HotZone

Theory of Operations

Configuration

Management

What is HotZone?

- ***Free*** low-interaction honeypot
- Source code available (not “Open Source™” but freely downloadable)
- Designed to be kitted up as a standalone system used in cooperation with a firewall or other intrusion detection capability
- Designed for easy customization
- Minimalist approach for maximum security

Basic Properties

- Automated Interaction
- Diagnosis
- Capture
- Automated Logging
- Easy Install
- Rule Sharing
- Easy Update

Basics: Automated Interaction

- Rather than exposing “real” applications to attack, emulate them well enough to fool automated tools
- Emulation performed in minimized “safe” executables that don’t have the security flaws in the target executables
- Emulation is driven by simple scripting language (Costume Shop) that can simulate many servers

Automated Interaction: Down Sides

- Since the system is not a “real” operating environment, a hacker should be able to pretty quickly realize that they are talking to an emulator
- Requires configuring emulators (rather than installing default buggy software)
- May not be able to emulate everything (some protocols that use encryption, etc)

Automated Interaction: Up Sides

- No worry about hackers using your honeypot as a jump-off point for further attack
- Very unlikely that it will be compromised past interaction layer
- Does not contain known vulnerabilities like a classic honeypot
- Easier to maintain and install

Basics: Diagnosis

- Identify external actions against internal knowledge of their meaning

```
Jul 4 17:30:23 hotzone ftpd[4003]: USER FTP
Jul 4 17:30:23 hotzone ftpd[4003]: PASS shellcode
Jul 4 17:30:23 hotzone ftpd[4003]: ANONYMOUS LOGIN FROM
   attacker.example.com [16.67.32.1] shellcode
Jul 4 17:30:23 hotzone ftpd[4003]: SITE EXEC (lines: 0)
   %.f%.f%.f%.f%.f%.%.f%.f… [repeated]
```

-> “Ramen Worm attack”

Basics: Capture

- Collect components of remote interactions for later analysis
- Application-level “traffic vacuum” - see what the Bad Guy has to say
  - Provide basic parsing of data into components for higher-level analysis

```
<rec>
<srcip>16.67.32.1</srcip>
<url/scripts/root.exe?c+dir</url>
</rec>
```

Basics: Capture

- State-driven interaction engine

```
state start {
    output “Login:”
    goto state username
}
state username {
    read line tokenize as “%s”
    username = $1
    output “Password:”
    goto state password
}
```

- Make it *look* like a login process!!

Basics: Automated Logging

- HotZone uses the “fargo” log processor
- Basic log processing should allow combination of multiple events into a single event record
  - Who wants 40 URLs if they all (taken together) mean “Code Red Scan”?
- Automate log rolling and cleaning
- Basic summarization and statistics

Basics: Easy Install

- HotZone boots from its own Cdrom
  - Operating system and executables pre-installed
  - No end-user configuration necessary (or even possible!)
  - No UNIX system administration headaches (no password file, inetd.conf, etc, etc)
  - Single purpose system simplifies internal configuration: make lots of assumptions

Basics: Rule Sharing

- HotZone rule-bases (configurations for costume shop and fargo) are automatically updated over the network (if you want) without requring intervention
- Rules are PGP-signed for authenticity and are checked before installing
  - You can specify whose rules you want to rely on

Basics: Easy Update

- HotZone executable programs are automatically updated over the network (if you want) without requiring intervention
  - This allows enhancement of capabilities / instant upgrade
- Executables are PGP-signed for authenticity and are checked before installing

Configuration

- Basic configuration uses get/put service
  - Copies configuration files to/from configuration directory
  - Configuration directory contents can be downloaded to your machine, edited, and pushed back onto HotZone
  - HotZone uses layered security model to prevent accidental access to critical files

Internal Security Model

- 4 Layers separated from each other
  - System Layer
  - Executable / Privileged Layer
  - Configuration Layer
  - Interaction Layer
- All operations are performed at a lower layer than they are started
  - Prevents security flaws from being exploited to gain increased access

System Layer

- Bootstrap, CDRom kernel, /device files
- Located both on hard disk(mutable) and on CDRom(immutable)
- Compromising this layer would compromise the entire system
  - Accessible only at boot-time and console: ***physical*** access *required*

(We take it as an item of faith that no system is secure *no matter how hard we try* if a bad guy gets his paws on it!)

Executable Layer

- All executable programs runnable on the system (get server, costume shop, fargo, etc) - ***also*** file logs and output
- When started chroot(2) to appropriate working directories
  - Any applications that have encryption keys move out of file tree where crypto keys reside once they have accessed the key(s)

Configuration Layer

- Configuration files used by fargo and costume shop
  - Each kept in separate chroot(2) area maintained by the executable layer
- Output produced by fargo and costume shop
  - Kept in an output directory for further analysis
  - Spool area for Email off system

Interaction Layer

- Totally unprivileged layer
- This is where costume shop runs and interacts with the Bad Guys
  - Process is chrooted into an area where there are no devices and hardly any files for it to tamper with
  - Logs are sent to fargo over a fifo - implementing a secure one-way data exchange across the chroot

Some Internals

- Sample File Tree:

```
/- System Layer
        /bsd    kernel image (if none, used version on CDRom)
        /dev    (devices - used for mounting f/s and console)
        Executable / Privileged Layer
        /exe    (executable images)
                /exe/fargo - logger
                /exe/getserv - remote file updater    … etc …
        /cf - Configuration Layer
                /cf/system - global configuration directory
                /cf/pgp - PGP key rings
                /cf/auth - encryption keys for file updater
        /cf/interact - Interaction Layer
                /cf/interact/logs
                /cf/interact/logs/fargo - fargo output
                /cf/interact/costumes - costume shop
                /cf/interact/wc.fifo - FIFO for fargo data
```

Internal Utilities

- No user level access to these but they're what keeps the system running inside
  - Get/Put and Getserv
  - Minimail
  - GPG
  - Monitor
  - Updater
  - Klogger

Get/Put and Getserv

- Encrypted command-line oriented non-interactive file transfer tool
- Used to get and put configuration files to system
  - This is the root of the entire “remote management” capability
  - All remote management below the System layer is done by modifying files in /cf/system
  - Can put wrappers (e.g. GUI) on top of that

Minimail

- Minimalist outgoing-only SMTP sender
  - Used to deliver outgoing mail over a subset of SMTP (RSET, MAIL, DATA, QUIT)
  - Primary mechanism for delivering real-time alerts or informational messages
  - Doesn’t use MX records or deliver to destination - only knows how to deliver to a “smart” mail gateway for further delivery
  - Will queue/retry if link to gateway is down

GPG

- Gnu Privacy Guard
  - By ***far*** the most complex piece of software in the system(!)
  - Used to check signatures on new rules
  - Used to check signatures on new executables
  - ***Not*** used for encryption

Monitor

- Oversees boot process, filesystem mounting, and run state
  - Replacement for UNIX init function
- Provides System Layer console interaction
  - Allows re-keying administrative access
  - Allows importation of new PGP keys and adjusting their trust level
- Displays system status while running

Updater

- Minimal HTTP client for pulling down new rules and executables
  - Proxy aware
  - Connects to HotZone update site and accesses new rules or executables
  - Downloads them
  - Checks their PGP signature for authorization
  - Installs them if authorized

Klogger

- Reads kernel logs (some of the honeypot is in kernel mode) and parses them into format suitable for fargo
  - Manages /dev/klog
  - Outputs XML based on its own configuration rules (/cf/klogger/rules)

Core Utilities

- These are the “guts” of the system - the parts that do the real work
  - Fargo
  - Costume Shop
  - In-kernel IP-level logging hacks

Fargo

- Advanced log processor
- Treats input records in a pseudo-XML markup language
- Extensive rules for matching, coalescing, rewriting, input and output
- Statistics keeping and analytics
- Extensive output capabilities (text or XML)

Costume Shop

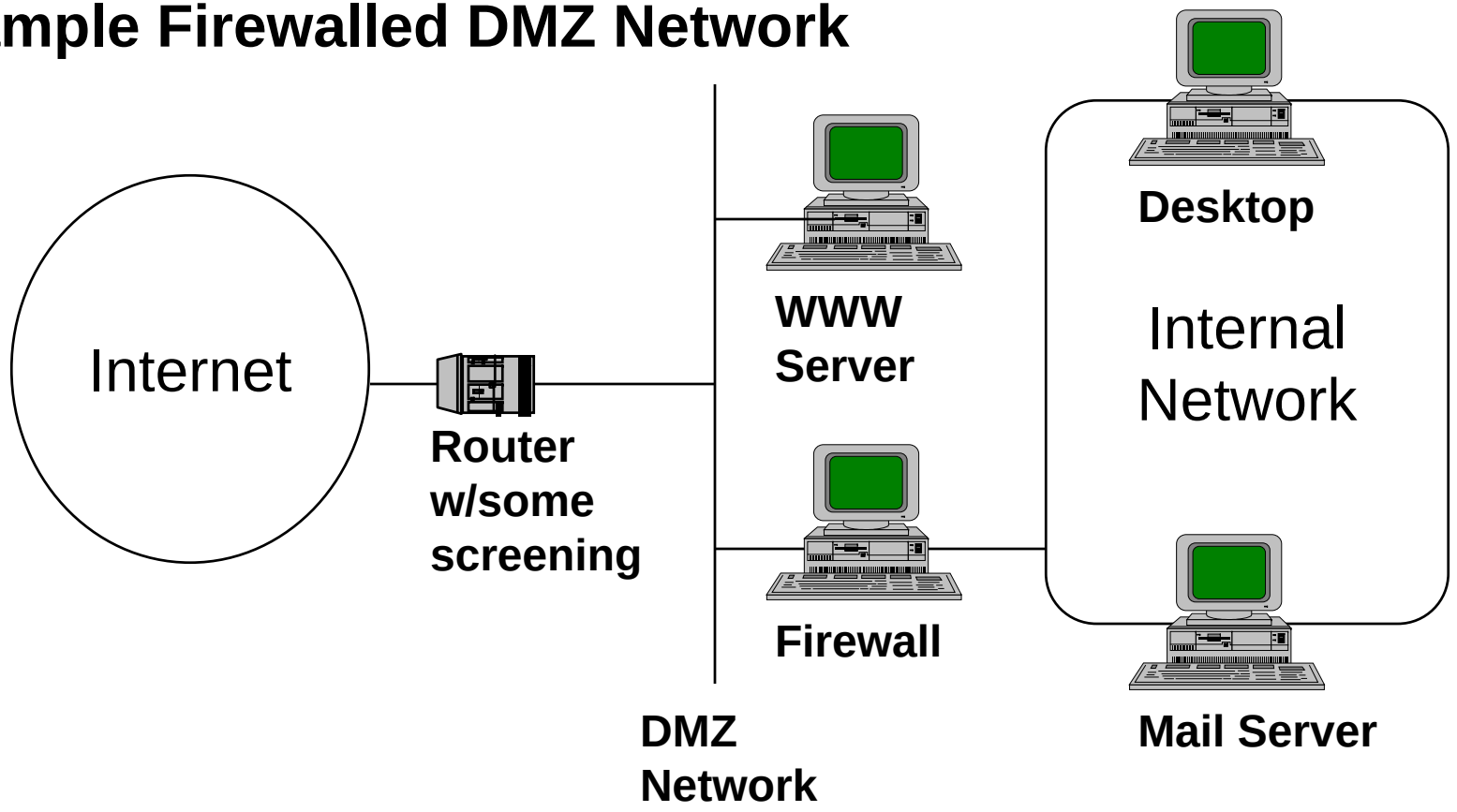
- Safe Interaction driver
- Uses state-driven scripts to interact with external processes
- Can record chunks of interactions and pass them to Fargo for processing
- Can emulate connection-oriented and packet-oriented services

In-Kernel hacks

- Modifications to underlying kernel (OpenBSD kernel) IP stack record unusual IP-level traffic that might indicate a scan
- Not programmable (static C code hacked into IP stack)
- Messages passed out of kernel via /dev/klog to Klogger into Fargo
- Useul for detecting scans and DOS

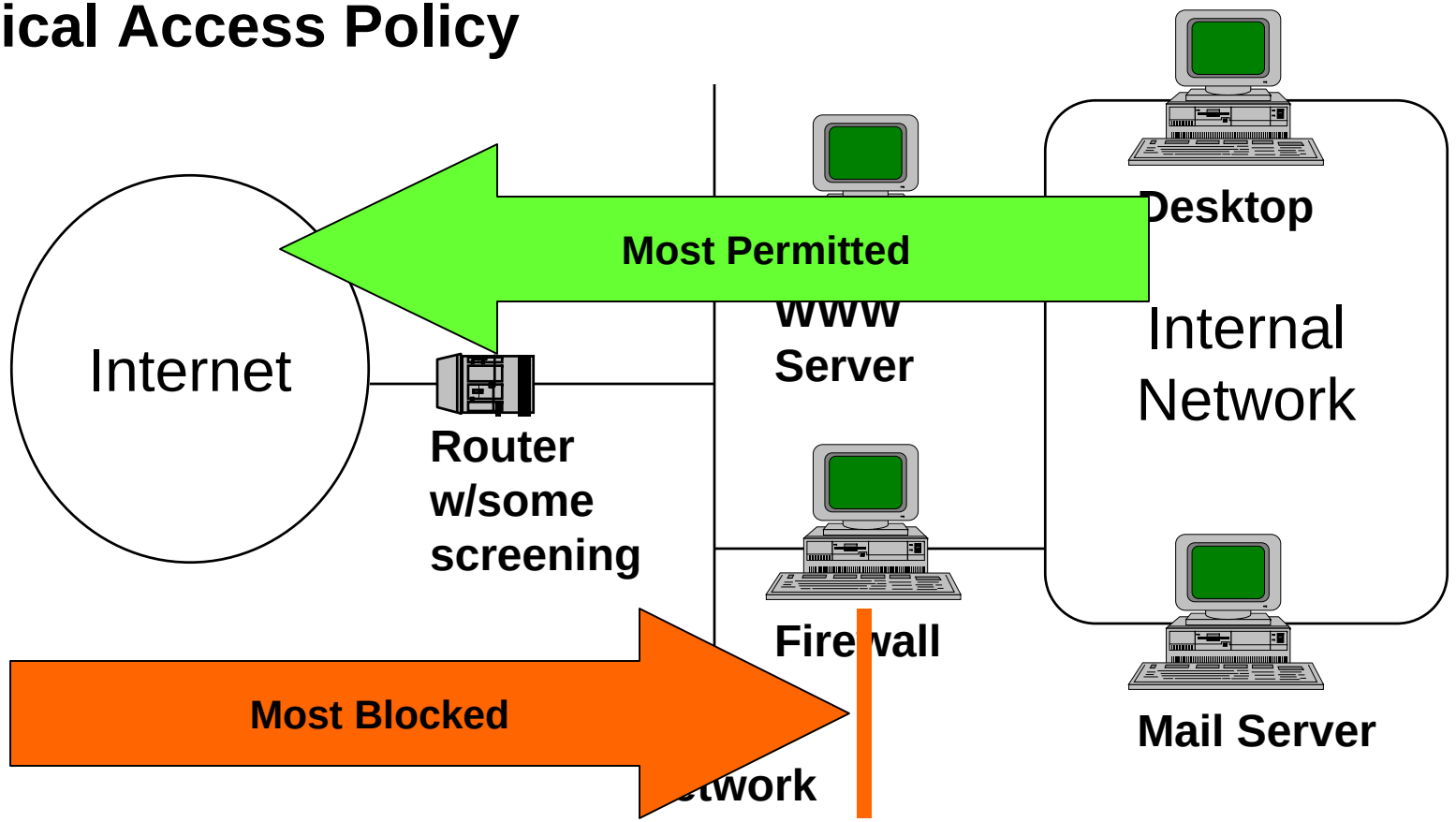
Setting up a HotZone

Example Firewalled DMZ Network



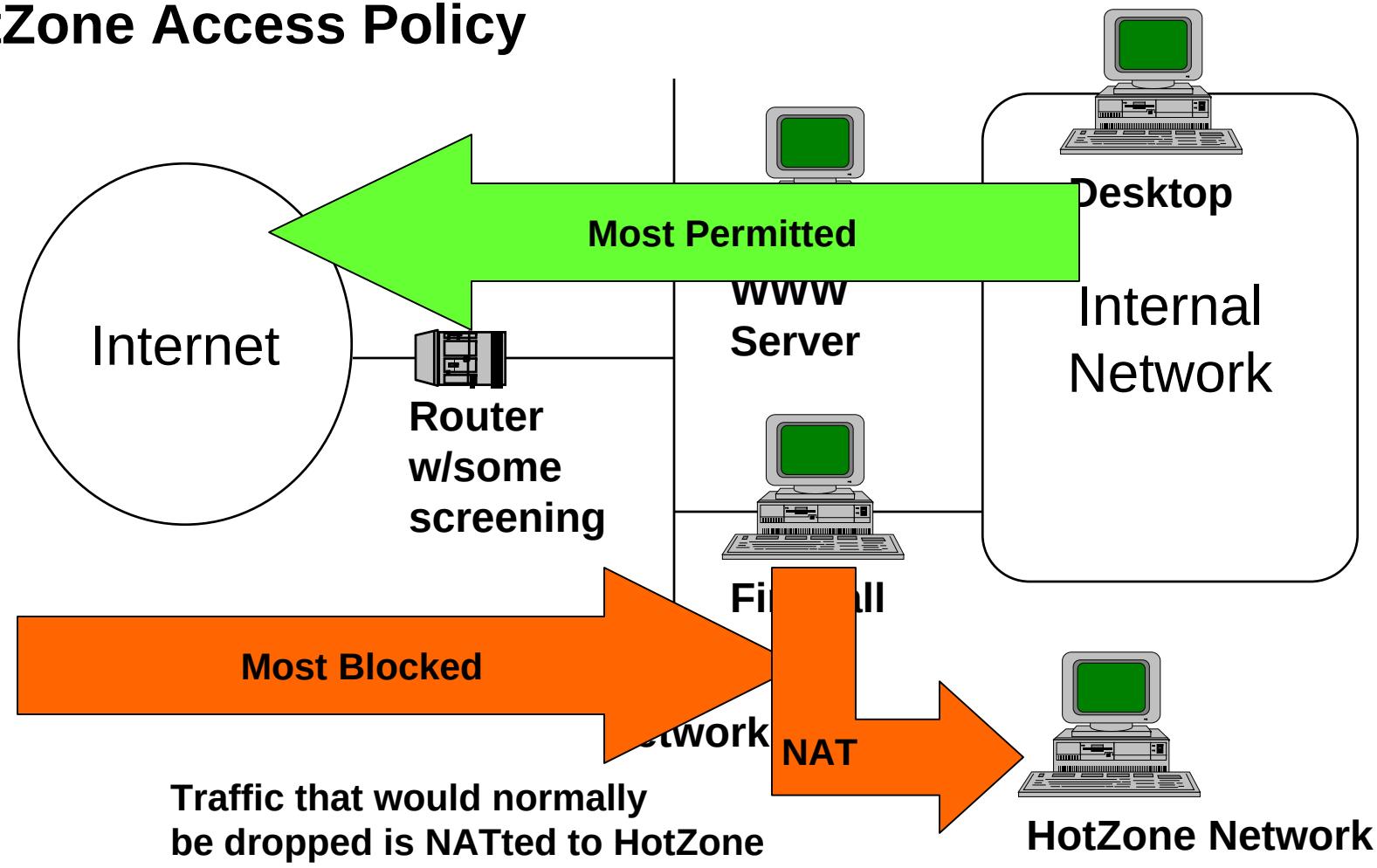
Setting up a HotZone

Typical Access Policy



Setting up a HotZone

HotZone Access Policy



Management

- For now management is pretty rudimentary
  - Console
  - Via get/put of configuration files
- Eventual plans are to write a GUI (in TCL/TK and in visual basic) on top of the basic get/put functionality
  - Volunteers? ;)

Summary

- This has been an overview of HotZone
- Typical deployment
- Purpose
- Internals
  - Processes
  - Layout of filesystems
  - Management