACT

This thesis focuses on the issue of covert channels using IPv6 and ICMPv6 protocols. To transmit packets between devices, we need an application written in Python and also running IDS software on the receiver side. The theoretical part treats the issues of IPv6, ICMPv6 protocols and the possibility of hidden channel transmission in them. It deals with the details of their design and possible implementations. The practical part discusses a scenario that describes how the communication between the sender of packets and their receiver will take place. It also describes how to create a hidden channel and how to send it correctly. It also describes the structure of the application, both for sending the covert channel and for decrypting it. The final chapter describes programs for creating and decrypting covert channels, their communication behavior, and their strengths and weaknesses. According to the analysis of this thesis, it can be concluded that covert channels in IPv6/ICMPv6 are not yet so well understood that they are almost non-existent. This is especially true for extension headers, which are barely used in normal communication. They may be the very source, for the transmission of hidden information. The Suricata IDS has also been able to pass most packets to the receiver without alerting the receiver that the packet is suspicious.

KEYWORDS

Covert channels, extension headers, ICMPv6, IDS Suricata, IPv6, Kali Linux, Python, Scapy, Wireshark.

Vysázeno pomocí balíčku `thesis` verze 4.09; `https://latex.fekt.vut.cz/`

LOHUNKOV, Ivan. *Konstrukce skrytých kanálů IPv6 a jejich vliv v informačním systému*. Bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2024. Vedoucí práce: Ing. Viet Anh Phan

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Ivan Lohunkov
VUT ID autora:	241069
Typ práce:	Bakalářská práce
Akademický rok:	2023/24
Téma závěrečné práce:	Konstrukce skrytých kanálů IPv6 a jejich vliv v informačním systému

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing.Viet Anh Phan, za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Úvod **11**

1 IPv6 **12**
1.1 Vznik . . . 12
1.2 Vlastnosti . . . 12
1.2.1 Výhody a nevýhody . . . 13
1.3 Struktura . . . 13
1.3.1 Druhy adresování IPv6 . . . 14
1.3.2 Zapis IPv6 . . . 15
1.4 Rozšiřující záhlaví (Extension Headers) . . . 15
1.4.1 Hop-by-hop options header . . . 16
1.4.2 Routing Header . . . 17
1.4.3 Fragment Header . . . 17
1.4.4 Destination options header . . . 17
1.4.5 Authentication Options header . . . 17
1.4.6 Encapsulating Security Payload header . . . 17
1.5 Internet Control Message Protocol pro IPv6 . . . 18
1.5.1 Format ICMPv6 . . . 18

2 Skryté kanály v IPv6 **19**
2.1 Hop Limit skrytý kanál . . . 19
2.1.1 Důvod použití Hop Limit jako skrytého kanálu . . . 19
2.1.2 One-bit-per-packet channel . . . 20
2.2 Flow Label skrytý kanál . . . 20
2.2.1 Důvod použití Flow Label . . . 20
2.3 Destination Options extension Header – DOH . . . 21
2.3.1 Důvod použití DOH . . . 21
2.3.2 Vytvoření skrytého kanálu pomocí Unknown Option v záhlaví DOH . . . 21
2.4 Skryté kanály ve zprávách ICMPv6 . . . 21
2.4.1 Destination unreachable . . . 21
2.4.2 Time exceeded . . . 22
2.4.3 Neighbor discovery redirect . . . 22

3 Uvedení použitých nástrojů, platforem a aplikaci **23**
3.1 Kali . . . 23
3.2 Cisco Cloud Services Router (CSR) 1000v . . . 23

3.3 Wireshark 23
3.4 Python 24
3.5 Suricata IDS 24
3.6 Scapy 24

4 Scenář **25**

5 Struktura **27**
5.1 Struktura aplikace na straně odesílatele paketu 27
5.2 Struktura aplikace na straně příjemce paketu 27

6 Implementace **30**
6.1 hop_limit.py 31
6.2 flow_label.py 33
6.3 destination_options_header.py 35
6.4 destination_unreachable.py 36
6.5 time_exceeded.py 38
6.6 neighbor_discovery_redirect.py 40

Závěr **42**

Literatura **43**

Seznam symbolů a zkratek **46**

7 Obsah elektronické přílohy **48**

Seznam obrázků

1.1 Struktura hlavičky protokolu IPv6. 14
1.2 Struktura celého paketů IPv6. 16
1.3 Struktura hlavičky protokolu ICMPv6. 18
3.1 Příklad spuštění IDS Suricata na operačním systému Kali Linux. . . . 24
4.1 Průběh komunikace. 25
4.2 Diagram případů užití aplikace. 26
5.1 Zjednodušená struktura aplikace pro odesílání paketů na Python, strana odesílatele. 28
5.2 Zjednodušená struktura aplikace pro dešifrování paketů na Python, strana příjemce. 28
5.3 Příklad detekce paketů se skrytým kanálem v IDS Suricata na operačním systému Kali Linux. 29
6.1 Příklad spuštění aplikace `covert.py` pro vytvoření skrytého kanálu v jazyce Python. 30
6.2 Příklad spuštění programu `decoder.py` pro dešifrování paketů v jazyce Python. 30
6.3 Příklad zachycených paketů se skrytým kanálem Hop Limit v programu Wireshark. 32
6.4 Příklad dešifrovaného skrytého kanálu v Hop Limit. 33
6.5 Příklad zachyceného paketů se skrytým kanálem Flow Label v programu Wireshark. 34
6.6 Příklad dešifrovaného paketu se skrytým kanálem v Flow Label. . . . 34
6.7 Příklad odesílání paketů se skrytým kanálem Destination Options Header. 35
6.8 Příklad zachycených paketů se skrytým kanálem Destination Options Header v programu Wireshark. 36
6.9 Příklad zachycených paketů s dekódovanou zprávou se skrytým kanálem Destination Options Header. 36
6.10 Příklad odesílání paketů se skrytým kanálem Destination Unreachable. 36
6.11 Příklad zachyceného paketů se skrytým kanálem Destination Unreachable v programu Wireshark. 37
6.12 Příklad zachycených paketů s dekódovanou zprávou v Destination Unreachable. 38
6.13 Příklad odesílání paketů se skrytým kanálem Time Exceeded. 38
6.14 Příklad zachycených paketů se skrytým kanálem Time Exceeded v programu Wireshark. 39
6.15 Příklad zachycených paketů s dekódovanou zprávou v Time Exceeded. 39

6.16 Příklad odesílání paketů se skrytým kanálem Neighbor discovery redirect. 40
6.17 Příklad zachyceného paketů se skrytým kanálem Neighbor discovery redirect v programu Wireshark. 40
6.18 Příklad zachycených paketů s dekódovanou zprávou v Neighbor discovery redirect 40

Úvod

Protokol IPv6 je stále poměrně mladý a neprozkoumaný. Ve srovnání s protokolem IPv4 obsahuje mnohem více rezervačních adres, které protokolu verze 4 v současné době chybí. Bude trvat ještě hodně času, než se s protokolem IPv6 lépe seznámíme, ale v současné době jej lidé využívají a vytvářejí skryté kanály uvnitř tohoto paketu. Skrytý kanál může být buď v hlavní hlavičce protokolu IPv6, nebo v rozšířené hlavičce. Výjimkou nejsou ani zprávy protokolu ICMPv6, který s výše uvedeným protokolem úzce souvisí.

Cílem této práce je nastudovat konstrukce skrytých kanálů a ukázat, jak mohou ovlivnit informační systém. V rámci analýzy budou zohledněny skryté kanály v obou protokolech, IPv6 i ICMPv6. Pomocí softwaru IDS Suricata a nástroje pro zachytávání paketů v programu Wireshark můžeme zjistit, jak probíhá komunikace mezi oběma zařízeními. Pomocí knihovny Scapy byly implementovány dva programy v jazyce Python, z nichž jeden vytváří skrytý kanál se zprávou a druhý tuto zprávu dešifruje.

V kapitole 1 je popsán vznik protokolu IPv6, jeho rozdělení, různé výhody a nevýhody a co mohou obsahovat jeho rozšiřující hlavičky. Příští kapitola 2 je podrobnější popis skrytých kanálů v různých částech paketu IPv6. Následující kapitola 3 popisuje programy a operační systémy, které budou použity v průběhu komunikace. Kapitola 4 popisuje komunikační scénář, který popisuje celou cestu paketu. V kapitole 5 má v sobě strukturu aplikace obou programů, popisuje fungování a jejich správnou funkci. Kapitola 6 obsahuje implementaci obou aplikací, podrobnější popis vytvářeného skrytého kanálu a jeho dešifrování. Na základě analýz jednotlivých skrytých kanálů byly také popsány jejich výhody a nevýhody.

1 IPv6

1.1 Vznik

Koncem 20. století zaznamenal protokol IPv4 obrovský nárůst uživatelů, načež svět začal přemýšlet, jak vyřešit problém s omezeným počtem adres. V 90. letech byla také zahájena soutěž o vytvoření rozsáhlejšího internetového protokolu, který by byl lepší než verze IPv4. Společnost IETF, která ji vyhlásila, schválila několik kandidátů, kteří by mohli nahradit starý protokol. Do roku 1996 schválila řadu RFC definujících již dobře známý internetový protokol verze 6 neboli IPv6. Tento protokol má verzi 6, protože verze 5 byla experimentálním protokolem určeným pro přenos videa a zvuku [1].

Od počátku roku 2010 se protokol IPv6 začal rychle používat po celém světě. Používají ho velké kampaně, jako například Google a Microsoft. Protokol IPv4 zatím stále vede nad protokolem IPv6, ale bude trvat, než celý svět v pomalu přejde na nový protokol, což bude nadlouho, pokud se vůbec podaří.

1.2 Vlastnosti

Délka adresy byla čtyřikrát delší oproti IPv4. To znamená, že dokázal uložit o 2^{128} více než předchozí protokol, který ukládal 2^{32} adres. Hlavička internetového protokolu byla také změněna, aby vypadala jednodušeji. Díky snížení velikosti směrovacích tabulek ve směrovačích může efektivněji kombinovat počet adres a také lépe plnit požadavky na hierarchii adresování [2].
Vlastnosti:

- Záhlaví bylo pro lidi jednodušší, proto bylo s vyplňováním těchto položek méně starostí.
- Podporování nových protokolu ICMPv6 a DHCPv6.
- Fungování se třemi druhy adres – unicast, multicast, anycast.
- Díky technologii SLAAC (Stateless Address Autoconfiguration) je možné konfigurovat adresy automaticky, bez potřeby DHCPv6 serveru. To výrazně zjednoduší plánování sítě (Network Management).
- Přidání podpory šifrování přenosů IPsec, které umožní zabezpečení nového protokolu. [2]

1.2.1 Výhody a nevýhody

Výhody

- „Čitelnější“ hlavička protokolu.
- Obrovské množství dostupných adres.
- Lepší fungování multicastových přenosů.

Nezjevné výhody

- Sítě typu peer-to-peer, tj. sítě, v nichž mohou zařízení komunikovat přímo mezi sebou, budou fungovat lépe.
- Odstranění problémů s nedostatkem adres, nyní bude každé zařízení mít svojí vlastní adresu.

Nevýhody

- Nový internetový protokol je mladý, takže kvůli nedostatku času na testování se lidé stále nenaučili konfigurovat IPv6, protože málo rozumí jeho funkcím.
- Neexistuje zpětná kompatibilita s protokolem IPv4, což vede k použití režimu dual-stack, aby bylo možné podporovat oba protokoly.
- Nákladný přechod na protokol IPv6. Pro firmy není výhodné přecházet na nový protokol, protože vzniká problém s jeho údržbou, zabezpečením a instalací.
- Zavedení nového protokolu přinese 2x vyšší náklady na práci a čas pro oba protokoly na jednom zařízení, a těchto zařízení může být tisíc, a také přinese více různých útoků.

1.3 Struktura

Struktura protokolu IPv6 podle obr. 1.1 je následující:

1. **Verze** (4 bity) – určuje, která verze internetového protokolu je v paketů. Zde to bude verze 6.
2. **Třída provozu** (8 bitů) – Umožňuje konfigurovat stanovení priority v rámci paketů.
3. **Identifikace toku dat** (20 bitů) – jednoduché směrování paketů po stejné cestě díky stejné hodnotě těchto paketů. Pokud je hodnota 0, pak nepatří do žádné cesty.
4. **Celková délka přenášených dat** (16 bitů) – délka přenášených dat bez základního záhlaví.
5. **Next Header** (8 bitů) – vložená informace v záhlaví, například záhlaví rozšíření. Záhlaví pro informace o mezisměrném spojení, které následuje po IPv6 adresy. Pokud není použita, znamená to, že se jedná o přítomnost protokolu vyšší vrstvy (nejčastěji TCP nebo UDP) [2].

6. **Limit počtu skoků** (Hop limit) (8 bitů) – Stejné jako u protokolu IPv4 pole TTL (Time to live). Zobrazuje maximální počet skoků, které může paket udělat. Hlavním účelem tohoto pole je nezacyklit paket a ty, které nelze doručit, jsou jednoduše odstraněny.
7. **IPv6 adresa odesílatele nebo příjemce paketů** (128 bitů) – informace o adrese odesílatele nebo příjemce paketů.

Ze specifikace je patrné, že celková velikost základní hlavičky je 40 bajtů. Na rozdíl od protokolu IPv4, kde byla délka variabilní, je v protokolu IPv6 fixní. Základní část hlavičky IPv4 má velikost 20 bajtů, což je polovina velikosti hlavičky IPv6. I když je délka adres IPv6 čtyřikrát delší, nejedná se o výrazný nárůst.

Obr. 1.1: Struktura hlavičky protokolu IPv6.

1.3.1 Druhy adresování IPv6

V současné době máme tři typy adresování [3]:

1. Individuálné **(Unicast)** – které umožňuje komunikaci s jedním počítačem na straně příjemce.
2. Skupinové **(Multicast)** – které odesílá pakety na tuto cílovou adresu a posílá je všem možným členům skupiny.
3. Výběrové **(Anycast)** – podobný princip jako adresování multicast, ale s tím rozdílem, že paket je odeslán pouze jednomu členu skupiny, nejčastěji tomu, který je nejblíže k odesílateli.

1.3.2 Zapis IPv6

Adresa protokolu IPv6 se skládá z 8 skupin (16 bitů na skupinu) po 4 hexadecimálních číslech, která jsou oddělena dvojitou tečkou, na stejném principu jako v protokolu IPv4, např.:

4000:0000:0000:0000:0BCA:AFDE:0000:8E7F.

IPv6 se od IPv4 liší, vypadá mnohem delší, a tak bylo rozhodnuto se to změnit, aby se formát zmenšil na čitelnější:

4000::0BCA:AFDE:0000:8E7F.

Můžeme zkrátit pouze ty skupiny, které mají v sobě nuly, pokud existuje několik takových skupin a jsou uspořádány za sebou, můžeme to zkrátit pomocí „::“, že jsou v nich skryté nuly. Tuto zkratku můžeme použít pouze jednou na straně s více nulami. Je také možné zkrátit úvodní nuly v každé skupině, aby se dále zkrátila délka adresy protokolu IPv6 nebo pokud máme skupinu čtyř nul na jiném místě, můžeme je zjednodušit na jednu:

4000::BCA:FDE:0:8E7F.

1.4 Rozšiřující záhlaví (Extension Headers)

Existují také různé rozšiřující hlavičky protokolu IPv6, které budou používat síťová zařízení (směrovače, přepínače atd.), pak se proběhne rozhodnutí, poslat dál, nebo zpracovat. Rozšiřující hlavičky mají délku který bude dělitelný osmi oktetů.

Podle RFC 2460 [7] máme k dispozici šest různých rozšiřujících záhlaví:

- Hop-by-hop options header,
- směrovací hlavička (routing header),
- hlavička fragmentace (fragment header),
- destination options header,
- hlavička autentizace (authentication header),
- hlavička pro zabezpečený přenos dat (encapsulating security payload header).

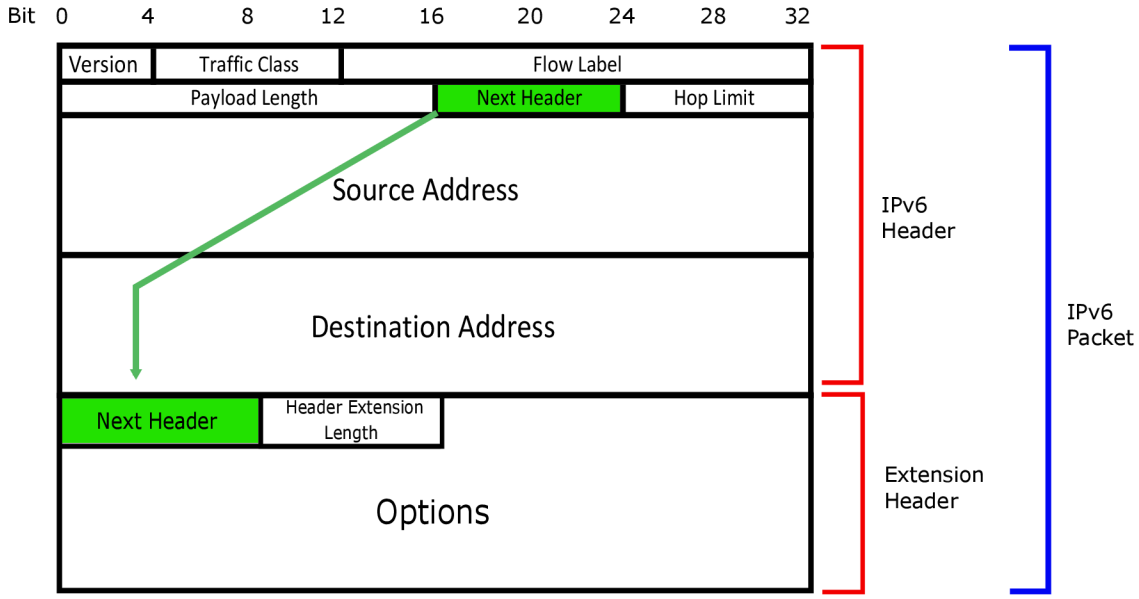
Mají také stanovenou pořadí, která je uvedená v tab. 1.1

Tab. 1.1: Pořadí rozšiřujících záhlaví IPv6

Číslo pořadí	Název rozšiřující záhlaví
1	IPv6 header
2	Hop-by-hop options header
3	Destination options header
4	Routing Header
5	Fragment Header
6	Authentication Options header
7	Encapsulating security payload header
8	Destination options header
9	Hlavička vyšší vrstvy

1.4.1 Hop-by-hop options header

Tato hlavička nese informaci, kterou si může ověřit každý uzel, jímž tento paket prochází. To může být upozorněním pro směrovač, že paket obsahuje data, která by ho mohla zajímat. Jelikož se jedná o první hlavičku rozšíření, měla by být vždy na prvním místě, nikdy, takže Hop-by-hop nebude po jiných hlavičkách rozšíření, pokud existují. Díky různým typům definovaných i nedefinovaných možností a variabilní délce poskytuje tato rozšiřující hlavička možnosti pro vysokorychlostní skryté kanály v protokolu IPv6. Jeho přítomnost v záhlaví uvidíme v položce Next Header s hodnotou 0, jak se uvedeno na obr. 1.2.



Obr. 1.2: Struktura celého paketů IPv6.

1.4.2 Routing Header

Směrovací hlavička se používá k určení jednoho nebo více mezilehlých uzlů, kterými má paket na své cestě k cíli projít. Definuje také směrování s přísným zdrojem a směrování s volným zdrojem pro paket. Pokud je směrování rozložené, musí být každé zprostředkující zařízení vzdáleno jeden skok [5]. V Next Header mají hodnotu 43.

1.4.3 Fragment Header

Tato hlavička má v sobě informace, aby cílovému uzlu sdělil velikost paketů, který byl fragmentován, a cílový uzel tak mohl paket znovu složit. Oproti protokolu IPv4 se fragmentace paketů provádí vždy ve zdrojovém uzlu. Dochází k fragmentaci paketů, pokud MTU (Maximum Transmission Unit) linky není dostatečně velká [6]. V Next Header je definována hodnotou 44.

1.4.4 Destination options header

Obsahuje informace určené pouze pro cílové uzly. Podle tab. 1.1 k projevu této rozšířené hlavičky v zásobníku hlaviček IPv6 dochází dvakrát:

1. Téměř na začátku, za hlavičkou Hop-by-hop.
2. Na konci tohoto zásobníku.

Vzhledem k tomu, že tato hlavička má stejný formát jako Hop-by-hop, skryté kanály pro tyto hlavičky budou navzájem podobné. V Next Header bude mít svou hodnotu 60.

1.4.5 Authentication Options header

Poskytuje autentizační mechanismy pro ověření integrity a původu dat. Jedna ze záhlaví tvořících protokol IPsec. Za účelem kontroly integrity bez připojení vypočítává tato hlavička (ICV – integrity check value) pro každý paket na základě určitých polí, včetně hlavičky IPv6 a dalších rozšiřujících hlaviček. Má hodnotu 51 v Next Header.

1.4.6 Encapsulating Security Payload header

Tato rozšiřující hlavička je také součástí IPsec jako hlavička autentizace, ale má jiný účel. Zajišťuje šifrování nebo ověřování pro bezpečný přenos informací v end-to-end komunikaci. V Next Header uvidíme s hodnotou 50.

Každá z těchto rozšiřujících hlaviček má specifickou funkci a může být přítomna nebo chybět v IPv6 protokolu podle požadované funkcionality a nastavení. Používání těchto hlaviček umožňuje flexibilitu a rozšiřitelnost IPv6 pro různé aplikace a scénáře síťové komunikace.

Určité rozšířené záhlaví plní svou zamýšlenou funkci. V IPv6 nejsou povinné v závislosti na požadované funkci. Poskytují dostatečnou rozšiřitelnost protokolu, ale také zvyšují počet skrytých kanálů v IPv6.

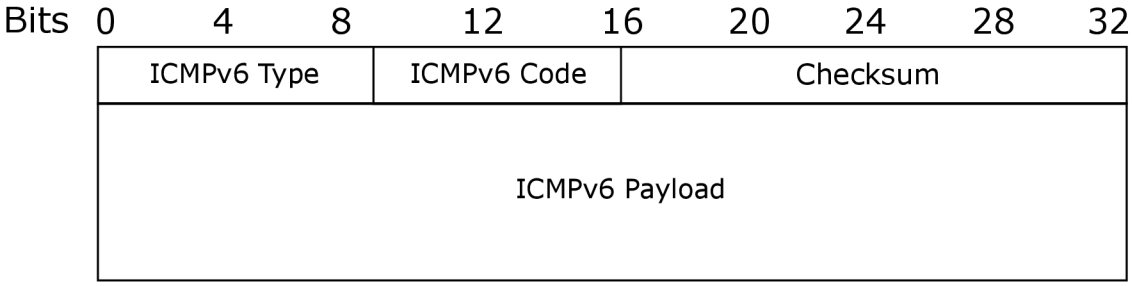
1.5 Internet Control Message Protocol pro IPv6

Tento protokol je nedílnou součástí IPv6. Používá se k indikaci chyb v komunikačním procesu nebo při zpracování stejného paketů. Je to základní protokol, protože provoz IPv6 vyžaduje použití zpráv ze současného ICMPv6 pro různé důležité procesy, jak je popsáno v RFC 4443 [3]. Má v sobě nové funkce IPv6, např. MTU Path Discovery a Neighbor Discovery [8]. Je poměrně jednoduchý a může být předmětem mnoha zneužití a útoků. Má hodnotu 58 v Next Header.

1.5.1 Format ICMPv6

Struktura protokolu ICMPv6 podle obr. 1.3 je následující [3]:

1. **Typ** (8 bitů), obsahuje informace o typu zprávy. Chybové zprávy se pohybují v rozmezí hodnot 0–127 a informační zprávy v rozmezí 128–255.
2. **Kod** (8 bitů), závisí na typu zprávy. Používá se k vytvoření další úrovně podrobnosti zprávy.
3. **Pole kontrolního součtu** (16 bitů), slouží k detekci poškození dat ve zprávě ICMPv6.



Obr. 1.3: Struktura hlavičky protokolu ICMPv6.

2 Skryté kanály v IPv6

Při analýze je důležité vyhodnotit obě strány bezpečnosti počítačových systémů. Mohou být veřejné i skryté. **Veřejný kanál** – je kanál, který je určen pro autorizovaný přenos dat. **Skrytý kanál** – je určen k přenosu skrytých informací, které jsou pro ostatní osoby neviditelné. Je určena pouze osobě, která ji má přijmout. Jde o to, že při běžné komunikaci je obtížné zaznamenat jakékoli změny, například v protokolu IPv6. Lze je rozdělit na **ukládací** a **synchronizační** kanály [6].

- **Ukládací kanál** je přímý zápis do úložného místa jedním procesem a jeho přímé čtení.
- **Synchronizační kanál** zahrnuje signalizační mechanismy, jako je procesor nebo čas, takže změna doby odezvy pozorovaná jiným procesem může předat informaci.

Skryté kanály, které obsahují skryté informace pro nezúčastněné osoby, nejsou určeny k omezení obvyklého přístupu k informacím v procesu komunikace, ale k zajištění toho, aby nepřitahovaly pozornost uživatelů, stejně jako k zajištění bezpečnosti samotných skrytých informací [9].

Skrytý kanál lze vytvořit pomocí řídicích bitů síťových protokolů, které jsou obsaženy v záhlaví. Řídicí informace v protokolu IPv6 jsou rozděleny na fixní pole, která je povinná, a rozšiřující hlavičky, které jsou nepovinné. Většinu skrytých kanálů lze vytvořit pomocí rozšiřujících záhlaví [10].

2.1 Hop Limit skrytý kanál

Hop limit funguje podobně jako pole hlavičky IPv4 TTL. Neumožňuje nekonečné cyklování paketů mezi uzly. Obvykle se nastavuje na 64 nebo 128 uzlů, což by mělo pro jakoukoli komunikaci stačit k přenosu paketů.

2.1.1 Důvod použití Hop Limit jako skrytého kanálu

Existuje několik důvodů, které naznačují, že použití tohoto pole pro skrytý kanál je výhodné [11]:

1. Je to rozmanitost tohoto vlastního pole, které je považováno za normální. To znamená, že pokud změníme výchozí hodnotu na něco jiného, co chceme, je to považováno za normální chování. Skládá se z 8 bitů (tj. hodnota 255), což je maximální počet uzlů, kterými může paket projít. Vzhledem k častým změnám směrovacích informací v síti nebudou změny tohoto pole podezřelé.
2. Lze stále použít pro účely nalezení nejbližšího serveru ve skupině jiných serverů, které mají stejné služby. Hop Limit odešle požadavek na nejbližší server s

hodnotou 1 a podívá se, kdo odpoví, pokud není zaznamenána žádná odpověď, následuje další požadavek s hodnotou +1 od předchozího a tak dále, dokud nepřijde požadovaná odpověď. Tímto způsobem může hostitel zjednodušit vyhledávání, aby nezatěžoval více vzdálených serverů.

2.1.2 One-bit-per-packet channel

Základní myšlenkou je zakódovat informace uvnitř tohoto pole. Metoda spočívá v nahrazení posledního bitu pole jeho jedním bitem informace. Tato metoda má své výhody i nevýhody [11]:

- Výhodou je, že se hůře detekuje skrytý kanál v cílovém uzlu.
- Nevýhodou je, že pokud změníme hodnotu posledního bitu na požadovanou, může to ovlivnit správnost kanálu, protože pokud potřebujeme přenést například jedničku, musíme pro větší správnost kanálu zvolit vhodné číslo mezi 0 a 255, aby tento vypadal přirozeněji. To nemůžeme udělat, protože se mění pouze poslední bit.

2.2 Flow Label skrytý kanál

Toto pole používá uzel k označování IPv6 protokolu. Nulové hodnoty Flow Label ukazují pakety, které nebyly označeny. Je možné, aby klasifikátory paketů používaly trojici z Flow Label, adresy odesílatele a příjemce paketů k identifikaci toku, ke kterému konkrétní paket IPv6 patří [12]. Podle [13] by mezilehlé uzly neměly měnit hodnotu Flow Label. Tato hodnota by měla být rovnoměrně rozložena, aby se minimalizovala možnost jejího uhodnutí. Pomocí toku protokolu UDP, jejichž hodnota Flow Label bude obsahovat skryté informace. Změna nenulových hodnot v izolovaných paketech UDP však bude považována za podezřelou. Pokud provést sdílení Flow Label v náhodných paketech, bude to poměrně obtížné odhalit.

2.2.1 Důvod použití Flow Label

Podle [14][15] může být Flow Label implementovaný v různých operačních systémech nedokonalý, takže ověření jeho chování může dále zlepšit celkové zabezpečení. Toto pole záhlaví protokolu IPv6 se v reálných podmínkách používá málo, proto je vhodné mít možnost s ním manipulovat a odesílát informace.

2.3 Destination Options extension Header – DOH

V seznamu pořadí záhlaví se zobrazí dvakrát, zpracovává je cílový uzel. Informace, které jsou v DOH, jsou zakódovány ve formátu TLV (Type-Lenght-Value). Při nesprávném použití možností záhlaví se může stát skrytým kanálem pro předání požadovaných informací příjemci.

2.3.1 Důvod použití DOH

V této rozšířené hlavičce IPv6 je několik metod, které umožňují skryté kanály. To naznačuje, že tato hlavička stále čeká na výzkum a další revize, aby se zamezilo těmto v současnosti skrytým kanálům.

2.3.2 Vytvoření skrytého kanálu pomocí Unknown Option v záhlaví DOH

Odesílatel, tj. v našem případě útočník, může do typu „Option“ vložit náhodná data nebo odeslat „Unknown Option“ s počátečním číslem dvě nuly. Uzel tuto volbu nerozpoznává – přeskočí ji a pokračuje [7][10]. To má za následek vytvoření skrytého kanálu. Rozsah, který není přidělen Internet Assigned Numbers Authority **(IANA)** [16], lze použít pro skrytý kanál. Pokud odešlete paket se skutečnou IP adresou a neznámou volbou IANA v záhlaví, pak jej cílový operační systém po přijetí paketů bude ignorovat a předá jej vyšší vrstvě a nebude jej považovat za podezřelý. Touto metodou skrytého kanálu lze tedy předávat informace protokolu vyšší vrstvy bez většího podezření.

2.4 Skryté kanály ve zprávách ICMPv6

Níže jsou uvedeny typy zpráv protokolu ICMPv6, které mohou sloužit jako skrytý kanál pro přenos informací. Odesílatel vkládá tajnou zprávu do pole záhlaví ICMPv6 a odešle ji příjemci.

2.4.1 Destination unreachable

Chybové zprávy „Destination unreachable“ v ICMPv6 jsou přenášeny mezi jednosměrovými adresami. Tuto zprávu může vygenerovat kterýkoli uzel během vzájemného přenosu, když nemůže předat paket z jiného důvodu než z důvodu překročení. Tyto zprávy jsou důležité pro urychlení cyklu předávání paketů [17].

Tato zpráva obsahuje vyhrazené pole o délce 4 bajty. Odesílatel skryté zprávy může do pole vložit data, což umožňuje skrytou propustnost 32 bitů/paket [9].

Systém IDS často rozpozná takový paket jako podezřelý, protože jen velmi zřídka dochází k jakýmkoli změnám ve rezervovaném pole.

2.4.2 Time exceeded

Taková zpráva se může objevit ve dvou případech, popsaných v RFC 4890 [18]:

1. Kód **0** je generován v libovolném uzlu na cestě, po které paket prochází, a odesílá se od libovolného uzlu k cíli mezi unicastovými adresami, pokud hodnota Hop Limit v tomto uzlu klesne na nulu.
2. Kód **1** je generován, když mezi koncovými uzly (end-to-end unicast) nejsou přijaty všechny segmenty fragmentované zprávy během doby opětovného sestavení.

Zpráva s kódem **0** může být vyžadována během procesu navazování spojení, pokud cesta k určitému cíli vyžaduje velký počet skoků mezi uzly.

Nevýhodou je, že množství informací, které lze přenášet, je značně malé. Protože bychom posílali zprávu v binární podobě, jejíž jeden znak by mohl při přenosu potenciálně představovat 4-5 paketů. Proto lze tento skrytý kanál použít pouze v případě, že je třeba přenést málo informací.

2.4.3 Neighbor discovery redirect

Tato zpráva obsahuje také pole o délce 4 bajty, do kterého lze vložit data se skrytou přenosovou šířkou pásma 32 bitů/paket. Může poskytovat funkce pro autokonfiguraci adres IPv6, zjišťování sousedů nebo směrovačů, prefixů, diverzity adres, duplicit a udržování informací o dosažitelnosti aktivních sousedů. Pokud je informací více než 4 bajty, budou rozděleny do několika paketů v závislosti na velikosti.

Pokud je informace s velikostí 12 bajtů, odešlou se 3 pakety po 4 bajtech. Může být odesláno v malém náhodném časovém intervalu, aby je systém IDS nepovažoval za podezřelé.

3 Uvedení použitých nástrojů, platforem a aplikaci

3.1 Kali

Kali Linux je open-source distribuce Linuxu založená na Debianu, která je určena pro různé úkoly v oblasti informační bezpečnosti, jako je penetrační testování, bezpečnostní výzkum, počítačová forenzní analýza a reverzní inženýrství [19]. Tento systém je hodně využíván v bezpečnostním průmyslu k zajištění ochrany dat. Výhodou tohoto systému je, že je vhodný pro více programů, které jsou potřebné pro lidi z oblasti bezpečnosti, a proto je v těchto popularita Kali zřejmá. V této práci používá Kali verze 2023.3 na obou virtuálních počítačích.

3.2 Cisco Cloud Services Router (CSR) 1000v

Použity 2 virtuální směrovače společnosti Cisco, které umožňují uživateli odesílat pakety mimo lokální síť. Umožňují také vytvořit vlastní síť, po které se Kali Linux automaticky připojí ke směrovači před volnou adresou IPv6 pomocí technologie SLAAC (Stateless Address Autoconfiguration). Oba směrovače jsou připojeny na straně odesílatele/příjemce jako výchozí brána. Je použita verze směrovače Cisco CSR1000V Software, 15.4(1)S2.

3.3 Wireshark

Wireshark je přední světový analyzátor síťových protokolů. Vývoj programu Wireshark se rozvíjí díky příspěvkům síťových odborníků z celého světa. Je pokračováním projektu, který byl zahájen v roce 1998 [20]. Tento program nám umožní zachytit přenášené pakety, které budou skrytými kanály. Umožňuje hledat chyby, které by bez programu pro zachytávání paketů bylo velmi obtížné odhalit. Je použita verze Wireshark 4.0.7.

Hlavní výhody používání programu Wireshark jsou [20]:

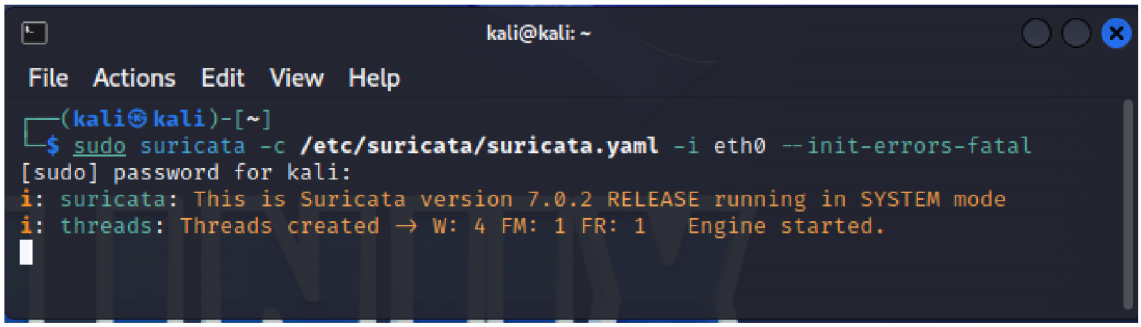
1. Živé snímání a offline analýza.
2. Multiplatformní: běží na Windows, Linuxu, OS a dalších platformách.
3. Podpora dešifrování pro mnoho protokolů, včetně IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP, WPA/WPA2.
4. Na seznam paketů lze aplikovat pravidla pro rychlou a intuitivní analýzu.

3.4 Python

Python je známý programovací jazyk, který si oblíbilo mnoho programátorů. Vytvořil ji Guido Van Rossum a poprvé vyšla v roce 1991 [21]. Obsahuje velké množství standardních knihoven, které programátorům výrazně usnadňují práci. V současné době je nejrozšířenější verzí je 3 a vyšší. V této práci je Python použit k vytvoření hlavního programu, který bude používat knihovnu Scapy, a také k vytvoření skrytého kanálu. Verze Python 3.11.4 pro vytvoření aplikace o skrytých kanálech.

3.5 Suricata IDS

Suricata je vysoce výkonný open source software pro analýzu sítě a detekci hrozeb, který používá většina organizací soukromého a veřejného sektoru [22]. Pro tuto práci budeme tento program potřebovat k analýze podezřelého provozu v komunikačním procesu. Umožňuje nastavit vlastní pravidla, která během komunikace detekují příslušné pakety. Konfigurace a nastavení pravidel v tomto programu zůstává standardní. Podle obr. 3.1 na straně příjemce se používá Suricata verze 7.0.2.



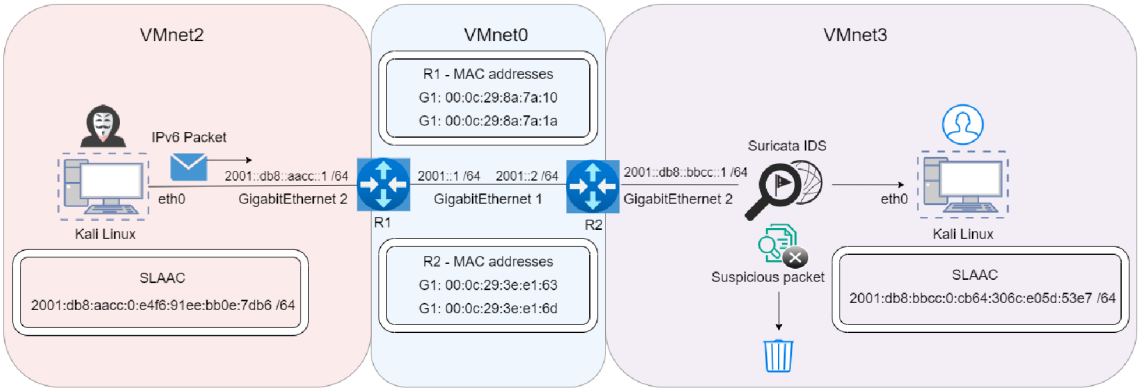
Obr. 3.1: Příklad spuštění IDS Suricata na operačním systému Kali Linux.

3.6 Scapy

Scapy – je výkonná interaktivní knihovna pro manipulaci s pakety napsaná v jazyce Python. Scapy dokáže falšovat nebo dekódovat pakety velkého množství protokolů, porovnávat požadavky a odpovědi [23]. Tato knihovna je považována za hlavní knihovnu pro tuto práci. Funkce knihovny nám umožní analyzovat nebo vytvořit potřebné pakety IPv6.

4 Scenář

Podle obr. 4.1 vidíme průběh komunikace mezi odesílajícím a přijímajícím zařízením.

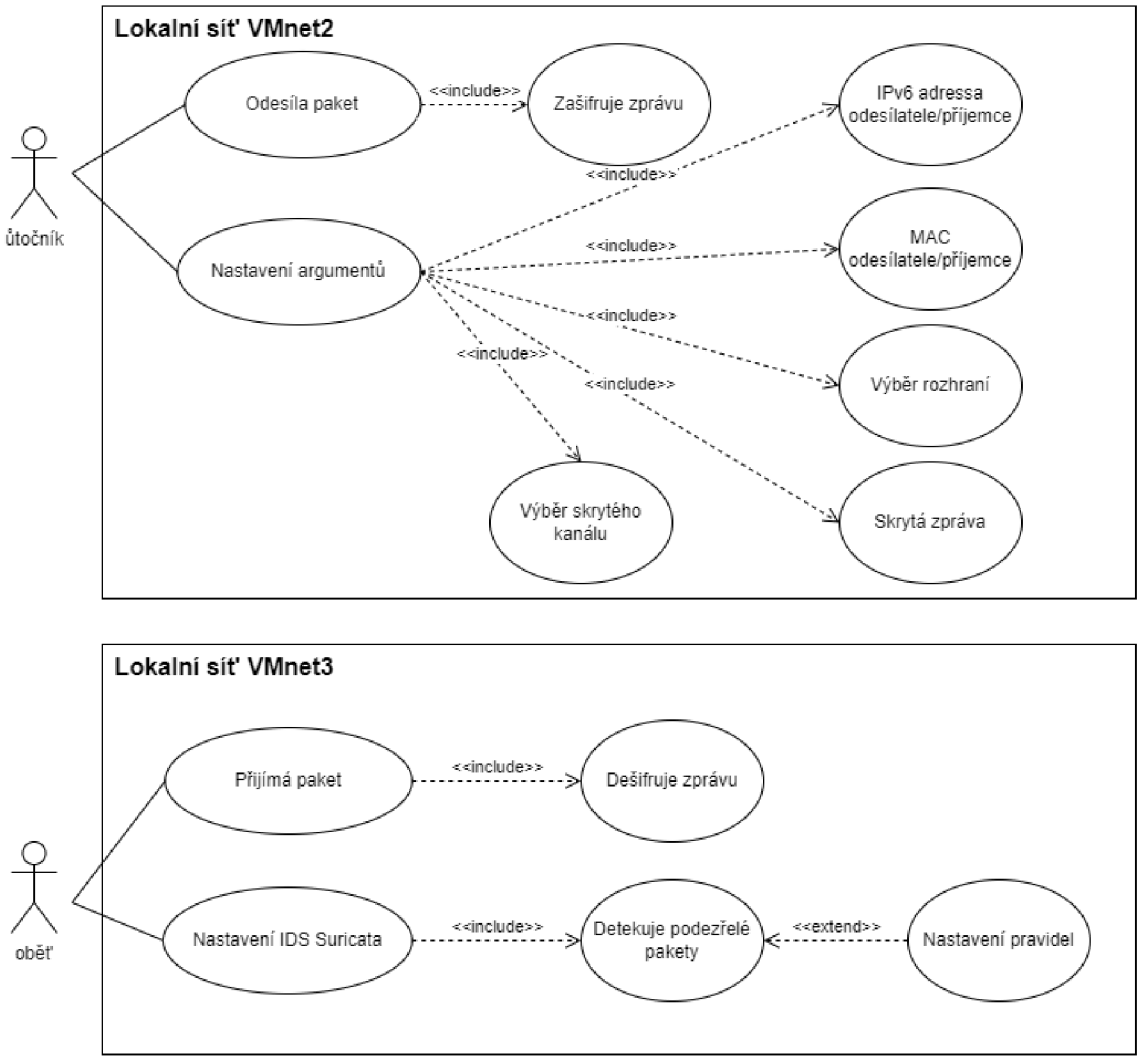


Obr. 4.1: Průběh komunikace.

- **VMnet2 (síť: 2001:db8:aacc:: /64)**:
  1. **Kali Linux**: Generuje IPv6/ICMPv6 protokol se skrytým kanálem. Zařízení je připojeno přes síťové rozhraní **eth0** a používá SLAAC pro automatickou konfiguraci své IPv6 adresy. K vytvoření skrytého kanálu použije program napsaný v jazyce **Python** a vybere, který chce vytvořit. Vybere **zdrojovou a cílovou** IPv6 adresu a také zadá **MAC adresy**, jedna je zdrojová a druhá je adresa směrovače, protože to je další zařízení, přes které paket projde. V závislosti na typu skrytého kanálu také zapíše **skrytou zprávu**.
  2. Rozhraní **GigabitEthernet 2** na **R1**: Přijme paket se skrytou zprávou pro další komunikaci. Pro úspěšnou komunikaci v síti je třeba nakonfigurovat příkazy `ip route`, které směrovači určí, kam má být paket směrován.
- **VMnet0 (síť: 2001:: /64)**:
  1. Rozhraní **GigabitEthernet 1** na **R1**: Přes toto rozhraní se odešle na další směrovač, protože jsou ve stejné síti s **R2**.
  2. Rozhraní **GigabitEthernet 1** na **R2**: přijme paket od **R1**.
- **VMnet3 (síť: 2001:db8:bbcc:: /64)**:
  1. Před zahájením odesílání je třeba spustit **dešifrovací program**, aby bylo možné analyzovat pakety se skrytou zprávou. Vybírá **síťové rozhraní**, **typ skrytého kanálu** pro dešifrování a **čas analýzy** doručených paketů. Je také nutné spustit aplikaci **Suricata**, která analyzuje tok paketů a upozorňuje na podezřelé. Doporučeno je také spustit **Wireshark** pro další analýzu paketů a skrytého kanálu v něm.

2. Rozhraní **GigabitEthernet 2** na **R2**: odešle paket přímo příjemci, v našem případě systému **Kali Linux**, přes systém **IDS**.
3. **Suricata IDS**: určí, zda je přijatý paket podezřelý, a pokud ano, smaže jej. Nebo poskytne informaci, že v paketů bylo něco změněno, a uvede typ hlášení.
4. **Kali Linux**: SLAAC se také používá pro automatickou konfiguraci své IPv6 adresy a je připojeno přes **eth0**. Po ověření systémem IDS, přijme protokol IPv6/ICMPv6 se skrytým kanálem. Poté aplikace dešifruje pakety, které obsahují očekávanou zprávu.

Na obr. 4.2 je ukázán diagram případů užití, který popisuje pochopení funkčnosti systému z pohledu odesílatele/příjemce. Příjemce paketů se skrytou zprávou může také konfigurovat pravidla pro systém IDS.



Obr. 4.2: Diagram případů užití aplikace.

5 Struktura

5.1 Struktura aplikace na straně odesílatele paketu

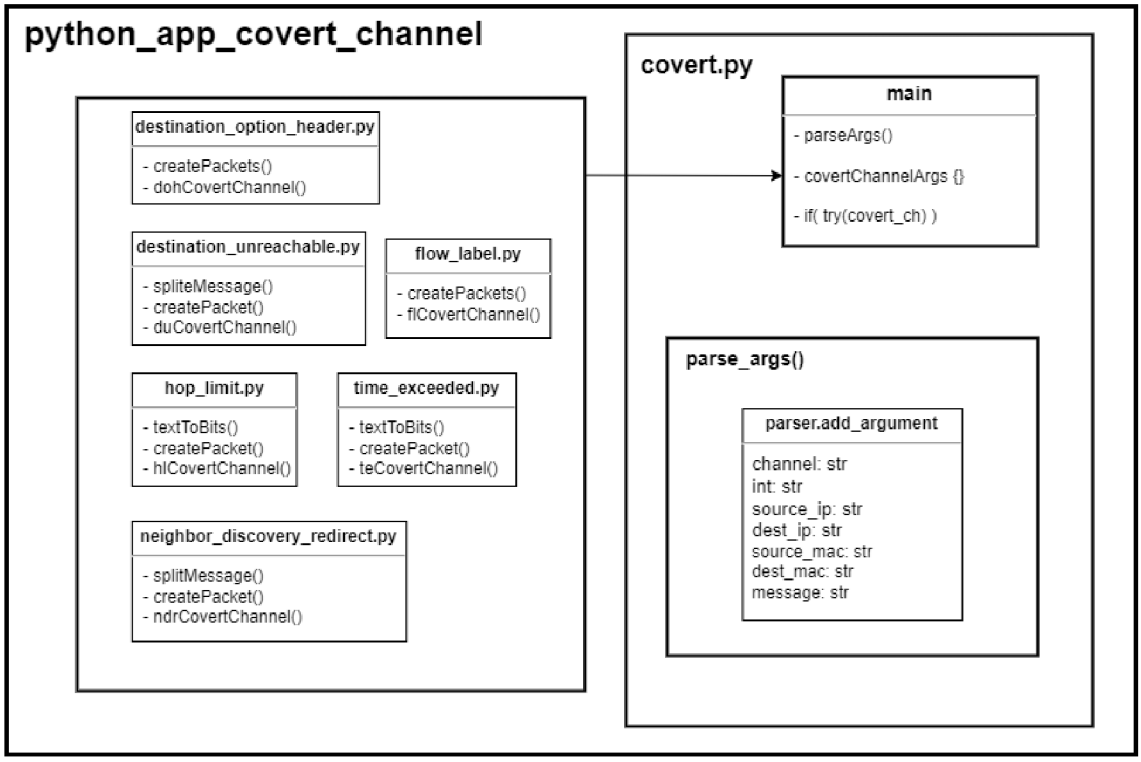
Skript `covert.py` je určen k vytvoření skrytého kanálu mezi dvěma uživateli pomocí IPv6/ICMPv6 se skrytou zprávou. Struktura podle obr. 5.1 vypadá následovně:

1. Hlavní knihovnou je Scapy pro vytvoření správného paketů pro skrytý kanál.
2. Funkce `parseArgs` je zodpovědná za použití nástroje ArgumentParser k analýze argumentů příkazového řádku. Definuje a analyzuje argumenty týkající se konfigurace sítě, včetně síťového rozhraní, výběr skrytého kanálu, zdrojové a cílové IP adresy, zdrojové a cílové MAC adresy a zprávy, která má být skrytě přenášena.
3. Napsané skryté kanály v jednotlivých souborech, které jsou pak volány v metodě `__main__` a budou podrobněji popsány v kapitole 6.
4. Má slovník, který pomáhá orientovat se ve skrytých kanálech a jejich pravopisu, který si může uživatel zavolat.
5. Vysvětlen popis jednotlivých argumentů, které je třeba zapsat, aby bylo možné vytvořit skrytý kanál.
6. Pro lepší pochopení toho, jak tuto aplikaci spustit a o čem je, byl vytvořen soubor `README.md`, kde je stručně napsáno, jak je program zahájen.

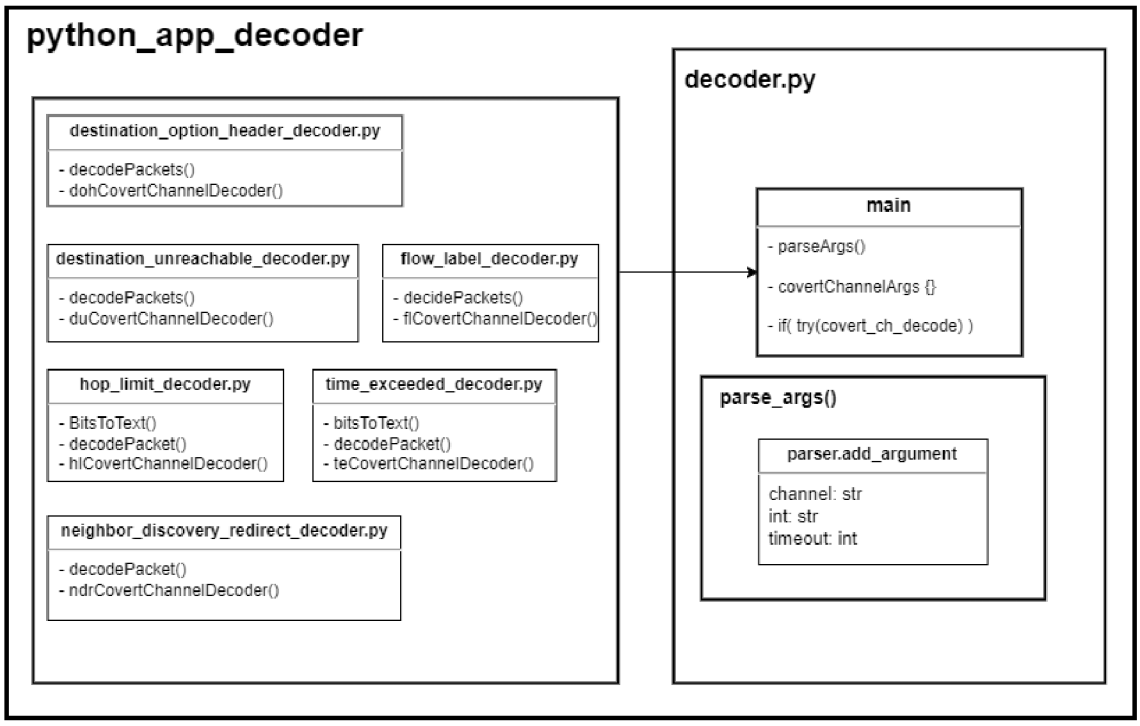
5.2 Struktura aplikace na straně příjemce paketu

Soubor `decoder.py` je určen k dešifrování přijatých paketů se skrytou zprávou umístěnou v protokolech IPV6/ICMPv6. Struktura podle obr. 5.2 vypadá následovně:

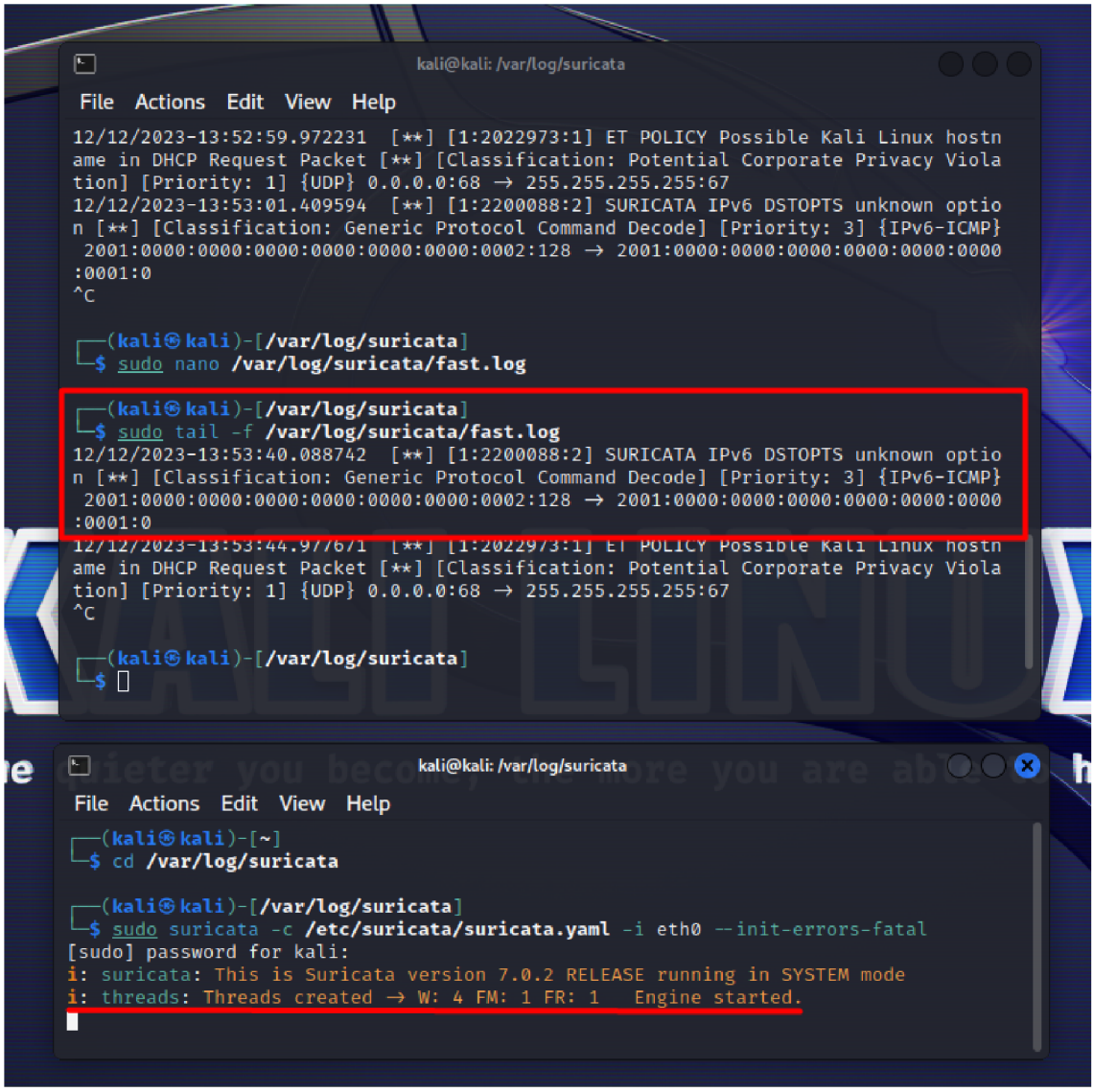
1. Hlavní knihovnou je Scapy, kde k analýze přijatých paketů používáme metodu `sniff`.
2. Funkce `parseArgs` se rovněž používá stejným způsobem jako ve struktuře aplikace odesílatele. Z argumentů můžete vybrat síťové rozhraní, skrytý kanál a čas, po který budou pakety analyzovány.
3. V metodě `__main__` se také volá potřebný dekodér, který vypíše přijatou zprávu v původní podobě.
4. Slovník poskytující různé možnosti záznamu názvů argumentů a skrytých kanálů.
5. K dispozici je také interaktivní zadávání argumentů.
6. Soubor `README.md` je vytvořen za účelem seznámení se správným spuštěním programu.



Obr. 5.1: Zjednodušená struktura aplikace pro odesílání paketů na Python, strana odesílatele.



Obr. 5.2: Zjednodušená struktura aplikace pro dešifrování paketů na Python, strana příjemce.



Obr. 5.3: Příklad detekce paketů se skrytým kanálem v IDS Suricata na operačním systému Kali Linux.

6 Implementace

Podle obr. 6.1 6.2 je příklad správného zápisu příkazu pro spuštění programu pro vytvoření a dešifrování skrytého kanálu v protokolech IPv6/ICMPv6. Zde je implementace skrytého kanálu se zprávou ve Flow Label.

```
root@kali: /home/kali/Desktop/python_app_covert_channel
File  Actions  Edit  View  Help
└─# python covert.py -h
usage: covert.py [-h] [-c CHANNEL] [-i INTERFACE] [-sip SOURCE_IP] [-dip DEST_IP] [-smac SOURCE_MAC] [-dmac DEST_MAC] [-m MESSAGE]

Covert channel application

options:
  -h, --help            show this help message and exit
  -c CHANNEL, -ch CHANNEL, --channel CHANNEL
                        one of the following covert channels (doh, DOH, destination-option-header, hl, HL, hop-limit, fl, FL, flow-label,
                        te, TE, time-exceeded, ndr, NDR, neighbour-discovery-redirect, du, DU, destination-unreachable)
  -i INTERFACE, -int INTERFACE, --interface INTERFACE
                        interface from which packet will be sent
  -sip SOURCE_IP, --source_ip SOURCE_IP
                        source ip address
  -dip DEST_IP, --dest_ip DEST_IP
                        destination ip address
  -smac SOURCE_MAC, --source_mac SOURCE_MAC
                        source mac address
  -dmac DEST_MAC, --dest_mac DEST_MAC
                        destination mac address
  -m MESSAGE, -msg MESSAGE, --message MESSAGE
                        message to be sent using covert channel

┌──(root㉿kali)-[/home/kali/Desktop/python_app_covert_channel]
└─# python covert.py -c fl -i eth0 -sip 2001:db8:aacc:0:e4f6:91ee:bb0e:7db6 -dip 2001:db8:bbcc:0:cb64:306c:e05d:53e7 -smac 00:0c:29:74:69:11
-dmac 00:0c:29:8a:7a:1a -m "Covert Message"
```

Obr. 6.1: Příklad spuštění aplikace `covert.py` pro vytvoření skrytého kanálu v jazyce Python.

```
┌──(root㉿kali)-[/home/kali/Desktop/python_app_decoder]
└─# python decoder.py -h
usage: decoder.py [-h] [-c CHANNEL] [-i INTERFACE] [-t TIMEOUT]

Decoder application

options:
  -h, --help            show this help message and exit
  -c CHANNEL, -ch CHANNEL, --channel CHANNEL
                        one of the following covert channels decoder from (doh,
                        DOH, destination-options-header, hl, HL, hop-limit, fl,
                        FL, flow-label, te, TE, time-exceeded, ndr, NDR,
                        neighbor-discovery-redirect, du, DU, destination-
                        unreachable)
  -i INTERFACE, -int INTERFACE, --interface INTERFACE
                        interface which will be sniffed for packets
  -t TIMEOUT, -tout TIMEOUT, --timeout TIMEOUT

┌──(root㉿kali)-[/home/kali/Desktop/python_app_decoder]
└─# python decoder.py -i eth0 -t 10 -c fl
```

Obr. 6.2: Příklad spuštění programu `decoder.py` pro dešifrování paketů v jazyce Python.

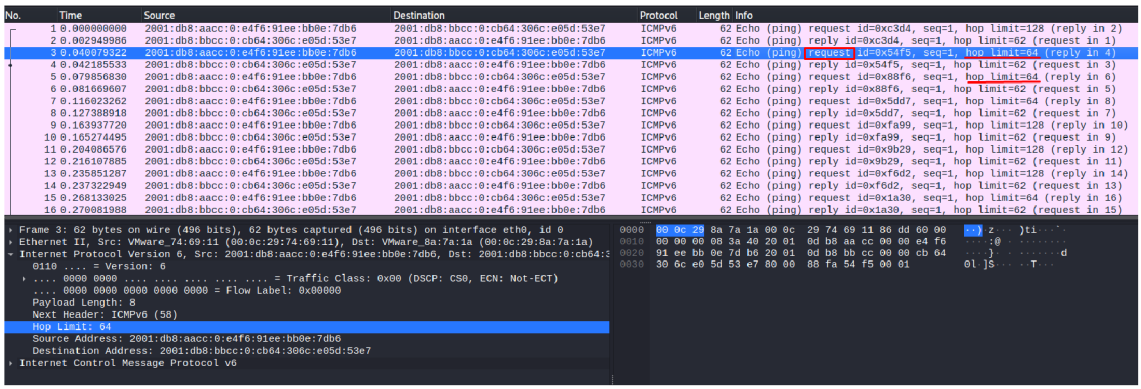
Uživatel v závislosti na aplikaci zapíše všechny požadované agrumenty:

- `-c, -ch, -channel`: Výběr skrytého kanálu.
- `-i, -int, -interface`: Výběr síťového rozhraní, v tomto případě `eth0`.
- `-sip, -source_ip`: Nastavení zdrojové adresy, ze které se paket začne odesílat. Zde je adresa Kali Linux v síti VMnet2.
- `-dip, -dest_ip`: Nastavení cílové adresy, kam má být paket odeslán. Použitá adresa je v síti VMnet3.
- `-smac, -source_mac`: Nastavení zdrojové MAC adresy. Zde je MAC adresa Kali Linux v síti VMnet2.
- `-dmac, -dest_mac`: Nastavení cílové MAC adresy, která je nakonfigurována na adresu směrovače na rozhraní **GigabitEthernet 1**, protože se jedná o další adresu, protože k cíli paketu není nakonfigurována žádná cesta s MAC adresou, takže je důležité uvést další adresu zařízení.
- `-m, -msg, -message`: Jakákoli zpráva, která bude dále vložena do paketu. Zde je příklad „Covert Message“.
- `-t, -tout, -timeout`: Nastavení doby analýzy příchozích paketů v sekundách. Toto nastavení je nepovinné, protože program lze zastavit ručně.

6.1 hop_limit.py

Tento soubor obsahuje program pro vytvoření Hop Limit skrytého kanálu, který funguje na principu one–bit–per–packet. Jinými slovy, do jednoho paketů vložíme jeden bit informace, a totiž hodnotu Hop Limit. V závislosti na množství informací, které potřebujeme přenést, vytvoříme tolik paketů, kolik potřebujeme, dokud se nenaplní celá zpráva zapsaná uživatelem.

Podrobněji je to vidět podle obr. 6.3. Tento snímek obrazovky byl pořízen ze strany odesílatele, aby ukázal, jak vypadají pakety od začátku odesílání až po příjemce. V této situaci musíme analyzovat Echo Request, protože tam se skrývá skrytá zpráva. Byly zvoleny dvě různé hodnoty Hop Limit, které mají být zašifrovány podle ASCII znaku na jeho binární reprezentaci, čímž zajistíme, že výsledkem bude 8bitový binární řetězec. Tyto hodnoty jsou 64 a 128, které jsou považovány za standardní při běžné komunikaci zařízení. Byla zašifrována zpráva „qw“, která po převedení do binární podoby vypadá jako `0111000101110111`, kde 64 se chová jako `1`, 128 naopak jako `0`. Je důležité si uvědomit, že ze strany příjemce to vypadá trochu jinak, protože v závislosti na tom, kolik zařízení mezi nimi stojí, se o tolik sníží Hop Limit. V našem případě jsou mezi odesílatelem a příjemcem 2 směrovače, což znamená, že pokud spustíte Wireshark, aby zachytil pakety, ukáže tyto hodnoty o dvě menší, tj. 62 a 126.



Obr. 6.3: Příklad zachycených paketů se skrytým kanálem Hop Limit v programu Wireshark.

Na straně příjemce podle obr. 6.4 je zobrazeno, jak byly jednotlivé pakety analyzovány s hodnotou Hop Limit. Dešifrování proběhlo úspěšně a program byl schopen rozpoznat odeslanou zprávu. Ono se provádí tak, že se analyzuje každý paket se zprávou Echo Request a jeho hodnotou Hop Limit. Pokud je to 62, je to 1, pokud 126, je to 0. Tímto způsobem je určen každý bit. Z přijatých bitů se pak vytvoří kousky po 8 bitech, které tvoří jeden bajt informace. Po složení těchto kousků se převede na celočíselné desítkové číslo, načež se porovná s tabulkami ASCII a výsledek se předá ve formě symbolu.

Tento skrytý kanál je snadno implementovatelný, protože standardní Hop Limit je 64 nebo 128, z čehož lze předpokládat, že je možné jednoduše brát hodnoty mnohem menší, na které standardní hodnota pravděpodobně neklesne, ne všude se cesta paketu skládá z mnoha zařízení. Nebo vzít o trochu vyšší hodnotu, to také nebude vypadat podezřele v procesu komunikace.

Stojí za pozornost také to, že tento skrytý kanál nebyl systémem Suricata zaznamenán jako podezřelý kvůli obvyklým hodnotám Hop Limit a typu zprávy Echo Request/Reply. V reálných podmínkách se doporučuje používat jiné hodnoty.

Nevýhodou je, že velký objem informace se obtížně přenáší, takže tento skrytý kanál lze použít k přenosu malé zprávy.

Na základě výsledků této analýzy bylo rozhodnuto, že tento skrytý kanál je považován za snadno dostupný pro přenos skrytých informací, i když ne ve velkém objemu.

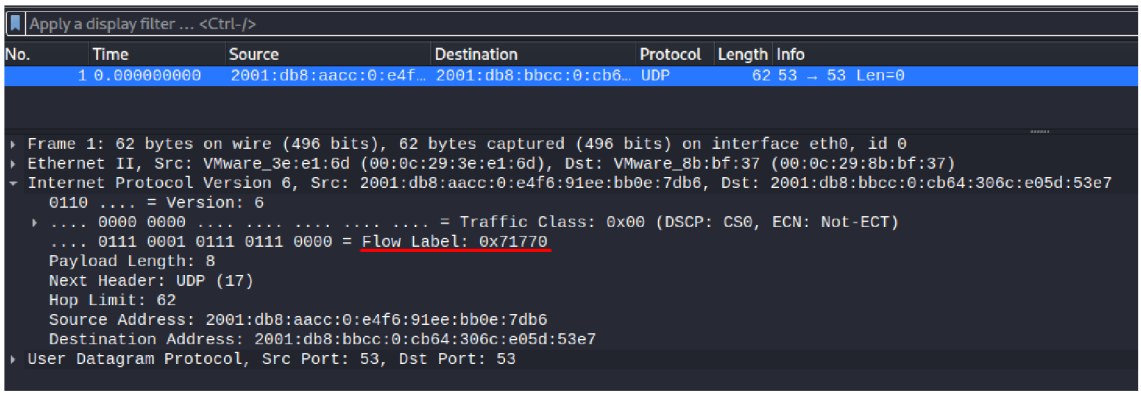
```
┌──(root㉿kali)-[/home/kali/Desktop/python_app_decoder]
└─# python decoder.py -i eth0 -t 10 -c hl
Captured bit: 0
Captured bit: 1
Captured bit: 1
Captured bit: 1
Captured bit: 0
Captured bit: 0
Captured bit: 0
Captured bit: 1
Captured bit: 0
Captured bit: 1
Captured bit: 1
Captured bit: 1
Captured bit: 0
Captured bit: 1
Captured bit: 1
Captured bit: 1
Captured bits: 0111000101110111
Decoded message: qw
Packet(s) decoded!

┌──(root㉿kali)-[/home/kali/Desktop/python_app_decoder]
└─#
```

Obr. 6.4: Příklad dešifrovaného skrytého kanálu v Hop Limit.

6.2 flow_label.py

Při vytváření tohoto skrytého kanálu použijeme pole Flow Label, který nám umožňuje odeslání skryté zprávy. K zajištění utajení opět použijeme tabulku ASCII. Podle obr. 6.5 vytvoříme skrytý kanál se stejnou zprávou jako v kapitole 6.1. Abychom mohli data zašifrovat, musíme naši zprávu nejprve převést do binární podoby, pak musíme na konec zprávy přidat 4 nuly, protože naše zpráva se skládá ze dvou 8bitových symbolů, které dohromady dávají 16 bitů, když tento pole obsahuje 20 bitů. V binárním tvaru pak posuneme první číslo o 12 číslic a druhé číslo o 4 číslice, což zajistí dostatečnou vzdálenost mezi nimi, aby se nepřekrývaly. Pro posun se používá dekadický počet symbolů `q` a `w`, což 113 a 119. Přidáme tedy tyto nuly na konec, abychom se vyhnuli chybám. Pak tuto hodnotu převedeme na hexadecimální, protože tak se zapisuje hodnota Flow Label.

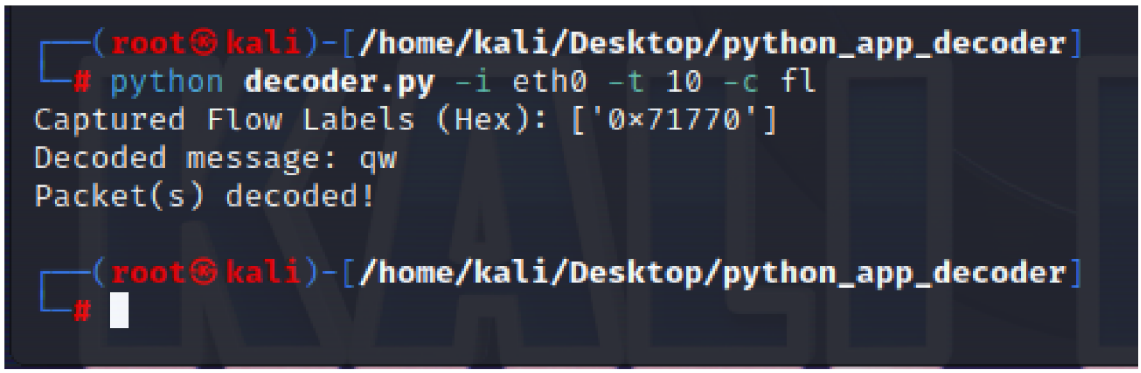
- Postup:
  1. Zpráva „qw“ podle tabulky ACSII vypadají takto: `0111000101110111`.
  2. `q` symbol posunutý o 12 znaků, `w` o 4 znaků.
  3. Přidání nul na konec.
  4. Hexadecimální převod: `0x71770`.



Obr. 6.5: Příklad zachyceného paketů se skrytým kanálem Flow Label v programu Wireshark.

Tento skrytý kanál je přenášen pomocí protokolu UDP uvnitř protokolu IPv6. Vzhledem k tomu, že Flow Label může obsahovat pouze dva znaky po 8 bitech, zbytek projde jinými pakety. Pokud potřebujete přenést například šestipísmenné slovo, bude odesláno třemi různými pakety.

Podle obr. 6.6 dešifrování probíhá v podobných krocích, jako je popsáno výše, pouze v opačném pořadí. Nyní musíme bity přesunout zpět na 12 a 4 pozice a zkontrolovat, zda nejsou nulové, aby nebyly v konečném výsledku vyvedeny. V terminálu lze také ověřit, které pakety byly zachyceny, a jejich vzhled v hexadecimální soustavě, která se používá v poli Flow Label.



Obr. 6.6: Příklad dešifrovaného paketu se skrytým kanálem v Flow Label.

Systém IDS nedokázal rozpoznat nic podezřelého, protože tato hodnota v podobě zašifrované zprávy vypadá zcela standardně. Důležité je, aby se tato hodnota neměnila po celou dobu komunikace mezi zařízeními.

Na rozdíl od předchozího skrytého kanálu tento kanál umožňuje předávat více informací bez jakéhokoli podezření.

Nevýhodou je, že tento program nevypočítá požadované pakety Flow Label při datovém toku z protokolů UDP. Tento problém můžete vyřešit tak, že vymyslíte jiný způsob ověřování paketů zavedením několika dalších změn, díky kterým bude dešifrovací software vědět, které pakety má očekávat. Mohlo by se jednat o změnu stejné hodnoty v poli Hop Limit na neobvyklou hodnotu, a pak nechat zachytit takové specifické pakety a načíst Flow Label. Pokud to nestačí, lze nastavit, aby první znak každého paketu byl v poli Flow Label stejný, pak po kontrole hodnot Hop Limit a Flow Label bude program s velkou pravděpodobností předpokládat, že se jedná o paket, který je třeba dešifrovat.

6.3 destination_options_header.py

Podle obr. 6.7 na tomto skrytém kanálu pošlete o něco větší zprávu.

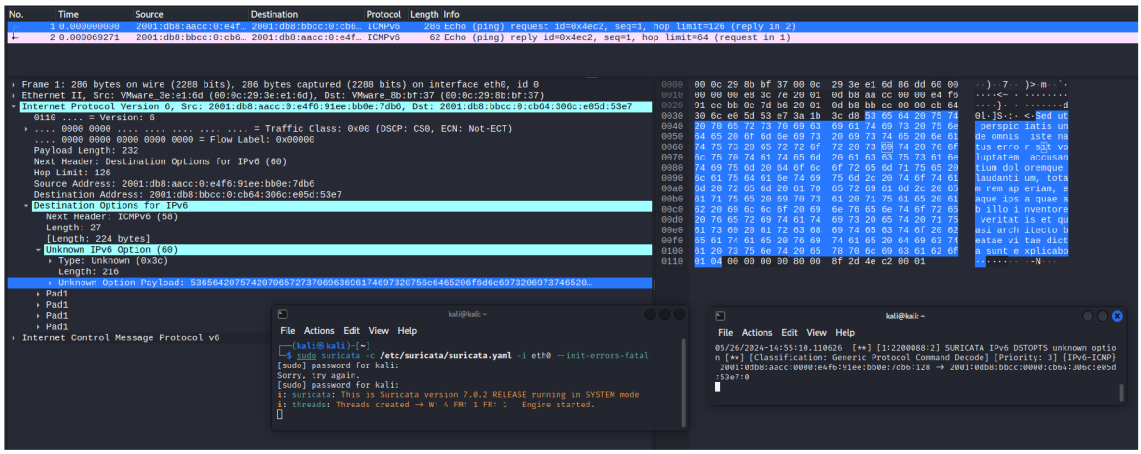
```
┌──(root㉿kali)-[/home/kali/Desktop/python_app_covert_channel]
└─# python covert.py -i eth0 -sip 2001:db8:aacc:0:e4f6:91ee:bb0e:7db6 -dip 2001:db8:bbcc:0:cb64:306c:e05d:53e7 -smac 00:0c:29:74:69:11 -dmac
00:0c:29:8a:7a:1a -m "Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, ea
que ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo" -c doh
Packet Sent!
```

Obr. 6.7: Příklad odesílání paketů se skrytým kanálem Destination Options Header.

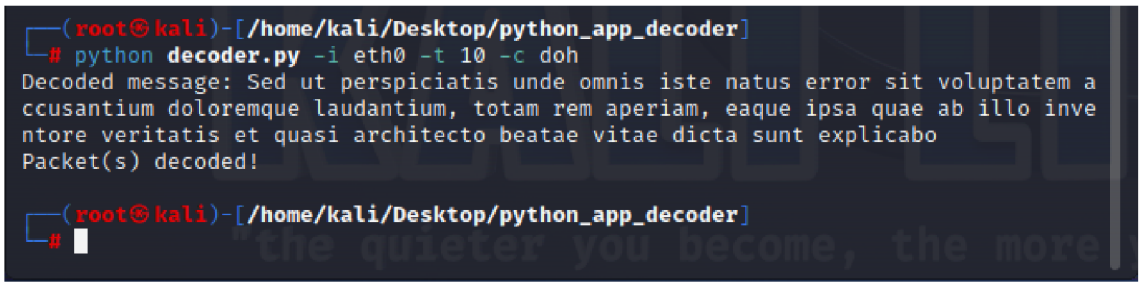
DOH umožňuje přenos dalších parametrů nebo instrukcí, které by měl zpracovat pouze koncový uzel, do kterého je paket směrován. Jak bylo uvedeno v kap. 2.3.2 používá Unknown Option k vytvoření skrytého kanálu. Podle obr. 6.8 je možné vidět, jak k tomu dochází přímo v IPv6. Text je rozdělen na části po 253 bajtech, protože každý Option má 2bajtovou hlavičku. Pro každé pole je vytvořena vlastní Option s unikátním typovým kódem `otype`.

Tento kanál se dešifruje čtením Unknown Option. Vzhledem k tomu, že toto pole je v reálných podmínkách poměrně vzácné, lze jednoduše analyzovat pakety, ve kterých toto pole obsahuje alespoň nějaká data, ve většině případů se bude jednat o skrytý kanál, který je třeba dešifrovat. Podle obr. 6.9 je vidět úspěšné dešifrování zprávy.

V tomto případě je kvůli použití tohoto pole Suricata schopna rozpoznat, že se nejedná o běžný paket. Je to proto, že používáme Unknown Option. Protože Unknown Option je často nepoužívaná, systémy IDS ji odhalí snadněji. Na základě



Obr. 6.8: Příklad zachycených paketů se skrytým kanálem Destination Options Header v programu Wireshark.

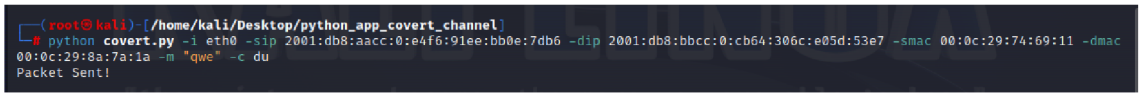


Obr. 6.9: Příklad zachycených paketů s dekódovanou zprávou se skrytým kanálem Destination Options Header.

výsledků tohoto skrytého kanálu byl paket doručen. Díky tomu může fungovat tak, jak bylo zamýšleno.

Tento skrytý kanál je nebezpečný, protože do jednoho paketu se vejde velké množství informací, i když jsou systémy schopny paket zablokovat, přesto je předán dál, což může útočníkům umožnit jeho zneužití, ale pouze jednou, poté lze problém pravděpodobně odstranit.

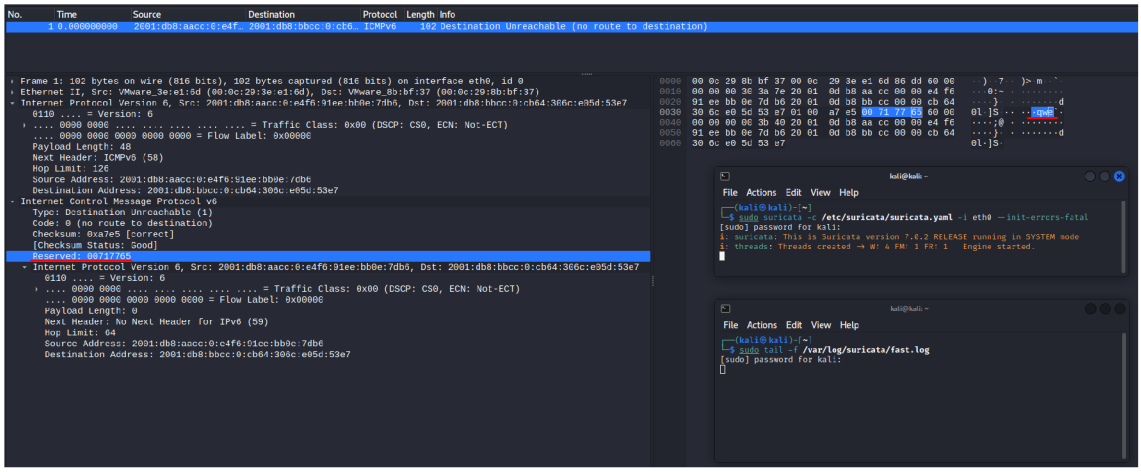
6.4 destination_unreachable.py



Obr. 6.10: Příklad odesílání paketů se skrytým kanálem Destination Unreachable.

Podle obr. 6.10 používá se skrytý kanál v protokolu ICMPv6 s hlášením Destination Unreachable.

Zde používáme male pole Reserved, abychom skrýt v něm informace. Celkem obsahuje až 4 bajty dat. Informace jsou rovněž uloženy v hexadecimální podobě.



Obr. 6.11: Příklad zachyceného paketů se skrytým kanálem Destination Unreachable v programu Wireshark.

V tomto případě na obr. 6.11 mohou být v tomto pole umístěny pouze 3 bajty, protože první bajt je obsazen nulami. To je způsobeno zvláštností vytváření paketů ve Scapy. Protože při kontrole tohoto pole před odesláním se zobrazí výsledek, kde všechny 4 bajty nesou skrytou informaci. Na tom však není nic špatného, protože tento program rozděluje zprávu do paketů tak, aby nic nechybělo. Abyste se při komunikaci vyhnuli podezření, je možné nakonfigurovat odesílání všech paketů v různých malých časových intervalech.

`Chunk`, které obsahují 2 bajty, jsou převedeny na bajtový objekt pomocí kódování UTF–8. Poté převede bajtový objekt na celé číslo interpretací bajtů jako čísla v big–endian[1] pořadí v hexadecimálně soustavě.

Na obr. 6.12 zpráva byla úspěšně dešifrována. Big–endian se používá pro dešifrování i šifrování.

Vzhledem k tomu, že tento paket je považován za oznámení o chybě, bude přenos většího počtu takových paketů obtížný ve srovnání s pakety s Echo Request/Reply jako předchozími skrytými kanály. Jelikož Destination Unreachable je chybové hlášení, bude přenos velkého množství informací podezřelý. Pokud nastavit časový interval pro odesílání těchto paketů, může to pomoci omezit podezřelý provoz paketů.

[1]Big-endian je definován jako pořadí bajtů, kdy je nejvýznamnější bajt (neboli "velký konec") vícebajtové datové hodnoty uložen na nejnižší paměťové adrese.

```
┌──(root㉿kali)-[/home/kali/Desktop/python_app_decoder]
└─# python decoder.py -i eth0 -t 10 -c du
Decoded message: qwe
Packet(s) decoded!

┌──(root㉿kali)-[/home/kali/Desktop/python_app_decoder]
└─#
```

Obr. 6.12: Příklad zachycených paketů s dekódovanou zprávou v Destination Unreachable.

V knihovně Scapy mohou být tyto poli Reserved volány různě. Například ve skrytém kanálu, který bude popsán v 6.6 je toto pole označeno jako „res“, zatímco v tomto je označen jako „unused“. Proto je vytvoření každého skrytého kanálu vždy unikátní.

Tento skrytý kanál dokázal, že je možné přenášet bez podezření přes IDS, hlavním problémem je počet paketů, které budou vypadat podezřele, pokud přijdou k příjemci v řadě za sebou ve velkém počtu. Malá delegace odesílání paketů pomůže spolehlivěji odeslat skrytou zprávu.

6.5 time_exceeded.py

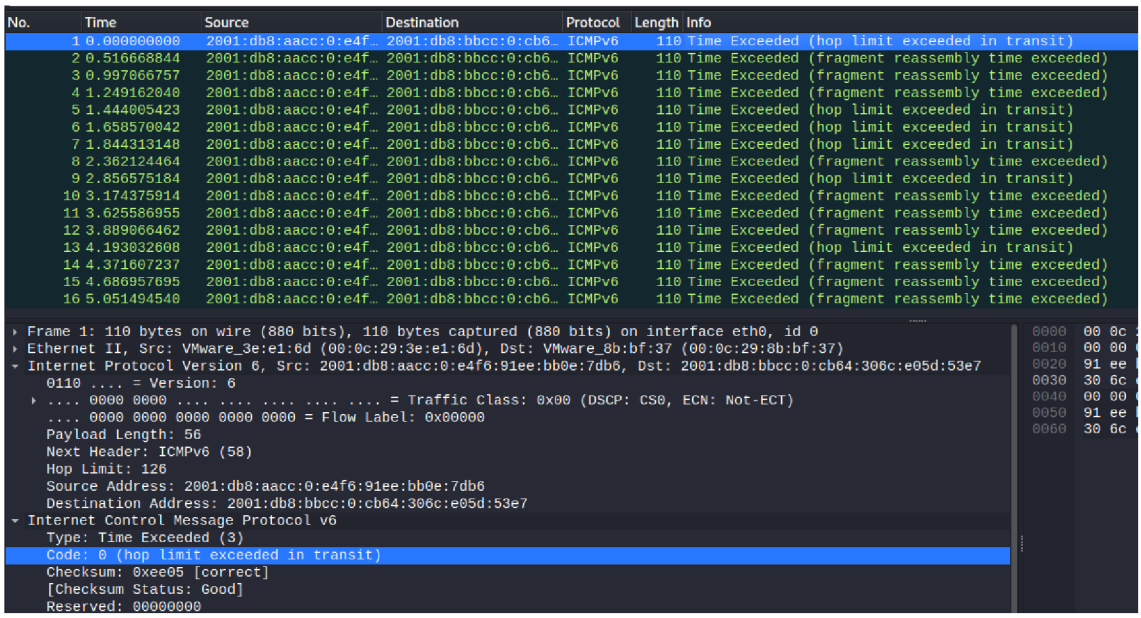
```
┌──(root㉿kali)-[/home/kali/Desktop/python_app_covert_channel]
└─# python covert.py -i eth0 -sip 2001:db8:aacc:0:e4f6:91ee:bb0e:7db6 -dip 2001:db8:bbcc:0:cb64:306c:e05d:53e7 -smac 00:0c:29:74:69:11 -dmac
00:0c:29:8a:7a:1a -m "qw" -c te
Packet Sent!
```

Obr. 6.13: Příklad odesílání paketů se skrytým kanálem Time Exceeded.

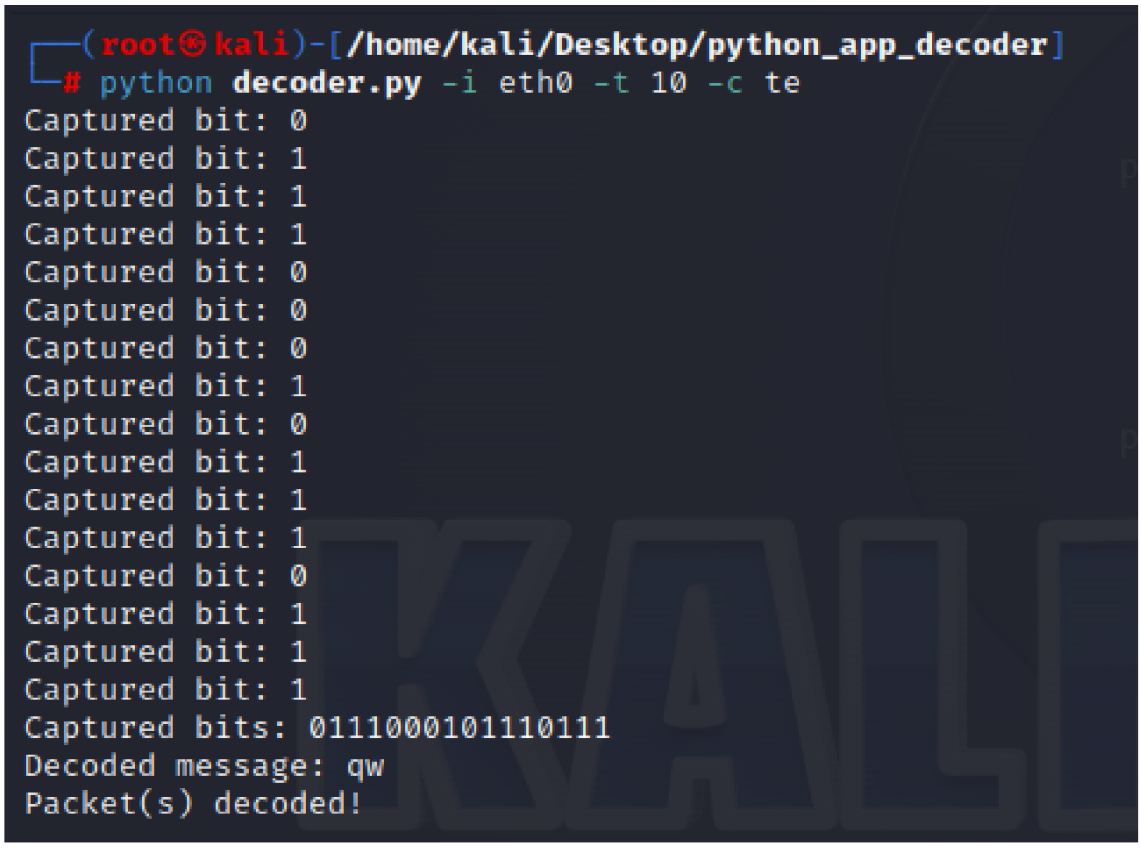
Podle obr. 6.14 byly zachyceny pakety se skrytým kanálem Time Exceeded. Zde používáme pole Code, ve kterém může být umístěna 0 nebo 1. Musíme tedy data převést do binární soustavy, kterou vložíme do těchto paketů. Zde je také pro snížení podezření pro systém žádoucí posílat tyto pakety v časových intervalech.

K Data také přidáme 8 bajtů informací, abychom simulovali, že se jedná o zcela běžný paket, který nese informace o chybách. To dává možnost zobrazit tento paket jako normální.

Příjem zprávy podle obr. 6.15 proběhl úspěšně. Pro dešifrování program vytvoří 8bitové kousky, které pak spojí do jednoho. Poté dojde k překladu na celé číslo, které se nachází v tabulce ASCII, a odtud se pak převezme symbol určený pro toto číslo. Výsledkem je zobrazení skryté zprávy.



Obr. 6.14: Příklad zachycených paketů se skrytým kanálem Time Exceeded v programu Wireshark.



Obr. 6.15: Příklad zachycených paketů s dekódovanou zprávou v Time Exceeded.

Systém IDS při příjmu paketů nerozpoznal nic podezřelého. Pro odesílání paketů se rovněž je doporučeno používat časové intervaly. Složitost spočívá ve velkém množství přenášených informací. Na druhou stranu je snadné tento skrytý kanál vytvořit. Pro odesílání paketů se rovněž doporučuje používat časové intervaly.

Nevýhodou je také typ zprávy, který neumožňuje přenášet příliš mnoho paketů najednou. Podle výsledků této práce je tento kanál považován za nejhorší skrytý kanál jako skrytý přenos informací.

6.6 neighbor_discovery_redirect.py

```
┌──(root㉿kali)-[/home/kali/Desktop/python_app_covert_channel]
└─# python covert.py -i eth0 -sip 2001:db8:aacc:0:e4f6:91ee:bb0e:7db6 -dip 2001:db8:bbcc:0:cb64:306c:e05d:53e7 -smac 00:0c:29:74:69:11 -dmac
00:0c:29:8a:7a:1a -m "qwer" -c ndr
Packet Sent!
```

Obr. 6.16: Příklad odesílání paketů se skrytým kanálem Neighbor discovery redirect.

Podle obr. 6.16 byl vytvořen skrytý kanál v protokolu ICMPv6 pomocí zprávy Neighbor discovery redirect. Zde se také používá rezervované pole o velikosti 4 bajty. V tomto případě přeloží zapsanou zprávu do bajtů, pak ji v big-endian pořadí převede na jedno celé číslo a zapíše ji do pole Reserved.

```
No.  Time          Source                  Destination             Protocol  Length  Info
   1 0.000000000   2001:db8:aacc:0:e4f…    2001:db8:bbcc:0:cb6…    ICMPv6      94    Redirect

▸ Frame 1: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface eth0, id 0
▸ Ethernet II, Src: VMware_3e:e1:6d (00:0c:29:3e:e1:6d), Dst: VMware_8b:bf:37 (00:0c:29:8b:bf:37)
▸ Internet Protocol Version 6, Src: 2001:db8:aacc:0:e4f6:91ee:bb0e:7db6, Dst: 2001:db8:bbcc:0:cb64:306c:e05d:53e7
▾ Internet Control Message Protocol v6
    Type: Redirect (137)
    Code: 0
    Checksum: 0x5c1a [correct]
    [Checksum Status: Good]
    Reserved: 71776572
    Target Address: 2001:db8:aacc:0:e4f6:91ee:bb0e:7db6
    Destination Address: 2001:db8:bbcc:0:cb64:306c:e05d:53e7

0000  00 0c 29 8b bf 37 00 0c  29 3e e1 6d 86 dd 60 00   ··)··7·· )>·m··`·
0010  00 00 00 28 3a 7e 20 01  0d b8 aa cc 00 00 e4 f6   ···(:~ · ********
0020  91 ee bb 0e 7d b6 20 01  0d b8 bb cc 00 00 cb 64   ····}· · *******d
0030  30 6c e0 5d 53 e7 89 00  5c 1a 71 77 65 72 20 01   0l·]S··· \·qwer ·
0040  0d b8 aa cc 00 00 e4 f6  91 ee bb 0e 7d b6 20 01   ******** ****}··
0050  0d b8 bb cc 00 00 cb 64  30 6c e0 5d 53 e7          *******d 0l·]S·
```

Obr. 6.17: Příklad zachyceného paketů se skrytým kanálem Neighbor discovery redirect v programu Wireshark.

```
┌──(root㉿kali)-[/home/kali/Desktop/python_app_decoder]
└─# python decoder.py -i eth0 -t 10 -c ndr
Decoded message: qwer
Packet(s) decoded!
```

Obr. 6.18: Příklad zachycených paketů s dekódovanou zprávou v Neighbor discovery redirect

Dešifrování proběhlo úspěšně podle obr. 6.18 a získali tak požadovanou zprávu. Princip je stejný, pouze v obráceném pořadí. Je zde také kontrola nul, aby se zabránilo jejich vypsání spolu se zprávou.

Oproti skrytému kanálu Destination Unreachable je možné do pole vložit všechny 4 bajty informací. Jak je uvedeno v 6.4, záleží na zpracování paketů Scapy. V tomto případě s tím nebyly žádné problémy.

Nejprve byl vytvořen tento skrytý kanál, který pracoval podle stejného schématu šifrování skryté zprávy jako Destination Unreachable. Po podrobném prostudování knihovny Scapy se však zjistilo, že zde vytvoření a odeslání paketu vypadá jinak, protože pole Reserved může obsahovat pouze 3 bajty. Z toho vyplývá, že i když dva různé skryté kanály s technikou šifrování zprávy vypadají stejně, neznamená to, že v procesu komunikace bude vše stejné. Proto je nutné otestovat všechny možné varianty skrytých kanálů v různých oblastech, i když metoda vytvoření skrytého kanálu vypadá stejně.

Závěr

V teoretické části této práce byla analyzována problematika IPv6 a ICMPv6. Byly analyzovány rozšířené hlavičky paketů IPv6 a jejich možné využití pro skryté kanály. V kapitole 2 bylo z teoretického hlediska vysvětleno, jak mohou tyto skryté kanály vypadat. Byl zde také představen scénář v kapitole 4, který ukazuje celý proces od odeslání až po příjem paketu se skrytým kanálem. Byla popsána struktura aplikace v kapitole 5, a to jak na straně odesílatele, tak na straně příjemce. Implementace této práce je popsána v poslední kapitole, která zohledňuje jak odesílání a vytváření paketů, tak i jejich dešifrování. Byly napsány dvě aplikace, jedna pro vytvoření skrytého kanálu a jeho implementaci v protokolech IPv6/ICMPv6, druhá pro dešifrování odeslané zprávy. K tomuto účelu byl použit programovací jazyk Python s hlavní knihovnou této práce Scapy. Byla provedena analýza fungování skrytých kanálů, která ukazuje, že k dnešnímu dni stále existuje mnoho různých možností vytváření skrytých kanálů, zejména v málo používaných polích protokolů IPv6/ICMPv6 a také v málo používaných rozšiřujících hlavičkách protokolu IPv6. Tyto programy mohou být v budoucnu aktualizovány v závislosti na situaci.

Literatura

[1] WIKIPEDIA CONTRIBUTORS. *IPv6 Wikipedia, The Free Encyclopedia.* Online. 2023. Dostupné z: `https://en.wikipedia.org/wiki/IPv6`. [cit. 2023-10-23].

[2] JEŘÁBEK, J. *Pokročilé komunikační techniky.* Skripta. Brno: FEKT VUT v Brně, 2023, s.101–137.

[3] GUPTA, M., CONTA, A. *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification.* Online. Datatracker, 2006. Dostupné z: `https://datatracker.ietf.org/doc/html/rfc4443#section-2`. [cit. 2023-10-24].

[4] HINDEN, B., DEERING, S. *Internet Protocol, Version 6 (IPv6) Specification.* Online. Datatracker, 1998. Dostupné z: `https://datatracker.ietf.org/doc/html/rfc2460#section-4`. [cit. 2023-10-24].

[5] JUNIPER NETWORKS, INC. Online. *IPv6 Flow-Based Processing Overview.* USA: Sunnyvale, California, 2023. Dostupné z: `https://rb.gy/ylcs3b`. [cit. 2023-10-25].

[6] LUCENA, N.B., LEWANDOWSKI, G., CHAPIN, S.J. 2006. Online. *Covert Channels in IPv6.* In: DANEZIS, G., MARTIN, D. (eds) Privacy Enhancing Technologies. PET 2005. Lecture Notes in Computer Science, vol 3856. Springer, Berlin, Heidelberg. Dostupné z: `https://doi.org/10.1007/11767831_10`. [cit. 2023-10-25].

[7] HINDEN, B., DEERING, S. *Request for Comments.* Online. Datatracker, 1998. Dostupné z: `https://doi.org/10.17487/RFC2460`. [cit. 2023-10-27].

[8] PHAN, VIET ANH. *Generování IPv6 a ICMPv6 paketů a jejich vliv na fungování zařízení v síti.* Online. Diplomová práce, vedoucí Jan Jeřábek. Brno: FEKT VUT v Brně, 2023. Dostupné z: `http://hdl.handle.net/11012/210094`. [cit. 2023-10-27].

[9] J. WANG, L. ZHANG, Z. LI, Y. GUO, L. CHENG and W. DU. Online. *CC-Guard: An IPv6 Covert Channel Detection Method Based on Field Matching*, 2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application, Hainan, China, 2022, s. 1416–1421. Dostupné z: `https://doi.org/10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00219`. [cit. 2023-10-30].

[10] M. MAVANI and L. RAGHA. Online. *Covert channel in IPv6 Destination option Extension header*, 2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA), Mumbai, India, 2014, s. 219–224. Dostupné z: `https://doi.org/10.1109/CSCITA.2014.6839262`. [cit. 2023-11-2].

[11] HAIPENG QU, PURUI SU & DENGGUO FENG. Online. *A typical noisy covert channel in the IP protocol*, 38th Annual 2004 International Carnahan Conference on Security Technology, 2004., Albuquerque, NM, USA, 2004, s. 189–192. Dostupné z: `https://doi.org/10.1109/CCST.2004.1405390`. [cit. 2023-11-3].

[12] SHANE A., RAJAHALME J., BRIAN E. CARPENTER and SHENG JIANG. Online. *IPv6 Flow Label Specification*. Datatracker, 2011. Dostupné z: `https://datatracker.ietf.org/doc/html/rfc6437#section-2`. [cit. 2023-11-3].

[13] SHANE A., RAJAHALME J., BRIAN E. CARPENTER and SHENG JIANG. Online. *Covert Channel Risk*. Datatracker, 2011. Dostupné z: `https://datatracker.ietf.org/doc/html/rfc6437#section-6.1`. [cit. 2023-11-3].

[14] L. CAVIGLIONE, M. ZUPPELLI, W. MAZURCZYK, A. SCHAFFHAUSER & M. REPETTO. Online. *Code Augmentation for Detecting Covert Channels Targeting the IPv6 Flow Label*, 2021 IEEE 7th International Conference on Network Softwarization (NetSoft), Tokyo, Japan, 2021, s. 450–456. Dostupné z: `https://doi.org/10.1109/NetSoft51509.2021.9492661`. [cit. 2023-11-5].

[15] J. BERGER, A. KLEIN & B. PINKAS. Online. *Flaw Label: Exploiting IPv6 Flow Label*, 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2020, s. 1259–1276. Dostupné z: `https://doi.org/10.1109/SP40000.2020.00075`. [cit. 2023-11-6].

[16] *Internet Protocol Version 6 Parameters*. Webové sídlo. Online. Dostupné z: `https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xml`. [cit. 2023-11-8].

[17] ELWYN B. DAVIES and JÁNOS MOHÁCSI. Online. *Recommendations for Filtering ICMPv6 Messages in Firewalls*. Datatracker, 2007. Dostupné z: `https://datatracker.ietf.org/doc/html/rfc4890#appendix-A.1`. [cit. 2023-11-9].

[18] ELWYN B. DAVIES and JÁNOS MOHÁCSI. Online. *Time Exceeded Error Message*. Datatracker, 2007. Dostupné z: `https://datatracker.ietf.org/doc/html/rfc4890#appendix-A.3`. [cit. 2023-10-9].

[19] *KALI LINUX. The most advanced Penetration Testing Distribution.* Online. 2023. Dostupné z: `https://www.kali.org`. [cit. 2023-11-11].

[20] *ABOUT WIRESHARK. About Wireshark.* Online. 2023. Dostupné z: `https://www.wireshark.org/about.html`. [cit. 2023-11-11].

[21] *PYTHON.* Online. 2023. Dostupné z: `https://www.python.org`. [cit. 2023-11-11].

[22] *SURICATA IDS. Observe. Protect. Adapt.* Online. 2023. Dostupné z: `https://suricata.io`. [cit. 2023-11-11].

[23] *SCAPY.* Online. 2023. Dostupné z: `https://scapy.net`. [cit. 2023-11-11].

Seznam symbolů a zkratek

ASCII American Standard Code for Information Interchange

CSR Cloud Services Router

DOH Destination option Extension Header

DHCPv6 Dynamic Host Configuration Protocol version 6

IANA Internet Assigned Numbers Authority

ICMPv6 Internet Control Message Protocol version 6

ICV Integrity Check Value

IDS Intrusion Detection System

IETF Internet Engineering Task Force

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

IPsec IP security

ISAKMP Internet Security Association and Key Management Protocol

MAC Media Access Control

MTU Maximum transmission unit

OS Operating System

RFC Request for Comments

SLAAC StateLess Address Auto Configuration

SNMPv3 Simple Network Management Protocol version 3

SSL/TLS Secure Sockets Layer/Transport Layer Security

TCP Transmission Control Protocol

TLV Type-Lenght-Value

TTL Time-To-Live

UDP User Datagram Protocol

WEP Wired Equivalent Privacy

WPA Wi-Fi Protected Access

7 Obsah elektronické přílohy

Obsah elektronické přílohy, která byla odeslána prostřednictvím informačního systému VUT. Existuje jak složka **python_app_covert_channel** pro odesílatele paketu, tak složka **python_app_decoder** pro příjemce paketu.

Systém Suricata IDS stačí stáhnout do virtuálního počítače nebo počítače.

Vše bylo testováno na Pythonu verze 3.11.4, Suricara verze 7.0.2, Kali Linux verze 2023.3, Cisco CSR1000V Software 15.4(1)S2, Wireshark 4.0.7.

```
covert_channel_app ........................ kořenový adresář přiloženého archivu
├── python_app_covert_channel .... složka s aplikací pro vytvoření skrytého kanálu
│   ├── covert.py ............................................ spouštěcí soubor
│   ├── destination_options_header.py
│   ├── destination_unreachable.py
│   ├── flow_label.py
│   ├── hop_limit.py
│   ├── neighbor_discovery_redirect.py
│   ├── time_exceeded.py
│   └── README.md ........................... krátké uvedení pro spuštění programu
├── python_app_decoder ........... složka s aplikací pro dešifrování skrytého kanálu
│   ├── decoder.py ........................................... spouštěcí soubor
│   ├── destination_options_header_decoder.py
│   ├── destination_unreachable_decoder.py
│   ├── flow_label_decoder.py
│   ├── hop_limit_decoder.py
│   ├── neighbor_dicovery_redirect_decoder.py
│   ├── time_exceeded_decoder.py
│   └── README.md ........................... krátké uvedení pro spuštění programu
└── routers ........................................ configurace routeru soubory
    ├── R1 ............................................ configurace routeru R1
    │   ├── R1_1.png
    │   ├── R1_2.png
    │   ├── R1_3.png
    │   ├── R1_4.png
    │   └── R1_5.png
    └── R2 ............................................ configurace routeru R2
        ├── R2_1.png
        ├── R2_2.png
        ├── R2_3.png
        ├── R2_4.png
        └── R2_5.png
```