



Sicurezza dei sistemi e delle reti¹

Mattia Monga

Dip. di Informatica
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

a.a. 2014/15

¹ © 2011–15 M. Monga. Creative Commons Attribuzione — Condividi allo stesso modo 4.0 Internazionale. <http://creativecommons.org/licenses/by-sa/4.0/deed.it>. Derivato con permesso da © 2010 M. Cremonini.



Lezione XXI: Virtual Private Network



Virtual Private Network

VPN

Una rete “overlay” che costituisce un dominio amministrativo sostanzialmente indipendente dalla topologia effettiva della rete sottostante

- Le comunicazioni sono criptate
- Molto usate per permettere a utenti roaming di accedere alle risorse delle reti aziendali



Tunnel

Si basano sul concetto di tunnel: ossia incapsulano i pacchetti in un altro protocollo (che rispetta la topologia della rete “fisica” sottostante)

Differiscono per il livello di rete virtuale che offrono.

Encryption



Sicurezza delle reti
Monga
VPN

Anche la confidenzialità e integrità dei pacchetti può essere ottenuta a diversi livelli

- IPSec
- SSL/TLS
- Protocolli proprietari
- SSH

464

SSH



Sicurezza delle reti
Monga
VPN

OpenSSH può essere usato per fare tunneling di altri protocolli in SSH

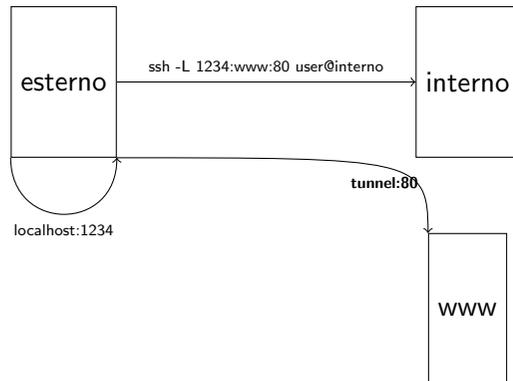
- Port forwarding
- Vera e propria VPN

465

Port forwarding



Sicurezza delle reti
Monga
VPN



www vede una connessione da interno, **il pezzo criptato è solo esterno/interno**

466

Una vera VPN con SSH



Sicurezza delle reti
Monga
VPN

Per avere una vera VPN bisogna appoggiarsi sui livelli piú bassi dello stack. Con Linux esiste un device tun che serve proprio a questo scopo (Tunneling/NAT regolato da applicazioni utenti)

```

+-----+ OpenSSH 4.3 +-----+
| Machine A | tun0 -- Tunnel Interface -- tun0 | Machine B | |
| Has a tunnel | <-----> | Has a tunnel |
| and ethernet | 10.0.0.100 | 10.0.0.200 | and ethernet |
+-----+ point to point connection +-----+
eth0 | creates a bridge | eth0
10.0.0.100 | that plugs machine B | 192.168.0.100
port 22 | into network A |
forwarded | |
here | |
+-----+ +-----+ +-----+
| Network A | 1.2.3.4 | The Internet | Network B |
| 10.0.0.1/24 | | | 192.168.0.1/24 |
| Has internet | <-----> | Has internet |
| NAT gateway | Routable | NAT gateway |
+-----+ Address +-----+
  
```

467

Setup di TUN/TAP



Sicurezza delle
reti

Monga

VPN

Se non è già attiva, bisogna creare l'interfaccia TUN/TAP
In Linux potrebbe essere necessario

- 1 `mknod /dev/net/tun c 10 200`
- 2 `modprobe tun`

468

Configurazione



Sicurezza delle
reti

Monga

VPN

```
# A
iface tun0 inet static
  address 10.0.0.100
  pointopoint 10.0.0.200
  up arp -sD 10.0.0.200 eth0 pub
# B
iface tun0 inet static
pre-up ssh -f -w 0:0 1.2.3.4 'ifdown tun0; ifup tun0'
  address 10.0.0.200
  pointopoint 10.0.0.100
  up ip route add 10.0.0.0/24 via 10.0.0.200
  up ip route add 1.2.3.4/32 via 192.168.0.1
  up ip route replace default via 10.0.0.1
  down ip route replace default via 192.168.0.1
  down ip route del 10.0.0.0/24 via 10.0.0.200
  down ip route del 1.2.3.4/32 via 192.168.0.1
```

469

Riassumendo



Sicurezza delle
reti

Monga

VPN

Le VPN

- Permettono di usare una rete potenzialmente ostile, con alcune garanzie di confidenzialità e integrità
- Si basano sul tunnel di protocolli e possono essere realizzate in molti modi diversi

470

SSHuttle



Sicurezza delle
reti

Monga

VPN

Iptables + SSH tunnelling

```
1 Chain PREROUTING (policy ACCEPT 33 packets, 1604 bytes)
2 pkts bytes target prot opt in out source destination
3 33 1604 sshuttle-12300 all -- * 0.0.0.0/0 0.0.0.0/0
4
5 Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
6 pkts bytes target prot opt in out source destination
7
8 Chain OUTPUT (policy ACCEPT 24 packets, 1568 bytes)
9 pkts bytes target prot opt in out source destination
10 48 3008 sshuttle-12300 all -- * 0.0.0.0/0 0.0.0.0/0
11
12 Chain POSTROUTING (policy ACCEPT 48 packets, 3008 bytes)
13 pkts bytes target prot opt in out source destination
14
15 Chain sshuttle-12300 (2 references)
16 pkts bytes target prot opt in out source destination
17 0 0 RETURN tcp -- * 0.0.0.0/0 127.0.0.0/8
18 24 1440 REDIRECT tcp -- * 0.0.0.0/0 0.0.0.0/0 TTL match TTL != 42 redir ports 12300
```

471

OpenVPN



Sicurezza delle
reti

Monga

VPN

Un programma open-source che permette di costruire VPN basate su tunnel TCP o UDP. (La porta assegnata da IANA a OpenVPN è 1194)

Anche in questo caso l'implementazione sfrutta il driver TUN/TAP.

472

Autenticazione



Sicurezza delle
reti

Monga

VPN

L'autenticazione può avvenire in due modi

Static key pre-shared key

TLS tramite una sessione SSL/TLS (su UDP) con certificati e scambio di chiavi

473

Tunnel



Sicurezza delle
reti

Monga

VPN

Il tunnel può essere TCP (sconsigliato per ragioni di efficienza) o UDP (default)

Si può anche scegliere se usare l'interfaccia TUN (livello network) TAP (livello link).

474

Configurazione



Sicurezza delle
reti

Monga

VPN

Punto a punto, TAP (indirizzi assegnati con DHCP)

```
1 client
2 remote 172.16.0.3
3 dev tap
4 tls-client
5 ca client.crt
6 auth-user-pass
```

```
1 dev tap
2 remote 172.16.0.1
3 tls-server
4 ca server.crt
5 auth-user-pass-verify script
```

con la modalità server è possibile definire VPN con topologie complicate

475



OpenVPN è facile da gestire con i firewall, perché usa un'unica porta (generalmente UDP 1194)

```
iptables -A INPUT -p udp --dport 1194 -j ACCEPT
```

E poi

```
iptables -A INPUT -i tun+ -j ACCEPT
```

(o l'equivalente con tap)



Ogni pacchetto ha un message authentication code calcolato con HMAC, per cui i pacchetti non integri vengono scartati

$$HMAC = H(K \oplus OPAD | H(K \oplus IPAD | m))$$

(RFC2104: $OPAD = 0x5c5c5c \dots 5c5c$, $IPAD = 0x363636 \dots 3636$)

- Autenticazione più forte che basata su IP sorgente
- Ogni pacchetto che arriva a tun/tap è già stato controllato



È opportuno usare HMAC e non MAC più semplici perché essendo i pacchetti di dimensione fissa sarebbero possibili attacchi del tipo hash extension, se l'hash è di tipo Merkle-Damgård (MD4-5, SHA0-2)

- $H(K|m)$: extension attack se nota la lunghezza di m
- $H(m|K)$: meglio, ma dipende fortemente dalle tecniche di collisione di H (birthday attack)
- $H(K|m|K)$ ancora meglio, ma manca una dimostrazione del grado di efficacia



Reso famoso da un attacco a Flickr (2009... vulnerabilità nota dal 1992!)

Se si sa $len(K|m)$ è possibile generare $H(K|m|m')$ senza conoscere K . Il motivo è che gli H Merkle-Damgård spezzano lo stream in blocchi e applicano una funzione di compressione ad ogni blocco **indipendentemente**.

Con lunghezza si sa quale funzione applicare a blocchi aggiunti dall'attaccante (estensione).



OpenVPN permette di costruire VPN

- autenticazione con chiave condivisa o certificati
- tunnel UDP o TCP

480



Caricare una pagina web da un server HTTP espone moltissimi *dati personali* (alcuni addirittura *sensibili*)

- IP del client (pseudonym)
- IP del server (il client è interessato a quel server)
- identità (vedi <https://panopticklick.eff.org>)
- Dati del browser (history, cookie,...)
- System profile
- Dati trasmessi con form,...

481



- 1 Un eavesdropper può osservare il traffico: anche quando è criptato (`https`) gli IP e il system profile è accessibile
- 2 Il server web conserva i dati riguardanti il client
- 3 Il gestore della rete osserva e/o censura il traffico

482



- 1 HTTPS
- 2 Proxy, cambiando proxy server (e browser!)
- 3 Onion Routing

483

Proxy



Sicurezza delle
reti

Monga

VPN

```
...  
REMOTE_ADDR = 194.85.1.1  
HTTP_ACCEPT_LANGUAGE = ru  
HTTP_USER_AGENT = Mozilla/4.0  
HTTP_HOST = www.webserver.ru  
HTTP_VIA = 194.85.1.1 (Squid/2.4.STABLE7)  
HTTP_X_FORWARDED_FOR = 194.115.5.5  
...
```

484

proxy



Sicurezza delle
reti

Monga

VPN

Transparent proxy HTTP_X_FORWARDED_FOR è l'IP del client originale!

Anonymous Proxies IP nascosto

Distorting Proxies IP falso

High Anonymity Proxies HTTP_VIA vuoto

485

Privoxy



Sicurezza delle
reti

Monga

VPN

Privoxy è un proxy web con avanzate capacità di filtraggio progettato per la privacy

- gestisce i cookie
- javascript
- personalizzabile

Questa categoria di prodotti è detta protocol cleaner

486

Onion Routing



Sicurezza delle
reti

Monga

VPN

Onion Routing (OR) è una tecnica sviluppata dal Naval Research Laboratory di Washington.

Il traffico viene instradato in una serie (mutevole) di *onion router* in maniera tale da rendere difficile il tracciamento delle attività.

Gli onion router costituiscono dei mix di Chaum.

487

Mix di Chaum



Sicurezza delle reti

Monga

VPN

Mix di Chaum

Un mix riceve messaggi di lunghezza fissa, li cripta, aspetta di averne in numero sufficiente da garantire un certo livello di anonimato e inoltra i messaggi (in ordine arbitrario) ad altri mix.

488

Onion router



Sicurezza delle reti

Monga

VPN

- Gli onion router intermedi non hanno informazione sufficiente per tracciare mittente/destinatario e traffico
- Rimane una certa criticità degli exit node e (minore) dei nodi d'entrata

489

Riassumendo



Sicurezza delle reti

Monga

VPN

L'anonimato nella navigazione web è un problema piuttosto complesso. Varie difese:

- HTTPS
- Proxy
- Onion Routing

490

TOR



Sicurezza delle reti

Monga

VPN

TOR (The Onion Router project) è l'evoluzione più recente del concetto di OR. Sviluppato da NRL, open source, supportato da EFF.

Il progetto fa parecchi sforzi per rendere il prodotto comprensibile e utilizzabile anche da utenti inesperti (il numero di utenti è un valore per l'anonimato).

491



How Tor Works

How Tor Works: 1



How Tor Work



Monga

VPN

L'instradamento di ogni messaggio viene detto circuito

- Ogni nodo del circuito conosce solo il nodo precedente e successivo (non origine e destinazione)
- Molte richieste diverse vengono multiplexate in un unico circuito
- Robusto rispetto alla compromissione o l'introduzione di onion router malevoli
- Nodi trusted operano da *directory server* iniziali



Monga

VPN

- Nodi utenti hanno un Onion Proxy (OP)
- Onion Router (OR) connessi tra loro con TLS
- Gli OR hanno una long-term key e short-term "onion" key
- L'unità di trasmissione è la cella, di dimensione fissa di 512 byte



Monga

VPN

La *long-term identity key* viene usata per

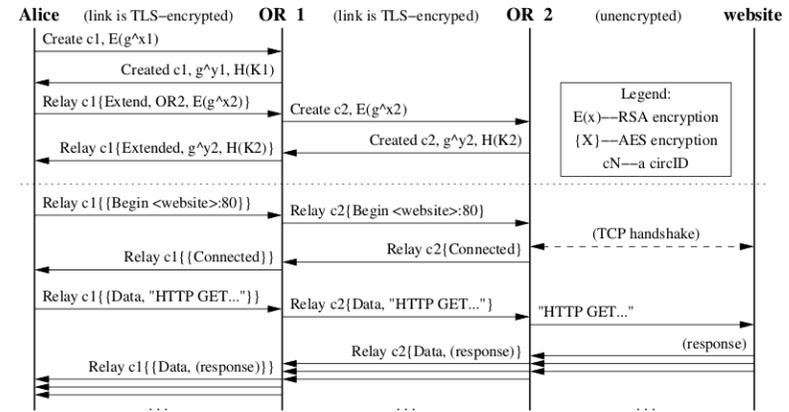
- firmare la descrizione del router: certificati TLS, chiavi, metadati, *exit policy*
- firmare gli elenchi di router

La *short-term key* (onion key) per:

- decrittare le richieste di circuiti
- negoziare chiavi una tantum (ephemeral key) che garantiscono la *forward secrecy*



- l'OP costruisce un circuito in background, e diversi stream utente vengono multiplexati sullo stesso circuito
- Ogni minuto viene creato un nuovo circuito



La prima fase *estende* il circuito, poi c'è il relay del traffico



L'OP sceglie quale OR può fare da exit node (ognuno ha una *exit policy*)
Solo TCP stream possono essere creati (UDP, e quindi le risoluzioni DNS, rimangono problematiche: vedi <http://code.google.com/p/torsocks/> per una soluzione)
Un protocol cleaner è necessario per evitare che informazioni rilevanti finiscano nello stream.



TOR

- il maggior progetto di Onion Routing
- permette la creazione di circuiti anonimi
- necessita di un protocol cleaner