Fakulta matematiky, fyziky a informatiky
Univerzita Komenského, Bratislava

Experimentálne porovnanie výkonu OpenVPN pri použití rôznych kryptografických algoritmov

2-INF-262 Bezpečnosť IT infraštruktúry
(Seminárna práca)
`https://github.com/laciKE/openvpn_test/`

Ladislav Bačo, Michal Petrucha
Vedúci práce: RNDr. Jaroslav Janáček, PhD.

25. mája 2014 **Bratislava**

1 Úvod

OpenVPN [2] je populárny multiplatformový open-source nástroj na vytváranie virtuálnych privátnych sietí. Na zaručenie dôvernosti a integrity dát tečúcich cez takúto sieť sú použité kryptografické prostriedky, ktoré vyžadujú netriviálne výpočty. V prípade dôvernosti ide o šifrovací algoritmus, na integritu a autenticitu správ využíva OpenVPN funkciu HMAC v kombinácii s voliteľnou hašovacou funkciou.

Cieľom tejto práce je experimentálne určiť vplyv použitých nastavení OpenVPN na priepustnosť vytvorenej siete. Konkrétne ide o použitý šifrovací algoritmus, použitú hašovaciu funkciu a vplyv použitia kompresie na prenesené dáta.

2 Spôsob merania

V tejto sekcii popíšeme metódy použité pri meraní a prostredie, v ktorom boli merania vykonané.

2.1 Nastavenia OpenVPN

OpenVPN poskytuje viacero spôsobov na vytvorenie spojenia medzi koncovými uzlami, medzi ktorými sa vytvára tunel. Tradičný spôsob využíva PKI na overenie autenticity účastníkov a vygenerovanie symetrických kľúčov pre konkrétnu session.

Druhá možnosť je vopred vygenerovať priamo symetrické kľúče a cez bezpečný kanál ich distribuovať medzi účastníkov. Pri tejto metóde odpadá nutnosť vytvárania certifikátov, ale je obmedzená na práve dvoch účastníkov, na rozdiel od PKI, kedy môže jeden server obsluhovať koľkokoľvek klientov.

Keďže v tejto práci využívame iba dvoch účastníkov, pre jednoduchosť sme zvolili statický zdieľaný kľúč podľa [3].

Merali sme vplyv troch parametrov na priepustnosť: šifrovací algoritmus, hašovacia funkcia a použitie kompresie. Pre každú kombináciu týchto parametrov sme vytvorili dvojicu konfiguračných súborov pre OpenVPN podľa šablóny 1.

Zoznam podporovaných šifrovacích algoritmov a hašovacích funkcií je možné získať príkazmi `openvpn --show-ciphers` a `openvpn --show-digests`. Použitá verzia OpenVPN podporuje 58 rôznych šifrovacích algoritmov a 25 hašovacích funkcií, čo vyžadovalo vysokú mieru automatizácie.

2.2 Nástroje a dáta

Keďže jedným z cieľov je zistiť, ako sa prejavuje kompresia, je očakávateľné, že výsledky sa budú líšiť v závislosti od toho, či prenášané dáta sú dobre komprimovateľné, alebo nie. Za

```
dev tun
remote $SERVER_IP
ifconfig $CLIENT_VPN_IP \
         $SERVER_VPN_IP
secret $STATIC_KEY
auth $DIGEST
cipher $CIPHER
comp-lzo [yes|no]
```

```
dev tun
ifconfig $SERVER_VPN_IP \
         $CLIENT_VPN_IP
secret $STATIC_KEY
auth $DIGEST
cipher $CIPHER
comp-lzo [yes|no]
```

Obr. 1: Šablóny konfiguračných súborov pre OpenVPN klienta (vľavo) a server (vpravo).

týmto účelom sme zvolili dva súbory: jeden obsahujúci náhodné dáta a jeden obsahujúci iba nuly.

Pri prvotných testoch sa ukázalo, že systémový pseudonáhodný generátor `/dev/urandom` na testovacích počítačoch stíhal produkovať výstup rýchlosťou najviac okolo 5 MB/s, preto bolo potrebné náhodný súbor vopred vygenerovať. Keďže rýchlosť čítania z tradičných platňových diskov iba zriedkavo dosahuje hodnoty blížiace sa ku 100 MB/s, aby sme vylúčili vplyv rýchlosti čitania z disku, testovacie súbory boli uložené v ramdisku.

Veľkosť súborov bola zvolená na 320 MB.

Na prenos súborov po sieti bol použitý nástroj netcat. Každý súbor bol prenesený raz od klienta na server a raz zo servera ku klientovi, pričom príjemca vždy súbor zahodil (`/dev/null`). Keďže je pomerne náročné spoľahlivo merať čas prenosu na strane príjemcu iba s použitím shellscriptu, merali sme iba čas behu `nc` na strane odosielateľa, pričom na tejto strane `nc` skončilo akonáhle odoslalo posledný byte vstupu.

Na meranie času sme použili príkaz `date '+%s.%N'` tesne pred a tesne po príkaze `nc`, čo je výrazne jednoduchšie, než parsovať výstup príkazu `time`.

2.3 Testovacie prostredie

Na testovanie bola použitá dvojica identických počítačov s procesorom Intel Pentium 4 s taktovacou frekvenciou 2.80GHz, 1GB RAM a operačným systémom Debian GNU/Linux 7.4. Počítače sú spojené 1 GBit sieťou, čo v kombinácii so starším procesorom zaručilo, že prenosová rýchlosť bola pri všetkých testoch obmedzená výpočtovým výkonom a nie kapacitou linky.

Použitá verzia OpenVPN je 2.2.1 a keďže OpenVPN využíva implementácie kryptografických algoritmov z OpenSSL, použitá verzia tejto knižnice je 1.0.1e-2+deb7u6.

Použité počítače nedisponujú žiadnymi prostriedkami na akceleráciu kryptografických operácií, či už periférne moduly, alebo rozšírenia inštrukčnej sady procesora, preto všetky operácie museli prebehnúť softwarovo.

Meranie v tomto prostredí trvalo takmer 30 hodín, počas ktorých bolo prostredníctvom OpenVPN prenesených 1000 GB dát medzi testovacími počítačmi.

Tabuľka 1: Vplyv kompresie na čas prenosu. Hodnoty sú škálované tak, aby čas prenosu bez kompresie bol 1.

súbor	**avg**	**stdev**	**median**	**min**	**max**
random	1.0027	0.002	1.0027	0.9796	1.0075
zero	0.6614	0.0943	0.6627	0.5164	0.84

2.4 Komplikácie

Počas púšťania testov sme narazili na dve komplikácie hodné zmienky.

Prvou bolo, že napriek hláseniu OpenVPN o 58 rôznych podporovaných šifrovacích algoritmov, v skutočnosti bolo možné použiť iba 16 z nich. Súčasťou špecifikácie šifrovacieho algoritmu totiž je aj použitý blokový mód a v našej konfigurácii OpenVPN umožňovalo použiť iba mód CBC.

Druhou komplikáciou bol zvláštne sa prejavujúci race condition. Po dokončení všetkých prenosov sa totiž reštartovala služba OpenVPN s novými nastaveniami. Na strane odosielateľa sa to ale stávalo hneď po zapísaní posledného bytu vstupu do socketu, čo bolo občas skôr, než stihol príjemca spracovať celý súbor. Výsledkom bolo, že na strane príjemcu ešte `nc` čítalo posledné byty prenášaného súboru, ale na strane odosielateľa bol zatiaľ VPN tunel ukončený. V dôsledku tohto ostával proces `nc` na strane príjemcu visieť v nedefinovanom stave, snažiac sa čítať vstup zo socketu, ktorému ale medzičasom prestal existovať sieťový interface, teda tento proces ostával visieť nadobro.

Tento race condition sme eliminovali uspaním odosielateľa na niekoľko sekúnd po poslednom prenose.

3 Výsledky

Výsledky sme spracúvali s pomocou databázy SQLite [5] s rozšírením pre matematické a reťazcové funkcie `extensions-functions.c` [6]. Namerané dáta spolu s SQL dotazmi použitými na vypočítanie štatistík popísaných v tejto časti sú dostupné v projektovom repozitári [1].

3.1 Vplyv kompresie

Pre určenie vplyvu kompresie na prenosovú rýchlosť sme pre každú trojicu šifry, hašovacej funkcie a súboru spočítali pomer medzi časom bez kompresie a časom s kompresiou. Pre náhodný aj nulový súbor sme následne spravili štatistiku týchto pomerov cez všetky dvojice šifry a hašovacej funkcie. Výsledok je možné vidieť v tabuľke 1.

Ako tabuľka naznačuje, pre náhodný súbor, ktorý je veľmi ťažko komprimovateľný, použitie kompresie prinieslo iba zanedbateľné zrýchlenie vo veľmi obmedzenom počte prípadov;

Tabuľka 2: Vplyv šifrovacieho algoritmu na čas prenosu. Hodnoty sú škálované tak, aby čas prenosu pri použití BF-CBC bol 1.

šifra	**avg**	**stdev**	**median**	**min**	**max**
BF-CBC	1.0	0.0	1.0	1.0	1.0
CAMELLIA-128-CBC	1.0148	0.0036	1.014	1.0096	1.0236
CAMELLIA-256-CBC	1.0573	0.0044	1.0583	1.0456	1.0655
CAMELLIA-192-CBC	1.058	0.0046	1.0599	1.0467	1.0663
CAST5-CBC	1.0781	0.0078	1.0799	1.0603	1.0894
DES-CBC	1.1026	0.0119	1.1034	1.0776	1.1402
SEED-CBC	1.1045	0.0068	1.1058	1.0852	1.1147
DESX-CBC	1.1134	0.0109	1.1159	1.0887	1.1321
AES-128-CBC	1.2823	0.0221	1.2867	1.2235	1.3078
AES-192-CBC	1.3687	0.0293	1.376	1.2925	1.4016
RC2-64-CBC	1.4211	0.0364	1.4294	1.3309	1.4679
RC2-40-CBC	1.4213	0.0368	1.4306	1.3314	1.4694
RC2-CBC	1.4213	0.0365	1.4315	1.3318	1.4687
AES-256-CBC	1.4553	0.0376	1.4671	1.3593	1.498
DES-EDE-CBC	1.5821	0.0508	1.5934	1.4585	1.6539
DES-EDE3-CBC	1.5824	0.0504	1.592	1.4599	1.6551

skôr naopak, väčšinou spôsobilo veľmi mierne spomalenie, čo je v súlade s dokumentáciou OpenVPN [4], podľa ktorej použitá LZO kompresia môže zväčšiť packet o jeden byte pri nekomprimovateľných údajoch.

V prípade nulového súboru, ktorý je veľmi dobre komprimovateľný, sme očakávali značné zrýchlenie. Z tabuľky vyplýva, že zrýchlenie bolo celkom konzistentné o približne jednu tretinu, s relatívne malou disperziou.

3.2 Porovnanie šifrovacích algoritmov

Štatistiky pre toto porovnanie sme vytvárali iba na základe prenosu náhodného súboru bez kompresie.

Najprv sme identifikovali algoritmus Blowfish ako konzistentne najrýchlejší. Následne sme pre každý hašovací algoritmus vydelili čas prenosu časom pre šifru Blowfish, čím sme vyjadrili relatívne spomalenie každého šifrovacieho algoritmu oproti Blowfish pri danej hašovacej funkcii.

Nakoniec sme pre každý šifrovací algoritmus spočítali štatistiku relatívnych spomalení, ktorú možno vidieť v tabuľke 2.

Ako najpomalší algoritmus sa ukázal trojitý DES vo variantoch s jedným a tromi kľúčmi. AES a RC2 boli tiež výrazne pomalšie, než Blowfish, zatiaľ čo Camellia je iba nepatrne

pomalšia oproti Blowfish.

O algoritme Blowfish je známe, že preň existuje trieda slabých kľúčov, ktoré umožňujú kryptoanalýzu [9], preto treba vytváranie kľúča vykonávať opatrne. Nepodarilo sa nám zistiť, nakoľko tento problém rieši OpenVPN a OpenSSL. Pre lepšiu záruku dôvernosti je preto možné použiť šifru Camellia, pre ktorú zatiaľ neboli publikované podobné zraniteľnosti.

3.3 Porovnanie hašovacích funkcií

Pri tomto porovnaní sme opäť vychádzali iba z časov prenosu náhodného súboru bez kompresie.

Postup pri vytváraní štatistík bol podobný ako v časti 3.2, s tým rozdielom, že tentokrát sme identifikovali ako najrýchlejšiu funkciu MD4 a časy prenosu sme normalizovali na relatívne spomalenie oproti tejto funkcii. Výsledok je v tabuľke 3.

V jednom prípade prebehol prenos najrýchlejšie pri použití funkcie MD4 v kombinácii s podpisom pomocou RSA, čo je dôvod pre minimum menšie ako 1 v druhom riadku tabuľky. Vo všetkých ostatných prípadoch samotná funkcia MD4 prebehla najrýchlejšie.

Z nám neznámych dôvodov knižnica OpenSSL poskytuje aplikáciám hašovacie funkcie v kombinácii s podpisovými schémami RSA a DSA, pričom z hľadiska aplikačného rozhrania ich neodlišuje od klasických hašovacích funkcií. Nepodarilo sa nám zistiť, aký kľúč sa pri takýchto funkciách používa, ani akú výhodu majú oproti samotným hašovacím funkciám (keďže sú používané čisto ako hašovacie funkcie, nie na podpis).

Každopádne, podpísanie výsledku sa vo všetkých prípadoch prejavilo iba minimálne, niekedy dokonca prebehlo rýchlejšie, než bez podpisu.

Funkcie MD4 a MD5 boli suverénne najrýchlejšie, ibaže v dnešnej dobe sa už neodporúča používať ktorúkoľvek funkciu v nových protokoloch a aplikáciách [7, 8].

Funkcia SHA1 sa ukázala byť iba mierne pomalšia. Keďže podľa našich informácií pre túto funkciu doteraz nevyšlo odporúčanie nepoužívať ju ako hašovaciu funkciu pre HMAC, javí sa táto funkcia ako dobrý kandidát poskytujúci dostatočnú bezpečnosť bez výrazného negatívneho vplyvu na výkon. RIPEMD160 je tiež vhodný kandidát, ktorý je iba nepatrne pomalší oproti SHA1 a pre ktorý nie je zatiaľ známy žiaden útok.

Varianty SHA s väčšou dĺžkou bloku sa ukázali ako nezanedbateľne pomalšie a Whirlpool sa umiestnil na poslednom mieste.

4 Záver

V tejto práci sme experimentálne odmerali vplyv kompresie, použitej šifry a použitej hašovacej funkcie na priepustnosť OpenVPN tunelov.

V prípade kompresie sa ukázalo, že pri prenose veľmi dobre komprimovateľných dát stúpla rýchlosť približne na jedenapolnásobok. Vzhľadom na to, že tieto dáta sa dajú skomprimovať veľmi rýchlo na približne 1/900, tento výsledok nás mierne sklamal. Napriek tomu však

Tabuľka 3: Vplyv hašovacej funkcie na čas prenosu. Hodnoty sú škálované tak, aby čas prenosu pri použití MD4 bol 1.

haš	**avg**	**stdev**	**median**	**min**	**max**
MD4	1.0	0.0	1.0	1.0	1.0
RSA-MD4	1.0027	0.0066	1.0013	0.9985	1.0272
MD5	1.0082	0.0019	1.0084	1.0039	1.0111
RSA-MD5	1.0092	0.002	1.0092	1.005	1.0114
ecdsa-with-SHA1	1.0494	0.0098	1.0509	1.0312	1.0643
DSA-SHA1-old	1.0498	0.0094	1.0512	1.0328	1.0631
SHA1	1.0502	0.0094	1.0507	1.032	1.0627
DSA-SHA	1.0505	0.0099	1.0523	1.0328	1.0649
RSA-SHA1-2	1.0505	0.0095	1.0525	1.033	1.0656
RSA-SHA1	1.0506	0.0097	1.0524	1.0324	1.065
DSA-SHA1	1.0507	0.0102	1.051	1.0327	1.0654
DSA	1.0509	0.0095	1.052	1.0336	1.0644
RIPEMD160	1.0741	0.014	1.0746	1.0492	1.0936
RSA-RIPEMD160	1.0746	0.0147	1.0761	1.0503	1.0949
RSA-SHA	1.0846	0.0161	1.0873	1.0574	1.1073
SHA	1.0857	0.0165	1.0863	1.0582	1.1101
RSA-SHA224	1.133	0.0246	1.1346	1.0907	1.1695
SHA224	1.1331	0.0241	1.1346	1.0911	1.1687
SHA256	1.1367	0.0254	1.1369	1.0942	1.1735
RSA-SHA256	1.1369	0.0253	1.1369	1.0944	1.174
RSA-SHA512	1.2056	0.0383	1.2098	1.1504	1.2561
SHA512	1.206	0.0379	1.2113	1.1506	1.2593
SHA384	1.2106	0.0397	1.2164	1.1521	1.2651
RSA-SHA384	1.2108	0.0396	1.2163	1.1525	1.2635
whirlpool	1.327	0.0589	1.3344	1.2401	1.4065

Tabuľka 4: Celkové poradie vybratých konfigurácií.

	šifra	haš	čas
0	—	—	0.1227
1	BF-CBC	MD4	1.0
2	BF-CBC	RSA-MD4	1.0027
3	BF-CBC	MD5	1.0102
4	BF-CBC	RSA-MD5	1.0114
...	...	...	...
11	BF-CBC	SHA1	1.0612
...	...	...	...
41	CAMELLIA-128-CBC	RIPEMD160	1.1087
...	...	...	...
395	DES-EDE-CBC	SHA384	1.9065
396	DES-EDE3-CBC	SHA384	1.9069
397	DES-EDE3-CBC	RSA-SHA384	1.9082
398	AES-256-CBC	whirlpool	1.9115
399	DES-EDE-CBC	whirlpool	2.051
400	DES-EDE3-CBC	whirlpool	2.053

isté zlepšenie pre takéto dáta kompresia poskytla a keďže sa ukázalo, že pre nekomprimovateľné dáta bolo spomalenie v dôsledku kompresie zanedbateľné, z hľadiska priepustnosti siete nevidíme dôvod odporúčať nepoužívať kompresiu.

V prípade šifrovacích algoritmov bol najrýchlejším Blowfish, ktorý je v dnešnej dobe stále považovaný za bezpečný, za predpokladu opatrne zvoleného kľúča. Alternatívne šifra Camellia poskytuje porovnateľne dobrý výkon bez známych útokov.

Pokiaľ ide o hašovacie funkcie, najlepšie dopadli MD4 a MD5, ktoré ale už nie sú odporúčané z hľadiska bezpečnosti. Funkcia SHA1 poskytuje o niečo lepšie záruky s iba miernym spomalením.

V tabuľke 4 zhŕňame celkové poradie jednotlivých kombinácií šifry a hašovacej funkcie spolu s relatívnym spomalením oproti najrýchlejšej kombinácii. Navyše prikladáme pre porovnanie aj rýchlosť prenosu mimo VPN tunela, ktorá bola približne osemkrát vyššia, než najrýchlejšia kombinácia nastavení.

Predvolené hodnoty v OpenVPN sú Blowfish ako šifrovací algoritmus a SHA1 ako hašovacia funkcia. Tieto nastavenia sa ukázali ako najrýchlejšie s rozumnou zárukou dôvernosti, autentickosti a integrity so spomalením približne 1.06 oproti Blowfish s MD4. Pre ešte lepšie záruky je vhodná kombinácia šifry Camellia a funkcie RIPEMD160, ktorej spomalenie oproti najrýchlejšej kombinácii je približne 1.11.

Na opačnom konci spektra sa umiestnili kombinácie trojitej šifry DES a AES s 256-bitovým kľúčom s funkciou Whirlpool, ktoré poskytujú približne polovičnú priepustnosť

oproti najrýchlejšej konfigurácii s diskutabilnými bezpečnostnými výhodami v porovnaní s vyššie spomínanými kombináciami.

Existuje viacero smerov, ako ďalej pokračovať v tejto práci. Jednou možnosťou je preskúmať vplyv adaptívnej kompresie OpenVPN, ktorá sa snaží heuristicky komprimovať iba tie dáta, ktoré sú dobre komprimovateľné, a ostatné prenášať priamo. Ďalšou možnosťou je preskúmať účinok rozšírení inštrukčnej sady moderných procesorov o akceleráciu operácií využívaných v šifre AES; tieto rozšírenia pravdepodobne posunú AES prinajmenšom do hornej polovice tabuľky. Je však možné, že zrýchlia aj niektoré iné šifry, keďže viacero testovaných šifrovacích algoritmov používa niektoré rovnaké operácie ako AES.

Literatúra

[1] `https://github.com/laciKE/openvpn_test` – Repozitár projektu.

[2] `http://openvpn.net/index.php/open-source.html` – Domovská stránka OpenVPN.

[3] `http://openvpn.net/index.php/open-source/documentation/miscellaneous/78-static-key-mini-howto.html` – Návod na jednoduché nastavenie statického zdieľaného kľúča pre OpenVPN.

[4] `https://community.openvpn.net/openvpn/wiki/Openvpn22ManPage` – Manuálové stránky OpenVPN.

[5] `http://sqlite.org/` – Domovská stránka SQLite.

[6] `http://www.sqlite.org/contrib` – Domovská stránka rozšírení SQLite.

[7] `http://tools.ietf.org/html/rfc6150` – RFC 6150: MD4 to Historic Status.

[8] `http://tools.ietf.org/html/rfc6151` – RFC 6151: Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms.

[9] O. Kara and C. Manap. "A new class of Weak Keys for Blowfish". In: *Fast Software Encryption*. 2007, pp. 167–180.