Firewall Piercing

Alon Altman
Haifa Linux Club

Introduction

Topics of this lecture

- Basic topics
  - SSH Forwarding
  - PPP over SSH
  - Using non-standard TCP ports
- Advanced topics
  - TCP over HTTP
  - Tunneling over UDP
  - Tunneling over DNS

Definitions

- The machine requiring access (your laptop) will be called the **client**.
- The client is connected to the Internet behind a **firewall**.
- You may be able to administer another machine on the Internet which is the **server**.
- Finally, you want to connect to some service on the Internet – the **target**, which runs on the **target host**.

SSH Forwarding

- **Problem:** The firewall blocks outgoing connections to certain target hosts or ports.
  - Note this exactly the same problem as when both the server and the target are behind a firewall blocking incoming connections.
- We assume SSH to the server is open.
- **Solution:** Use ssh's built-in forwarding support to forward your connections to the target.

SSH Local Forwarding

- If you need to connect to a specific port on a specific target host, use SSH *local forwarding*.
- Assume you wish to connect to irc.inter.net.il on port 6667 (irc).
- You can forward local port 9000 to port 6667 on irc.inter.net.il:
  - ssh -l user server -L 9000:irc.inter.net.il:6667
- Now, connect to localhost:9000 in your IRC software.

SSH Dynamic Forwarding

- **Problem**: What if you need to access to more than one target host and more than one target service?
- **Solution**: SSH dynamic forwarding simulates a SOCKS4 proxy on the local host.
- Simply run SSH with dynamic forwarding:
  - ssh -l user server -d 1080
- And then, configure your application to use a SOCKS4 proxy on localhost:1080

PPP over SSH

- **Problem**: I want to run ICQ (or any other non-SOCKSable) service over SSH.
- **Solution**: Run PPP(point to point protocol) over the SSH tunnel.
- Allocates a fake IP to the client.
- All network activity is transparent.
- Care needs to be taken in order to keep a working route to server the DNS servers.

PPP over SSH - configuration

- In the ppp provider configuration:
  - debug
  - nodefaultroute
  - noauth
  - pty "/usr/bin/ssh -t user@server slirp ppp"
- Allow root to access user@server without a password (using public keys).
- In the client, setup host specific routes to server, and then add only a default route via ppp0.

Using non-standard ports

- **Problem**: The firewall blocks access to a specific service (say, SSH) running on the server.
- In this case the target host and the server are the same (Your machine).
- **Solution**: Run the target service on a port which is open in the firewall and connect via this port.
- This can be applied also when the server is behind a firewall blocking incoming ports.

Example: SSH on port 80

- Suppose your university blocks all ports except 80 (http).
- You wish to have SSH access to your server.
- You can open an SSH server on port 80.
  - In the sshd configuration change “Port” to 80.
- In the client (your laptop), connect using the non-standard port:
  - ssh -p 80 -l user server
  - scp -P 80 file user@server:directory/

Non-standard ports: Problem

- **Problem**: What if you want to run a web server as well?
- **Solution**: Use an iptables based firewall on the server to redirect port 80 only for your university IP address.
  - iptables -t nat -A PREROUTING -p tcp --dport 80 -s client.ip.addr -j REDIRECT --to-ports 22
- Even if you use a different port, this technique allows you to have the one SSH server listening on several ports

Example – Some faculty's WiFi

- Limited to students only.
- Any client who knows the ESSID can get an IP address without authentication.
- Authentication should be done against a special web server.
- Unauthenticated clients' access is blocked.
- However, outgoing TCP (and UDP) port 53 is open to the entire network of the institution.

Faculty WiFi – The bypass

- Client is unprivileged laptop in faculty.
- Server is inside the institution, with SSH open on port 53.
- Client runs PPP, buffering the data over the SSH connection.
- Server runs slirp – a simple user-only PPP emulator with NAT.
- Speed is very fast, as the underlying network is local and high-speed.

Advanced Topics

TCP over HTTP

- **Problem**: Web-only Internet connection via a proxy (say, in a public library).
- Proxy can be transparent or opaque.
- Can't run ssh on port 80(HTTP), because proxy accepts only real HTTP connections.
- **Easy Solution:** Use HTTP CONNECT. HTTP proxies may allow direct connections via a special CONNECT mechanism. I won't discuss this here.

TCP over HTTP – brute force

- The idea: Run a special web server on the server and send special web requests from the client.
- Outgoing packets can be encoded as requests. Incoming packets encoded as replies.
- Overhead is quite low.
- Implementation: GNU httptunnel

Tunneling over UDP

- **Problem**: Firewall blocks TCP connections but allows UDP connections to a specific port.
- For example, a restrictive firewall which only allows DNS queries, but does not limit the target.
- Sometimes found in login-only or for-pay WiFi services.
- **Solution**: Tunnel TCP (or anything else) inside UDP packets.

TCP over DNS
(or how to save €10 per hour)

- **Problem**: A for-pay WiFi service blocks all services except DNS for non-paying users.
- Access is allowed only to a specific DNS server, which responds only to real DNS queries.
- **Solution:** Tunnel your data over DNS.