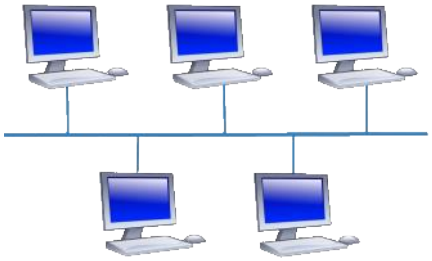
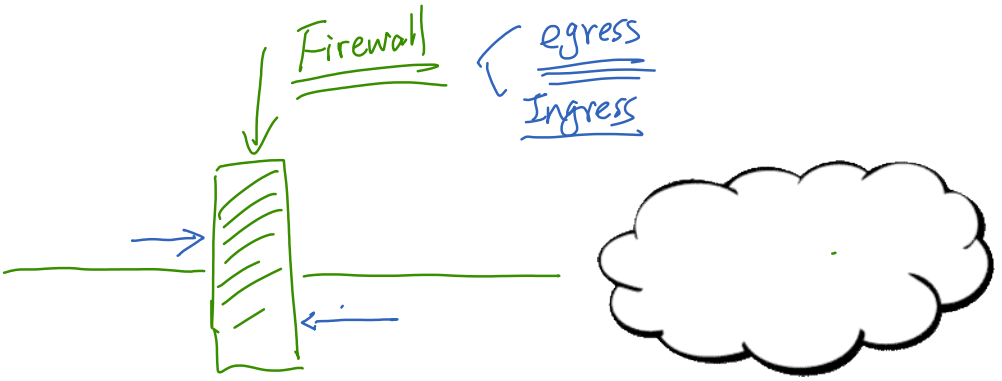
Internet Security

Firewall
Bypassing Firewall

Overview of Firewall





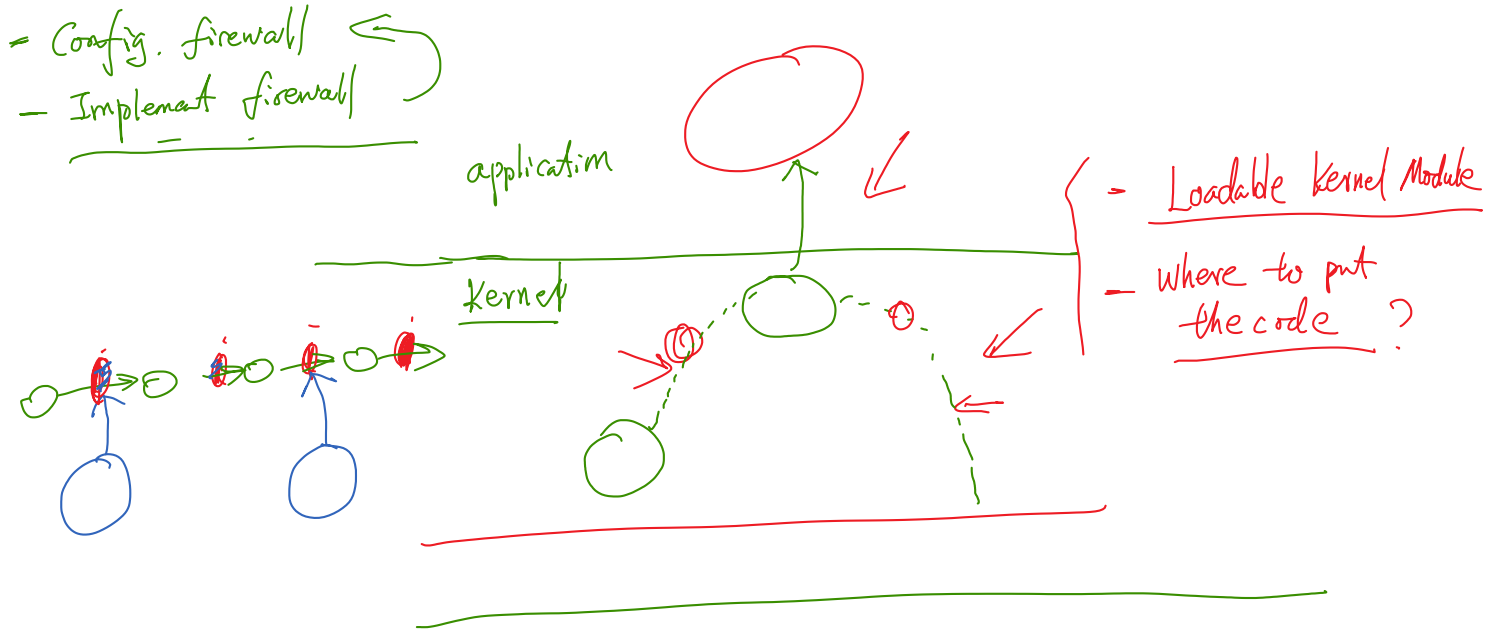
packet
- packet filter
- Stateful filter (connection-based)

- Application-Level Firewall

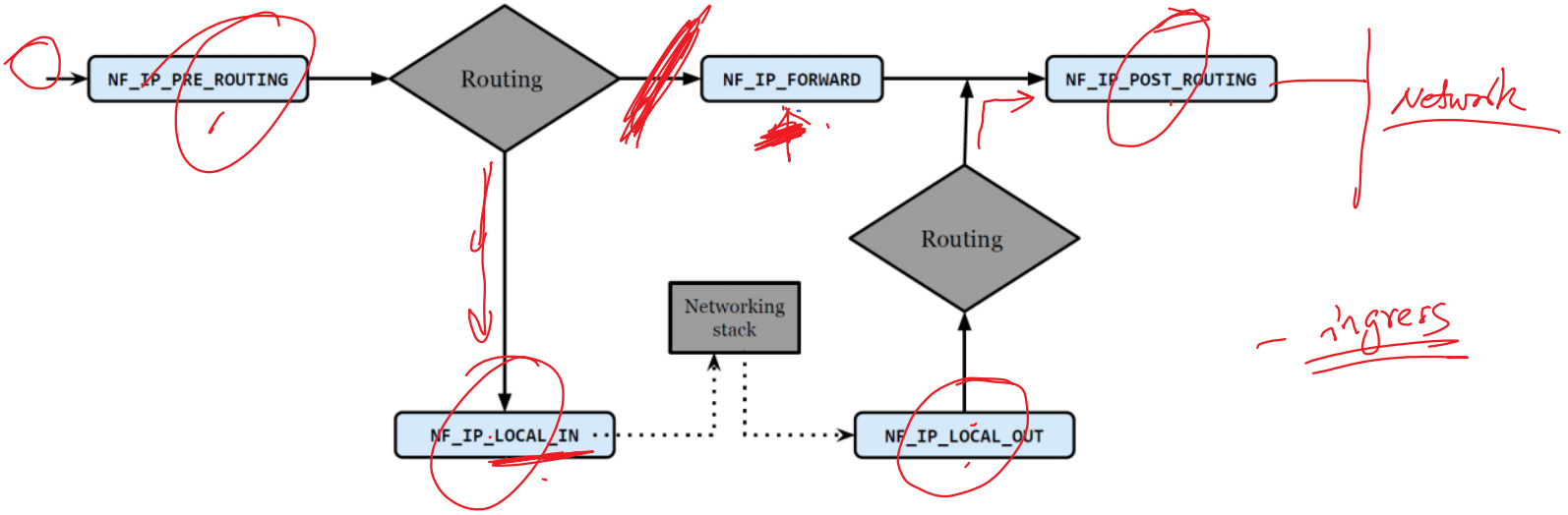
web proxy

Types of Firewalls

Linux Firewall Implementation



Netfilter Hooks



Kernel Module

```
#include <linux/module.h>

static int kmodule_init(void) {
        printk(KERN_INFO "Initializing this module\n");
        return 0;
}

static void kmodule_exit(void) {
        printk(KERN_INFO "Module cleanup\n");
}

module_init(kmodule_init);
module_exit(kmodule_exit);

MODULE_LICENSE("GPL");
```

```
// Insert the kernel module into the running kernel.
$ sudo insmod kMod.ko

// List kernel modules
$ lsmod | grep kMod
kMod                   12453  0

// Remove the specified module from the kernel.
$ sudo rmmod kMod
```

```
$ dmesg
......
[65368.235725] Initializing this module
[65499.594389] Module cleanup
```

Netfilter: Implement a Simple Firewall (`minifirewall`)

❖ Hooking filter code to one of the netfilter hooks

```
static struct nf_hook_ops telnetFilterHook;

int setUpFilter(void) {
    printk(KERN_INFO "Registering a Telnet filter.\n");
    telnetFilterHook.hook = telnetFilter;
    telnetFilterHook.hooknum = NF_INET_POST_ROUTING;
    telnetFilterHook.pf = PF_INET;
    telnetFilterHook.priority = NF_IP_PRI_FIRST;
    nf_register_hook(&telnetFilterHook);
    return 0;
}

void removeFilter(void) {
    printk(KERN_INFO "Telnet filter is being removed.\n");
    nf_unregister_hook(&telnetFilterHook);
}

module_init(setUpFilter);
module_exit(removeFilter);
```

12.04

16.04

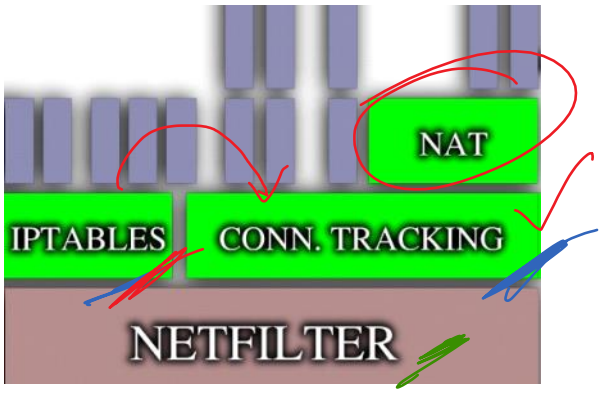
❖ Implementation of the filter

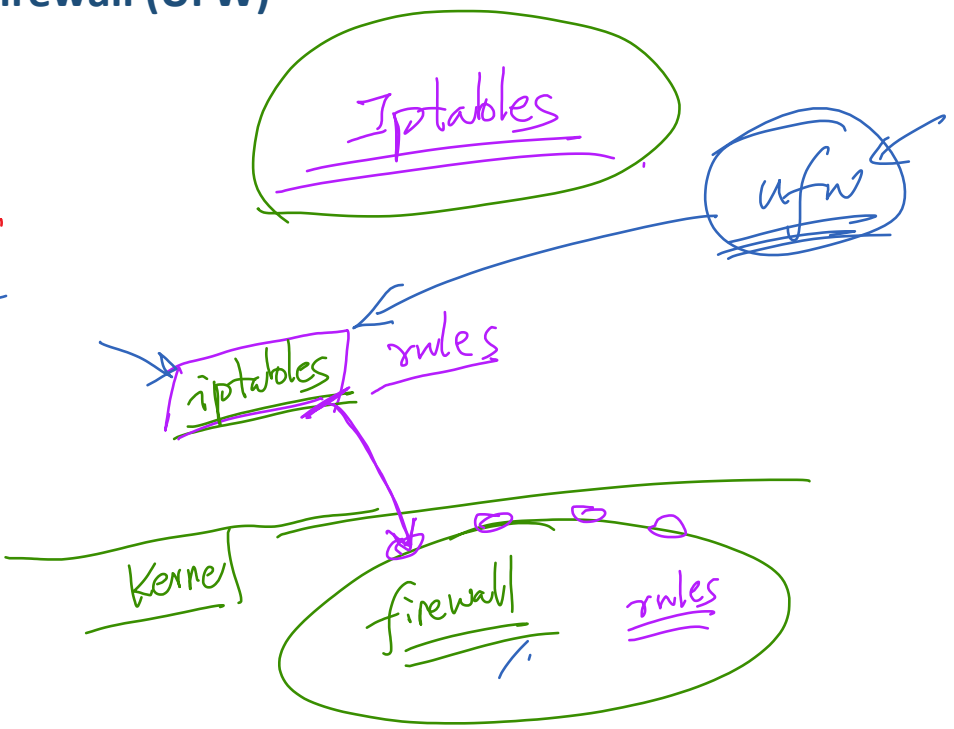
```
unsigned int telnetFilter(unsigned int hooknum, struct sk_buff *skb,
        const struct net_device *in, const struct net_device *out,
        int (*okfn)(struct sk_buff *)) {
    struct iphdr *iph;
    struct tcphdr *tcph;

    iph = ip_hdr(skb);
    tcph = (void *)iph+iph->ihl*4;

    if (iph->protocol == IPPROTO_TCP && tcph->dest == htons(23)) {
        printk(KERN_INFO "Dropping telnet packet to %d.%d.%d.%d\n",
                ((unsigned char *)&iph->daddr)[0],
                ((unsigned char *)&iph->daddr)[1],
                ((unsigned char *)&iph->daddr)[2],
                ((unsigned char *)&iph->daddr)[3]);
        return NF_DROP;
    } else {
        return NF_ACCEPT;
    }
}
```

❖ Header files

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/netfilter.h>
#include <linux/netfilter_ipv4.h>
#include <linux/ip.h>
#include <linux/tcp.h>
```

Iptables and Uncomplicated Firewall (UFW)

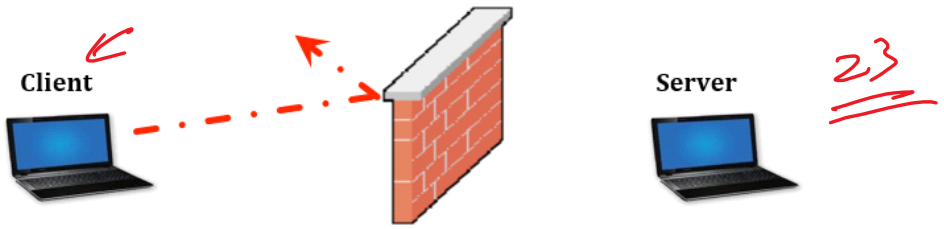




UFW: Using UFW to Set up Firewall Rules

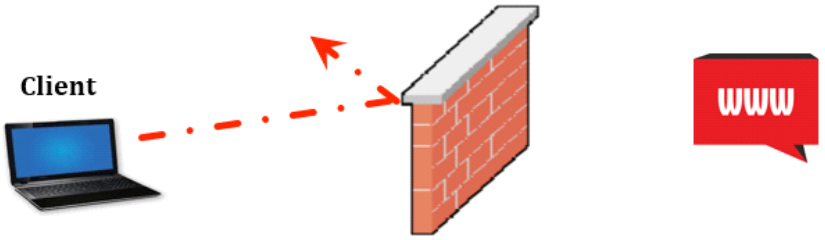
```
ufw <action> <direction> <service>
ufw (allow | deny) (in | out) from (src) to (dest) port (portNo)
```

#1 *"Prevent client machine from telnetting to any external machine"*




```
sudo ufw deny out from Client_IP to any port 23
```

#2 *"Prevent client machine from accessing a website"*



More UFW Commands

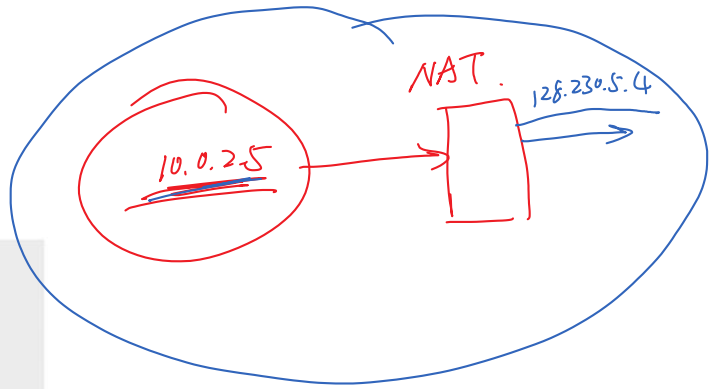
```
$ sudo ufw enable          // this will enable the firewall.
$ sudo ufw disable         // this will disable the firewall.
$ sudo ufw status numbered // this will display the firewall rules.
$ sudo ufw delete 3        // this will delete the 3rd rule.
```

The iptables Firewall

❖ iptable Tables and Chains

Table	Chain	Functionality
filter	INPUT FORWARD OUTPUT	Packet filtering
nat	PREROUTING INPUT OUTPUT POSTROUTING	Modifying source or destination network addresses
mangle	PREROUTING INPUT FORWARD OUTPUT POSTROUTING	Packet content modification

NAT



NAT

❖ Examples

```
// Allow all incoming TCP packets bound to destination port 22.
// -A INPUT: Append to existing INPUT chain rules.
// -p tcp: Select TCP packets
// -dport 22: Select packets with destination port 22.
// -j ACCEPT: Accept all the packets that are selected.
$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT

// Similarly, accept all packets bound to destination port 80.
$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

```
// Allow all outgoing TCP traffic.
// -A OUTPUT: Append to existing OUTPUT chain rules.
// -p tcp: Apply on TCP protocol packets
// -m tcp: Further apply matching rules defined in 'tcp' module.
// -j ACCEPT: Let the selected packets through.

$ sudo iptables -A OUTPUT -p tcp -m tcp -j ACCEPT
```

```
// -t mangle = Add this to 'mangle' table
// -A PREROUTING = Append this rule to PREROUTING chain

iptables -t mangle -A PREROUTING -j TTL --ttl-inc 5
```

Connection Tracking

An Example of Statefull Firewall

"connection"

TCP

UDP

ICMP

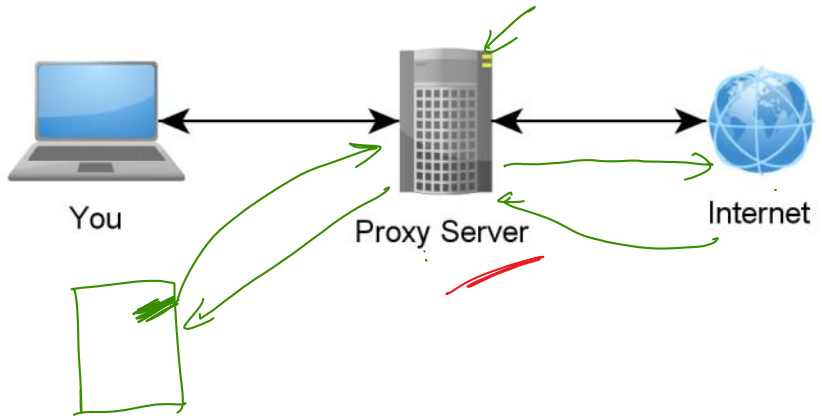
❖ Without considering connections

```
// Allow all outgoing TCP traffic.
// -A OUTPUT: Append to existing OUTPUT chain rules.
// -p tcp: Apply on TCP protocol packets
// -m tcp: Further apply matching rules defined in 'tcp' module.
// -j ACCEPT: Let the selected packets through.

$ sudo iptables -A OUTPUT -p tcp -m tcp -j ACCEPT
```

❖ Considing connections

```
// -A OUTPUT: Append to existing OUTPUT chain rules.
// -p tcp: Apply on TCP protocol packets.
// -m conntrack: Apply the rules from conntrack module.
// --ctsate ESTABLISHED,RELATED: Look for traffic in ESTABLISHED or RELATED
   states.
// -j ACCEPT: Let the selected packets through.

$ sudo iptables -A OUTPUT -p tcp -m conntrack --ctstate ESTABLISHED,RELATED
   -j ACCEPT
```

Application Firewall: Web Proxy

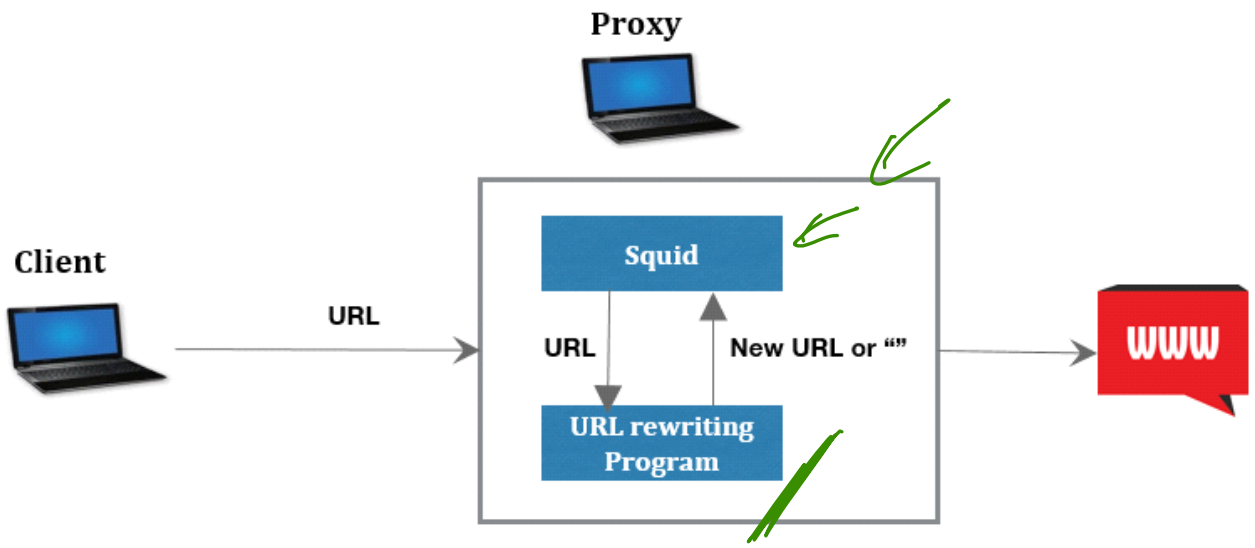


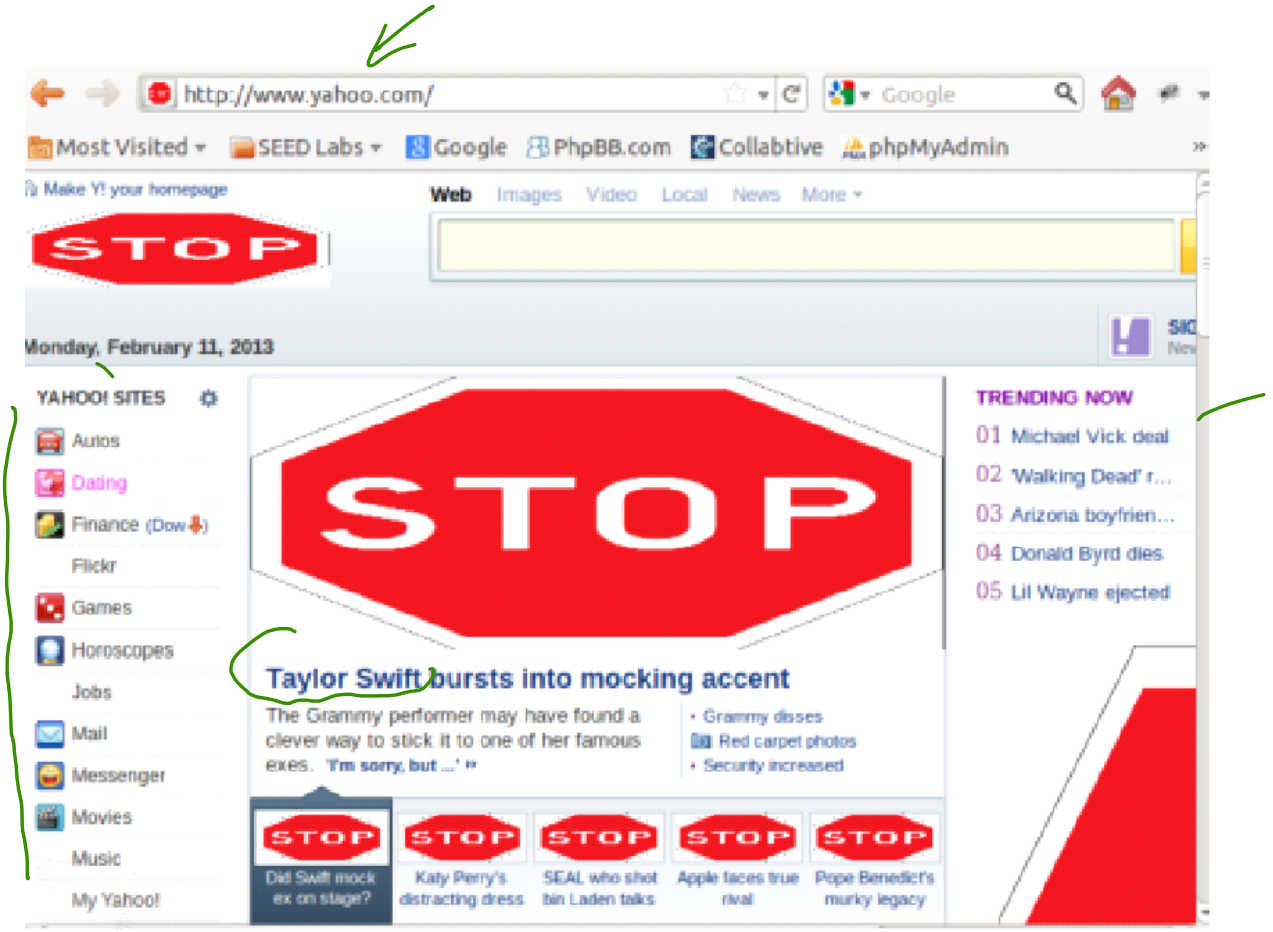
Web Proxy: Squid

What is Squid?

Squid is a fully-featured HTTP/1.0 proxy which is almost (but not quite - we're getting there!) a fully-featured HTTP/1.1 proxy. Squid offers a rich access control, authorization and logging environment to develop web proxy and content serving applications. Squid offers a rich set of traffic optimization options, most of which are enabled by default for simpler installation and high performance.

Squid: Redirect Traffic

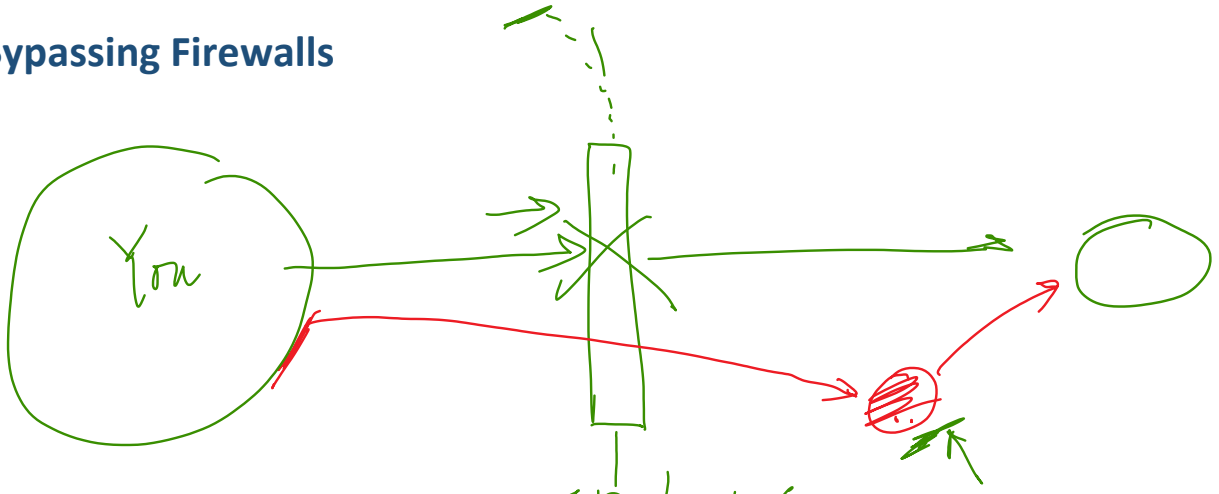




Squid: URL Rewriting Code

```
#!/usr/bin/perl -w
use strict;
use warnings;
# Forces a flush after every write or print on the STDOUT
select STDOUT; $| = 1;
# Get the input line by line from the standard input.
# Each line contains an URL and some other information.
while (<>) {
    my @parts = split;
    my $url = $parts[0];
    # If you copy and paste this code from this PDF file,
    # the ~(tilde) character may not be copied correctly.
    # Remove it, and then type the character manually.
    if ($url =~ /\.(jpg|bmp|gif|jpeg)/) {

        # URL Rewriting
        print "http://mars.syr.edu/html/seed/stopsign.png\n";
    }
    else {
        # No Rewriting.
        print "\n";
    }
}
```

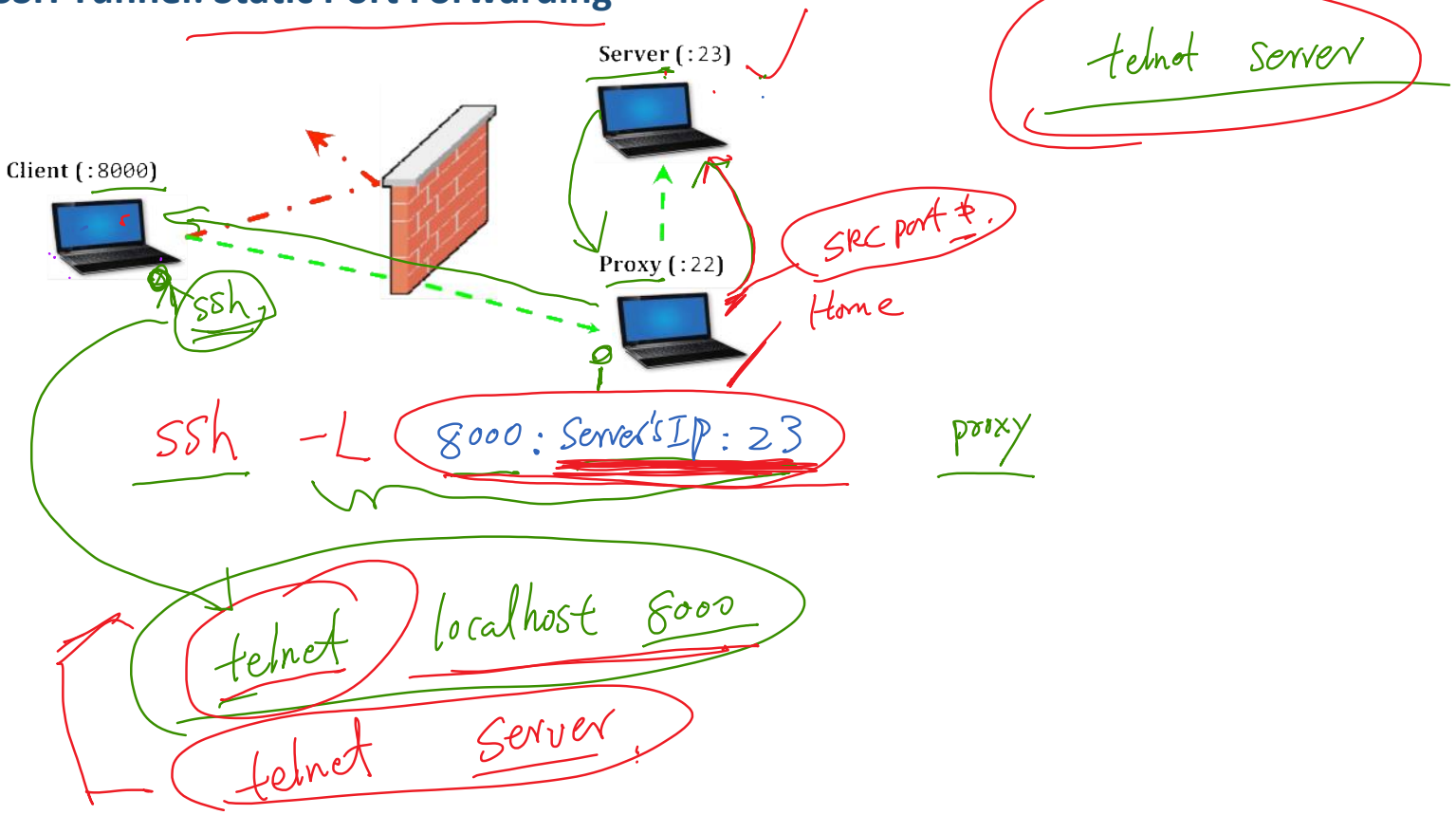
Bypassing Firewalls



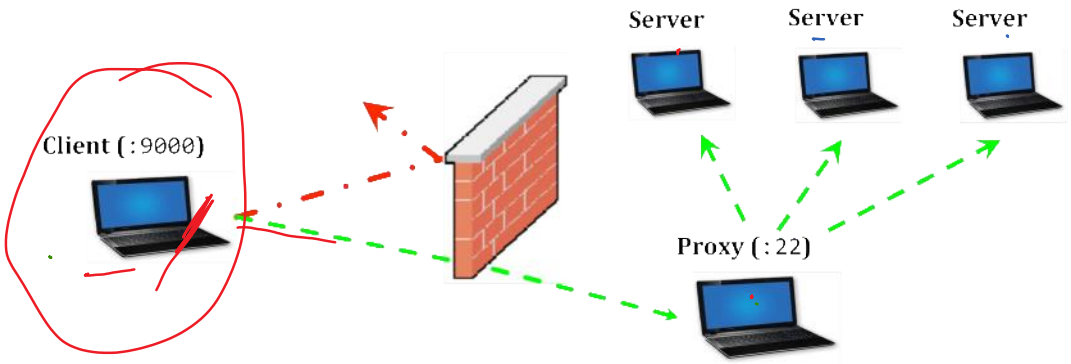
- VPN : IP Layer
- SSH Tunnel (poor man's VPN) : Transport Layer

Tunneling Techniques

SSH Tunnel: Static Port Forwarding



SSH Tunnel: Dynamic Port Forwarding

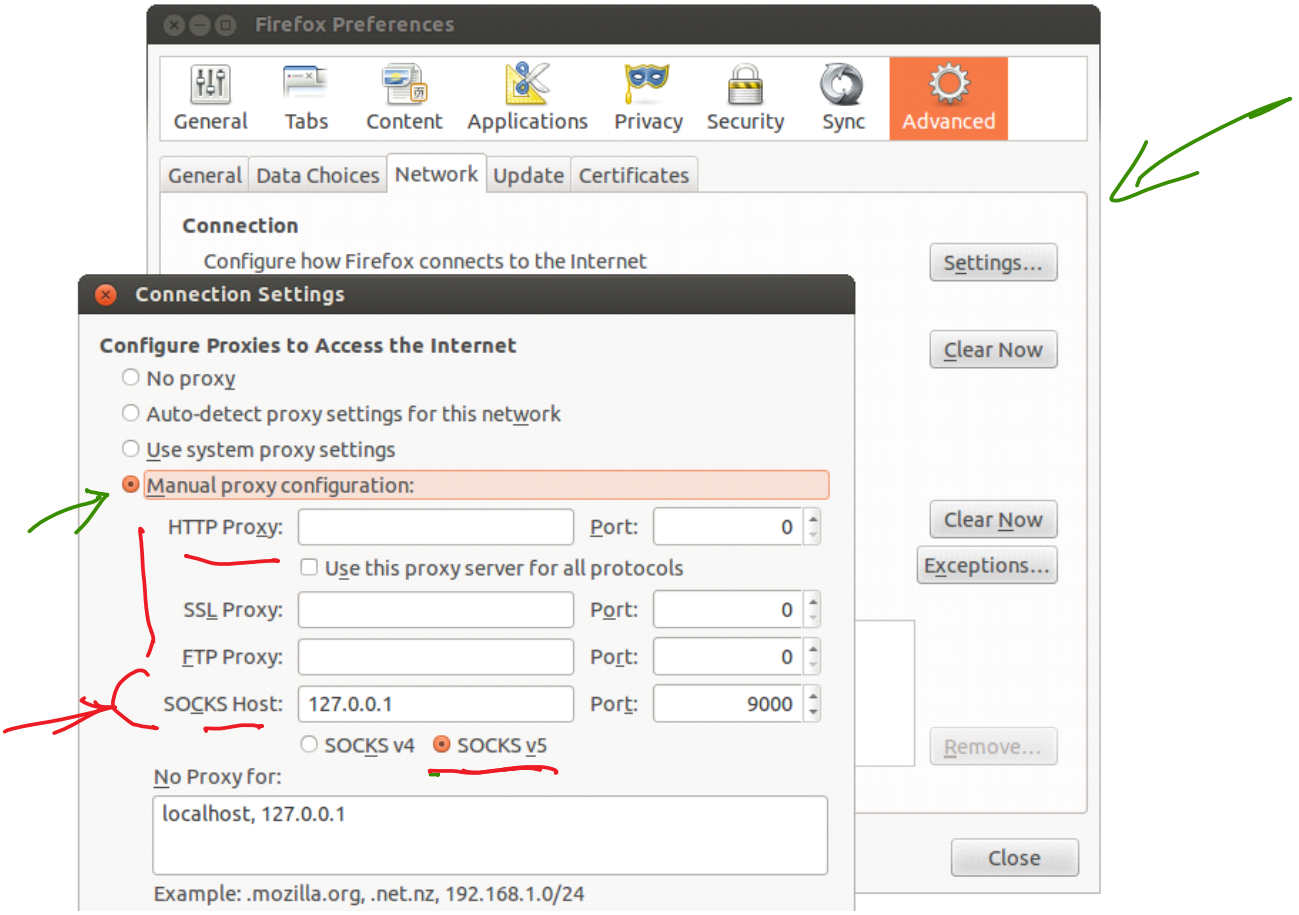


Browser

ssh -D 9000 proxy

telnet localhost 9000

Configuring Browser to use Dynamic Port Forwarding



SSH Tunnel: Remote Port Forwarding

IP Tunneling

SURA: Before Running VPN

❖ Interfaces

```
PS C:\Users\kevin> ipconfig

Windows IP Configuration


Wireless LAN adapter Wireless Network Connection 2:

   Media State . . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wireless Network Connection:

   Connection-specific DNS Suffix  . : syr.edu
   Link-local IPv6 Address . . . . . : fe80::30c5:d02c:ed1d:2d2e%13
   IPv4 Address. . . . . . . . . . . : 10.1.56.64
   Subnet Mask . . . . . . . . . . . : 255.255.192.0
   Default Gateway . . . . . . . . . : 10.1.0.1
```

❖ Routing table (Windows: Route PRINT)

```
IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0         10.1.0.1       10.1.56.64     25
         10.1.0.0    255.255.192.0         On-link        10.1.56.64    281
       10.1.56.64  255.255.255.255         On-link        10.1.56.64    281
      10.1.63.255  255.255.255.255         On-link        10.1.56.64    281
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    306
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    306
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    306
    192.168.147.0    255.255.255.0         On-link     192.168.147.1    276
    192.168.147.1  255.255.255.255         On-link     192.168.147.1    276
  192.168.147.255  255.255.255.255         On-link     192.168.147.1    276
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    306
        224.0.0.0        240.0.0.0         On-link     192.168.147.1    276
        224.0.0.0        240.0.0.0         On-link        10.1.56.64    281
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    306
  255.255.255.255  255.255.255.255         On-link     192.168.147.1    276
  255.255.255.255  255.255.255.255         On-link        10.1.56.64    281
===========================================================================
```

SURA: After Running VPN

❖ Interfaces

```
PS C:\Users\kevin> ipconfig

Windows IP Configuration


PPP adapter Syracuse University Remote Access VPN:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 128.230.153.98
   Subnet Mask . . . . . . . . . . . : 255.255.255.255
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Wireless Network Connection 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wireless Network Connection:

   Connection-specific DNS Suffix  . : syr.edu
   Link-local IPv6 Address . . . . . : fe80::30c5:d02c:ed1d:2d2e%13
   IPv4 Address. . . . . . . . . . . : 10.1.56.64
   Subnet Mask . . . . . . . . . . . : 255.255.192.0
   Default Gateway . . . . . . . . . : 10.1.0.1
```

❖ Routing table

```
IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0         10.1.0.1       10.1.56.64     25
         10.1.0.0    255.255.192.0         On-link        10.1.56.64    281
       10.1.56.64  255.255.255.255         On-link        10.1.56.64    281
      10.1.63.255  255.255.255.255         On-link        10.1.56.64    281
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    306
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    306
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    306
      128.230.0.0      255.255.0.0   128.230.153.30   128.230.153.98     21
   128.230.153.11  255.255.255.255         10.1.0.1       10.1.56.64     26
   128.230.153.98  255.255.255.255         On-link    128.230.153.98    276
    192.168.147.0    255.255.255.0         On-link     192.168.147.1    276
    192.168.147.1  255.255.255.255         On-link     192.168.147.1    276
  192.168.147.255  255.255.255.255         On-link     192.168.147.1    276
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    306
        224.0.0.0        240.0.0.0         On-link     192.168.147.1    276
        224.0.0.0        240.0.0.0         On-link        10.1.56.64    281
        224.0.0.0        240.0.0.0         On-link    128.230.153.98    276
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    306
  255.255.255.255  255.255.255.255         On-link     192.168.147.1    276
  255.255.255.255  255.255.255.255         On-link        10.1.56.64    281
  255.255.255.255  255.255.255.255         On-link    128.230.153.98    276
===========================================================================
```