Masarykova univerzita
Fakulta informatiky

Rozpoznanie užívateľa na základe informácií o HTTP komunikácií

Diplomová práca

Matej Majdiš

Brno, jar 2017

Namiesto tejto stránky vložte kópiu oficiálneho podpísaného zadania práce a prehlásenie autora školského diela.

Prehlásenie

Prehlasujem, že táto diplomová práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Matej Majdiš

Vedúci práce: doc. RNDr. Vlastislav Dohnal Ph.D. title

Poďakovanie

TODO

Zhrnutie

TODO

Kľúčové slová

keyword1, keyword2, …

Obsah

Zoznam tabuliek

Zoznam obrázkov

1 Úvod

Problematika jednoznačnej identifikácie používateľa je dnes veľmi dôležitou a riešenou témou. Jedným z hlavných dôvodov je fakt, že väčšina dnešných existujúcich, prípadne novo vznikajúcich systémov a aplikácií je nejakým spôsobom zapojená do Internetu. Zároveň zaznamenávame nárast aplikácií, ktoré poskytujú užívateľom webové rozhranie a ústup takzvaných desktopových aplikácií.

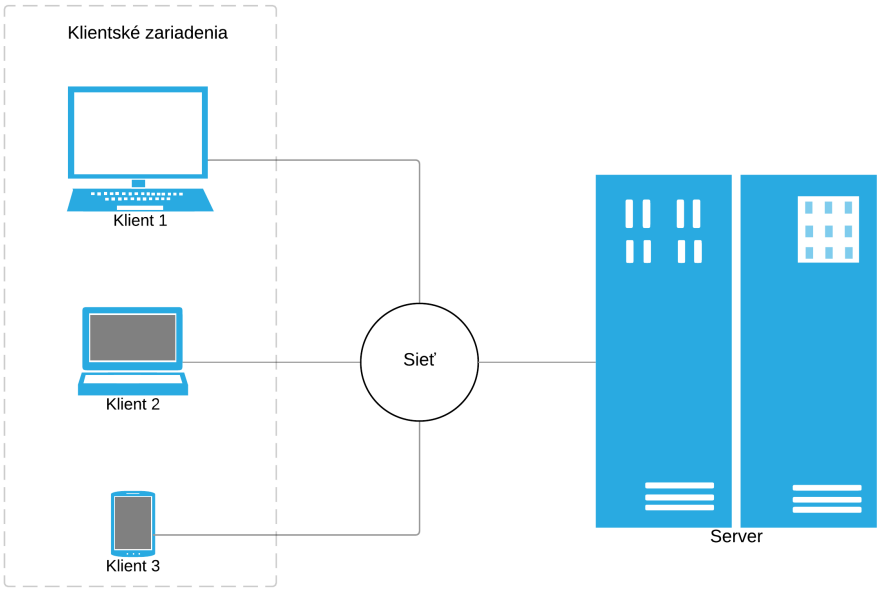
Obr. 1.1: Vizualizácia pomeru počtu užívateľov webových a desktopových aplikácií v čase, zdroj: vlastné spracovanie

Z tohto vyplýva potreba rozoznania a identifikácie používateľov, ktorý s danou aplikáciou interagujú. Existuje niekoľko rôznych prístupov k identifikácií, od mapovania IP adries sieťovej vrstvy až po aplikačnú správu užívateľských účtov. Podrobne sa nimi zaoberá kapitola 4. Cieľom tejto práce je vytvoriť unikátny identifikátor na základe informácií dostupných z *HTTP* protokolu. Pred zostavením samotného algoritmu je preto dôležité popísať niektoré kľúčové oblasti a postupy.

Nasledujúce kapitoly sa preto budú stručne zaoberať fungovaním aplikácií typu klient-server, modelom sieťových vrstiev či útokmi typu *Denial of Service*. Ďalej v práci popíšem spomínané existujúce prístupy a vlastný návrh algoritmu identifikácie užívateľa.

1.1 Aplikácie typu Klient-Server

S pokračujúcim vývojom nových technológií sa Web stáva stále väčšou súčasťou našich životov. Web taktiež už nie je limitovaný prehliadaním na počítačoch. Musí sa prispôsobovať rôznym novým technológiám, ako sú napríklad mobilné, či iné zriadenia. Najčastejšie používaným modelom komunikácie pre architektúru webových aplikácií je tzv. Klient-Server model. Základnou myšlienkou tohto modelu je zaslanie požiadavku (*requestu*) klientom na server, ktorý vystupuje ako poskytovateľ služby.



Obr. 1.2: Schéma znázorňuje základnú architektúru modelu Klient-Server, zdroj: vlastné spracovanie

1.1.1 Klient-Server model

Pretože Klient-Server model je používaný rôznymi typmi aplikácií bolo nutné použiť štandardizované protokoly, na základe ktorý ch bude možné komunikovať. Základné používané protokoly sú: *FTP*

(File Transfer Protocol), *Simple Mail Transfer Protocol (SMTP)* a *Hypertext Transfer Protocol (HTTP)*. Bližšie sieťové vrstvy a jednotlivé protokoly popisuje kapitola 2.

1.1.2 Architektúra

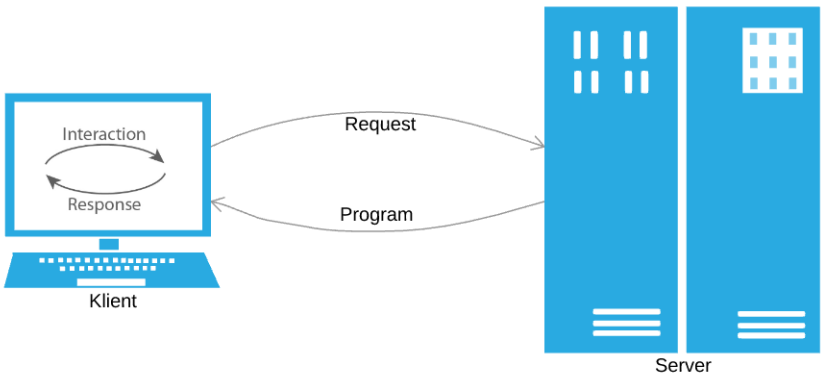
Architektúra modelu Klient-Server sa vo všeobecnosti typicky skladá z troch častí:

- Aplikačný server
- Databázový server
- Zariadenie klienta

Zároveň Existujú dva základné typy architektúr:

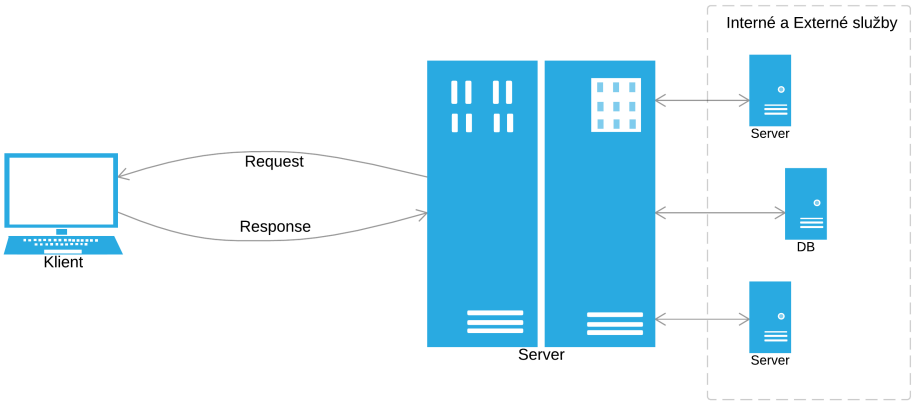
- 2-stupňová *(2-tier)*
- 3-stupňová *(3-tier)*

2-tier architektúra zahrna len zariadenie klienta a databázový server. U tohoto typu architektúry je aplikácia spustená na zariadení klienta, ktoré sa následne pripája priamo na server. Zariadenie tak obsluhuje zároveň *business* logiku aj zobrazovanie aplikácie. Inak tento typ architektúry nazývame aj tučný klient(*thick client*).



Obr. 1.3: Grafické znázornenie a popis priebehu komunikácie 2-tier architektúry, zdroj: vlastné spracovanie

3-tier architektúra, ktorou sa budem zaoberať v tejto práci sa od *2-tier* líši najmätým, že okrem zariadenia klienta a databázového servera zahŕňa aj aplikačný server. Tento je následne používaný na obsluhu *business* logiky aplikácie a komunikáciu s databázou, pričom zariadenie klienta slúž len na zobrazovanie. Iný názov pre takýto typ architektúry je tenký klient (*thincient*).



Obr. 1.4: Grafické znázornenie a popis priebehu komunikácie 3-tier architektúry, zdroj: vlastné spracovanie

Nasledujúce kapitoly tejto práce sa budú zaoberať jedným z najdôležitejších problémov Webových aplikácií, ktorým je identifikácia užívateľa. Najskôr kapitola 2 popisuje jednotlivé sieťové vrstvy a protokoly, ktorých informácie je možné použiť ka následnej identifikácií. Ďalšou časťou je zhrnutie existujúcich prístupov k rozoznaniu užívateľov v kapitole 4 a popis útokov typu DOS v kapitole 3. Najdôležitejšou časťou je však samozrejme kapitola 5, ktorá popisuje návrh samotného algoritmu identifikátoru.

2 Sieťové vrstvy

Nasledujúca kapitola stručne popisuje jednotlivé sieťové vrstvy TCP/IP modelu a ich protokoly. Zameriava sa na štruktúry a informácie, ktorých pochopenie je dôležité pre ďalšie zložky teoretickej a praktickej časti práce. Ide najmä o konkrétne informácie jednotlivých protokolov skutočne použité v implementácií identifikačného algoritmu.

Obr. 2.1: Vizualizácia vrstiev TCP/IP modelu, jeho najpoužívanejších protokolov a prenosových štruktúr, zdroj: vlastné spracovanie

2.1 Aplikačná vrstva

Na vrchole hierarchie je Aplikačná vrstva abstrakciou, ktorá špecifikuje konkrétne protokoly a metódy používané hostiteľskými uzlami v sieti. Táto vrstva je definovaná jednak v TCP/IP ale aj OSI *(Open Systems Interconnection)* modele sieťovej komunikácie.

Táto práca narába s modelom TCP/IP, v ktorom aplikačná vrstva definuje práve protokoly a metódy rozhraní pre komunikáciu medzi jednotlivými procesmi spájaných strán v sieti. Samotná aplikačná vrstva však len štandardizuje formu komunikácie, pri čom ustálenie uceleného dátového spojenia a správu prenosu dát u aplikácií typu klient-server a *peer-to-peer* ponecháva na protokoloch nasledujúcej -

transportnej vrstvy. Keďže aplikačná vrstva nijakým spôsobom nepopisuje a nešpecifikuje konkrétne pravidlá pre formu prenášaných dát, aplikácie samotné musia obsahovať logiku, ktorá zabezpečí, že obe komunikujúce strany budú v tomto ohľade navzájom kompatibilné.

Aplikačná vrstva definuje veľké množstvo protokolov, ako napríklad:

- HTTP/HTTPS
- TLS/SSL poskytujúci zabezpečenú vzdialenú komunikáciu po sieti
- FTP využívaný na prenos súborov
- SMTP určený pre emailovú komunikáciu
- DNS realizujúci systém hierarchie doménovým mien, a iné...

Pri tvorbe identifikátoru sa však budeme venovať najmä informáciám z protokolov rodiny HTTP, konkrétne teda hlavičkám a atribútom HTTP a HTTPS.

2.1.1 HTTP

HTTP *(Hypertext Transfer Protocol)* je distribuovaný protokol určený pre spoluprácu a prenos informácií medzi jednotlivými systémami určenými na prácu s hyper-médiami. Tento protokol je bez-stavový a vo všeobecnosti použiteľný nielen pre prenos hypertextu, ale aj správu DNS serverov, prípadne zložitejších objektov a systémov, v ktorých uplatní rozsiahlu škálu svojich hlavičiek a chybových hlášok. Charakteristickým znakom tohto protokolu je aj možnosť voľby reprezentácie dát, čo dáva systémom veľkú nezávislosť na prenášanom formáte.

HTTP funguje na princípe požiadavku a odpovede *(request - respone)* založenom na fungovaní samotného klient-server modelu. Klientom môže byť napríklad webový prehliadač, serverom zasa aplikácia spustená na počítači hostiteľského uzla. V tomto prípade teda webový prehliadač zašle správu vo forme HTTP požiadavku na server. V ideálnom prípade server tento požiadavok *(request)* spracuje a vygeneruje odpoveď (napríklad HTML stránku), ktorú vráti klientovi ako správu

vo formáte HTTP odpovede *(response)*. Odpoveď pre klienta vždy obsahuje takzvaný *(status)*, ktorý informuje o dokončení požiadavku a prípadne samotné telo so správou odpovede.

Celá relácia HTTP spojenia medzi dvoma uzlami je teda sekvenciou niekoľkých takýchto *(request - respone)* transakcií. Samotný prenos dát však nie je realizovaný na úrovni HTTP protokolu, ale prostredníctvom protokolu TCP na úrovni transportnej vrstvy. HTTP Klient iniciuje ustálenie TCP spojenia pre špecifický port serveru (typickými portmi sú 80, 443 alebo 8080). HTTP server počúva na danom porte a čaká na správu požiadavku klienta. Po prijatí správy ju už server štandardne spracuje a vráti odpoveď.

Klient a server komunikujú prostredníctvom zasielanie plain-textových správ. Požiadavok pozostáva z nasledujúcich informácií:

- Definícia požiadavku
- HTTP hlavičky (napríklad: *Accept-Language: en*)
- Prázdny riadok
- Prípadné telo správy

Takisto správa odpovede má definovaný formát podobný formátu požiadavku, ktorý sa skladá z nasledujúcich položiek:

- Status spracovania požiadavku a dôvod
- HTTP hlavičky odpovede (napríklad: *Content-Type: text/html*)
- Prázdny riadok
- Prípadné telo správy odpovede

Prvý riadok definície a každá z hlavičiek musia byť zakončené znakmi <*CR*><*LF*> (znak pre *carriage return* nasledovaný znakom *line feed*). Prázdny riadok musí zároveň pozostávať výlučne zo znakov <*CR*><*LF*>. Z pohľadu identifikácie budú neskôr veľmi dôležité práve HTTP hlavičky. U HTTP verzie 1.1 je jedinou povinnou hlavičkou *Host* definujúci server, na ktorý bol požiadavok odoslaný.

Obr. 2.2: Príklad štruktúry HTTP požiadavku a odpovede

2.2 Transportná vrstva

2.2.1 Protokol TCP

K samotným dátam aplikačnej vrstvy preto TCP pripája ku každému paketu aj takazvanú TCP halvičku *TCP Header*, ktorá ho dopĺňa o potrebné informácie.

— TODO obrázok TCP header —

— TODO obrázok 3-way handshake —

2.2.2 Protokol UDP

— TODO obrázok UDP header —

2.3 Sieťová vrstva

2.3.1 Protokol IP

— TODO obrázok IP header —

2.4 Vrstva sieťového rozhrania

Vrstva sieťového rozhrania funguje v TCP/IP modeli na najnižšej úrovni. Táto vrstva popisuje samotnú sieťovú architektúru a jej komunikáciu na úrovni fyzického pripojenia. Ide o skupinu metód a komunikačných protokolov, ktoré teda operujú na fyzickom spojení dvoch uzlov.

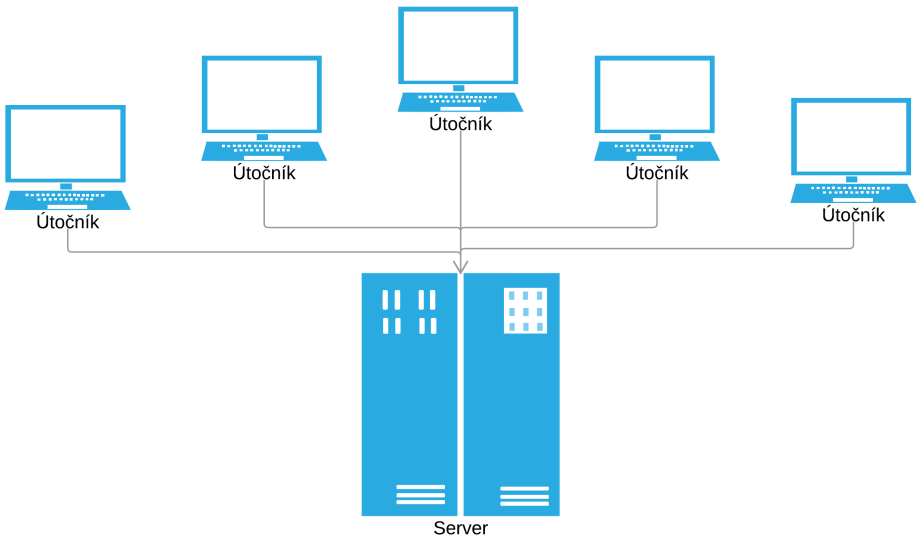
Niektorými z jej dôležitých protokolov sú: ARP *(Address Resolution Protocol)*, RARP *(Reverse Address Resolution Protocol)*, NDP *(Neighbor Discovery Protocol)*, OSPF *(Open Shortest Path First)* a iné.

Informácie týchto protokolov však operujú na príliš nízkej úrovni sieťovej infraštruktúry, preto ich nebude možné využiť pri identifikácií. V prehľade je teda vrstva sieťového rozhrania uvedené len pre úplnosť.

3 Útoky typu *Denial of Service*

Jedným z hlavných dôvodov identifikácie užívateľov je prevencia proti útokom. Medzi najznámejšie z útokov patrí tzv. *Denial of Service*, ďalej len *DoS*.

Vo všeobecnosti je za *DoS* útok považovaná snaha utočníka zabrániť oprávneným užívateľom v prístupe k informáciám, prípadne službám poskytovateľa. Snahou útočníka je znefunkčniť pripojenie neustálym narúšaním služby serveru, prípadne sieťovej infraštruktúry, v dôsledku čoho môže dôjsť k čiastočnej, či úplnej strate internetového pripojenia hostiteľa.



Obr. 3.1: Schéma DoS útoku, zdroj: vlastné spracovanie

U distribuovaného *DoS* útoku môže útočník použiť k útoku na server počítače klientov. Nad týmito zariadeniami je možné prevziať kontrolu využitím bezpečnostných chýb alebo nedostatkov. Takto je následne možné donútiť počítač posielať obrovské množstvo dát na webové servery, prípadne odosielanie nevyžiadanej pošty na konkrétne e-mailové adresy. Útok sa nazýva "distribuovaný", pretože útočník používa viac zariadení na začatie útoku *denial-of-service*.

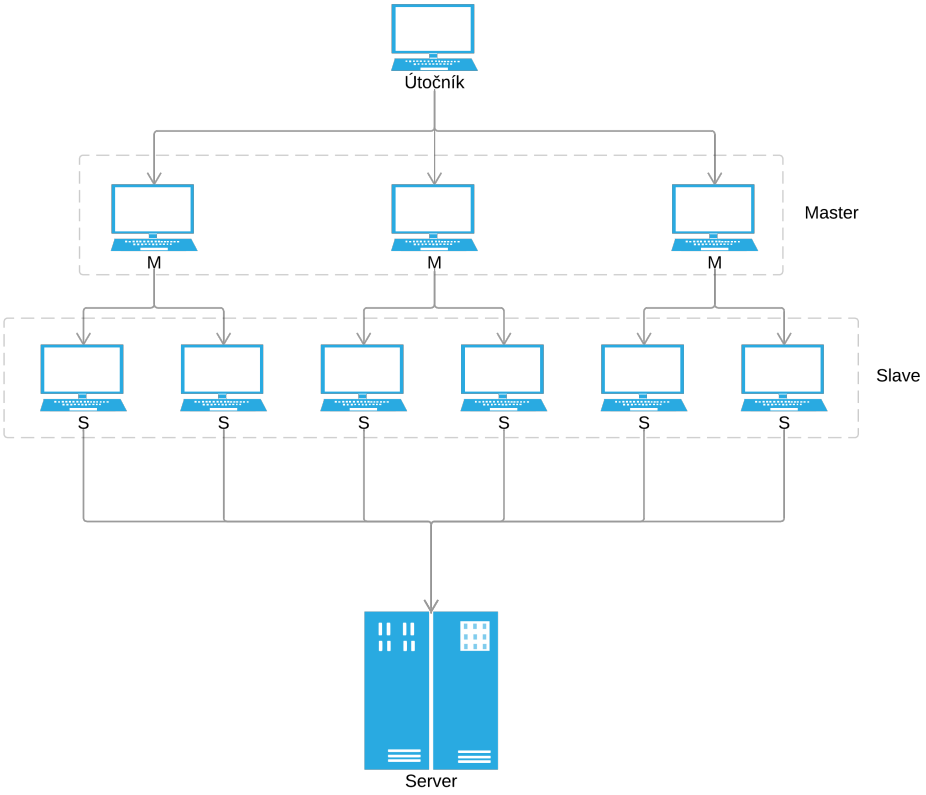
DoS útok je podobný veľkej skupine ľudí, ktorá sa zhromažďuje pri vstupe do obchodu a bráni vo vstupe skutočným zákazníkom, ktorých záujmom sú reálne služby. Útočníci vykonávajúci tieto útoky sa často zameriavajú na webové služby a servery, ktoré sú poskytované vysoko profitujúcimi inštitúciami, ako sú napríklad banky, prípadne platobné brány.

3.1 Základne typy a techniky

Nasledujúce odseky popisujú najpoužívanejšie typy a techniky vykonávania *DoS* útokov a identifikujú prostriedky, ktorými je im možné zabrániť.

3.1.1 Distribuované DoS útoky

O distribuovaných *DoS* útokoch hovoríme v prípade, že viaceré zariadenia zaplavia celú šírku pásma prípadne zdrojov cieľového systému, ktorým je zvyčajne jeden, alebo viacero serverov. Takýto útok je často dôsledkom použitia rôznych systémov a zariadení (napríklad *botnetu*), ktoré sa snažia vyťažiť cieľový systém. *Botnet* je rozsiahla virtuálna sieť umelých (*zombie*) počítačov, ktorých cieľom je prijímať príkazy bez vedomia majiteľa. Keď cieľový systém spotrebuje všetky voľné spojenia, ďalšie (nové) už nie je možné nadviazať. Hlavné výhody útočníka pri využití Distribuovaného *DoS* útoku spočívajú v skutočnostiach, že viaceré zariadenia dokážu generovať väčšiu záťaž ako jedno, pričom použitie množstva systémov zabezpečuje omnoho ťažšiu detekciu útočníka a správanie sa každého z týchto zariadení je menej pozorovateľné čo sťažuje obranu voči útočníkovi. Tieto výhody taktiež spôsobujú vývoj obranných mechanizmov. Na strane cieľového serveru už nebude stačiť jednoduché zvýšenie šírky pásma nad hranicu momentálnej veľkosti útoku, pretože útočník môže napríklad zvýšiť počet zapojených zariadení čím by taktiež spôsobil zaťaženie a výpadok systému.



Obr. 3.2: Schéma rozloženia DDoS útoku, zdroj: vlastné spracovanie

3.1.2 Sémantické DoS útoky

Sémantické útoky využívajú špecifickú funkcionalitu, alebo implementačnú chybu aplikácie, prípadne protokolu zariadenia obete na zneužitie určitého množstva jeho zdrojov. Napríklad v prípade *TCP SYN* útoku je touto zneužitou funkcionalitou alokácia značného množstva priestoru v zozname pripojení ihneď po potvrdení *TCP SYN requestu*. Útočník otvorí viaceré spojenia, ktoré nikdy neuzavrie, čím zahlcuje server.

Pri *CGI* útoku je cieľom útočníka takýmto spôsobom zahltiť procesor viacerými *CGI requestami*.

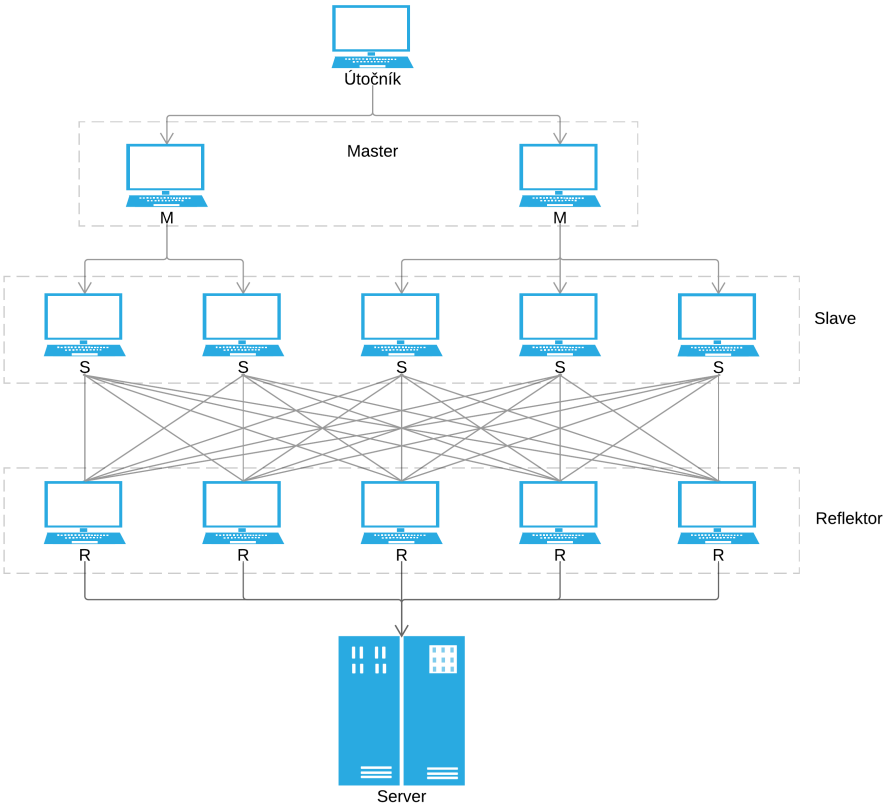
Jedným z obzvlášť nebezpečných útokov je *NAPTHA* útok, ktorý sa zameriava na *TCP* protokol. Inicializuje mnoho *TCP* spojení, ktoré zaplnia zdroje serveru.

3.1.3 Brute-force útoky

TODO
Source ref: https://www.eecis.udel.edu/ sunshine/publications/ccr.pdf

3.1.4 Reflexia a zosilnenie

TODO



Obr. 3.3: Schéma rozloženia DRDoS útoku, zdroj: vlastné spracovanie

3.1.5 HTTP POST DoS útoky

TODO

3.2 DoS nástroje a DoS as a Service

Typickou metódou prenosu mechanizmov *DDoS* útokov je malvér. Jedným z príkladov bol takzvaný *MyDoom*. Ide o *DoS* mechanizmus, ktorý sa spúšťal vo predom naplánovanom čase. Tento útok zahŕňal nastavenie hodnoty *IP* adresy cieľového systému pre nasadenie malvéru, pričom pre spustenie útoku nebola potrebná žiadna interakcia s používateľom.

Ďalším spôsobom zneužitia systému pre *DDoS* útok je použitie skrytej časti softvéru tretej strany, ktorý umožní útočníkovi stiahnutie *zombie* agenta, prípadne ho už softvér sám obsahuje. Útočník môže preniknúť do systému taktiež pomocou automatizovaných nástrojov, ktoré zneužívajú chyby v programoch počúvajúcich vzdialené pripojenia. Tento scenár zasahuje primárne systémy, ktoré sa správajú ako webové servery. Typickým príkladom *DDoS* nástroja z tejto oblasti je takzvaný *Stacheldraht*. *Stacheldraht* využíva vrstevnatú štruktúru, v ktorej útočník používa program klienta na pripojenie sa k *handlerom*, ktoré sú zneužívané na prenos príkazov k *zombie* agentom. Agenti následne vykonávajú samotný *DDoS* útok. Agenti sú cez handleri zneužívané útočníkom za pomoci použitia automatizovaných algoritmov na vyhľadávanie zraniteľností v programoch, ktoré prijímajú vzdialené pripojenia. Každý *handler* je schopný kontrolovať až tisíc agentov.

DDoS nástroje ako *Stacheldraht* stále používajú klasické *DoS* metódy zamerané na zosilnenie a podvrhovanie IP adries ako napríklad útok vyťaženia šírky pásma. Ďalšou z možností je zahltenie zdrojov - *SYN Flood* útok. Novšie nástroje používajú na *DoS* útoky taktiež *DNS* servery.

Nástroje ako *MyDoom* môžu byť použité voči ľubovoľnej IP adrese. Menej skúsení útočníci ich používajú k znemožneniu dostupnosti populárnych a známych webových serverov. Naopak sofistikovanejší útočníci používajú tieto nástroje na vydieranie, napríklad voči svojim obchodným protivníkom.

V niektorých prípadoch sa však môže zariadenie stať časťou *DDoS* útoku zámerne - so súhlasom majiteľa. Príkladom je distribuovaný útok *Operation Payback* organizovaný skupinu *Anonymous*.

—

TODO DoSaaS

3.3 DoS - Zhrnutie

TODO

4 Existujúce prístupy k identifikácií

Nasledujúca kapitola sa venuje popisu, porovnaniu a hodnoteniu existujúcich prístupov, ktoré sú momentálne využívané na účely identifikácie používateľa. Ide najmä o techniky využívajúce: protokoly sieťovej vrstvy (najmä IP adresy), monitorovanie TCP komunikácie a vlastné aplikačné identifikátory.

4.1 Využitie sieťovej vrstvy

Jedným z najtypickejších spôsobov identifikácie užívateľa a jeho zriadenia je IP protokol sieťovej vrstvy. Táto technika je veľmi rozšírená napríklad pre účely blokovania prístupu na server u systémových *firewallov* a podobne. TODO - rozšíriť

4.1.1 Internet Protocol (IP)

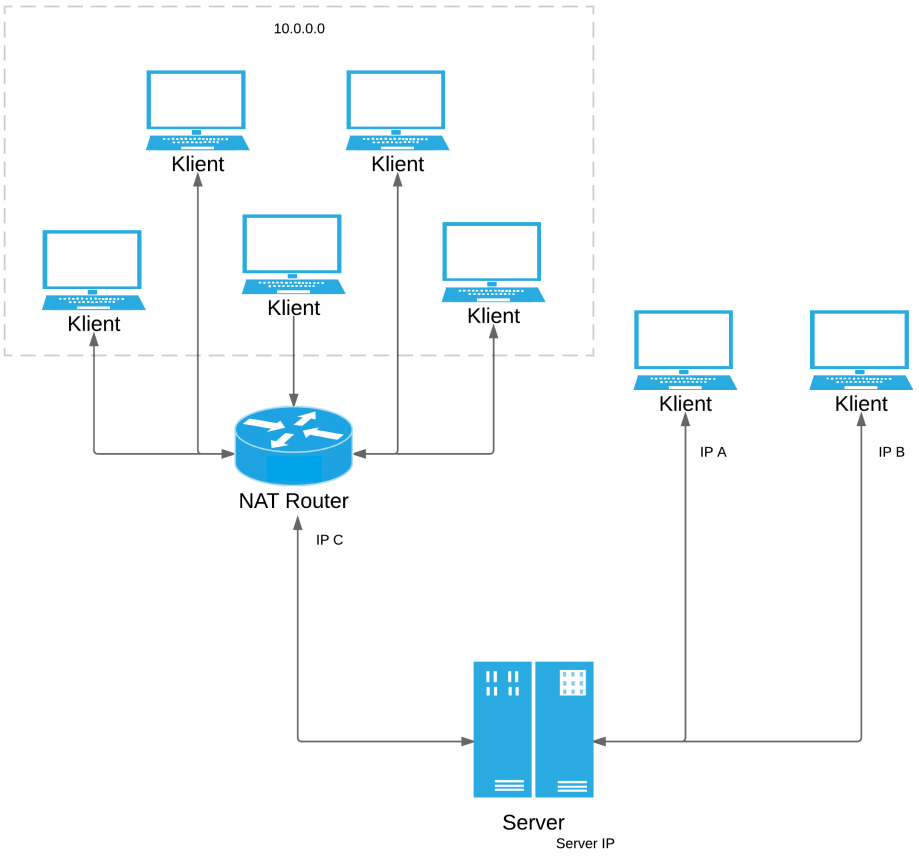
Ako popisuje kapitola 2, Internet Protocol slúži na prenos paketov medzi jednotlivými sieťovými uzlami - zariadeniami, ktoré sú identifikované IP adresami. V ideálnom by bolo každé zariadenie v sieti identifikované jednou statickou IP adresou. Bolo by teda možné užívateľa rozpoznať len na základe množiny jeho adries. Bohužiaľ adresový priestor protokolu IPv4 je obmedzený na 3.7 miliardy verejne dostupných adries [1]. Z toho vyplýva potreba metód ich dynamického prideľovania, proxy serverov a prekladu, čo identifikáciu značne komplikuje.

U dynamického prideľovania adries ide o jednoduché mapovanie pre zariadenia na úrovni smerovača. Identifikácia je teda stále možná, len sťažená o vyšší počet možných adries pre jedno zariadenie.

Preklad (*Network Address Translation* IP adries však identifikáciu prakticky znemožňuje. Vo všeobecnosti ide o techniku prekladu jednej podmnožiny IP adries na inú pomocou zmeny informácie o sieťovej adrese v datagrame IP protokolu. Pôvodne sa táto technika používala pre zjednodušenie presmerovania komunikácie bez nutnosti opätovnej adresácie každého uzlu. V pokročilých implementáciách však dnes

1. Uvedené množstvo zahŕňa všetky možné dostupné IPv4 adresy - približne 4.2 miliardy po odpočítaní rezervovaných adries - 588 miliónov.

NAT predstavuje veľmi populárnu možnosť takzvaného IP maskovania. Maskoavanie IP adries je technika zdieľania jednej verejnej IP adresy celou privátnou podsieťou. Adresný priestor privátnej podsiete, ktorá má byť skrytá je teda vždy mapovaný na verejnú IP adresu smerovača. Táto adresa samotná je taktiež zvyčajne súčasťou väčšej podmnožiny adresného priestoru. Pojem prekladu (*NAT*) IP adries sa dnes už stal synonymom ich maskovania. Paket odoslaný zo zariadenia, ktoré je prekryté *NAT* smerovačom bude teda ako zdrojovú adresu niesť namiesto IP adresy zariadenia z ktorého pochádza hodnotu verejnej IP adresy smerovača. Identifikácia na základe IP adresy preto v tomto prípade stráca význam.



Obr. 4.1: Vizualizácia rozloženia IP adresového priestoru pri použití techniky prekladu *NAT*, zdroj: vlastné spracovanie

4.1.2 Výhody a nevýhody

Hlavnými výhodami použitia IP adries sú jednoduchosť, rýchlosť a flexibilita. Napríklad pre účely znemožnenia prístupu stačí filtrovať len povolené IP adresy, prípadne neumožniť prístup tým zakázaným. Oba spôsoby sú technicky veľmi ľahko implementovateľné.

Ich nevýhodou je najmä rastúci trend použitia proxy serverov a *NAT* prekladu, čo má napríklad pri banovaní za následok znemožnenie prístupu aj užívateľom, u ktorých to nie je potrebné, ale stoja za rovnakou IP adresou.

Vo všeobecnosti je teda identifikácia zariadení používateľa na základe IP adresy možná. V prípade, že sa však zariadenie skrýva za proxy serverom, prípadne sú adresy prekryté *NATom* bude táto technika viesť takisto k prekrytiu jednotlivých užívateľov v rámci podsiete.

4.2 Monitoring TCP

Ako popisuje predchádzajúca kapitola, TCP *(Transmission Control Protocol)* je jedným z protokolov transportnej vrstvy sieťovej hierarchie. Jeho najtypickejším znakom je na rozdiel od UDP garancia spoľahlivého doručenia paketov v presnom poradí. Prijatie každého z paketov musí byť potvrdené príjemcom, inak je paket odoslaný znovu. Poradie jednotlivých paketov je zaručené ich sekvenčnými číslami. Tieto vlastnosti robia z TCP jasnú voľbu pre aplikácie, ktoré vyžadujú nulovú stratovosť dát. Aplikácie typicky zasielajú na TCP vrstvu *stream* dát pomocou takzvaných *stream socketov*, čím sú dáta *streamu* rozdelené na primerane veľké segmenty. K segmentom je zároveň spočítaný kontrolný súčet *(TCP checksum)*, pomocou ktorého je možné určiť či dáta neobsahujú poškodené pakety. Kontrolné súčty však nie sú v žiadnom ohľade kryptograficky zabezpečené. Podrobný popis štruktúry TCP paketu znázorňuje ==obrazok kapitola 2?==

//TODO dokončiť

4.2.1 Vyhody a nevyhody

TODO

4.3 Aplikačné identifikátory

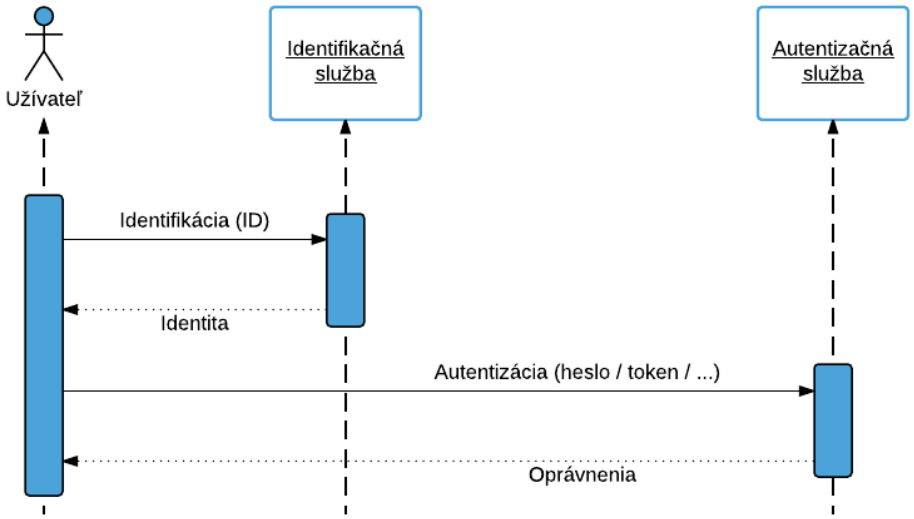
Na aplikačnej úrovni je identifikácia užívateľa oveľa komplexnejším a do istej miery aj abstraktnejším procesom. Z pohľadu systému sú jeho používateľmi typicky ľudia, prípadne procesy iných služieb, ktoré existujú mimo neho. Identifikátorom (ďalej len ID) užívateľa je teda projekcia aktuálneho jednotlivca, prípadne procesu do počítačového systému. Systém používa typicky abstraktný objekt - účet užívateľa, ktorý obsahuje množinu atribútov pre každého jednotlivca, prípadne proces. Tento objekt má jednoznačné a unikátne ID, prípadne meno, ktorým je reprezentovaný v rámci systému. Ďalej môže objekt obsahovať dodatočné atribúty, ktoré ho popisujú. Atribúty môžu, ale nemusia zahŕňať osobné údaje jednotlivca. Odhliadnuc od ID objektu, bezpečný systém typicky priradí každému z používateľov jednoznačný identifikátor (číslo, alebo reťazec), ktorým sa odkazuje na abstraktný objekt reprezentujúci aktuálnu entitu. Tvorba unikátneho abstraktného objektu vo forme užívateľského účtu pre každého jednotlivca, alebo proces komunikujúci so systémom je na aplikačnej úrovni veľmi dôležitá. Tento objekt je následne používaný na identifikáciu užívateľa v rámci celého systému. Zároveň tento objekt slúži ako odkaz pre jednotlivé akcie systému umožňujúce prístup k údajom komunikujúcej entity. ID používateľa sa tak stáva základom pre kontrolu prístupu. Z toho dôvodu je nevyhnutné uchovávať unikátne ID pre každého používateľa, keďže každý z nich môže mať rozličné požiadavky a individuálne zodpovednosti v rámci akcií tohto systému.

Metóda identifikácie systému poskytuje na aplikačnej úrovni jednoznačnú identitu používateľa. Táto identita je typicky reprezentovaná jeho identifikátorom. Systém preto prehľadáva všetky dostupné abstraktné objekty a vráti objekt vyhovujúci privilégiám a atribútom entity s ktorou aktuálne komunikuje. Po úspešnom dokončení tohto procesu je užívateľ jednoznačne identifikovaný.

Po úspešnej identifikácií typicky nasleduje krok validácie získanej identity, vo všeobecnosti nazývaný autentizácia používateľa. Fakt, že užívateľ tvrdí, že je reprezentovaný špecifickým abstraktným objektom nemusí totiž nutne znamenať, že je to pravda. Pre potvrdenie, že aktuálny používateľ môže byť skutočne mapovaný na konkrétny abstraktný objekt, čim mu budú udelené jeho práva a privilégiá musí komunikujúci preukázať svoju identitu systému. Autentizácia je teda

procesom validácie danej poskytnutej identity. Informácie, ktoré predkladá komunikujúca entita sa nazývajú poverenia (*credentials*). Tieto poverenia sa môžu v rôznych systémoch líšiť, prípadne môžu niektoré systémy vyžadovať ich väčšie množstvo. Najčastejšími formami týchto údajov sú: meno, heslo, pin, token, prípadne certifikáty, a iné.

Akonáhle prebehne úspešná autentizácia môže používateľ vykonať akcie, na ktoré ma oprávnenia. Všetky akcie ktoré vykoná sú zviazané s jeho identitou a je ich preto možné dodatočne trasovať.



Obr. 4.2: Grafické znázornenie procesu identifikácie užívateľa na aplikačnej úrovni a jeho následnej autentizácie, zdroj: vlastné spracovanie

4.3.1 Výhody a nevýhody

Identifikátory na aplikačnej úrovni sú veľmi jednoznačné a presné z hľadiska priradenia účtu užívateľovi. Je pomocou nich preto následne možné presne sledovať jeho akcie a manipulovať s privilégiami.

Tieto identifikátory sa však v abstrakcií komunikácie nachádzajú príliš vysoko, preto nie je pomocou nich možné ovplyvniť napríklad DoS útoky popísané v kapitole 3.

4.4 Zhrnutie

Na identifikáciu užívateľa, prípadne jeho zariadenia sú v súčasnosti používané rôzne techniky na rôznych vrstvách sieťovej infraštruktúry. Od IP adries sieťovej vrstvy cez atribúty TCP protokolu až po komplexné aplikačné identifikátory. V praktickej časti tejto práce, ktorá sa zaoberá tvorbou unikátneho identifikátoru, bude použitá kombinácia informácií zo všetkých spomínaných vrstiev (IP adresy, informácie z HTTP a TCP protokolu, ...) tak, aby vznikol čo najpresnejší možný odtlačok aktivity jedného užívateľa.

5 Tvorba unikátneho identifikátoru

//TODO - Úvod

5.1 Možnosti protokolu HTTP

5.2 Možnosti TCP

5.3 Popis Algoritmu

6 Záver

A Príloha

Appendices of thesis.