A Simple, Configurable, and Adaptive Network Firewall for Linux

James M. Westall

Department of Computer Science
Clemson University
Clemson, SC 29634

email: westall@cs.clemson.edu
http://www.cs.clemson.edu/˜ westall/homepage.html

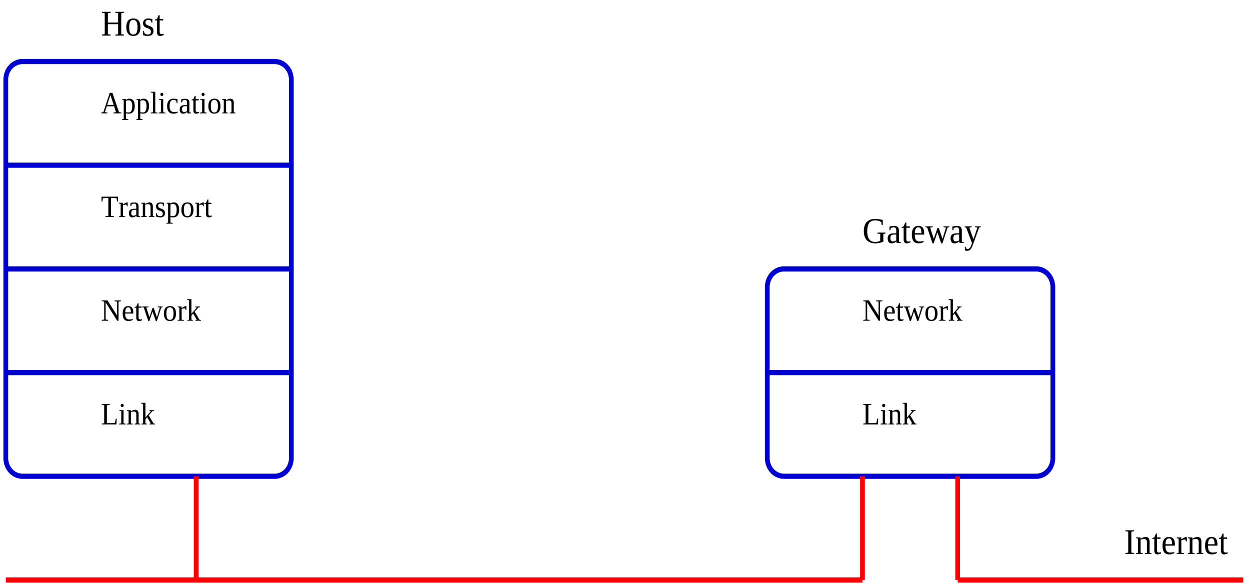
The Nature of the Problem

- 116,209,789 Internet hosts at 3:00pm on 12 March *
- Present growth rate is about 1 per second
- Most are end-user administered
- Hack scripts and virus kits are widely available
- So are ”script weenies”
- Law enforcement is overextended at best ...
- ... uninterested at worst

*As reported by http://www.netsizer.com

Defending Against the Problem

- *Fix* the application and system software!
- Prevent the attack from reaching the defective software.
- Use a *Defense in depth*



The term *Firewall* is used to describe any mechanism used to prevent the delivery of a packet associated with an a attack to its target.

Objectives of Defense Mechanisms

Desirable properties:

- Safety
- Unobtrusiveness
- Simplicity
- Efficiency

Fundamental tensions:

- Safety vs Unobtrusiveness
- (Safety + Unobtrusiveness) vs (Simplicity + Efficiency)

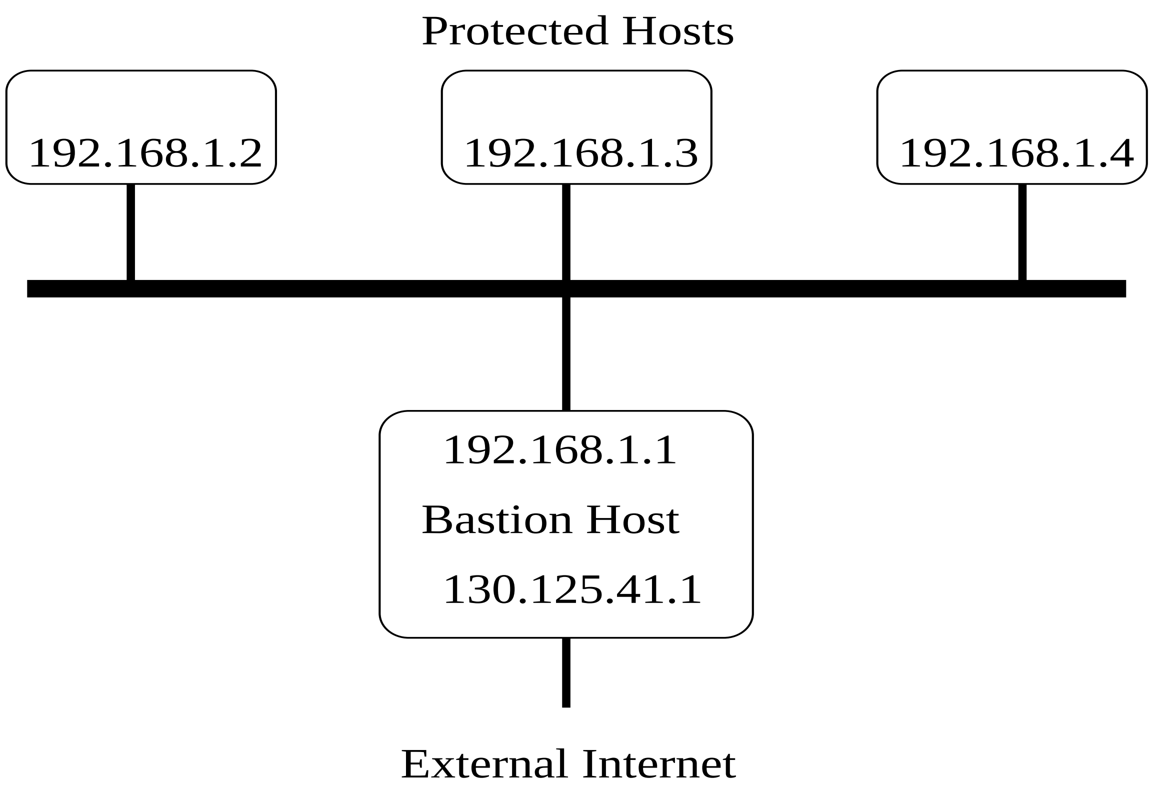
Security objectives *should be* site dependent

Network Defense Mechanisms

- Packet filters
  - Host or router based
  - Driven by *filter rules*
  - Rule matching driven by pkt hdr contents
  - Permissive and non-permissive rules
    - Permissive rules: unobtrusive but not safe
    - Non-permissive rules: safe but obtrusive
  - Static rule sets are typical
- Stateful inspection filters
  - Dynamic rule sets supported
  - Rule matching logic includes connection state
  - Tend to become complex

Network Defense Mechanisms

Network Address Translation (NAT) gateways



IP-In	Port-In	IP-Out	Port-Out
192.168.1.2	1027	201.14.12.1	1035
192.168.1.3	1027	211.12.12.4	1036
192.168.1.4	1027	211.12.12.1	1037
192.168.1.2	1028	201.14.12.9	1038

Disadvantages:

- Require a dedicated bastion host
- How to defend the bastion??
- Can be obtrusive to some apps

Firewalls in Linux

Packet filters and NAT gateways supported

- *ipfwadm* in kernel 2.0.x
- *ipchains* in kernel 2.2.x
- *iptables* in kernel 2.4.x

Limitations

- Packet filters either permissive or non-permissive
- NAT gateways need bastion host
- NAT is non-trivial to configure

A solution: *ad hoc* firewalls

Firewalls in Linux

Firewalls export three packet handlers

```
struct firewall_ops fw_ops =
{
  0,               /* Next firewall */
  fw_forward,      /* Forward       */
  fw_input,        /* Input         */
  fw_output,       /* Output        */
  PF_INET,         /* PF            */
  255              /* Priority      */
};
```

A firewall can be built as an installable module.

The packet handlers are registered at install time.

```
int init_module(void)
{
   int rc;
   rc = register_firewall(PF_INET, &fw_ops);

/* Kernel routines use printk to  */
/* print to the system log.       */

   printk("Reg_Fw returned %d \n", rc);
   return(rc);
}
```

Firewalls in Linux

A permissive input packet filter:

```
int fw_input(
struct firewall_ops *this,
int pf,
struct device *dev,
void *phdr,
void *arg)
{
   unsigned int     addr;
   struct iphdr     *iph;
   struct fwnetype  *ne;
   int              rc = FW_ACCEPT;

   iph = (struct iphdr *)phdr;
   addr = ntohl(iph->saddr);

   if (is_badguy(addr))
      rc = FW_BLOCK;

   return(rc);
}
```

The *fw* firewall

Design objectives

- Safety $\approx$ that of a non-permissive firewall
- Obtrusiveness $\approx$ that of a permissive firewall
- Simplicity and Efficiency $\approx$ that of simple packet filter

Design approach

- Dynamic rule creation
  - Don't talk to me unless I talk to you first
- Soft state
  - Your privilege to talk to me expires in n seconds...
  - ... unless I renew it by talking to you.

The *fw* firewall

The rules that control the operation of *fw* are structures consisting of four elements.

```
typedef struct
{
   unsigned int prefix; /* IP addr pfx */
   int          pfxlen; /* Pfx length  */
   unsigned int action; /* Action bits */
   unsigned int timeout;/* Expiry time */
}  fw_rule_t;
```

Rule matching is based upon *remote IP address* with usual longest-prefix-match wins tiebreaker. Action bits dictate responses to matched rules

```
#define DENY    1 /* Drop IN and OUT (mod CREATE) */
#define ALLOW   2 /* ~Deny (redundant)            */
#define LOG     4 /* Log rejections and creations */
#define DYNAM   8 /* Dynamically created rule     */
#define CREATE 16 /* Create new rule on OUT       */
```

When the *CREATE* bit is present in a *DENY* rule matching an *OUTPUT* packet, a new rule is created with:

- action = DYNAM
- prefix = destination IP address/32
- timeout = 120 seconds

When the *DYNAM* bit is present in a rule matching an *OUTPUT* packet, the timeout is refreshed.

The *fw* firewall

A sample rule set:

```
fw_rule_t rule_base[MAX_RULES] =
{
  0x00000000,   0,   CREATE | DENY | LOG, -1, /* All     */
  0x00000000,  32,   DENY | LOG, -1,    /* 0.0.0.0      */
  0x3f0a0000,  16,   DENY | LOG, -1,    /* UUnet DHCP   */
  0x827f3000,  24,   ALLOW,      -1,    /* 130.127.48   */
  0x827f3800,  24,   ALLOW,      -1,    /* 130.127.56   */
  0xc0a80100,  24,   ALLOW,      -1,    /* ATM CLIP net */
  0xc0a80200,  24,   ALLOW,      -1,    /* ATM LANE net */
  0x7f000000,   8,   ALLOW,      -1,    /* Local host   */
  0x827f0e0e,  32,   ALLOW,      -1,    /* Mickey       */
};
```

- First rule matches any IP address (but loses to any other matching rule
- First rule allows dynamic rule creation
- Second rule matches only address 0.0.0.0
- Second and third rules block input and output
- Remaining rules enumerate trusted nets and hosts

Performance Evaluation

Safety

- Who are we vulnerable to?
- How long does vulnerability last?

Assuming things work as advertised... *fw*, NAT gateways, and SIFWs reduce exposure

- From 10's of millions of hosts to
- 10's of recently contacted hosts

But...

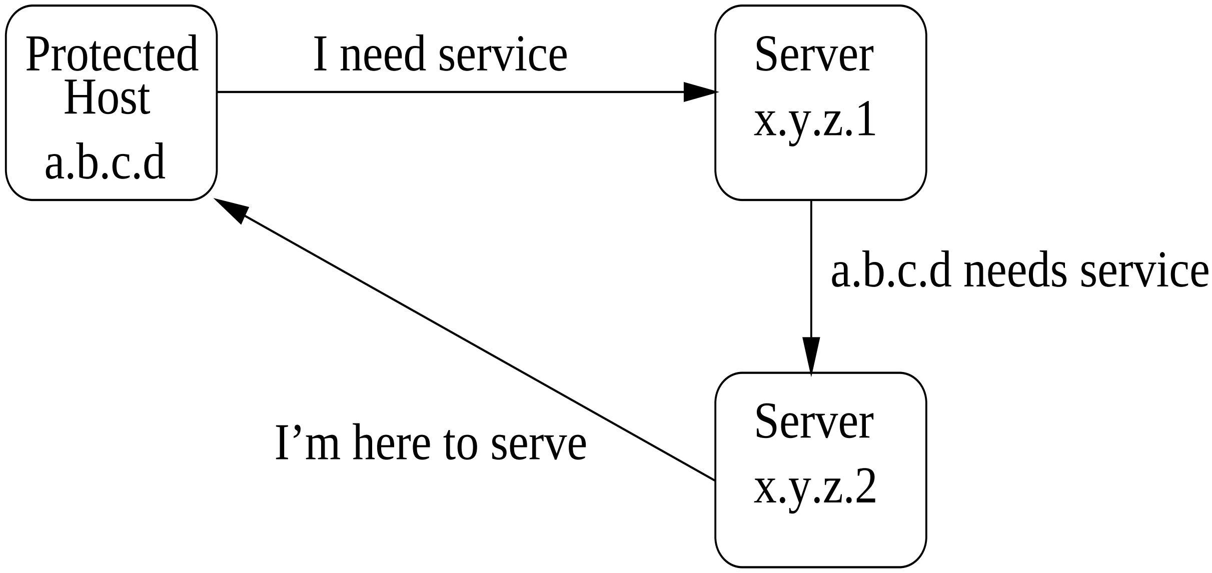
- port and state constraints of NATGW's and SIFWs...
- provide marginally better protection than *fw*

All limit exposure *time* to short timeout period.

Performance Evaluation

Obtrusiveness

- *fw*, unlike NAT, supports port/ip in data stream
- NAT doesn't support "handoff"
- *fw's* handoff support depends on default prefix len



But handoff rejection has advantages....

In conclusion...

- No system running *fw* is known to have been hacked
- Ensuring that *fw* *is* installed *is* harder than expected.
- Open source ENABLES INNOVATION...
- ... OS/360 *was* open source to those who ran it!!