VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zvýšení odolnosti SIP Proxy proti DoS útokům

Improvements of SIP Proxy Robustness Against DoS Attacks

2011

Bc. Jakub Šafařík

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

V Ostravě 3. května 2011 .....

Rád bych poděkoval svému vedoucímu diplomové práce, panu doc. Ing. Miroslavu Vozňákovi, za jeho pomoc, vstřícný přístup a trpělivost při tvorbě mé práce.

Abstrakt

Nárůst popularity VoIP v posledních letech nezbytně vedl i k zájmu hackerů o tuto novou platformu. Jedním z nejvíce užívaných útoků je DoS, zejména díky jednoduchosti a vysokému dopadu na danou službu. Diplomová práce popisuje nejen vlastnosti SIP protokolu, ale i bezpečnostní hrozby týkající se VoIP řešení. Tyto znalosti slouží k testování odolnosti SIP proxy, stejně tak i pro následné bezpečnostní opatření. Každý útok je detailně popsán i s návrhem obrany. Cestou k zvýšení odolnosti je nasazení IPS systému na bázi aplikací Snort, SnortSam a IPtables. Práce obsahuje také popis dalších bezpečnostních kroků, zvyšující bezpečnost v síti i vlastní SIP proxy.

Klíčová slova: SIP, bezpečnost, Snort, SnortSam, IPtables, DoS, IPS, obrana proti DoS

Abstract

Increasing rate of popularity of VoIP solution in last few years lead to hackers interest for this new platform. One of the most used attack nowadays is DoS, because of its simplicity and big impact. This diploma thesis describes features of SIP protocol and main security threats in VoIP. These knowledge are used for testing robustness of SIP proxy server. Attacks are described in detail, there is made a security precaution for each of them. Way, how to defend SIP proxy against attacks is creating an IPS system, composed as combination Snort, SnortSam and IPtables applications. Part of thesis is also proposal for security steps, increasing inside network safety and SIP proxy security.

Keywords: SIP, security, Snort, SnortSam, IPtables, DoS, IPS, DoS defense

Seznam použitých zkratek a symbolů

3DES	–	Triple DES - varianta DES
AES	–	Advanced Encryption Standard
AH	–	Authentication header
AP	–	Access point
CoS	–	Class of Service
CPU	–	Central Processing Unit
DAQ	–	Data acquisition API
DDOS	–	Distributed denial of service
DES	–	Data Encryption Standard
DNS	–	Domain Name System
DOS	–	Denial of service
ESP	–	Encapsulating Security Payload
IDS	–	Intrusion Detection System
IETF	–	Internet Engineering Task Force
IP	–	Internet protocol
IPS	–	Intrusion Prevention System
IPsec	–	IP security
IVR	–	Interactive voice response
LDAP	–	Lightweight Directory Access Protocol
MITM	–	Man in the middle
NAT	–	Network address translation
P2P	–	peer to peer
QoS	–	Quality of service
RFC	–	Request for Comments
RTCP	–	RTP Control Protocol
RTP	–	Real-time Transport Protocol
SAML	–	Security Assertion Markup Language
SDP	–	Session Description Protocol
SIP	–	Session Initiation Protocol
SIPS	–	Označuje zabezpečenou verzi SIP protokolu
SNMP	–	Simple Network Management Protocol
SPIT	–	Spam over Internet Telephony
SRTCP	–	Secure RTP Control Protocol

SRTP	–	Secure Real-time Transport Protocol
SSI	–	označuje kombinaci aplikací: Snort, SnortSam, Iptables
SSL	–	Secure Sockets Layer
TLS	–	Transport Layer Security
TTL	–	time to live
UA	–	user agent
URI	–	Uniform Resource Identifier
URL	–	Uniform Resource Locator
VLAN	–	Virtual Local Area Network
VoIP	–	Voice over internet protocol
WiFi	–	Wireless fidelity
ZRTP	–	rozšíření SRTP funkcionality

Obsah

Seznam obrázků

1 Úvod

Technologie VoIP zažívají v současné době svá zlatá léta. Dochází k jejich hromadnému nasazování, existuje nepřeberné množství dostupných variant řešení i protokolů zprostředkovávajících VoIP služby. Doba, kdy dojde k úplnému nahrazení za stávající PSTN sítě není daleko. Zvyšuje se také tlak vývoje na integraci telefonních služeb (a nejen těchto) do řešení tzv. sítí nové generace, kde přenášená data nebudou závislá na transportní technologii. Uživatelům tak mohou poskytovatelé nabízet služby, které lze v původním PSTN řešení pouze obtížně nasazovat, případně tyto služby nelze zavádět vůbec.

S rostoucí popularitou vzniklo mnoho řešení, mezi protokoly dochází k značnému upřednostňování protokolu SIP, v současnosti snad nejpoužívanějšímu protokolu pro VoIP služby. Tento růst má za následek mnoho různých implementací, dostupné jsou samozřejmě i open–source řešení. Během vývoje však po dlouho dobu nebyla bezpečnost a zabezpečení hlavním cílem, mnohdy byly odsouvány na druhou kolej. Vývoj se soustředil především na rozšiřování stávající funkcionality, přinášení nových služeb.

Vysoká popularita ale přitahuje pozornost i potencionálních útočníků a potřeba zabezpečení VoIP služeb začala být velmi aktuální. Útočníci měli také značně zjednodušenou situaci díky otevřenosti protokolu, značné podobnosti s *STMP* a *HTTP* protokoly, jenž přinesla většinu zranitelností těchto protokolů i do internetové telefonie. Na druhou stranu však rychle vzniklo mnoho řešení, jak dané problémy vyřešit.

Z hlediska oblíbenosti typů útoků mezi hackery vede zajisté DoS – útok na zamezení přístupu k službě (ať už úplné zamezení či jen částečné). Na špičce druhů útoků se nedrží náhodou, toto postavení způsobuje především jejich vysoká efektivita kombinovaná s relativně snadnou cestou provedení těchto útoků.

Diplomová práce shrnuje vlastnosti protokolu SIP se zaměřením na metody zvýšení odolnosti vůči DoS útokům. Po stručném popisu vlastností protokolu SIP a popsání principu jeho funkce přecházíme ke shrnutí bezpečnostních rizik týkajících se současných VoIP řešení. Popsáno je také možné dělení DoS útoků i s vlastnostmi různých variant tohoto útoku. Díky znalostem principu funkce protokolu SIP a DoS útoků, dojde k otestování odolnosti SIP proxy vůči těmto zranitelnostem. Každý útok obsahuje zhodnocení efektivnosti, dopad na SIP proxy server i návrh vhodného způsobu obrany. Důkladně popsáno je nasazení bezpečnostních opatření, nemělo by být tedy problémem použít těchto kroků pro přenesení obraného systému k jiným řešením. Závěrečné shrnutí se ohlíží po provedených útocích a navrhuje další možné kroky pro vytvoření bezpečnější síťové topologie a méně zranitelného řešení VoIP.

2 SIP protokol, jeho vlastnosti a využití

Protokol SIP, vyvíjený od roku 1996 v rámci IETF, je řídícím protokolem zajišťujícím vytváření, dohled i ukončení telefonních hovorů na bázi VoIP. Specifikován byl standardem RFC 3261 v roce 2002. Od této doby vzniká řada dalších RFC, přímo se týkajících SIP protokolu či rozšířením jeho vlastností.

2.1 Vlastnosti SIP protokolu

Jak bylo zmíněno, SIP protokol slouží pro řízení hovorů. Pracuje na aplikační vrstvě, přičemž byl při jeho vývoji kladen důraz zvláště na snadnou rozšiřitelnost, implementaci a také flexibilitu. V kombinaci se SIPem jsou používány také další protokoly – SDP a RTP.

SDP protokol využíváme pro popis vlastností účastníků komunikace, k vyjednávání parametrů spojení mezi všemi účastníky komunikace (popsán v RFC 2327). RTP pak slouží k přenosu multimediálních dat v reálném čase. Na transportní vrstvě pak k přenosu využívá protokol UDP.

SIP vychází z HTTP protokolu, je tedy textově orientovaný. Průběh komunikace klientů se serverem probíhá formou požadavků a odpovědí (HTTP), v hlavičce paketu se pak nalézají pole *From*, *To* a *Subject*, obdobně jako u e-mailové komunikace protokolem SMTP.

Koncová zařízení znají jednotlivé stavy komunikace, díky čemuž bývá protokol označován také jako signalizační protokol typu end–to–end. Díky tomu se zvýší odolnost proti chybám, na druhou stranu ale stoupá i režie spojená s hlavičkami jednotlivých SIP zpráv. Tímto se SIP výrazně odlišuje od původní PSTN sítě, která je založena na opačném modelu – tam je logika řízení v síti a koncová zařízení mohou být značně jednoduchá, primitivní. SIP tak nabízí vyšší výkonnost a nové typy služeb, které v PSTN nemohou být nasazeny vůbec či s velkými obtížemi.

Klienti používající SIP protokol se nachází uvnitř domén, každá doména je vázána ke svojí SIP proxy. Komunikace probíhající mezi doménami probíhají mezi různými SIP proxy servery (pokud není použita tzv. multidoménová SIP proxy), komunikace v rámci vlastní domény pak pouze s použitím výchozí SIP proxy.

K identifikaci jednotlivých klientů – SIP entit, slouží *SIP URI*.

```
sip:user:password@host:port;uri-parameters?headers
```

User část identifikuje uživatele, host se vztahuje k určité doméně (hostiteli), poskytující prostředky k zajištění komunikace. Část pro heslo *password* nemusí být použita, ze zjevných důvodů není ani doporučováno její použití. Standardní port pro SIP server je 5060, použitý protokol pak UDP. Případné parametry musíme oddělovat pomocí středníku. Většinou však bývá použita pouze kratší varianta URI.

```
sip:user@host
```

Aby při zpracovávání SIP zpráv nedocházelo ke vzniku smyček, používá SIP dva mechanismy. Stateful proxy pomocí atributu *branch* určují příslušnost zprávy k transakci a může tak následně ovlivňovat další zprávy. Druhou metodou je dekrementace hodnoty u parametru *Max-Forwards*, fungujících na podobném principu jako hodnota *TTL* v protokolu IP.

2.2 Prvky SIP topologie

Jednotlivé subjekty komunikující prostřednictvím SIP protokolu lze rozdělit na dvě skupiny:

- koncová zařízení (označovaná také UA, klient)
- SIP servery – proxy, register, redirect, ...

Ačkoliv je možné vytvořit síť pouze prostřednictvím dvou navzájem propojených klientů, typicky síť obsahuje i další zařízení – servery. Ty zajišťují logické operace v rámci celé sítě, obvyklá je také konsolidace těchto serverů do společného řešení. Zástupcem klienta může být hardwarový SIP telefon, softwarová aplikace (pro PC, mobilní zařízení), PSTN brána (gateway), IVR systém, ...

2.2.1 Druhy SIP serverů

Kostrou celé infrastruktury SIP topologie jsou SIP proxy servery. Klienti zasílají na proxy servery žádosti o spojení, server tyto žádosti vyhodnotí. Provádí tedy nejen směrování těchto žádostí (podle umístění klienta), ale také může provádět autentizaci, účtování či realizování dalších doplňkových služeb.

Při vytváření nového spojení je postup následující. Pokud SIP proxy nezná umístění daného koncového zařízení, prohledává další SIP proxy servery, dokud nenarazí na takový, který má informace o umístění daného zařízení. Následně přesměruje požadavek na cílového klienta, který buď hovor přijme nebo odmítne.

Dalšími typy SIP serverů jsou:

- Redirect server – pro přesměrovávání, navrací nové hodnoty SIP URI. V případě požadavku prohledá lokalizační databázi vytvářenou registar serverem. Seznam aktuálního umístění klienta zasílá formou odpovědi třídy 3xx, tedy přesměrování (viz. příloha B). Hovor tak může být spojen i pokud se cílový klient nachází v dané chvíli v cizí doméně.
- Registar server – obsluhující požadavky na registraci. Zprostředkovává mapování fyzických zařízení na logické URI adresy klientů, aktualizuje lokalizační databázi. Každý z klientů musí být registrován na určitém SIP registar serveru. Registrace klienta probíhá pouze na serverem určenou dobu (uvedena v hlavičce kontaktu). Když klient do vypršení doby svou registraci neobnoví, bude považován za nedostupné koncové zařízení.
- Location server – uchovává informace o umístění uživatelů a SIP proxy serverů. Tyto informace následně využívá SIP proxy pro spojování hovorů.
- B2BUA server – zvláštní druh serveru, ukončující stávající spojení a sestavující nové spojení na cíl. Pro klienty se chová obdobně jako SIP proxy, funkcionalita je ale značně omezena.

2.3 Módy SIP proxy serveru

U SIP proxy serverů rozlišujeme dva módy:

stateless neboli bezestavový

stateful obsahující informace o stavech

2.3.1 Stateless SIP proxy

Jedná se o jednoduché řešení přeposílající SIP zprávy bez ohledu na jejich vzájemné vazby. Nedokáží například kontrolovat zprávy z hlediska smysluplnosti, spoléhají na správnost významu i sledu zpráv. Bezestavové proxy servery také nezachytí replikaci zpráv, detekce smyček trvá mnohem déle než v případě stavových serverů. Vzhledem k stavovým serverům však nabízí vyšší rychlost, používají se tedy jako balanční servery, pro směrování nebo jednoduché překládání zpráv.

2.3.2 Stateful SIP proxy

Komplexní řešení zpracování SIP zpráv poskytují právě stavové proxy servery. Každá přijatá zpráva způsobí vytvoření údaje o stavu, v němž jsou drženy mnohé důležité informace. Délka, po kterou se informace uchovávají v paměti závisí na typu požadavku.

- transakční – udržují stav vzhledem k žádosti (např. vyřízení *INVITE* žádosti)
- dialogové – informace jsou uchovávány po celou dobu trvání dialogu (tj. celé spojení)

Právě díky transakcím dokáže server monitorovat průběh zpráv – disponuje tak možnostmi větvení, zaslání zpráv více klientům, detekcí opakovaného zaslání požadavku, používaní komplexních metod pro nalezení klienta (postupné vyzvánění na více koncových zařízeních), . . . V případě bezestavového serveru dojde k odeslání odpovědi a server nadále nemůže sledovat důsledky dané akce.

Většina SIP proxy serverů používá možností stateful serverů. Bohužel právě díky udržování informací o stavu a analyzování jednotlivých zpráv vzniká prostor pro specifické útoky.

SIP proxy servery byly vytvořeny také s ideou vzájemné spolupráce více proxy serverů vzájemně, respektive aby bylo možné komunikovat s koncovými zařízeními v různých doménách. Pokud je odeslán požadavek na spojení do cizí domény, pokusí se server prostřednictvím DNS serveru zjistit umístění SIP proxy pro zvolenou doménu. V případě, že doména existuje, zašle požadavek na spojení SIP serveru v odpovídající doméně. Cílový proxy server vyhledá daného uživatele a vzhledem k výsledku hovor spojí (tj. přepošle zprávu klientovi) nebo hovor odmítne.

2.4 SIP metody

Zprávy protokolu SIP probíhají formou požadavku a odpovědi, liší se však na několik metod sloužících různým účelům. Při vydání RFC 3261 existovaly pouze základní metody.

INVITE – žádost o vytvoření spojení, případně pak o změnu parametrů již probíhajícího hovoru

ACK – metoda potvrzující přijetí konečné odpovědi na zprávu *INVITE*. Vzhledem k využívání protokolu UDP na transportní vrstvě představuje tzv. 3–way handshaking.

BYE – zpráva použitá pro ukončení již probíhajícího spojení.

CANCEL – má stejný výsledek jako předchozí metoda, používá se však v případě, kdy není vytvořeno spojení. Volaný tedy ještě nepotvrdil zprávu *INVITE* konečnou odpovědí, dialog není sestaven.

REGISTER – slouží k zaregistrování (i odregistrování) klienta. Hodnoty, podle nichž je vytvořen záznam, obsahují pole *From* a *Contact* SIP hlavičky.

OPTIONS – speciální typ metody určené k detekování vlastností koncových zařízení. Strukturou se shoduje s *INVITE* zprávou, spojení však není vytvářeno. Využívána bývá pro zjišťování podporovaných funkcí, pro udržení záznamu při průchodu zpráv přes NAT nebo ověření dostupnosti zařízení před expirací registrace.

K těmto základním metodám byly následnými RFC přidány metody níže uvedené.

INFO – přenáší informace v průběhu vytvořeného hovoru (RFC 2976)

UPDATE – aktualizuje parametry spojení klienta (RFC 3311)

PRACK – dočasné potvrzení odpovědí z třídy 1xx (RFC 32622)

SUBSCRIBE – přihlášení k odběru informací o událostech (RFC 3265)

NOTIFY – k doručení odebíraných informací (RFC 3265)

MESSAGE – přináší podporu IM, přenáší text zprávy (RFC 3428)

REFER – oznamuje požadavek jiného klienta k relaci (RFC 3515)

PUBLISH – aktualizace stavu (prezence) serveru (RFC 3903)

Odpovědmi zasílanými na tento server jsou zprávy obsahující návratové kódy, obdobně jako u protkolu *HTTP*. Návratové kódy dělíme do šesti tříd dle typu odpovědi, více v příloze B.

1xx – informativní odpovědi, pokračuje zpracování SIP zprávy

2xx – konečná odpověď, pouze úspěšná oznámení

3xx – odpověď o přesměrování

4xx – negativní odpověď, došlo k chybě na straně odesílatele

5xx – došlo k problému na serveru, záporná odpověď

6xx – finální odpověď pro případ, kdy požadavek neakceptuje žádný server.

2.5 Příklady spojení

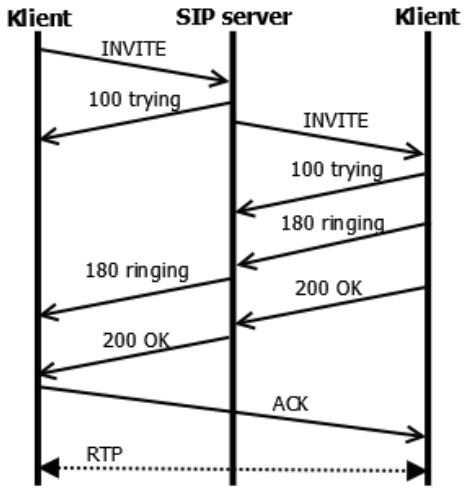
Vzhledem k útokům na SIP server popsaných v dalších kapitolách popíšeme základní průběh registrace klienta a výměnu zpráv při zahájení hovoru.

2.5.1 Registrace klienta

Průběh registrace odpovídá stavu popsanému výše. Doplňme jen, že v případě odhlášení registrace se v SIP hlavičce použije hodnota *expires* nastavená na hodnotu 0.

```
REGISTER sip:192.168.0.10 SIP/2.0
Via: SIP/2.0/UDP 192.168.0.3:15068;branch=z9hG4bK-d8754z-
    c36f4e576dcd2fee-1---d8754z-;rport
Max-Forwards: 70
Contact: <sip:1001@192.168.0.3:15068;rinstance=e7238becce042b76>
To: "1001"<sip:1001@192.168.0.10>
From: "1001"<sip:1001@192.168.0.10>;tag=f4c01cef
Call-ID: NDE5NGI0NTY0ZDk3NDNjNzg0ZGVlN2YyMjNmZmIxODk.
CSeq: 1 REGISTER
Expires: 3600
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY,
    MESSAGE, SUBSCRIBE, INFO
User-Agent: X-Lite 4 release 4.0 stamp 58832
Content-Length: 0

SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP 192.168.0.3:15068;branch=z9hG4bK-d8754z-
    c36f4e576dcd2fee-1---d8754z-;received=192.168.0.3;rport=15068
From: "1001"<sip:1001@192.168.0.10>;tag=f4c01cef
To: "1001"<sip:1001@192.168.0.10>;tag=as0bd246c3
Call-ID: NDE5NGI0NTY0ZDk3NDNjNzg0ZGVlN2YyMjNmZmIxODk.
CSeq: 1 REGISTER
Server: Asterisk PBX 1.6.2.5-0ubuntu1.3
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE,
    NOTIFY, INFO
Supported: replaces, timer
WWW-Authenticate: Digest algorithm=MD5, realm="asterisk",
```



Obrázek 1: Průběh sestavení hovoru

```
    nonce="5935b1a9"
Content-Length: 0
```

Při využití autentizace probíhá výměna zpráv obdobně jako na uvedeném příkladu. Vypsány jsou pouze hlavičky SIP zpráv *REGISTER* a odpovědi *401 Unauthorized*. Metody používáné k zabezpečení popisuje detailně kapitola o bezpečnostních rizicích, konkrétně pak část 3.1.1. Zpráva s odpovědí *401* obsahuje informace *realm* a *nonce* pro hashovací funkci klienta, který následně opovídá požadovaným způsobem – zasláním *REGISTER* zprávy s autentizačními údaji.

2.5.2 Spojení hovoru

Průběh výměny zpráv mezi jednotlivými účastníky komunikace znázorňuje obrázek 1. Komunikace probíhá v rámci domény, tj. pouze přes výchozí SIP proxy server. Z hlediska stateful proxy serveru představuje posloupnost zpráv, od prvotní *INVITE* až po přijetí *200 OK* klientem, jednu transakci. Během této doby tedy uchovává podstatné informace o vytvářeném spojení.

3 Bezpečnostní rizika a klasifikace DoS útoků

S rostoucí popularitou jednotlivých aplikací a zařízení na internetu stoupá i počet útoků vůči nim. Vzhledem k rostoucí popularitě VoIP řešení lze tedy očekávat, že se stane i častějším cílem útočníků. VoIP navíc přináší změny oproti původní spojově–orientované telefonii.

Neexistuje například žádná nadřazená entita kontrolující vývoj či nasazení VoIP řešení. V dnešní době není problém vybrat si z širokého pásma dostupných řešení a protokolů. Z bezpečnostního hlediska však tento stav danou situaci ještě více zhoršuje. Jednoduše dostupná koncová řešení, která nekladou na dodavatele vysoké nároky z hlediska znalostí, mohou vést ke špatně konfigurovaným řešením s řadou bezpečnostních chyb. Pokud se nadále tyto aplikace neudržují a neaktualizují, jedná se o vážnou bezpečnostní hrozbu.

Příkladem je i případ, který se odehrál v České republice během roku 2009. Majitelka malého hotelu se rozhodla pro nasazení VoIP řešení v rámci úspor. Úspory se sice dostavily, ale ne na dlouho. Neaktualizovaný Asterisk s bezpečnostní mezerou nalezl hacker, který během tří dnů způsobil škodu 1,4 miliónu Kč, pomocí volání na lichtenštejnské číslo [17].

Na jednu stranu je tato situace i důsledkem prudkého vývoje v daném odvětví. Při implementaci softwarových ústředen bylo dbáno především na širokou podporu kodeků, vývoj nových funkcionalit a služeb. Bezpečnost byla po dlouho dobu postavena na „vedlejší kolej“.

Masivním nasazením VoIP řešení se navíc dostaneme do stavu, kdy nahradí původní PSTN síť. Obyčejní uživatelé budou očekávat stejné chování i od nové technologie. Případné útoky a bezpečnostní incidenty by tak mohli narušit důvěru a spokojenost s VoIP řešením.

Vzhledem k výše zmíněným faktům se bezpečností začalo zabývat více společností, vznikla řada nástrojů určených právě pro VoIP. Neexistuje však žádné jednoduché řešení, které by pokrylo celou oblast problematiky VoIP.

3.1 Přehled zranitelností VoIP

- DOS – útočník se při tomto útoku snaží o přerušení služby, regulérní uživatelé pak nemohou služby využívat.
- MITM – jedná se o typ útoku , kdy se útočník nachází mezi dvěma komunikujícími subjekty a může komunikaci monitorovat, modifikovat, mazat, vytvářet či měnit zprávy, které si vzájemně vyměňují (tzn. odposlech hovoru, přesměrování hovoru, zasáhnutí do hovoru, . . .).
- Theft of service – označujeme takto stav, kdy útočník získá zdroje, které jsou zpoplatněné, limitované či je k nim omezený přístup.

- Eavesdropping – nejedná se přímo o útok, ale o monitorování sítě, nahlížení do session. Útočník sbírá informace o síti, komunikujích subjektech. Předchází většinou dalším útokům.
- Impersonating – útočník předstírá, že je někdo jiný. Souvisí s ukradením autentizačních údajů.
- DNS poisoning – útočník pozmění nebo podstrkuje falešné dns záznamy, za účelem přerušení či přesměrování komunikace.
- Session disruption – popisujeme takto snahy o narušení session či pokus o injekci falešných údajů. Do této kategorie lze zařadit i útoky na přerušení probíhajícího hovoru pomocí BYE zpráv.
- Replay attack – dochází k ukládání a následnému opětovnému zasílání korektních paketů či zpráv.
- SPIT – tedy rozesílání nevyžádané reklamy prostřednictvím VoIP. Obrana je velice komplikovaná.
- Voice phishing – jedná se o variantu sociální inženýrství, kdy je namísto mailu použit hovor. Jedinou obranou je školení uživatelů a obeznámení s praktikami phishingu.

Tento seznam by mohl být značně delší, zmíněné útoky však patří mezi nejkritičtější z hlediska útoků na VoIP. Díky struktuře SIP zpráv podobné protokolům *http* a *smtp* přebírá VoIP mnoho zranitelností i od těchto svých „předchůdců“, případně lze útoky modifikovat tak, aby zasahovali i VoIP služby. Z pohledu bezpečnosti požadujeme následující základní vlastnosti (de facto se také jedná o obecné cíle útoků).

důvěryhodnost – zajišťuje, že nelze získat informace obsažené uvnitř zprávy během přenosu mezi odesilatelem a příjemcem.

integrita – poslaná zpráva nebyla během přenosu nijak modifikována. Ověření probíhá na straně příjemce zprávy.

autentičnost – identita jednoho nebo obou účastníků je ověřena tak, aby nedošlo k přetvařování identity.

dostupnost – obecně garantuje dostupnost dané služby, tedy možnost ji využívat.

Samozřejmě existují ještě další formy rozdělení. Musíme si také uvědomit, že pokud budeme diskutovat o bezpečnosti VoIP, nesmíme zapomenout i na bezpečnostní rizika spojená s použitým operačním systémem, stějne tak jako aplikacemi a službami na něm běžícími. Právě komplexita VoIP vytváří značné množství zranitelností. SIP protokol nabízí sám dva druhy zabezpečení, a to signalizace a hovoru.

3.1.1 Zabezpečení signalizace SIP

Díky tomu, že struktura SIP zpráv vychází z modelu *HTTP*, lze všechny bezpečnostní mechanismy dostupné pro *HTTP* aplikovat i pro SIP.

3.1.1.1 HTTP Basic Authentication Základní autentizace zajišťující autentizaci pouze na základně sdíleného hesla. To je zaslané v otevřeném tvaru. Díky tomu může být snadno zachyceno a proto tedy tato metoda představuje i vážné bezpečnostní riziko. Použití v SIP verzi 2 není schváleno.

3.1.1.2 HTTP Digest Authentication Vylepšením předchozí metody se stává přístup používající hashů (MD5 i SHA–1). Hashovány jsou heslo a náhodný řetězec (pro případ použití slabého hesla). I přes tato vylepšení však nelze tuto metodu označovat za bezpečnou. Použitím nebezpečného MD5 hashe či slovníkovým útokům na hashe lze komunikaci následně narušit či dokonce nalézt původní hesla. Metoda také poskytuje pouze autentizaci stran, nikoliv šifrovací mechanismus nebo integritu.

3.1.1.3 PGP Může být použita pro autentizaci a volitelně i jako šifrování obsahu, ale pro verzi SIP zpráv 2 byla nahrazena metodou S/MIME.

3.1.1.4 Secure MIME Formát *MIME* používaný u e–mailové komunikace k definici obsahu zpráv obsahuje i mechanismy pro zabezpečení integrity a důvěryhodnosti zároveň. Toho je dosaženo prostřednictvím certifikátů (uživatelé identifikováni pomocí SIP URI). Tělo zprávy je šifrováno pomocí symetrické šifry (DES, 3DES, AES). Hlavička šifrována není, zpráva však obsahuje kontrolní hash hlavičky. Důležitá je spolupráce s důvěryhodnou certifikační autoritou, která zajišťuje výměnu veřejných klíčů a potvrzuje správnost bezpečnostních certifikátů.

Díky neexistenci globální autority poskytující certifikáty pro konečná řešení, zranitelnosti vlastnoručně generovaných certifikátů proti MiTM, nedošlo k plnému využití této metody a její použití je pouze výjimečné.

3.1.1.5 SIPS URI – TLS Použitím *SIPS URI* v poli *from* SIP zprávy INVITE označuje požadavek na použití TLS cestou k cíli. Díky možnosti přidávat informace do SIP hlaviček, musí být spojení navázáno mezi jednotlivými účastníky (SIP proxy) celého přenosu na tzv. hop–by–hop bázi.

TLS vychází z protokolu SSL a pro přenos používá transportního protokolu TCP. Nutná je také podpora pro správu certifikátů.

3.1.1.6 IPsec Poslední možností zabezpečení je IPsec pro realizování šifrovaných spojení na síťové vrstvě. Obdobně jako u TLS je spojení vytvářeno na hop–by–hop bázi. Skládá se ze dvou protokolů – AH a ESP, přičemž mohou být použity i oba zároveň. IPsec poskytuje důvěryhodnost, integritu dat i autentičnost přenesených zpráv. Využívat můžeme předsdílených klíčů i certifikátů.

3.1.2 Zabezpečení hovorů SIP

Multimediální streamy jsou ve VoIP přenášeny pomocí protokolu RTP, založeném na UDP. Pro účely informování o kvalitě hovoru, účtování poplatků, atd. mohou být přenášeny i tyto informace prostřednictvím RTCP šifrovaně. Díky vysoké citlivosti na zpoždění a jitter nepřichází v úvahu algoritmy výrazně ovlivňující právě tyto parametry.

3.1.2.1 SRTP Vzniká rozšířením RTP protokolu, cílem je poskytnout RTP i RTCP šifrování obsahu, intergritu a autentizaci. Přenos je šifrován algoritmem AES v režimu CTR, integritu zajišťují hashe SHA-1. K datům se přidává také autentizační hlavička.

3.1.2.2 IPsec Pro bezpečný přenos RTP lze použít, podobně jako u SIP signalizace, i IPsec. Využívá už jednou vyjednané asociace (pro přenos signalizace SIP zpráv), přidává však značné zatížení k provozu. Při použití zvukového kodeku μ–law a 80 vzorcích v RTP paketu dosahuje IPsec zatížení kolem 30–50%.

3.1.2.3 ZRTP Vzniká rozšířením SRTP o mechanismy pro výměnu symetrických klíčů, pro obranu vůči MiTM.

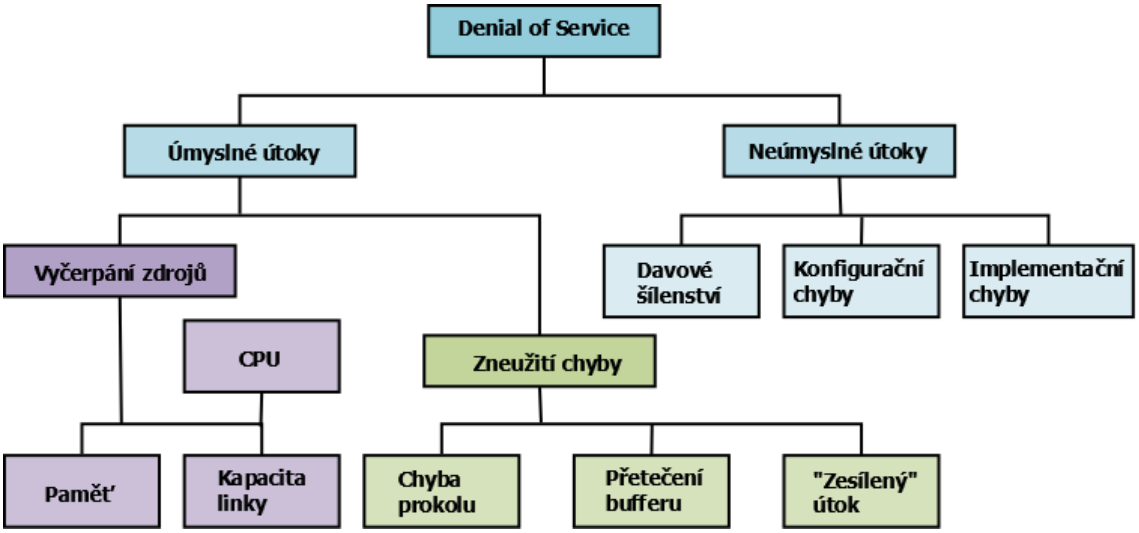
3.2 Klasifikace DoS útoků

Cílem DoS útoků je zamezení užívání služby legitimním uživatelům. Toho lze dosáhnout několika způsoby – zaplavováním serveru poškozenými, modifikovanými či naprosto bezcennými pakety pro vyčerpání zdrojů tohoto stroje. Zasažený server je následně tak vytížen, že není schopen obsluhovat legitimní požadavky.

Bezpečnostní hrozby, zvláště pak typu DoS, se v předchozí generaci PSTN vyskytovaly pouze v minimálním množství. Způsobeno to bylo hlavně uzavřenou síťovou topologií určenou výlučně k účelu přenášení hlasu. S rozvojem VoIP se ale tato situace začala rychle měnit, jak již bylo zmíněno.

Rozlišujeme různé druhy DoS útoků (viz. obrázek 2), které by se daly obecněji rozdělit pouze na tři skupiny.

- Útoky zahlcením – zaměřené především na zdroje serveru, tedy CPU, pamět či kapacitu linky.
- Útoky typu zneužití – útočník použije SIP zprávu upravenou takovým způsobem, aby mohl například ukončit cizí hovor, přesměrovat hovor, nelegálně užívat služeb. Nemusí dojít k úplnému odstřižení služby, blokována může být pouze úzká část uživatelů.
- Nepřímé útoky – pro správnou funkci VoIP je zapotřebí ještě řada dalších služeb (překlad adres, účtování hovorů, ...). Na tyto služby se útočník zaměří za účelem přerušení podporované služby. Může tak dojít k zmatení služby či omezení dostupnosti některých služeb.



Obrázek 2: Klasifikace DoS útoků

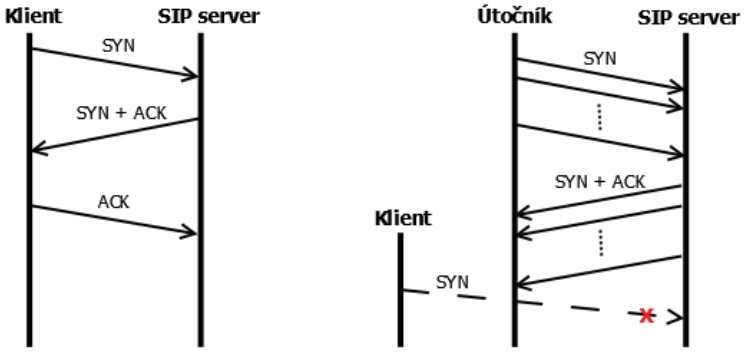
Účinek DoS útoku závisí také na povaze oběti, tedy proti komu je tento útok veden. Zaměřením se na klienta způsobíme, že nebude schopen VoIP využívat. Naproti tomu útokem na SIP proxy můžeme docílit stavu, kdy službu nemůže využít žádný z uživatelů. Útoky na SIP server navíc poškozují i dobré jméno poskytovatele, který tak může přijít o řadu současných i budoucích zákazníků.

V několika posledních letech se také DoS útoky stávají majoritním problémem, s rostoucí frekvencí, dopadem i komplexností. Musíme však rozlišovat mezi úmyslnými a neúmyslnými provedeními. Dále v textu se budeme zabývat pouze úmyslně cílenými útoky, vzhledem k rychlému vývoji VoIP řešení však nejsou neobvyklé ani případy, kdy došlo ke špatné konfiguraci či systém havaroval díky skryté chybě v implementaci serveru. Posledním typem neúmyslného útoku označeného jako „davové šílenství“ popisujeme stav, kdy dojde k přerušení služby jejím nadměrným používáním. Takovéto stavy nastávají při přírodních katastrofách, svátcích či jiném podnětu vyvolávajícím hromadnou potřebu komunikovat v rámci sítě. VoIP server se pak není schopen s náhlým nárůstem zpráv vyrovnat a někteří uživatelé nejsou obslouženi. To vede k dalším pokusům o spojení hovoru a zhoršování situace. Většinou se takový stav rychle vrátí k normálu.

3.2.1 Útoky na paměť serveru

Při zpracovávání SIP zpráv dochází k uložení části informací. Doba po jakou jsou uloženy i množství informací, které jsou uchovávány, závisí na módu, v němž server běží – stavový a bezestavový. Během trvání transakce tyto informace ponechává v paměti, vymazány jsou až s ukončením transakce. Nejčastějším útokem zastupujícím tento typ je *TCP SYN flood* útok.

Regulérní průběh 3–way handshaku, potřebného pro vytvoření TCP session, vidíte na obrázku 3 vlevo. Klient zasílá zprávu s příznakem SYN, server si uloží informace



Obrázek 3: 3–way handshake, korektní průběh a stav při TCP SYN útoku

do paměti a odpovídá vlastní zprávou obsahující příznaky SYN a ACK pro potvrzení klientovy zprávy. Klient následně odpovídá serveru ACK – potvrzuje tak přijetí zprávy od serveru.

V případě útoku (stav zobrazen vpravo) dochází k zahlcení serveru SYN požadavky. Server tyto požadavky zpracovává a snaží se o navázání session pomocí odpovědí SYN + ACK. Útočník však na tyto zprávy neodpovídá a pokračuje v zahlcování dalšími a dalšími pakety. Server se dostává do stavu, kdy již není schopen obsluhovat žádné další požadavky na TCP spojení, dochází k odmítnutí komunikace s legitimním uživatelem.

Dalším druhem útoku na pamět je zasílání vysoce fragmentovaných paketů, přičemž některé části útočník záměrně vynechá. Server se snaží vyžádat chybějící části a obdržené pakety uchovává v paměti. Dokud nevyprší čas, po který jsou pakety udržovány v paměti, zabírají zbytečně místo.

3.2.2 Vyčerpání prostředků CPU

Zatížením procesoru na serveru omezuje útočník schopnost SIP serveru zpracovávat legitimní požadavky uživatelů. Zvýšení zátěže může být způsobeno jak zvýšením počtu požadavků, tak i komplexními požadavky spotřebovávajícími více systémových prostředků. Pomocí zahlcení serveru ICMP pakety lze zvýšit zátěž až na 100%, stejných hodnot však lze dosáhnout i použitím zpráv nutících server k dalším výpočtům. V takovém případě stačí k podobnému výsledku mnohem menší množství zpráv (viz. REGISTER flood kapitola 6.1).

Po přijetí zprávy musí dojít k její analýze a odpovídající akci. I přesto, že jsou servery schopné obsluhovat stovky regulérních zpráv, dokážeme pomocí zpráv obsahující neplatné údaje, neexistující uživatelské účty či zmanipulované hlavičky donutit server k vysoké zátěži a tedy i neschopnosti zpracovávat regulérní SIP zprávy. Situace je o to horší, že se server snaží obsloužit i zprávy s nesprávnými hlavičkami, což ho nutí k větší výpočetní aktivitě.

Také zapnutá autentizace uživatelů přivádí server k náročnějším operacím, stav tak paradoxně ještě více zhoršuje. V případě použití certifikátů může útočník zaslat zprávu, tvářící se jako zpráva od legitimního uživatele. Odkazuje však na neplatný certifikát.

Server sice zjistí, že certifikát je špatný, ale mezitím útočníkova zpráva zabírá místo v paměti i spotřebovává procesorový čas.

Pokud útok skončí, server se většinou po určité době navrátí k původnímu stavu a obsluhuje uživatele stejně jako před útokem.

3.2.3 Útoky zahlcující linku

Zde není cílem spotřebovat zdroje stroje, na němž SIP server běží, ale zahltit linku, kterou je server připojen k síti. Následně dochází k zahazování i legitimních požadavků. Ty totiž nelze v záplavě záškodných paketů rozlišit. Situaci zhoršuje i vlastnost SIPu komunikovat prostřednictvím UDP. Pokud je totiž požadavek nedoručen, je zaslán opětovně znovu, což situaci rozhodně nezlepší.

K útoku se využívá UDP protokol, který není stavový, útočník tak zahlcuje server UDP pakety s pokud možno co největší velikostí. Tím dosáhne odříznutí regulérních uživatelů od serveru. Pokud navíc server nezná IP adresu zdroje, snaží se ji zjistit, což opět situaci ještě více komplikuje.

3.2.4 Útoky zneužívající chyb a vlastností SIPu

Narozdíl od útoků zahlcujících SIP server velkým množstvím paketů, útokům z následující části postačí paketů pouze několik. Využívají totiž slabostí cíle pro svůj prospěch. Rozlišujeme zde útoky proti operačnímu systému, implentaci TCP/IP protokolu v operačním systému a vůči SIP protokolu. Vzhledem k povaze práce se budeme zabývat pouze útoky na SIP protokol.

Metodou útoku zde tedy není zahlcení SIP serveru, naopak se útočník snaží znemožnit uživateli použití VoIP. Útok je tedy mnohem více cílen vůči uživateli či skupině uživatelů. Přestože je tento útok pro poskytovatele služby mnohem méně nepříjemný, musíme si uvědomit několik faktů. Útočník musí být například schopen zachytávat provoz v síti (tzv. *eavesdropping*, viz. výše), zasahovat do zpráv, maskovat se za legitimního uživatele, ...

Úspěšnost těchto útoků zavisí na umístění útočníka a jeho schopnosti sledovat provoz v síti. Pokud se bude útočník nacházet v blízkosti uživatele (například ve stejném segmentu sítě), může být schopen ovlivňovat zprávy tohoto uživatele. Pokud se však dokáže dostat k lince mezi SIP server a zbylou část sítě, představuje již vážný problém, díky možnosti sledovat a zasahovat do veškeré komunikace.

Typickým útokem bývá přesvědčení jedné z komunikujích stran o ukončení hovoru.

3.2.4.1 BYE útok V případě *BYE* útoku použije útočník odposlechnuté hodnoty proměnných v hlavičce SIP zprávy pro vytvoření podvrhnuté SIP zprávy ukončující hovor.

Obranou proti tomuto útoku je pouze použití šifrování, například pomocí TLS.

3.2.4.2 CANCEL útok Použitím *CANCEL* zprávy se útočník snaží o zabránění komunikace už během vytváření hovoru. Jakmile je zaznamenána *INVITE* zpráva, zasílá

útočník podvrženou *CANCEL* zprávu. Ta obsahuje všechny informace obsažené v *INVITE* zprávě i se stejným sekvenčním číslem. Tím vytváří dojem ukončení hovoru. Důležité je, aby útočníkův paket dorazil před finální odpovědí legitimního uživatele.

Díky použití stejného sekvenčního čísla není zpráva autentizována. Jedinou obranou zbývá zamezení odposlechnutí komunikace třetí stranou (například použitím *IPsec*).

3.2.5 Zesílený útok

Druh tohoto útoku již patří mezi zástupce DDoS. Útočník zašle pakety na broadcast adresu v dané síti s podvrženou zdrojovou adresou. Touto adresou je samozřejmě IP adresa cíle útoku. Po rozeslání paketu všem hostům v síti, začnou odpovídat na danou zdrojovou adresu.

Útočník přitom nemusí infiltrovat ostatní zařízení, využije jen ostatních zařízení. Příkladem těchto útoků jsou *Smurf attack* a *Fraggle attack*.

4 Návrh detekce DoS útoků pomocí nástroje Snort

Pro detekci útoků na SIP protokol, respektive SIP proxy server jsme zvolili open source řešení Snort. Tento nástroj pracuje na bázi IDS (NIDS), ale lze jej upravit i tak, že následně dokáže pracovat jako IPS systém (tak jsme jej také využívali). Termínem IDS označujeme techniky a metody, používané pro detekování podezřelé aktivity, a to jak na úrovni sítě, tak i uživatele.

Dále lze tyto systémy rozdělit na dva druhy. První se zakládá na signaturách, druhý pak na detekci anomálií. Aplikace Snort, využitá pro tuto práci, se řadí mezi signaturové IDS, pomocí pluginů však lze přidat i podporu detekce anomálií. Signaturou označujeme jistý vzor hledaný uvnitř paketů, typický pro detekci útoku.

4.1 Komponenty snortu

Před vlastním popisem obrany bych krátce popsal princip funkce nástroje snort. Ten by se dal rozdělit na několik logických komponent, které společně zajišťují rozpoznání rozličných typů útoků. Průběh zpracování paketu je zobrazen na obrázku 4.

4.1.1 Paketový dekodér

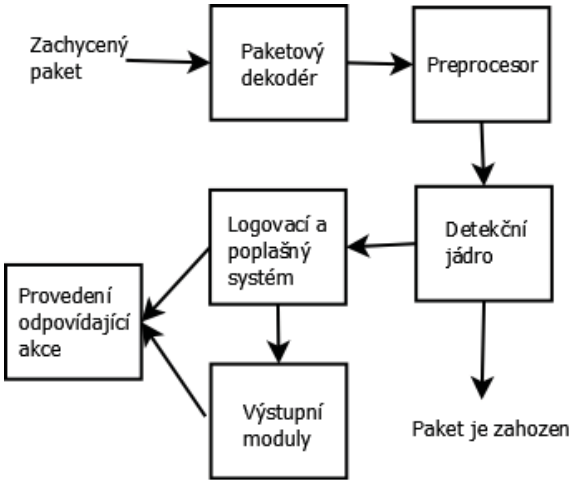
Zajišťuje sběr paketů ze síťových rozhraní, připravuje tyto pakety k předzpracování nebo k zaslání do detekčního jádra.

4.1.2 Preprocesor

Obsahuje komponenty a pluginy pro přípravu či modifikaci paketů před zpracováním. Může také provádět již výše zmíněňou detekci prostřednictvím monitorování výskytu anomálií v síti. Preprocesor patří mezi důležité součásti IDS systému, jelikož zvyšuje schopnost detekce útoku, provádí defragmentaci paketů, dekóduje http URI, znovuskládá TCP streamy, apod.

4.1.3 Detekční jádro

Nejdůležitější součástí aplikace Snort, zodpovídající za detekci narušitelovi aktivity v síti pomocí pravidel. Pravidlem jsou vnitřní struktury nebo řetězce, které srovnáváme vůči paketům. Jakmile paket odpovídá určitému pravidlu, spustí se odpovídající reakce. Zároveň však patří jádro z hlediska času mezi nejvíce kritickou část Snortu. V závislosti na výkonu stroje a množství pravidel dochází k různým dobám vyhodnocování rozdílných paketů. Při vysokém provozu uvnitř sítě může docházet k zahazování paketů, případně nedochází k real-time odpovědi na útok. Nejsou neobvyklé ani útoky mířené přímo proti IDS/IPS nástrojům tak, aby v zápětí nebyly schopné zaznamenat či reagovat na další útok.



Obrázek 4: Snort – průběh analýzy paketu

4.1.4 Logovací a poplašný systém

V závislosti na idetifikované hrozbě uvnitř paketu může docházet pouze k logování či vyvolání poplachu, což má na starosti právě tato komponenta. Při vlastním návrhu obrany pak využíváme stav, kdy je každý útočný paket zároveň i zalogován a následně uložen do mysql databáze. Díky tomu lze zpětně vyhodnocovat daný útok, přijímat nová protiopatření.

4.1.5 Výstupní moduly

Neboli také pluginy, mohout provádět různé operace podle požadavků bezpečnostního administrátora. Vzhledem k velikosti projektu Snortu je těchto pluginů dostupných velká řada – například lze zmínit pluginy pro vytváření SMTP trap, zasílání zpráv do syslogu, generovaní XML výstupů, úpravu v konfiguraci routerů, atd.

4.2 SnortSam

Jedná se o aplikaci fungující na principu klient–server. Slouží pro zásah do iptables na základě akcí ve Snort pravidlech. Umožňuje tedy, aby mohl IDS systém Snort jednat jako IPS. Vlastní aplikace se skládá ze serveru, klienta a patche pro aplikaci Snort. Server následně poslouchá na portu 898. Zprávy zaslané od aplikace Snort zpracovává a informuje o akci jednotlivé klienty. Tito klienti se pak nacházejí přímo na firewallu, kde převádí zprávy od serverů do podoby pravidel pro iptables. Vzhledem k zatížení stroje a případým přenosům zpráv mezi klienty a serverem dochází nutně ke zpoždění. O toto zpozdění probíhá detekovaný útok déle, dokud se nestačí aplikovat vhodná protiopatření.

SnortSam zprávy mezi klientem (označovaným také jako agent) a serverem jsou přenášeny šifrovaně, je tedy nutné mít správně nastavená hesla na obou stranách. K dis-

pozici je také white-list pro definování seznamu IP adres, které nebudou nikdy blokovány. Jestliže proběhne ověření zprávy v pořádku a IP adresa není zmíněna na white–listu, dochází k aktivaci pravidla omezujícího daný nežádoucí provoz. SnortSam navíc dokáže po uplynutí stanové doby (definované u pravidla, příp. nastavené jako default) provoz znovu povolit. Tím dochází k blokování jen toho provozu, který opravdu v danou chvíli ohrožuje chráněnou síť.

4.3 IPtables

Program fungující nejenom jako firewall, určený pro operační systémy na linuxovém jádru, poskytující následující možnosti.

- Stavové a bezestavové filtrování paketů (IPv4 i IPv6)
- Překlad síťových adres a portů (NAT, NAPT, pouze pro IPv4)
- Flexibilní a rozšiřitelnou infrastrukturu
- Velké množství pluginů v repozitáři

Právě zmíněný program IPtables budeme používat pro blokování zaznamenaných útoků.

4.4 Testovací prostředí

Vzhledem k možnostem dnešních embedded systémů a výkonostním parametrům těchto systémů, nás zaujala myšlenka umístění SIP serveru na tomto zařízení. S ohledem na cenu takovýchto zařízení se jedná i o velmi levnou a dostupnou metodu, jak řešit VoIP komunikaci v prostředí malých firem či detašovaný pracovišť. Navíc útoky proti tomuto zařízení mají mnohem větší dopad, díky omezenému výpočetnímu výkonu. Pro testování účinnosti útoků i následných protiopatření byla zvolena následující konfigurace SIP serveru.

- OS Ubuntu 10.04 LTS (32-bitová verze)
- 512 MB RAM
- 1 CPU
- cca 2 GB HDD

Obdobně vypadá konfigurace i u dalšího PC, které slouží pro provádění útoků cílených na SIP proxy. Obě PC jsou vzájemně spojeny prostřednictvím virtuálního switche, lze tedy očekávat i jisté dopady této virtualizace. Na SIP proxy serveru byla naistalována aplikace Asterisk 1.6.2.5, jako IPS systém byla zvolena kombinace Snort, SnortSam a firewall na bázi IPtables.

Zvažována byla i možnost využití programu fail2ban, který pracuje poněkud jiným způsobem. Sledováním logů a vlastním seznamem pravidel (regulární výrazy) je schopen

blokovat nežádoucí komunikaci na základě výskytu jistých zpráv v logu aplikace. Bohužel ale při některých útocích není do logu zapsána informace o IP adrese útočníka, nelze tedy vytvářet pravidla proti širšímu spektru útoků i komplexnějším variantám útoku. Posledním důvodem pro zamítnutí byla i větší míra prodlení než v případě reakce od aplikace Snort.

5 Realizace zabezpečení proti DoS nástroji Snort a IPtables

Průběh instalace OS Ubuntu 10.04 LTS je dosti intuitivní, dovolím si však shrnutí této instalace. Operační systém je k dispozici ke stáhnutí na adrese[1] (v době testování byla nejvyšší dostupná verze 10.04.1). Tento obraz byl použit pro instalaci operačního systému jak SIP serveru, tak i záškodnického PC, z něhož budou následně prováděny útoky. Vlastní instalace se skládá z několika kroků.

1. Zvolíme jazyk instalace: čeština
2. Nastavení rozložení klávesnice: Česko - qwerty, rozšířená klávesa zpětného lomítka
3. Nastavení jména počítače: *SIPproxy*
4. Potrvzení volby časové zóny
5. Rozložení disku: Asistované - použít celý disk a nastavit LVM
6. Následují další volby pro vytvoření diskových oddílů, ponechány defaultní hodnoty.
7. Nastavení jména uživatele: *sipproxy*
8. Nastavení hesla: *sipdos*
9. Zvolíme nešifrovat domovský adresář
10. Aktualizace: Instalovat bezpečnostní aktualizace automaticky
11. Ve výběru programů vybereme možnosti: LAMP, OpenSSH server
12. Nastevení root hesla pro MySQL: *mysql*
13. Nainstalujeme zavaděč GRUB do hlavního zaváděcího záznamu.

Po restartování máme nainstalován OS na počítači pro SIP server, přihlásíme se tedy pod uživatelským jménem *sipproxy* s heslem *sipdos*. Provedeme ještě aktualizaci systému pomocí nástroje *aptitude*, případně pomocí

```
apt-get update
apt-get upgrade
```

[1] http://www.ubuntu.cz/ziskejte/stahnout

5.1 Instalace aplikace Snort a jejích součástí

Instalace open source IDS Snort nemůže být provedena prostým způsobem pomocí programů *aptitude* či *apt*, neboť ji chceme využívat jako IPS a je tedy nutné provést instalaci ze zdrojových kódů. Tato nutnost vzniká právě díky použití s aplikací *SnortSam*, která vyžaduje opatchování zdrojových kódů Snortu. Patch je dostupný na stránkách[2], bohužel však pouze pro verzi Snort 2.9.0.3 (v době testování již byla dostupná vyšší verze Snort 2.9.0.4, patch pro tuto verzi však nikoliv). Zdrojové kódy pro Snort tedy stáhneme prostřednictvím příkazu *wget* následujícím způsobem.

```
wget http://www.snort.org/dl/snort-current/snort-2.9.0.3.tar.gz
    -O snort-2.9.0.3.tar.gz
```

Tímto máme stáhnuté zdrojové kódy pro Snort, verze od 2.9.0 výše ale využívají nové daq, stáhneme tedy i tyto zdrojové kódy.

```
wget http://www.snort.org/downloads/860 -O daq-0.5.tar.gz
wget http://www.snortsam.net/files/snort-plugin/
    snortsam-2.9.0.3.diff.gz -O snortsam-2.9.0.3.diff.gz
wget http://www.snortsam.net/files/snortsam/
    snortsam-src-2.70.tar.gz -O snortsam-src-2.70.tar.gz
```

Poslední dva výše zmíněné přikazy stáhnou zdrojové kódy aplikace *SnortSam* i patch potřebný pro *Snort*. Další aplikací, jejíž zdrojové kódy potřebujeme, je *libdnet*. Je sice dostupný i ve formě balíčků, ale vhodnější je provést instalaci ručně.

```
wget http://libdnet.googlecode.com/files/libdnet-1.12.tgz
    -O libdnet-1.12.tgz
```

Nyní již máme k dispozici všechny zdrojové kódy pro kompilaci Snortu, musíme ale ještě doinstalovat aplikace, které jsou pro běh Snortu nezbytné, případně jsou nutné pro vlastní kompilaci. Tyto aplikace lze ale instalovat prostřednictvím systémového správce balíků.

- libpcap0.8-dev
- libpcap-dev
- libmysqlclient15-dev (libmysqlclient-dev)
- libmysqld-dev
- bison
- flex
- libapache2-mod-php5

[2] http://www.snortsam.net/download.html

- php5-gd
- php5-mysql
- libtool
- libpcre3-dev
- pcre-dev
- php-pear
- gcc
- automake
- autoconf
- apache2
- mysql-client-5.1
- mysql-server-5.1

Některé z těchto programů již mohou být nainstalovány (zvláště pak poslední trojice, díky možnosti LAMP při instalaci operačního systému), žádný z nich by však neměl chybět.

5.1.1 Kompilace a instalace daq

Probíhá obvyklým způsobem - rozbalením archívu a tzv. „svatou trojicí“.

```
tar zxvf daq-0.5.tar.gz
cd daq-0.5
./configure
make
make install
```

5.1.2 Kompilace a instalace libdnetu

Stejný průběh jako u *daq*, posledním krokem je vytvoření symbolického linku.

```
tar zxvf libdnet-1.12.tgz
cd libdnet-1.12/
./configure
make
make install
ln -s /usr/local/lib/libdnet.1.0.1 /usr/lib/libdnet.1
```

5.1.3 Příprava zdrojových kódů Snortu, instalace SnortSam

Přejdeme do složky, kde máme stažený patch snortsam-2.9.0.3.diff.gz. Začneme rozbalením archívu.

```
gunzip snortsam-2.9.0.3.diff.gz
```

Nyní nastává čas na konfiguraci kódů Snortu a aplikaci rozbaleného patche. Pokud se patch pro snort nenachází o úroveň adresáře výše než snort, zadejte v příkazu *patch* cestu k tomuto souboru.

```
patch -p1 < ../snortsam-2.9.0.3.diff
bash ./autojunk.sh
./configure --enable-ipv6 --enable-gre --enable-mpls
    --enable-targetbased  --enable-decoder-preprocessor-rules
    --enable-ppm --enable-perfprofiling --enable-zlib
    --enable-active-response --enable-normalizer --enable-reload
    --enable-react --enable-flexresp3 --with-mysql
make
make install
groupadd snort
useradd -g snort snort
mkdir /var/log/snort
chown snort:snort /var/log/snort
mkdir /etc/snort
mkdir /etc/snort/rules
cd rules/
cp * /etc/snort/rules
cd ../etc/
cp * /etc/snort/
```

Aplikaci SnortSam pak doinstalujeme následovně. Ve složce s rozbalenými kódy spustíme tyto příkazy.

```
chmod -x makesnortsam.sh
./makesnortsam.sh
cp snortsam /usr/bin/
```

V době testování byla používána verze 2.70

5.1.4 Konfigurace Snortu

Konfigurační soubor nacházející se v */etc/snort/snort.conf* musíme mírně upravit. Nastavíme proměnné *HOME_NET* a *EXTERNAL_NET*, zavedeme nové proměnné pro SIP proxy server.

```
var SIP_PROXY 192.168.0.10
portvar SIP_PORT 5060
var HOME_NET 192.168.0.0/24
var EXTERNAL_NET !$HOME_NET
```

Zprovozníme i spolupráci s mysql databází. V části pro výstupní pluginy vložíme následující řádek.

```
output database: log, mysql, user=snort password=snortsipdos
    dbname=snort host=localhost
```

Nadále nastavíme už pouze SnortSam agenta, resp. zadáme ip na níž naslouchá a heslo.

```
output alert_fwsam: 192.168.0.10/snortsipdos
```

Jako poslední nastavíme, které soubory s pravidly bude Snort používat. Pro testovací účely a omezení falešných poplachů byl povolen pouze soubor *local.rules*. Do tohoto souboru budeme následně ukládat pravidla pro Snort.

```
include $RULE_PATH/local.rules
```

5.1.5 Konfigurace SnortSam

Vzorový konfigurační soubor pro SnortSam obsahuje dokumentace programu SnortSam. Pro naše účely stačí vytvořit soubor */etc/snortsam.conf* s těmito řádky.

```
defaultkey snortsamdefault
accept 192.168.0.10, snortsipdos
logfile snortsam.log
loglevel 3
deamon
iptables eth0 syslog.info
```

Tímto jsme nastavili defaultní heslo agentů, povolili přijímání z ip adresy snort senzoru s heslem *snortsipdos* (musí se shodovat s heslem uvedeným v */etc/snort/snort.conf*). Akce SnortSam jsou logovány, aplikace se spouští na pozadí jako daemon a zásahy do *IPtables* provádí na rozhraní `eth0`.

5.1.6 Nastavení databáze mysql

Tento krok není nutný pro korektní práci Snortu, ale umožní nám ukládat zachycené pakety (tedy ty, jenž vyhoví pravidlům Snortu) do databáze mysql, pro zpětnou analýzu. Musíme však vytvořit správné uživatele a databázovou strukturu. První polovina příkazů vytvoří databázi *snort* a umožní uživateli *snort* s heslem *snortsipdos* pracovat s touto databází. Umístění souboru s databázovou strukturou se může lišit v závislosti na složce, do níž byly staženy zdrojové kódy.

```
mysql -p
create database snort;
grant CREATE,INSERT,DELETE,SELECT,UPDATE on snort.*
    to snort@localhost;
SET PASSWORD FOR snort@localhost=PASSWORD('snortsipdos');
exit
cd /usr/local/snort/snort-2.9.0.3/schemas
mysql -p < create_mysql snort
```

Pokud proběhne vše v pořádku, měla by databáze snort obsahovat 16 tabulek.

```
mysql -p
use snort;
show tables;
```

5.1.7 Zprovoznění gui BASE

Podobně jako u databáze se nejedná o povinnou součást, ale o webové rozhraní pro analýzu chování Snort. Plugin je ke stáhnutí na webu sourceforge[3] Ke svojí činnost však potřebuje ještě *ADODB*, k dispozici ke stažení opět na sourceforge[4]. Stažený soubor stačí pouze rozbalit do složky */var/www/*

```
cd /var/www/
wget  http://downloads.sourceforge.net/project/adodb/
    adodb-php5-only/adodb-511-for-php5/adodb511.zip
    -O adodb511.zip
gunzip adodb511.zip
rm adodb511.zip
```

V adresáři by se nyní měla nacházet složka *adodb5*. Přesuneme se do složky */var/www/html/* a stáhneme pomocí aplikace *wget* BASE.

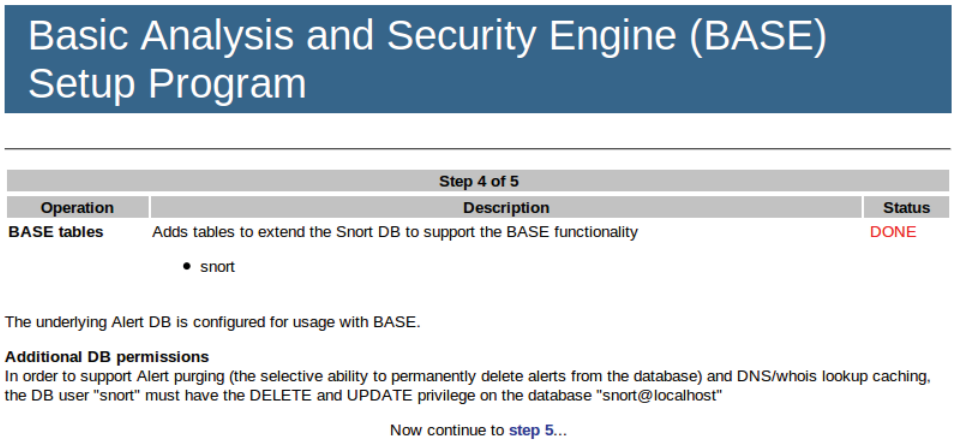
```
tar xvzf base-1.4.5.tar.gz
rm base-1.4.5.tar.gz
mv base-1.4.5 base
cd base/
cp base_conf.php.dist base_conf.php
```

Před konfigurací *BASE* je vhodné provést úpravy souboru */etc/php5/apache2/php.ini*. Na odpovídajících místech upravíme či odkomentujeme tyto řádky.

```
error_reporting = E_ALL & ~E_NOTICE
extension = mysql.so
extension = gd.so
```

[3] http://easynews.dl.sourceforge.net/sourceforge/secureideas/base-1.4.5.tar.gz

[4] http://downloads.sourceforge.net/project/adodb/adodb-php5-only/adodb-511-for-php5/adodb511.zip



Obrázek 5: BASE – stav na konci nastavení

Nyní stačí restartovat apache web server `\etc\init.d\apache2 restart`

Webové rozhraní pro analýzu Snort poplachů je dostupné na url ve tvaru ip adresy stroje + odpovídající cesty ke stránkám, například

`http://192.168.0.10/html/base`

5.1.7.1 Nastavení BASE Rozhraní je ale třeba ještě nakonfigurovat, aby bylo schopné spolupracovat se Snort a mysql databází. Konfiguraci lze provést ručně editací souboru *base_conf.php* nebo pomocí gui na

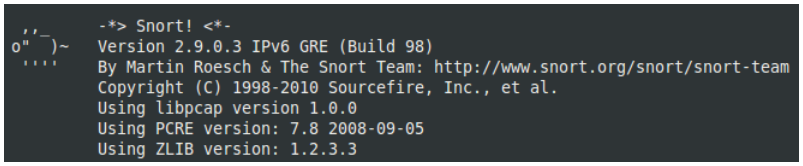
`http://192.168.0.10/html/base/setup`

První stránka obsahuje základní informace o php, klikneme na tlačítko *Continue*. BASE disponuje přeložením do českého jazyka, můžeme tedy na stránce s volbou jazyka zvolit možnost *czech*. Do řádku *Path to ADODB* vyplníme cestu do adresáře, kam jsme rozbalili ADODB, tedy */var/www/adodb5*. Pokračuje stisknutím *Continue*.

V druhém kroku je nutné provést nastavení připojení k mysql databázi.

- Database Name: *snort*
- Database Host: *localhost*
- Database Port: *necháme prázdné, doplní se defaultní hodnota*
- Database User Name: *snort*
- Database Password: *snortsipdos*

Možnost použití archivní databáze ponecháme nevyplněnu. Pokračujeme stisknutím *Continue*.



Obrázek 6: Výsledek příkazu snort -V

Třetím krok spočívá ve vyplnění uživatelského jména administrátora, jeho hesla a jména. Tato možnost však není povinná, v produkční verzi by ale tyto údaje měly být vyplněny. Pro potřeby testování nebylo ověřování nutné, pokračujeme *Continue*.

Čtvrtý krok z pěti přidá k databázi vlastní tabulky potřebné pro běh *BASE* kliknutím na tlačítko *Create BASE AG*. Pokračováním ke kroku 5 se již dostaneme k hlavnímu rozhraní *BASE*, konfigurace je ukončena.

5.2 Asterisk

Pro potřeby funkcionality SIP proxy serveru bylo vybráno řešení Asterisk 1.6 Jedná se o open–source implementaci VoIP ústředny, patřící mezi velmi oblíbené a často nasazované. Zdokumentovány jsou i útoky cílené přímo proti tomuto řešení.

Verze 1.6 přináší mnohé novinky, zvláště v rámci zabezpečení. Mezi nejvýraznější patří podpora TLS pro SIP protokol. Dalšími novinkami jsou zavedení SIP Session Timers definující ukončení SIP spojení, které bylo ukončeno síťovými problémy. Z hlediska sítě přibývá podpora CoS, umožňující zvýhodňování SIP provozu (v rámci QoS). Novým modulem je i spolupráce s LDAP.

5.3 Spuštění IPS Snort

Nyní ověříme, zda je *Snort* schopen běhu. Vypíšeme si informace o verzi Snortu a jeho součástech, která by měla vypadat jako na obrázku 6.

```
snort -V
```

Pokud příkaz skončí chybou, nejspíše bude nutné exportovat proměnnou *LD_LIBRARY_PATH*. Snort pak spustíme s parametry, které mu oznamují umístění konfiguračního souboru, uživatele, pod nímž má běžet, a síťové rozhraní pro sledování provozu.

```
LD_LIBRARY_PATH=/usr/local/lib/
export LD_LIBRARY_PATH
snort -u snort -i eth0 -c /etc/snort/snort.conf -D
```

Poslední parametr *-D* spustí Snort jako daemon na pozadí. Správnost konfigurace Snort lze jednoduše ověřit spuštěním pouze s parametrem určujícím umístění konfiguračního souboru. Z poskytnutého výpisu lze identifikovat chyby v pravidlech apod. Při spuštění ve formě daemona nejsou tyto informace vypisovány.

SnortSam má veškeré nastavení již uvedeno v souboru *snortsam.conf*, spustíme ho tedy prostým příkazem `snortsam`. Informace o práci snortsam agenta jsou ukládány do logovacího souboru *snortsam.log*, kde lze zjistit informace o případných chybách.

Pro ulehčení práce jsou příkazy pro spuštění obsaženy ve skriptu, který lze spouštět automaticky i při startu operačního systému.

5.4 Příprava PC útočícího na SIP server

Příprava útočníkova PC je proti SIP serveru mnohem jednodušší. Jak již bylo zmíněno výše, používá operační systém shodný se SIP serverem, u instalace tedy postupujeme stejně. Jedinou změnou je uživatelský účet *attacker* s heslem *sipdos*, název PC byl nastaven na *SIPattacker*.

Po instalaci systému pouze aktualizujeme a můžeme přejít k instalaci jednotlivých programů pro útoky na SIP server. Další řádky popisují jak tento software získat i nainstalovat. Použití těchto nástrojů může být mnohdy v konfliktu se současnou legislativou, používejte tedy tyto nástroje pouze k otestování uvnitř testovací topologie.

Pro účely testování bylo použito několik typů těchto nástrojů. Prvním typem jsou nástroje určené pro analýzu sítě či otevřených portů – zástupcem této kategorie je *fping* a *nmap*. Tyto nástroje slouží pro vyhledání „živých“ IP adres v síti a zmapování jejich otevřených portů. Aplikace lze ale použít i pro zvýšení bezpečnosti daného zařízení, dostupná je také instalace pomocí standardního správce balíků. Dále zmíněné typy už se plně orientují na provedení *DoS* útoku proti SIP serveru. Program *sipp*, ve správci balíků pojmenovaný jako *sip-tester*, slouží pro simulování hovorů. Scénáře lze ale upravit pro potřeby útoku a zasílat tak podvržené SIP zprávy. Podobným nástrojem je i *inviteflood*, ten se však soustředí pouze na zasílání *INVITE* zpráv. Nástroj *udpflood* řadíme mezi nástroje vyčerpávající přenosovou kapacitu linky k SIP serveru. Posledním typem, který byl pro útoky použit, jsou nástroje umožňující provést *TCP SYN flood*, konkrétně pak *flood2* a *juno* v různých úpravách.

5.4.1 Sipp, fping, nmap

Tyto nástroje můžeme stáhnout prostřednictvím standardního správce balíčků, jejich instalace nebude dále rozebrána.

5.4.2 inviteflood

Nástroj stáhneme ze stránek hacking voip[5]. Všechny nástroje budou stahovány do složky */usr/src/*.

```
wget http://hackingvoip.com/tools/inviteflood.tar.gz
    -O inviteflood.tar.gz
```

Pro správnou funkci musíme nainstalovat i následující závislosti.

[5] http://www.hackingexposedvoip.com

- libnet v1.1.2.1 (a vyšší verze)
- hack_library

Program po stažení rozbalíme a zkompilujeme. Korektní průběh pomocí *makefile* předpokládá, že se o adresář výše nachází složka *hack_library*, obsahující soubory *hack_library.o* a *hack_library.h*[6].

```
tar xzvf inviteflood.tar.gz
cd inviteflood
./configure
make
make install
```

Výsledkem by měl být program *inviteflood*, jehož použití bude popsáno samostaně v kapitole o útoku pomocí *INVITE* zpráv – kapitola 6.2.

5.4.3 udpflood

Stejně tak jako *inviteflood* je dostupný na webu Hacking VoIP exposed. Program zasílá 1400 bytů velké UDP zprávy s cílem zahlcení linky.

```
wget http://hackingvoip.com/tools/udpflood.tar.gz
    -O udpflood.tar.gz
tar xzvf udpflood.tar.gz
cd inviteflood
./configure
make
make install
```

5.4.4 flood2

Dostupný ze stránek[6], pouze ve formě zdrojového kódu, který autor záměrně poškodil tak, aby nebylo možné ho bez alespoň minimálních znalostí programování použít. Před kompilováním tedy otevřeme zdrojový kód a odkomentuje uzavírající závorku vykytující se ke konci kódu. Opravený soubor je součástí příloh.

```
mkdir flood2
cd flood2
wget http://dl.packetstormsecurity.net/DoS/flood2.c -O flood2.c
gcc flood2.c -o flood2
```

[6] http://packetstormsecurity.org

5.4.5 juno

Zástupce *TCP SYN flood* nástrojů stáhneme ze stejného zdroje jako *flood2*.

```
mkdir juno
cd juno
wget http://dl.packetstormsecurity.net/DoS/juno.c -O juno.c
gcc juno.c -o juno
```

Tímto získáme nástroj pro zasílání *TCP SYN flood* zpráv z podvrhnutých IP adres a náhodných portů. Vhodnou úpravou zdrojového kódu vytvoříme další varianty tohoto nástroje posílající zprávy

1. z jediné IP adresy a portu (*singlejuno*)
2. z jediné IP adresy a různých zdrojových portů (*ranportjuno*)

6 Zhodnocení dosažených výsledků

Pro otestování účinnosti obraného mechanismu tvořeného spojením aplikací *Snort*, *Snort-Sam* a *IPtables* bylo provedeno několik útoků na SIP server. Tyto útoky zkoumaly nejen míru dopadu útoku na výkon serveru, ale i chování klientských zařízení k tomuto serveru připojeným.

Musíme také uvést, že na serveru běžela jak aplikace *Asterisk*, sloužící jako SIP proxy server, tak i celý obraný mechanismus založený na IDS Snort. Tento postup byl zvolen s důrazem na to, zda je schopné vytvořit mechanismus obrany na vlastním zařízení.

Útočné schéma bylo pro všechny uvedené útoky stejné. Stav serveru byl monitorován po dobu 90 sekund, útok začal po uplynutí 10 sekund a trval následně dalších 50 sekund. Poslední půlminutu bylo sledováno chování SIP serveru po útoku, respektive schopnost vyrovnat se s útokem po jeho odeznění. Testovací scénáře probíhaly ve dvou či více vlnách, v závislosti na použitých nástrojích. Pokud byl útok prováděn bez obraného mechanismu běžícího na serveru, aplikace vůbec neběžely, tj. chování bylo stejné, jako by na daném stroji nebyly vůbec instalovány.

6.1 Útok REGISTER flood

Útok typu *REGISTER flood* spočívá v zahlcování SIP serveru žádostmi o registraci. K provedení útok byla použita aplikace *sipp*, konkrétně zasílající 250 žádostí o registraci za sekundu. Obsah *REGISTER* zprávy (tedy hlavičku této SIP zprávy) obsahoval soubor *reg_notag.xml*.

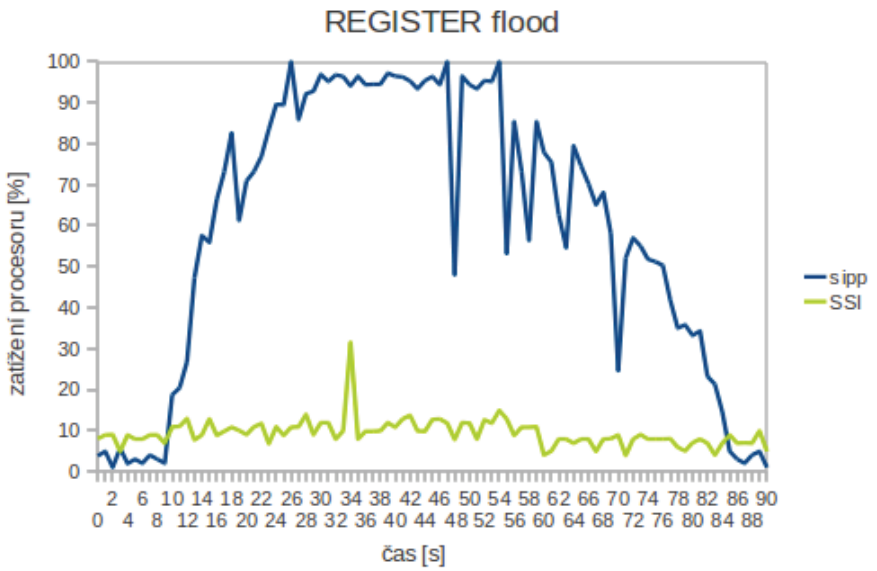
```
sipp -sn uac 192.168.0.10:5060 -sf reg_notag.xml -m 100000
    -r 250 -s 1003
```

Uvedeným příkazem provedeme útok, jednotlivé parametry označují použité schéma útoku, ip adresu SIP serveru (tj. cíl útoku), vlastní scénář zaslaných zpráv, počet celkově zaslaných zpráv (parametr *-m*), množství zpráv zaslaných během jedné sekundy *-r* i číslo stanice pokoušející se registrovat.

Dopad na server je patrný z obrázku 7. Útok na zařízení probíhá od času 10 s do 60 s, z grafu je také dobře patrné následné vyrovnávání serveru po útoku.

Jako obrana bylo zvoleno pravidlo sledující počet *REGISTER* zpráv z jednoho zdroje. V případě překročení přípustné hranice se aktivuje zablokování tohoto zdroje pomocí *IPtables* tak, aby nebyl *Asterisk* dále zatěžován útočnými zprávami. Do souboru pravidel Snortu přídáme pravidla.

```
alert udp $EXTERNAL_NET any -> $SIP_PROXY $SIP_PORT (msg:"SIP
    DoS attempt(registerflood)"; content:"REGISTER sip";
    detection_filter:track by_src, count 50, seconds 5;
    classtype:misc-attack; sid:1000001; rev:1; fwsam:src, 5min;)
alert tcp $EXTERNAL_NET any -> $SIP_PROXY $SIP_PORT (msg:"SIP
    DoS attempt(registerflood tcp)"; content:"REGISTER sip";
    detection_filter:track by_src, count 50, seconds 5;
    classtype:misc-attack; sid:1000007; rev:1; fwsam:src, 5min;)
```



Obrázek 7: Zatížení procesoru SIP serveru během útoku REGISTER flood

Tato pravidla se soustředí na zprávy obsahující spojení *REGISTER sip* v hlavičce. Jejich blokování se provádí pomocí zdroje, z nějž přicházejí, po dobu 5ti minut. Čas, po který můžeme zdroj blokovat lze samozřejmě přizpůsobit vlastním požadavkům.

Průběh útoku s aktivním protiopatřením je v grafu na obrázku 7 označen zelenou barvou. Zkratka *SSI* označuje Snort, SnortSam a IPtables.

Vyšší zatížení procesoru před útokem způsobuje více běžících aplikací, špičku v době cca 34 sekund způsobil web server *Apache2*. V grafu lze pozorovat pouze nízké zvýšení aktivity procesoru v řádu pár procent, způsobených Snortem, analyzujícím provoz.

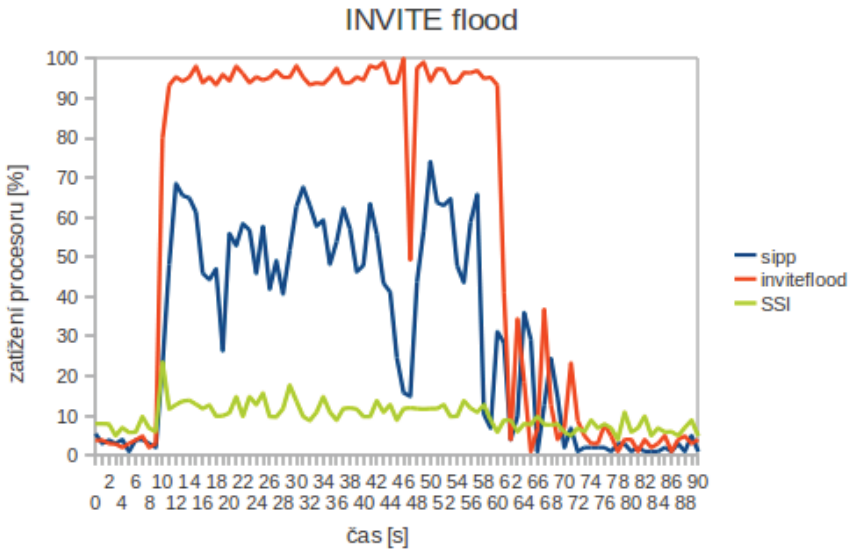
Z grafu vyčteme také, že serveru trvalo ještě dalších 25 sekund, než se po útoku vrátil do původního stavu. Během probíhajícího útoku nebyly narušeny již probíhající hovory (není třeba komunikovat s SIP serverem, RTP stream byl pouze mezi stanicemi), nové hovory či registrace však nebyly možné. S aktivním obraným mechanismem nebyla komunikace se SIP serverem nijak výrazně ovlivněna.

6.2 Útok INVITE flood

Dalším typem útoku zaměstnávajícím silněji SIP server je právě *INVITE flood*. Útok zasílá podobně jako *REGISTER flood* žádosti na SIP server, zde se však jedná o žádost vytvoření spojení. Pro porovnání míry dopadu jednotlivých druhů zpráv, zvolili jsme u sipp aplikace stejné nastavení počtu zpráv zaslaných za sekundu.

```
sipp -sn uac 192.168.0.10:5060 -sf invite.xml -m 100000
    -r 250 -s 1003
```

Pro útok byla použita i další aplikace *inviteflood* (viz kapitola 5.4.2). Vzhledem k rozdílným možnostem použití s velmi podobnými výsledky, spustíme následující příkaz.



Obrázek 8: Zatížení procesoru SIP serveru během útoku INVITE flood

```
./inviteflood eth0 1001 192.168.0.10 192.168.0.10 1000000
```

Syntaxe příkazu:

```
./inviteflood <rozhraní> <SIP číslo> <cílová doména>
    <cílová IP> <počet paketů>
```

Příkaz zasílá zprávy bez omezení množství, v grafu na obrázku 8 znázorněn oranžovou barvou.

Zajímavostí je určitě patrný pokles zatížení procesoru v době 47 s, kdy došlo u obou útoků ke krátkodobému poklesu. Tento pokles můžeme pozorovat i v předchozím případě útoku *REGISTER flood*.

Obraným mechanismem jsou pravidla:

```
alert udp $EXTERNAL_NET any -> $SIP_PROXY $SIP_PORT (msg:"SIP
    DoS attempt(inviteflood)"; content:"INVITE sip";
    detection_filter:track by_src, count 50, seconds 5;
    classtype:misc-attack; sid:1000002; rev:1; fwsam:src, 5min;)
alert tcp $EXTERNAL_NET any -> $SIP_PROXY $SIP_PORT (msg:"SIP
    DoS attempt(inviteflood tcp)"; content:"INVITE sip";
    detection_filter:track by_src, count 50, seconds 5;
    classtype:misc-attack; sid:1000008; rev:1; fwsam:src, 5min;)
```

Výsledné zatížení při aktivním obraném mechanismu obsahuje obrázek 8, značené zelenou barvou. Při počátku útoku lze spatřit vyšší zatížení, způsobené zpožděním mezi detekcí útoku a aktivováním pravidla v IPtables. Během této doby pronikaly na SIP server požadavky a proto tedy dané zvýšení.

Lze také pozorovat poměrně rychlejší zotavení po útoku, než v předcházejícím případě. Velmi podobný byl také dopad. Při útoku pomocí *sipp* s množstvím 250 zpráv za sekundu byl hovor po velmi dlouhé době vytvořen, většinou ale spojení vytvořit nešlo.

Při použití *inviteflood* se spojení vytvořit nepovedlo. Došlo však také k případu, kdy spojení nebylo vytvořeno, po ukončení útoku však náhle přišel požadavek na dané zařízení. Použitím nástroje *sipp* s vyšší hodnotou zasílání zpráv dosáhneme podobných výsledků jako u *inviteflood* (jedná se o hodnotu cca 2500 zpráv/s). Použitím uvedených pravidel docílíme úspěchu pouze částečně. V případě *inviteflood* útoku se schéma shoduje s *udpflood*, dochází tedy k blokování komunikace se SIP proxy.

6.3 Útok ACK,BYE a CANCEL flood

Při zasílání dalších typů zpráv, jako *ACK*,*BYE* a *CANCEL* již nemusí docházet k náročným operacím na straně SIP serveru. Útok provádíme obdobně jako v předchozích případech, opět v množství 250 zpráv za sekundu. Zasíláme pouze zprávy z podvrženého čísla, protože žádné takové hovory neprobíhají, nemůže dojít k jejich narušení. I v případě hovoru by bylo nutné znát hodnoty typické pro daný hovor (pokud není použito šifrování).

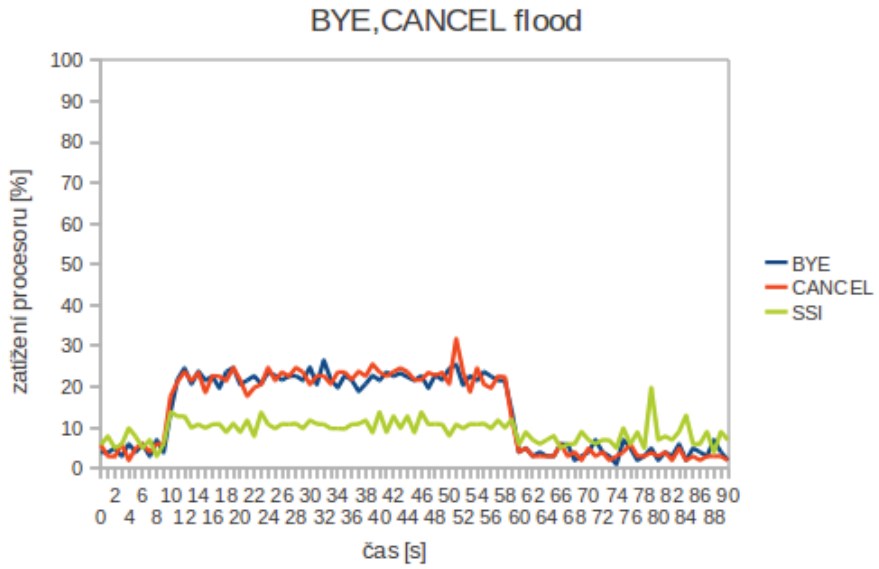
```
sipp -sn uac 192.168.0.10:5060 -sf ack.xml -m 100000
    -r 250 -s 1003
sipp -sn uac 192.168.0.10:5060 -sf bye.xml -m 100000
    -r 250 -s 1003
sipp -sn uac 192.168.0.10:5060 -sf cancel.xml -m 100000
    -r 250 -s 1003
```

Na grafu (obr. 9) můžeme pozorovat mnohem nižší zatížení procesoru tímto útokem. Patrný není ani výrazný rozdíl mezi těmito zprávami (*ACK flood* není v grafu pro přehlednost znázorněn).

Obranu tvoří příkazy pro Snort (varianta pro UDP přenos):

```
alert udp $EXTERNAL_NET any -> $SIP_PROXY $SIP_PORT (msg:"SIP
    DoS attempt(ackflood)"; content:"ACK sip";
    detection_filter:track by_src, count 50, seconds 5;
    classtype:misc-attack; sid:1000003; rev:1; fwsam:src, 5min;)
alert udp $EXTERNAL_NET any -> $SIP_PROXY $SIP_PORT (msg:"SIP
    DoS attempt(byeflood)"; content:"BYE sip";
    detection_filter:track by_src, count 50, seconds 5;
    classtype:misc-attack; sid:1000004; rev:1; fwsam:src, 5min;)
alert udp $EXTERNAL_NET any -> $SIP_PROXY $SIP_PORT (msg:"SIP
    DoS attempt(cancelflood)"; content:"CANCEL sip";
    detection_filter:track by_src, count 50, seconds 5;
    classtype:misc-attack; sid:1000005; rev:1; fwsam:src, 5min;)
```

Vzhledem k nízkému účinku na zatížení procesoru by bylo možné i tyto pravidla vynechat a použít obecné pravidlo, jak je tomu i u útoku udpflood (6.4). Komunikace se serverem není při použití uvedených pravidel nijak ovlivněna.



Obrázek 9: Zatížení procesoru SIP serveru během útoku BYE&CANCEL flood

6.4 Útok OPTIONS flood

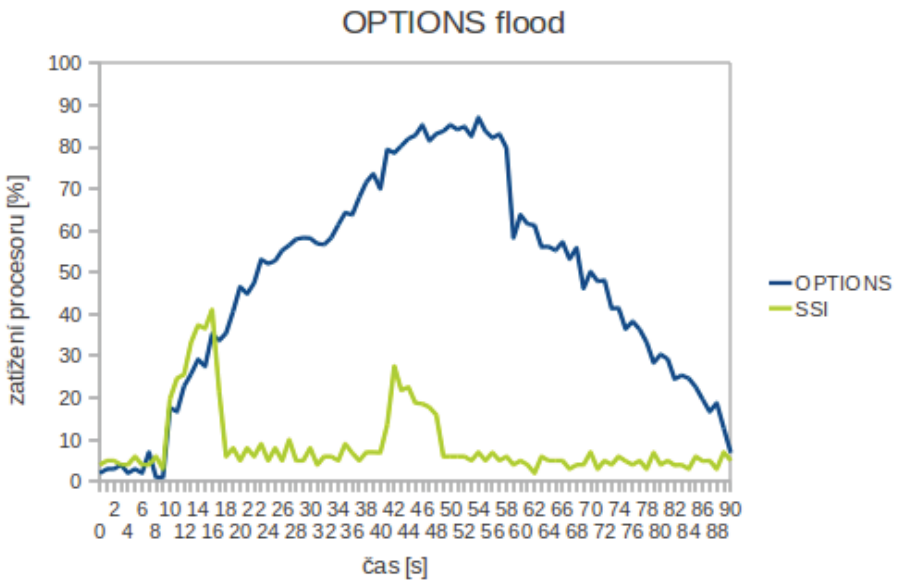
Posledním útokem na server pomocí SIP zpráv je zasílání *OPTIONS* zpráv. Ty slouží k zjišťování vlastností SIP zařízení strukturou podobnou zprávě *INVITE*. Můžeme tedy očekávat podobný dopad tohoto útoku. Pro útok použijeme opět programu *sipp* s odpovídajícím scénářem.

```
sipp -sn uac 192.168.0.10:5060 -sf options.xml -m 100000
    -r 250 -s 1003
```

Dopad na server se ale velmi liší od ostatních útoků. Na obrázku 10 vidíme průběh útoku na server. I přes počáteční nízké zatížení procesoru dochází k postupnému alokování dalších prostředků. Překvapivý je také poměrně dlouhý čas potřebný k zotavení serveru. Teprve po 35 sekundách od ukončení útoku se server vrátil k hodnotám zatížení před útokem.

Obranou je opět pravidlo pro *Snort* aplikující se prostřednictvím *SnortSam* do *IPtables*.

```
alert udp $EXTERNAL_NET any -> $SIP_PROXY $SIP_PORT (msg:"SIP
    DoS attempt(optionsflood)"; content:"OPTIONS sip";
    detection_filter:track by_src, count 50, seconds 5;
    classtype:misc-attack; sid:1000012; rev:1; fwsam:src, 5min;)
alert tcp $EXTERNAL_NET any -> $SIP_PROXY $SIP_PORT (msg:"SIP
    DoS attempt(optionsflood)"; content:"OPTIONS sip";
    detection_filter:track by_src, count 50, seconds 5;
    classtype:misc-attack; sid:1000013; rev:1; fwsam:src, 5min;)
```



Obrázek 10: Zatížení procesoru SIP serveru během útoku OPTIONS flood

Zajímavý je tentokrát i průběh zatížení při aktivní obraně (vyznačen zeleně v grafu na obr. 10). Počáteční velké zatížení způsobilo zřejmě prodlení v aktivování pravidla do *IP-tables*. Během této doby však procesor vytěžovalo více nástrojů – proto i větší zatížení v grafu. Špička znatelná v době mezi 40 a 50 sekundou byla způsobena aplikací *Snort* a pravděpodobně se jedná o následek počátečního zasažení serveru před aktivací obraných mechanismů.

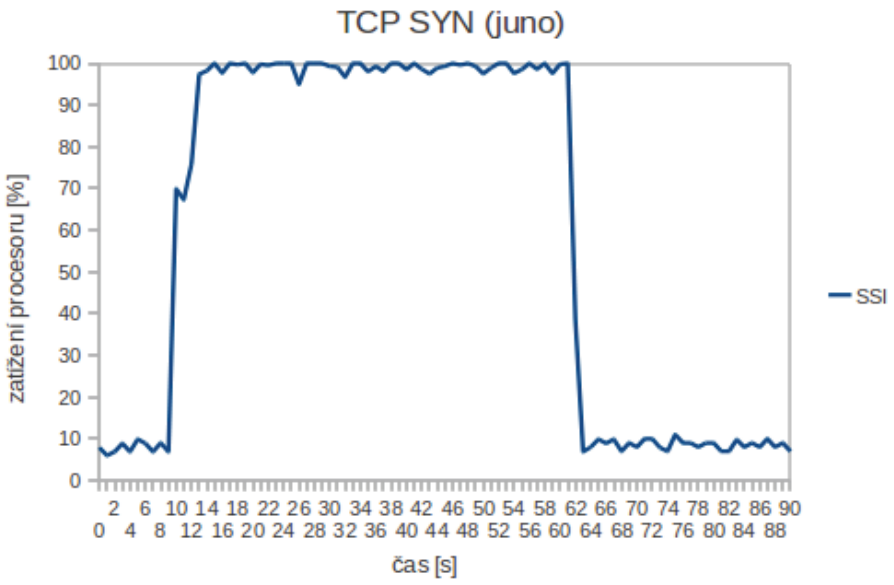
I přes to, že útok nevyčerpal úplně všechny zdroje serveru, bylo obtížné navázat hovor či zaregistrovat zařízení díky velkým prodlevám. Docházelo také k nemožnosti provést hovor nebo registraci, obdobně jako u *REGISTER* či *INVITE flood* útoku. Při vyšší hodnotě zasílaných zpráv dochází k rychlejšímu vyčerpání zdrojů. Použitím uvedené obrany dosáhneme stavu, kdy není legitimní komunikace zasáhnuta.

6.5 Zahlcení linky nástrojem udpflood

Narozdíl od všech předcházejících útoků používajích poškozené či zmanipulované SIP zprávy, nástroj *udpflood* se soustředí pouze na zahlcování cíle UDP pakety, které obsahují sekvenci od 1 do 9 následované nulami. Takto vytvořený paket o celkové velikosti 1400 bajtů následně zasílá na uvedenou cílovou adresu. Nástroj disponuje i možností podvržení zdrojové adresy, což ho činní ještě účinnějším (nebezpečnějším).

```
./udpflood 192.168.0.123 192.168.0.10 1234 5060 1000000
```

Syntaxe příkazu: zdrojová IP, cílová IP, zdrojový port, cílový port a množství paketů k odeslání.



Obrázek 11: Zatížení procesoru SIP serveru během útoku nástrojem juno

Při útoku na server dochází k zahlcení linky, dopad na procesor je minimální (Asterisk vytíží procesor v průměru o 7%). Linka k serveru je ale natolik zahlcena pakety, že není možné vytvářet nová spojení, registrovat zařízení na SIP server, apod. Jakákoliv interakce s serverem není možná. Obrana pomocí pravidla:

```
alert udp $EXTERNAL_NET any -> $SIP_PROXY $SIP_PORT (msg:"SIP
    DoS attempt(udpflood)"; detection_filter:track by_src,
    count 250, seconds 1; classtype:misc-attack; sid:1000006;
    rev:1; fwsam:src, 5min;)
```

Toto pravidlo zajistí blokování nežádoucích zpráv na rozhraní u SIP serveru. Zahlcena je však celá linka k serveru, účinek pravidla tedy mizí. Navíc Snort analyzující příchozí pakety odebírá zdroje počítače (podobně jako u *Asterisk*u se jedná o cca 7% výkonu).

Po útoku se server během okamžiku vrací do provozuschopného stavu, žádné zdlouhavé vracení k původním hodnotám zatížení před útokem nebylo pozorováno. Komunikace se serverem není možná ani při aktivní obraně.

6.6 TCP SYN flood útoky

Posledním typem provedených útoků proti SIP proxy serveru bylo zasažení zahlcením TCP SYN pakety. Toho bylo dosaženo nástroji *flood2* a variacemi nástroje *juno*(viz. kapitola 5.4.5). Konkrétně pak pomocí příkazů:

```
./flood2 192.168.0.10 5060
./juno 192.168.0.10 5060
```

```
./ranportjuno 192.168.0.123 192.168.0.10 5060
./singlejuno 192.168.0.123 1234 192.168.0.10 5060
```

Typicky programům předáváme IP adresu a port cíle, případně s IP adresou zdroje a zdrojového portu. Nástroj *flood2* je od tvůrce poškozen tak, aby nebyl jednoduše zneužitelný. Chyba byla částečně opravena, program byl schopen zasílat pakety, ty ale obsahovaly porušenou TCP hlavičku.

Nástroj *juno* patří mezi nejúčinnější programy pro cílený útok. Kód není třeba opravovat a lehkou úpravou vytvoříme vhodnou variantu nástroje. Program *juno* bez úprav zasílá TCP pakety s příznakem SYN na cílovou adresu a port z náhodně generovaných ip adres a portů (tj. útok na bázi DDoS). Úprava *ranportjuno* zasílá stejné pakety ze zvolené ip adresy a náhodných portů, *singlejuno* pak jen z pevně uvedené ip adresy a portu.

Použití nástroje *flood2* server silně zatěžuje, porušená TCP hlavička však neumožňuje použít příznaku *SYN*. Naproti tomu útoky nástrojem na bázi *juno* server naprosto ochromí. Bez ochranných opatření není možné se serverem komunikovat, a to ani prostřednictvím ssh (což vychází z podstaty SYN flood útoku). V případě TCP SYN útoku pocházejícího pouze z jednoho zdroje lze použít pravidlo pro filtrovaní dle zdrojové IP adresy.

```
alert tcp $EXTERNAL_NET any ->  $SIP_PROXY $SIP_PORT (msg:"SIP
    DoS attempt - tcp syn flood"; detection_filter:track by_src,
    count 20, seconds 2; classtype:attempted-dos; sid:1000009;
    flags:+S; rev:1; fwsam:src, 5min;)
alert tcp $EXTERNAL_NET any ->  $SIP_PROXY $SIP_PORT (msg:"SIP
    DoS attempt - tcp syn flood, spoofed src!";
    detection_filter:track by_dst, count 100, seconds 2;
    classtype:attempted-dos; sid:1000010; flags:+S;
    rev:1; fwsam:dst, 10min;)
```

Po zavedení tohoto protiopatření server naprosto vytěžuje aplikace Snort, která je zahlcena analýzou vzniklých požadavků. Při použití nástroje *juno* v jeho neupravené podobě navíc nelze blokovat pomocí zdroje, nezbývá tedy než blokovat veškerý provoz. Stav procesoru při útoku a spuštěné obraně zobrazuje obrázek 11.

Při útoku bez obrany i s obranou nebylo možné se serverem komunikovat.

6.7 Zhodnocení útoků

Jak dokázaly předcházející kapitoly, SIP proxy je velice zranitelná. I přes malý výkon stroje, na němž běžel SIP server, kombinace aplikací Snort, SnortSam a IPtables, bylo vidět, že z hlediska výkonu není analýza provozu náročná (až na výjimky, viz. kapitola 6.6), zvláště pokud útočník použije k útoku zmanipulované či podvržené SIP zprávy.

Bylo však ukázáno, že proti útokům typu *TCP SYN flood* a *udpflood* se na vlastním serveru bránit nelze. Je možné sice server připojit linkou s větší kapacitou či použít paralerních výpočtů na SIP serveru, situaci to však nereší, pouze tím můžeme zmírnit následky takového útoku.

Typ útoku	Dopad na SIP proxy	Stupeň ohrožení	Komunikace se SIP proxy
REGISTER flood	vysoké zatížení CPU i při nízkém počtu zpráv	velmi vysoký	znemožněna
INVITE flood	vytížení CPU (závislé na množství)	vysoký	znemožněna
ACK, BYE a CANCEL flood	malé vytížení CPU	nízký	velmi omezená, zpomalená
OPTIONS flood	postupné vysoké zatížení CPU i při nízkém počtu zpráv	velmi vysoký	znemožněna
UDP flood	malá zátěž CPU, linka k síti zahlcena	velmi vysoký	znemožněna
TCP SYN flood	vysoká zátěž CPU, neschopná navázat TCP spojení	velmi vysoký	znemožněna

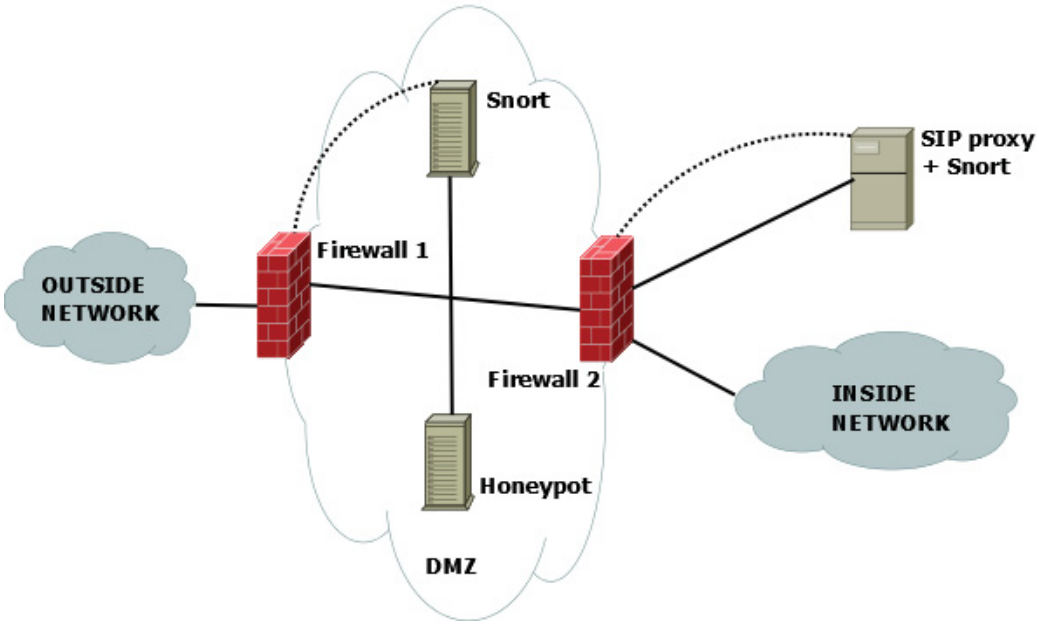
Tabulka 1: Dopad útoků na server

Dopad útoků i velikost ohrožení SIP proxy serveru zobrazuje tabulka 1. V průběhu útoků nikdy nedošlo k přerušení již probíhajících hovorů – hovor byl totiž sestaven a RTP stream probíhal pouze mezi klienty. K navázání hovorů, registraci či obsloužení dalších požadavků nedocházelo, pouze u útoku BYE,CANCEL a ACK flood docházelo k nízkému prodlení ve zpracování zpráv. Pokud by byl počet útočných zpráv větší, docházelo by logicky i k většímu zatížení, ale schéma by se již více podobalo UDP flood útoku. Po ukončení útoku se SIP proxy vždy vrátila do podobného stavu jako před útokem, rozdílná byla pouze doba zotavení.

Řešením je vybudování nové topologie, v níž se SIP server nachází. V této topologii pak zavedeme několik bezpečnostních úprav (obrázek 12). Hlavní změnou je použití tzv. „demilitarizované zóny (DMZ)“.

Demilitarizovaná zóna se nachází mezi dvěma firewally (vnější a vnitřní), jejím účelem je oddělení vnitřní „bezpečné“ části sítě od vnější. Uvnitř zóny se nachází první IPS, založená na stejném principu, jaký byl využit při obraně SIP serveru. Monitoruje tedy provoz mířící do vnitřní části sítě, případné útoky pak blokuje pomocí pravidel pro vnější firewall (na obrázku označen Firewall 1). Uvnitř této zóny se může také nacházet honeypot (například ve formě SIP serveru) či zde mohou být umístěny další servery. Na obou firewallech běží SnortSam agenti.

Vnitřní firewall (označen jako Firewall 2) pak slouží pro přímou ochranu SIP serveru, a to jak před útoky, které mohly proniknout přes vnější firewall, tak i před útoky z vnitřní sítě. Veškerý provoz v síti tak musí před zpracováním projít prověřením na alespoň jednom firewallu. Na SIP serveru běží samozřejmě další instance Snortu, zasahující pravidly pouze do vnitřního firewallu.



Obrázek 12: Návrh bezpečnější topologie

Samozřejmostí by měla být bezpečná vnitřní síť. Případný útok na zahlcení linky by totiž znemožnil použití pro celou vnitřní část, útok by tedy skončil úspěšně, i kdyby byl nyní zastaven na úrovni vnitřního firewallu a vnější část by mohla se SIP serverem komunikovat. Proto by základem této topologie měla být právě bezpečná vnitřní sít, používající metod *DHCP spoofing*u a *ARP inspection* znemožňující útočníkovi podvržení IP adresy. Dalším vhodným krokem je i umístění VoIP provozu do separátní *VLAN*.

V případě útoku s podvrženými adresami z vnější sítě bude útok zastaven na úrovni vnějšího firewallu. Útok se tak vnitřní části nemusí dotknout. Problém s odchozím voláním by mohl být řešen prostřednictvím *WHITELIST*u na vnějším firewallu. Také útok zahlcující linku zůstane zastaven na hranici u vnějšího firewallu.

Možným vylepšením, které by chránilo síť i proti dalším vynalézavým způsobům útoků na SIP server pomocí zahlcování pakety by mohlo zabránit použití *QoS* mechanismů právě v demilitarizovaném pásmu na provoz z vnější části. Tím by tak útočník nebyl schopen nikdy vyčerpat kapacitu uvnitř zabezpečené části sítě.

Inspiraci pro tvorbu dalších bezpečnostních opatření může poskytnout i *honeypot* umístěný v demilitarizované zóně.

7 Závěr

DoS útoky vyřazující z činnosti služby SIP proxy mohou k tomuto účelu používat mnohých přístupů. Práce zmapovala jednotlivé druhy těchto DoS útoků. Účinnost útoků, a tedy zranitelnost SIP proxy vůči těmto hrozbám, byla následně prakticky ověřena a zdokumentována.

Cílem práce bylo navrhnutí zabezpečení SIP proxy proti nejběžnejším DoS útokům na bázi aplikace Snort. Během analýzy byly zvažovány i další metody řešení IPS, nejvhodnějším provedením se ukázalo spojení několika způsobů navzájem. Výsledku bylo dosaženo kombinací Snortu a aplikace SnortSam, aplikované pomocí patche na IDS systém Snort. Program SnortSam, umožňující interakci s IPtables, využívá k detekci pravidel Snort, které jsou obohaceny o další parametry potřebné pro korektní funkci SnortSam. Blokování nežádoucího provozu zajišťuje open–source firewall IPtables. Doba, po kterou omezení platí, je ohraničena časovým limitem. Falešné poplachy tedy budou blokovány, stav se ale po uplynutí nastavené doby resetuje. Pokud útok trvá i během blokování, dochází k průběžnému navyšování doby blokace tohoto provozu.

Nevýhodou zvoleného řešení zůstává prodlení mezi detekcí útoku a následného zablokování. Tento interval je patrný na uvedených grafech zobrazujících průběh útoků (obr. 8 a 10). V případě komunikace s firewallem umístěném na jiném fyzickém zařízení toto prodlení ještě více naroste. Pokud by útočník dokázal odstavit detekční systém Snort, dojde ke zhroucení celého bezpečnostního mechanismu. Pro eliminování této hrozby zapojíme IDS/IPS systém pomocí *Stealth interface*, kdy dochází pouze k přijímaní paketů bez možnosti odesílání (využije se upraveného síťového kabelu). Druhou možností je využití switche podporujícího zrcadlení provozu v síti na určitý port (tím je k síti připojen i Snort).

Vůči SIP proxy serveru byla na základě provedené analýzy bezpečnostních rizik provedena řada útoků s rozdílným dopadem. Různorodé byly také cíle těchto útoků, tj. zaznamenány jsou útoky na paměť serveru, CPU i zahlcení linky. Každý z těchto útoků byl detailně popsán, popis se skládá nejen z postupu provedení útoku, ale také detekčního pravidla použitého pro obranu. Shrnutí všech útoků obsahuje kapitola 6.7 Práce tak může sloužit jako základ pro penetrační testování nebo vytváření dokonalejšího IPS systému. DoS útoky totiž nepatří mezi jediné hrozby týkající se VoIP. Jako nejúčinnější hrozby v rámci DoS se však řadí útoky *REGISTER* a *OPTIONS flood*, kde již nízké hodnoty zaslaných paketů během sekundy způsobují výrazné zvýšení zatížení procesoru. Vysoce nebezpečné zůstavají i útoky *UDP flood* a narušení 3–way handshake metody užitím *TCP SYN flood*.

Vzhledem k dopadu některých útoků na server obsahuje práce i návrh vytvoření odolnější síťové topologie. Důvodem byl zvláště účinek útoku zahlcujícím SIP proxy UDP pakety, vyčerpávající tak přenosovou kapacitu linky. Pro vyšší bezpečnost v síti zmiňujeme i další metody a mechanismy zabezpečení. Jejich aplikováním eliminujeme nejenom vlastní bezpečnostní díry, rostou také požadavky na znalosti případného útočníka a snižuje se tedy okruh potencionálních hackerů. Obecně nelze žádný systém považovat za bezpečný, neplikování žádných protiopatření pak připomíná ruskou ruletu,

kde v nejlepším případě dochází ke ztrátě důvěryhodnosti a poškození dobrého jména provozovatele.

Díky jednoduchosti, s níž můžeme provést DoS útok proti SIP proxy, patří mezi hackery k nejoblíbenějším útokům. Musíme si uvědomit, že zájem hackerů o danou službu se bude jejím rostoucím nasazováním pouze zvětšovat. Použitím posledních bezpečnostních mechanismů jakými jsou šifrování, autentizace a kontrola integrity zpráv docílíme větší bezpečnosti komunikace. Nesmíme však zapomenout chránit proti možným útokům i SIP proxy server – uzavřením otevřených portů, které nevyužíváme, udržováním operačního systému v aktualizovaném stavu.

Následky případného úspěšného útoku zmírní pouze výkonné servery využívající paralerních výpočtů, připojení do sítě pomocí linek s vyšší kapacitou (použití *Etherchannel*ů, optických vedení), load balancing mezi více SIP proxy servery, adaptivní firewally nové generace, ... Takováto opatření ale značně zvyšují cenu konečného řešení, vhodného zejména pro velké korporace. Prací navrhnuté řešení umožní implementaci alespoň základní obranné linie vhodné pro použití uvnitř malých a středních firem nebo detašovaných pracovišť.

Přínosem práce je shrnutí používaných DoS útoků a vzájemné porovnání efektivity těch nejnebezpečnějších z hlediska VoIP. Vzhledem k povaze útoků je navrhnuta změna topologie i postup nasazení IPS systému řešeného kombinací Snort, SnortSam a IPtables.

Bc. Jakub Šafařík

8 Reference

[1] JOHNSTON, A. B.; *SIP: Understanding the Session Initiation Protocol*. London: Artech house, 2009. 427 s.

[2] STEFFEN ,A.; KAUFMAN ,D.; STRICKER ,A.; *SIP Security*. Winterthur: Gesellschaft für Informatics, 2004. 14 s.

[3] REHMAN, Rafeeq Ur; *Intrusion Detection Systems with Snort*. New Jersey: Prentice Hall PTR, 2003.

[4] SISALEM D., KUTHAN J., ELHERT T. S., FRAUNHOFER F., *Denial od Service Attacks Targeting SIP VoIP Infrastructure: Attack Scenarios and Prevention Mechanisms*. IEEE Network, 2006. 275 s.

[5] SISALEM ,D.; FLOROIU ,J.; KUTHAN ,J.; ABEND ,U.; SCHULZRINNE, H.; *SIP SECURITY*. Wiley, 2009. 350 s.

[6] ENDLER ,D.; COLLIER ,M.; *Hacking Exposed VoIP*. McGraw–Hill Osborne Media, 2009. 539 s.

[7] VOZŇÁK, Miroslav; *Voice over IP*. Ostrava: VŠB – Technická univerzita Ostrava, 2009.

[8] SOURCEFIRE; *SNORT Users Manual*. Sourcefire, 2011. 199 s.

[9] ELHERT ,S.;ZHANG ,G.; GENEIATAKIS ,D.; KAMBOURAKIS ,G.; DAGIUKLAS ,M.; SISALEM ,D.; *Two layer Denial of Service prevention on SIP VoIP infrastructures*. Computer Communications, Elsevier B.V., 2008. 2443–2456.

[10] VOZŇÁK ,M.; ŘEZÁČ ,F.; *SIP Threats Detection System*. Portugal: University of Algarve, 2010. 119–124.

[11] CISCO SYSTEMS; *Denial of Service Protection*. Cisco IOS Software Configuration Guide, Cisco systems, 2009. Chapter 49, 20 s.

[12] VOZŇÁK ,M.; ŘEZÁČ ,F.; *SIP Penetration Test System*. Prague: CESNET: Technical Report 10/2010, 2010.

[13] VOZŇÁK ,M.; ŘEZÁČ ,F.; *Threats to Voice over IP Communications Systems* WSEAS TRANSACTIONS on COMMUNICATIONS, 2010. Volume 9, 1348-1358

[14] VOZŇÁK ,M.; ŘEZÁČ ,F.; *VoIP SPAM and a Defence against this Type of Threat*. Corfu: The 14th WSEAS International Conference on COMMUNICATIONS, 2010. 172–177.

[15] FreeSWITCH; *SIP TLS*. FreeSWITCH, 2009, `http://wiki.freeswitch.org/wiki/SIP_TLS`

[16] ČERVENKA M.; *Co nového přináší Asterisk 1.6*. root, 2008, `http://www.root.cz/clanky/co-noveho-prinasi-asterisk-1-6/`

[17] ZANDL P.; *Špatně zabezpečené VoIP stálo hoteliérku miliony. A nejenom ji!*. Lupa, 2009, http://www.lupa.cz/clanky/spatne-zabezpecene-voip-stalo-hotelierku-mil/