



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Auditing Systems, Applications, and the Cloud (Audit 507)"
at <http://www.giac.org/registration/gsna>

IPCop Audit: A Home User's Perspective

GSNA version 3.1

Option 1

Garret Cox

June 14, 2004

Abstract

I recently installed [IPCop](#) on a box to use as the [firewall](#) for my home network. I would like to insure that it is as secure as possible. What better way is there to accomplish this than to apply all of the nifty things I learned at the SANS Auditing course I attended recently?

Table of Contents

Section 1.1: Identify the System

Section 1.2: Evaluate the Most Significant Risk to the System

Possible Threat Vectors:

Services at Risk:

Vulnerabilities:

Section 1.3: Current State of Practice

Section 2: Audit Checklist

- 1) Perform a Vulnerability Scan
- 2) Ensure Regular Vulnerability Scans
- 3) Audit Firewall Rulebase
- 4) Audit Firewall Rulebase Changelog
- 5) Audit Firewall Rulebase Change Control Procedures
- 6) Scan IPCop for UnStealthed Ports
- 7) Audit Static Ingress Filtering
- 8) Audit Static Egress Filtering
- 9) Audit for Dangerous Ports in Both Directions
- 10) Ensure All Other Incoming Traffic is Controlled Statefully
- 11) Ensure the Firewall can Withstand a DoS attack
- 12) Ensure Up to Date Patch State
- 13) Audit Patch Update Policy
- 14) Ensure Firewall Logging and IDS are Enabled
- 15) Audit Firewall/IDS Log Checking Policy
- 16) Ensure Access Logging and File Verification are Enabled
- 17) Audit Access/File Verification Log Checking Policy
- 18) Ensure NTP is in use
- 19) Ensure Secure Administration

Section 3: Audit Testing, Evidence, Findings

- 1) Perform a Vulnerability Scan
- 3) Audit Firewall Rulebase
- 6) Scan IPCop for UnStealthed Ports

- 7) Audit Static Ingress Filtering
- 11) Ensure the Firewall can Withstand a DoS attack
- 12) Ensure Up to Date Patch State
- 14) Ensure Firewall Logging and IDS are Enabled
- 16) Ensure Access Logging and File Verification are Enabled
- 18) Ensure NTP is in use
- 19) Ensure Secure Administration

Section 4.1 Executive Summary

Section 4.2 Audit Findings

Section 4.3 Audit Recommendations

Section 1.1: Identify the System

IPCop: IPCop is a free Linux distribution that can be downloaded from www.ipcop.org.

The FAQ page on ipcop.org states “OLD PC + IPCOP = Secure Internet Appliance”. This motto is evident during the installation and configuration of [IPCop](http://ipcop.org), resulting in an intuitive installation processes and simple web based configuration GUI, reminiscent of those seen in the Linksys and D-Link Firewall/NAT devices on the market.

Unfortunately, this simplicity leads to a lack of control when configuring more specific options. The most evident example of this is the fact that there is no GUI for specifying inbound/outbound rulesets. Instead, [IPCop](http://ipcop.org) seems to take a “deny all” approach for incoming traffic originating from the outside, employing a stateful packet filter to ensure that only traffic requested from an internal client is allowed back through the [firewall](#). The most control the user has over this process is the ability to allow access to specific ports on the [firewall](#), while forwarding others to [internal clients](#) as needed.

However, unlike these appliances, with [IPCop](http://ipcop.org) we have the benefits inherent in using a secure Linux distribution as your [firewall](#): anything you don’t like, you can change! If [IPCop](http://ipcop.org) allows some dangerous incoming/outgoing traffic, we can append our own iptables rules. If it is responding to certain dangerous traffic, we can change the appropriate values in the `/proc/sys/net/ipv4` filesystem. If [IPCop](http://ipcop.org) is running a vulnerable service, we can patch it ourselves.

Role:

[IPCop](http://ipcop.org) is installed a single Pentium2/200 with 3 network interfaces to handle “everything” for a home office network.

In the home office, [IPCop](http://ipcop.org) specifically handles:

- Firewall to prevent external access to internal network.
- Support for a DMZ for an http/ftp server.
- IDS (Snort).
- DHCP for internal network.

NAT for internal network.

The home office **IPCop** box provides an internet connection for my entire network through a single public IP address. It automatically assigns and keeps track of various IP addresses for my internal network, allowing my dynamic IP laptops to connect to the network with ease. It also provides routing from the internet to a personal http/ftp server I have set up for friend/family use. It does all of this while (hopefully) protecting all of my internal clients from unauthorized persons and worm outbreaks.

IPCop setup:

IPCop refers to the 3 interfaces as red (external), yellow (dmz), and green (internal). Access is restricted red->green, red->yellow, and yellow-> green, but not so much so in the reverse. For testing, the interfaces are set with the following IP configurations.

Green: 192.168.0.1/255.255.255.0 subnet

Yellow: 192.168.1.1/255.255.255.0 subnet

Red: 192.168.2.1/255.255.255.0 subnet

I've temporarily set up the following testing environment:

A laptop, called "**redbox**" with IP Address **192.168.2.7** connected to a switch connected to the **IPCop**'s red interface. This laptop will mostly be used to attempt to gain some access to the yellow and green zones.

A desktop, called "**yellowbox**" with IP Address **192.168.1.7** connected to the yellow interface by a crossover cable. This desktop is running a simple WinXP Pro IIS http/ftp server. This machine will be used to detect intrusion from **redbox**.

A laptop, called "**greenbox**", with IP Address **192.168.0.7** connected to a hub connected to **IPCop**'s green interface. This box will mostly be used to detect intrusion from **redbox**. It will also be used to cause some havoc from inside the network, to see both what **IPCop**'s IDS detects, as well as what inappropriate activity **IPCop** will let an internal user get away with.

The scope of our audit will be the **IPCop** installation itself. As stated above, it is impossible to determine from the **IPCop** GUI exactly how the box is configured. This becomes the primary motivation for our audit: determining how **IPCop** filters various forms of inappropriate IP traffic, ensuring that a threat vector can not somehow circumvent **IPCop** to cause damage to my internal clients. I will also ensure that **IPCop** doesn't possess any vulnerabilities by which some threat vector could compromise or disable my **firewall**, cutting off the services outlined above.

To accomplish this goal, I will perform vulnerability and port scans (using NeWT and NMapWin) and send custom crafted packets (using Hping2), all while capturing network traffic with Ethereal and using the logging functionality built into [IPCop](#) to determine what effect such behavior has on [IPCop](#) and the internal clients it protects. I will also demonstrate how to access various aspects of [IPCop](#)'s GUI configuration interface, and offer recommendations targeted at ensuring the most secure possible configuration.

A parallel goal to this audit will be the creation of an audit checklist which a marginally tech savvy user could use to audit (and further secure) their own installation of [IPCop](#). This checklist will not only provide system recommendations to improve security, but also policy and procedure advice to ensure an [IPCop](#) installation stays secure and reaction time to an attack is minimized.

Section 1.2: Evaluate the Most Significant Risk to the System

Possible Threat Vectors:

Unauthorized Person

Description	An unscrupulous individual from somewhere on the internet could try to infiltrate and subvert my firewall /home network for a variety of nefarious purposes.
Consequences	Vandalism: Deleted files, Unusable PCs. Subversion: I could end up unknowingly hosting a warez or mp3 server. My bandwidth could be employed in a DDoS attack or my computer could serve as a spam bot.
Likelihood	Medium: I'm not exactly a target for industrial espionage. There is, however, definite motivation for an attacker to infiltrate my network, for the reasons stated above.
Severity	High: My home network contains half a terabyte of data that would be worthless to others, but is very valuable to me. In the cases of subversion, legal implications could result, especially if the RIAA got after me.

Worm

Description	A worm spreading haphazardly over the internet could exploit some known vulnerability in IPCop .
Consequences	Very similar to an unauthorized person. Infiltration by a worm could allow an unauthorized person access to my network.
Likelihood	Low-one doesn't see many worms of this type targeting non-MS systems.
Severity	High-as explained above

Administrator

Description	Being the only legitimate administrator of the firewall /home
-------------	---

	network, I could screw something up.
Consequences	Loss of IPCop functionality
Likelihood	Low
Severity	Low

Services at Risk:

This list is organized by scenario.

IPCop is shutdown/ inoperable

Consequences	Loss of internet access. Machines which depended on DHCP would loose network functionality. All access to ftp/http servers in the DMZ would be lost
Vectors	This could be cause by an unauthorized person, worm, or administrator
Likelihood	Medium-A worm or troublemaker would be more interested in subverting the firewall . There is always a good chance, however, that I will tweak the configuration into oblivion.
Severity	Low-to fix it, I just reboot/ reinstall IPCop

IPCop is subverted

Consequences	Firewall could be used to gain access to network resources, eventually resulting in either damage or subversion of network resources
Vectors	Unauthorized person or worm
Likelihood	Low (Hopefully)-the purpose of this audit is to ensure this.
Severity	High-irreplaceable loss of files could result, legal implications could ensue.

IPCop is circumvented

Consequences	If IPCop is misconfigured or simply vulnerable, an attacker could damage or subvert network resources without having to take it into consideration
Vectors	Unauthorized person or worm
Likelihood	Low (Hopefully)-the purpose of this audit is to ensure this
Severity	High-same as above

Vulnerabilities:

ISP Failure/ Denial of Service Attack

Description	Loss of Internet Access
Resulting Scenario	IPCop is inoperable
Likelihood	Low-Good ISP and little motivation for such an attack
Severity	Low-I'll just loose internet access for a little while.

Inadequate Physical Security

Description	If someone got into my office, they could tamper with IPCop directly.
Resulting Scenario	IPCop is inoperable/ subverted/ circumvented
Likelihood	Low-internal users are considered trustworthy
Severity	High-Someone of malicious intent could do a lot of damage

Administrative Misconfiguration

Description	I could configure IPCop incorrectly, either rendering it inoperable, or opening an avenue of attack
Resulting Scenario	IPCop is inoperable subverted/ circumvented
Likelihood	Medium-the audit should improve this
Severity	High-this could easily create an open door for attackers

Improper IPCop Implementation

Description	Despite being properly configured, IPCop could exhibit some specific vulnerability or inappropriate behavior. This may include behavior for which there is no user configurable option.
Resulting Scenario	IPCop is subverted/ circumvented
Likelihood	Low- IPCop prides itself on being a very secure distribution. The audit will verify (or disprove) this.
Severity	High- a known, possibly unfixable vulnerability is an open door for attackers

Out of Date Patch State

Description	Very similar to the above, except the behavior could be easily remedied by bringing IPCop up to date
Resulting Scenario	IPCop is subverted/ circumvented
Likelihood	Medium-Patches are released as needed, but no automatic update system exists
Severity	High-once a patch is created, a vulnerability is often more well known than before.

Inadequate State Detection/ Verification

Description	If IPCop is subverted, I would want to know about it.
Resulting Scenario	IPCop could be subverted, and I wouldn't know about it!
Likelihood	Medium-Access Logging is enabled by default, but a file verification system isn't even installed
Severity	Medium-This doesn't really take affect until a system is already subverted. However, if this remains undetected, bad can become worse.

Inadequate Environment Detection

Description	If IPCop or the home network comes under attack, I want to know about it, even if the attack fails
-------------	--

Resulting Scenario	IPCop could be circumvented, and I wouldn't know about it!
Likelihood	Medium-Traffic Logging is enabled by default, but an IDS isn't, though it is installed.
Severity	Medium-similar to above, you want to know the instant someone has access to the internal network.

Section 1.3: Current State of Practice

There is very little available information in this area specific to IPCop. IPCop's main page, www.ipcop.org, has some documentation detailing general installation[1], as well as an administration manual[2] which elaborates the various configuration options of IPCop's web based controls, but no "best practices" manual, which walks through how one would actually want these options set. I would imagine that the motivation behind this is the fact that for the most part, IPCop's default configuration is exactly how most users would want it. I did, however, discover a few issues with the default configuration that the security focused administrator/auditor would want changed. I'll detail these later in the checklist.

There is also no mention of IPCop being based on any other distributions. This further limits the resources available for vulnerability research.

In regard to firewalls in general, however, there are a plethora of resources. The primary references I used were the SANS Track 7 Manuals [3][4][5]. I also took suggestions from various practicals posted on the SANS website [6][7][8][9].

Section 2: Audit Checklist

Below, when referencing the SANS Manuals, the reference format is:

[manual reference]{chapter1:page1,page2 chapter2:page1-(through)page2}

When referencing other practicals:

[practical reference]{checklist item1,checklist item2}

1) Perform a Vulnerability Scan

Reference	[3]{Linux:6} [6]{6} [8]{27}
Risk	Inadequate State/Environment Detection
Explanation	This lets us know immediately if there any well known vulnerabilities could affect our system.
Testing Procedure	I prefer to use NeWT, a Windows Port of Nessus. It can be downloaded for free from http://www.tenablesecurity.com/newt.html Once installed on a machine on the external interface (redbox), start it up, choose New Scan Task->Input the IPCop IP External Address->Choose Enable All Plugins (Since downtime isn't an issue)->Choose Scan Now.
Test Nature	Objective
Evidence	

Findings	
2) Ensure Regular Vulnerability Scans	
Reference	[3]{Linux:6}
Risk	Inadequate State/Environment Detection
Explanation	We want to be aware of all possible vulnerabilities to the firewall , even as new vulnerabilities are discovered.
Testing Procedure	The auditor should interview the system administrator, and ensure that frequent (at least weekly) scans exist in his working schedule.
Test Nature	Subjective
Evidence	
Findings	
3) Audit Firewall Rulebase	
Reference	[4]{4:17-23} [6]{5}
Risk	IPCop could be circumvented due to Administrative Misconfiguration
Explanation	We should confirm that all unstealthed/forwarded ports are currently in use. This is only a check of the firewall 's configuration. Ensuring that the behavior of the firewall coincides with the configuration is accomplished by other checklist items.
Testing Procedure	The first step is to interview the system administrator to determine which ports should be open/forwarded. With his assistance, evaluate every rule in the IPCop interface as follows: From an internal machine, go to IPCop 's IP Address https://192.168.0.1:445 to access IPCop 's web based interface. Click Services->port forwarding. Ensure that every forwarded port is currently necessary. Eliminate any that are not. Now click on "external service access". Here you can see services that are directly available on the firewall . Ensure that these available services match up with what is expected. Eliminate any that do not.
Test Nature	Determining what services should be available is Subjective. Making sure the rules match up to this determination is Objective.
Evidence	
Findings	

4) Audit Firewall Rulebase Changelog	
Reference	[4]{4:15}
Risk	IPCop could be circumvented due to Administrative Misconfiguration
Explanation	Every rule should have the appropriate documentation
Testing Procedure	From an internal machine, go to IPCop 's IP Address https://192.168.0.1:445 to access IPCop 's web based interface.

	Click Services->port forwarding. IPCop has the capability to store a comment for each rule. Ensure that each rule has such a comment, explaining the purpose of the rule. No comment system exists on the "external service access" page, so an external changelog should be maintained. Ensure that this document exists and is correct/current.
Test Nature	Objective
Evidence	
Findings	

5) Audit Firewall Rulebase Change Control Procedures

Reference	[4]{4:13-15} [
Risk	IPCop could be circumvented due to Administrative Misconfiguration
Explanation	We need to ensure that thorough procedures have been defined for the process of updating the firewall rulebase and changelog.
Testing Procedure	Interview the system administrator, ensuring the following: 1) There is a set procedure for requests and authorization of firewall rulebase changes. 2) All changes are documented in the changelog, with at least the following information: -Name of individual who added/modified rule -Date -Reason for rule
Test Nature	Subjective
Evidence	
Findings	

6) Scan IPCop for UnStealthed Ports

Reference	[4]{4:6,7,24-33} [8]{1}
Risk	IPCop could be circumvented/subverted due to Administrative Misconfiguration or Improper IPCop Implementation
Explanation	An available service running on IPCop, or one forwarded to another machine, could be an avenue of attack.
Testing Procedure	Note: This test will have already been accomplished if the NeWT scan from 1) has been performed. I'm offering these instructions in the instance a user may just want to do a quick scan for this one vulnerability. With NMapWin installed on a machine connected to an external interface, do a SYN Stealth Scan against IPCop's external interface. (Simply select SYN Stealth under mode, and put in the external IP Address of IPCop under Host). IPCop should only respond to the SYN packets on ports which have been defined as available in the IPCop rulebase. Of course, the results of this test should be correlated with the settings in the

	rulebase to confirm this.
Test Nature	Objective
Evidence	
Findings	

7) Audit Static Ingress Filtering

Reference	[4]{1:10-13,16 3:69} [6]{8} [8]{2,3,4,6}
Risk	IPCop could be circumvented due to Administrative Misconfiguration or Improper IPCop Implementation
Explanation	<p>The following packets need to be blocked:</p> <p>SourceIP from internal address space: Prevents spoofing</p> <p>SourceIP from Loopback: Prevents some DoS attacks</p> <p>SourceIP from private/unallocated address space: Prevents spoofing.</p> <p>Source routed Packets: Makes spoofing difficult</p> <p>ICMP broadcasts</p> <p>ICMP redirects</p> <p>ICMP Timestamp requests</p>
Testing Procedure	<p>For each of these tests, we will use hping2 running on a Linux box connected to the external interface to send inappropriate traffic. Hping2 can be downloaded from http://wiki.hping.org/.</p> <p>We will also have ethereal running on a machine connected to the internal interface. Before each test, we start ethereal capturing by clicking capture->start, selecting the appropriate interface, and clicking OK. After each test, we click stop, and look for any traffic that matches what we just sent with hping. If at any point ethereal logs traffic that we have sent with hping, then the firewall has failed that aspect of the test.</p> <p>It may also be useful to check the logs on IPCop. From an internal machine, go to IPCop's IP Address https://192.168.0.1:445 to access IPCop's web based interface. Click Logs->firewall. If at any point the IPCop logs show traffic being redirected to an internal address, then the firewall has failed that aspect of the test. This is true even if the address the traffic reaches is invalid, since this could result in a vulnerability if IPCop is used with some different network configuration. It can also be informative to check the IDS logs (Logs->intrusion detection system), to see what IPCop reports as suspicious.</p> <p>Note that we would want this traffic blocked even on ports that have been forwarded to another machine from the firewall. For the duration of the test, we will configure a temporary port forward on some unused port to our machine running ethereal. All appropriate hping tests will use this port. See checklist item "Audit the Firewall Rulebase" on accessing IPCop's port forwarding interface.</p> <p>As an addendum to the hping/ethereal tests, I am providing some</p>

alternative tests. These tests simply involve checking iptables and /proc/sys/net/ipv4 settings. These tests do not provide the level of certainty as the hping ones, but require only access to the **firewall** console to perform. They also outline exactly how the machine should be configured, in case the auditor needs to advise the system administrator.

Use the following hping commands to perform the test.

Key:

<fp> is the port on the **firewall** that has been forwarded to our **box** running ethereal.

<ia> is an **internal IP address** from the internal network's IP range.

<ea> is the **external IP address** of the **firewall**

<ua> is an address from unallocated address space. For address allocation, check www.iana.org/assignments/ipv4-address-space

<eba> is the broadcast address for the subnet of which the **external address** is a member. A broadcast address is one in which the host portion of the IP address consists either of all 255's.

<hpa> the IP address of the **machine** sending spoofed packets with hping.

SourceIP from internal address space

Hping2 -S -p <fp> -a <ia> <ea>

Note: the -S results in Hping2 sending a SYN packet, so if **IPCop** is misconfigured, we may get a SYN/ACK back.

Check the current iptables ruleset (by typing iptables -L in the console on the **firewall**). Look for a rule similar to this one under the FORWARD chain:

REJECT tcp -- 192.168.0.0/24 anywhere reject-with icmp-port-unreachable

SourceIP from Loopback

Hping2 -S -p <fp> -a 127.0.0.1 <ea>

SourceIP from private/unallocated address space

Hping2 -S -p <fp> -a <ua> <ea>

Source routed Packets

Hping2 -S -p <fp> -lsrr <hpa> <ea>

Ensure /proc/sys/net/ipv4/conf/all/accept_source_route = 0

Do this by bringing up the console, and typing cat <the filename above>

These last three tests will target the **firewall** directly. As such, simply checking ethereal logs on the internal client will tell us nothings. Instead, we will simply have to inspect the **firewall** behavior.

ICMP broadcasts

Hping2 -C 8 <eba>

Note: be careful to only do this test when the network is not in heavy use. This is a dangerous test in a large environment. If **IPCop** responds, the test is a failure, as **IPCop** could become a participant in a SMURF attack.

Ensure `/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts = 1`

ICMP redirects

Hping2 -C 5 --icmp-gw <hpa> <ea>

To see if this has an effect, bring up a console on the **firewall**, and type `route -C`. Then run the above command from the machine on the external interface. Finally, type `route -C`, and see if there are any new entries. If there are, the test is a failure.

Ensure `/proc/sys/net/ipv4/conf/all/accept_redirects = 0`

ICMP Timestamp requests

Hping2 -C 13 <ea>

If **IPCop** replies, the test is a failure.

Check the current iptables ruleset (by typing `iptables -L` in the console on the **firewall**). Look for a rule similar to this one under the INPUT chain:

REJECT icmp -- anywhere anywhere icmp timestamp-request reject-with icmp-port-unreachable

Test Nature	Objective
Evidence	
Findings	

8) Audit Static Egress Filtering

Reference	[4]{1:10,14,16} [6]{4,8} 8{5,6}
Risk	IPCop could be circumvented due to Administrative Misconfiguration or Improper IPCop Implementation
Explanation	We want to only allow out packets from our address space. This ensures that our NAT doesn't leak out internal addressing. This will also ensure that outbound spoofing is blocked. If logged, it can allow us to track down compromised hosts. We should also block any ICMP unreachables, as these can allow an attacker to inverse map our network.
Testing Procedure	For each of these tests, we will use hping2 running on a Linux box connected to the internal interface to send inappropriate traffic. Hping2 can be downloaded from http://wiki.hping.org/ . We will also have ethereal running on a machine connected to the external interface . Before each test, we start ethereal capturing by clicking capture->start, selecting the appropriate interface, and clicking OK. After each test, we click stop, and look for any traffic

that matches what we just sent with hping. If at any point ethereal logs traffic that we have sent with hping, then the [firewall](#) has failed that aspect of the test.

Use the following hping commands to perform the test.

Key:

<ra> is some random, but valid IP address. This address should not coincide with any hosts involved with the test.

<ethrla> is the address of the host running ethereal.

A packet from with a source address not from our internal network.

Hping2 -c 1 -S -a <ra> <ethrla>

ICMP unreachables

Hping2 -c 1 -C 3 <ethrla>

Test Nature	Objective
Evidence	
Findings	

9) Audit for Dangerous Ports in Both Directions

Reference	[4]{1:15}
Risk	IPCop could be circumvented due to Administrative Misconfiguration or Improper IPCop Implementation
Explanation	We want to statically block in both directions: Windows: TCP&UDP 135-139 and 445 (these ports are used by netbios, for general windows networking) Unix: TCP 23(telnet), 512-514(rexec,rlogin,rsh), TCP&UDP 111(sunrpc),2049(nfs),6000-6255(XWindows) General: UDP 69(tftp), 161&162 (SNMP and SNMPTRAP), 514(syslog) This ensures that connections to these services can't be created, even when initiated by an internal client
Testing Procedure	We will be analyzing traffic in both directions. When auditing incoming port filtering, we will use a setup similar to the one used for the checklist item "Audit Static Ingress Filtering". The only addendum to this is that we will set up temporary port forwards for every port to be tested to the machine running ethereal. When auditing egress traffic, we will use a setup similar to the one used for the checklist item "Audit Static Egress Filtering". As before, if any of the hping traffic appears in the ethereal or IPCop logs, the firewall has failed that aspect of the test. Key: <ea> is the external IP address of the firewall <ethrla> is the address of the host running ethereal.

<tcpp> is some value in our list of dangerous tcp ports (23,111, 135-139, 445,512-514,2049,6000-6255)
<udpp> is some value in our list of dangerous udp ports (69,111,161,162,135-139,445,514,2049,6000-6255)

Each of the tests below should be run at least once for every port in the appropriate list.

Ingress filtering test (this test should be run from a machine connected to an **external interface**):

Hping2 -c 1 -S -p <tcpp> <ea>

Hping2 -c 1 -2 -p <udpp> <ea>

Egress filtering test (this test should be run from a machine connected to an **internal interface**):

Hping2 -c 1 -S -p <tcpp> <ethrla>

Hping2 -c 1 -2 -p <udpp> <ethrla>

Test Nature	Objective
Evidence	
Findings	

10) Ensure All Other Incoming Traffic is Controlled Statefully

Reference	[4]{1:17}
Risk	IPCop could be circumvented due to Administrative Misconfiguration or Improper IPCop Implementation
Explanation	IPCop should only allow in packets for which a client actually requested a connection, by keeping track of all outbound requests (not by just allowing SYN/ACK, or ACK flagged packets) One would also ensure that fragmented packets don't make it through.
Testing Procedure	NeWT has several tests to ensure stateful packet filtering is performing correctly. If one of these tests fail, then something is wrong with the stateful packet filter and the firewall has failed.

To ensure that stateful packet filtering is working even for forwarded traffic, we can use a setup similar to the one used for checklist item "Audit Static Ingress Filtering".

Key:

<fp> is the port on the **firewall** that has been forwarded to our box running ethereal.

<ea> is the external IP address of the **firewall**

We first use hping on the external interface to send an ACK packet

to the forwarded port on the server:

Hping2 -A -p <fp> <ea>

If stateful filtering is being used on this forwarded port, this packet will be dropped (since there was no SYN packet preceding it).

If, however, ethereal on the **internal server** shows any traffic, the test is a failure. We will most likely get an RA back from this server, so ethereal may be unnecessary.

Test Nature	Objective
-------------	-----------

Evidence	
----------	--

Findings	
----------	--

11) Ensure the Firewall can Withstand a DoS attack

Reference	[6]{7}
-----------	--------

Risk	IPCop is inoperable due to a DoS attack
------	--

Explanation	<p>We want to ensure that IPCop is resistant to, and continues operation during a Dos attack</p> <p>There are basically 2 classes of DoS attacks one must be prepared for. The first type exploits some vulnerability to somehow render the firewall (and any routing functionality) inoperative. The best way to ensure your firewall will withstand such an attack is to use a vulnerability scanner to test for most of the known vulnerabilities.</p>
-------------	--

The second type of DoS attack can occur if a server is simply flooded with more traffic than it can handle. In a typical environment, there are 3 points of failure:

- 1)The bandwidth available to the site is used up, preventing any legitimate traffic
- 2)The **firewall** receives so much traffic it can no longer filter or route traffic
- 3)Servers behind the **firewall** receive so much traffic they can no longer service legitimate traffic.

The ISP is not the target of the audit, so 1) isn't our concern.

We do, however, want to ensure that the **firewall** can handle the maximum amount of traffic that may come to it. We would also like to know if the **firewall** provides any mitigation for an attack on a server behind the perimeter.

Such attacks can take the form of a simple TCP or UDP flood, though such attacks aren't the most effective. We are more concerned with either a TCP SYN attack (a flood of TCP SYN packets which fills up the new connection queue on a server) or an ICMP flood (either a simple flood of pings, which will tie up a server trying to respond, or a SMURF attack, which sends a flood

of pings to some broadcast address, with a source address of the target, resulting in the server getting flooded with echo-replies).

The TCP SYN attack will only affect a machine with open TCP ports, so this will most likely target a server behind the [firewall](#), though it could target the [firewall](#) directly if remote administration on the [firewall](#) is enabled (the traffic could be directed at whichever port the OpenSSH server is awaiting a connection on).

That being said, we need to ensure the following:

The [firewall](#) should continue operation during an icmp flood (both a simple ping flood and a SMURF attack).

The [firewall](#) should continue operation during a TCP SYN attack on the [OpenSSH](#) port (if one is available).

The [firewall](#) should continue operation during a TCP SYN attack on a port forwarded to some [internal server](#).

The [firewall](#) should provide some mitigation to help internal servers cope with a TCP SYN attack.

A note on this last point: what can a [firewall](#) do to mitigate such an attack? If the attacker doesn't spoof the source address of the attack, then the attack will consistently originate from some set of addresses (if the attack is a DDos attack, there could be thousands of addresses in this set). The [firewall](#), ideally, would automatically block all traffic from these addresses after a few moments. If the attacker does spoof the source address, they will often cycle through some set of (in)valid addresses, to prevent the above method of mitigation from being effective. The [firewall](#), however, can still help the situation by blocking all traffic from unallocated, private, and internal address space. Traffic with a Loopback source address should obviously be blocked as well.

Testing
Procedure

To perform the vulnerability scan, I prefer to use NeWT, a Windows Port of Nessus.

It can be downloaded for free from

<http://www.tenablesecurity.com/newt.html>

Once installed on a machine on the external interface ([redbox](#)), start it up, choose New Scan Task->Input the [IPCop](#) external IP address->Choose "Define my own set of plugins"->under Families, check "Denial of Service", and uncheck "Do not use dangerous plugins even if they are selected" at the bottom of the window->click Scan Now.

If any applicable vulnerabilities are discovered by the scan, the [firewall](#) has failed the test.

During each of the following tests, we will test [firewall](#)/routing functionality in both directions, by transferring files from an external machine to an internal machine, and back again.

To do this, we need 3 machines connected to the [firewall](#). One machine should be attached to a hub/switch connected to [IPCop](#)'s external interface. This machine will need hping2, which we will use to simulate the various attacks. This machine should have a fairly fast processor, as we want it to send traffic more quickly than [IPCop](#) can process it. Another machine connected to this same hub/switch on the external interface should be serving a large file(100MB+) in some way it can be easily downloaded (http is generally easy to set up and route through a [firewall](#)). Finally, a third machine connected to [IPCop](#)'s internal interface should be running Ethereal and similarly hosting some large file. The appropriate port (in the case of http, port 80) should be forwarded from the [firewall](#) to this machine. See checklist item "Audit Static Ingress Filtering" on setting up hping2 and ethereal. See checklist item "Audit the [Firewall](#) Rulebase" on accessing [IPCop](#)'s port forwarding interface.

For an http server, one can simply use IIS if it is included with your version of Windows, or use one of the many freeware http servers.

Before performing any of the DoS attacks, copy the file once in each direction, making a note of how long this takes in each direction. This will be used to determine what performance hit, if any, our attacks have on [IPCop](#) and our internal server.

If during any of the tests below, it is impossible to make a connection in either direction, or if the file copies very slowly in either direction, then [IPCop](#) has failed that portion of the test.

Key:

<ea> is the external IP address of the [firewall](#)

<sshp> port on the [firewall](#) used for remote administration (OpenSSH)

<isp> port on the [firewall](#) forwarded to the internal server (probably port 80 if using http)

<spfda> spoofed address. This is used in most DoS attacks to prevent reply traffic from reaching the attacking machine, allowing the attacking machine to spend more resources pumping out more bogus traffic.

Simulating an ICMP flood:

Hping2 -C 8 -i u1 -a <spfda> <ea>

Simulating a TCP SYN flood on OpenSSH:

Hping2 -S -i u1 -p <sshp> -a <spfda> <ea>

Simulating a TCP SYN flood on the internal server:

Hping2 -S -i u1 -p <isp> -a <spfd> <ea>

Note: if this test fails, it is likely directly due to the internal server rather than **IPCop**. However, since we are interested in **IPCop**'s flood mitigation, the internal server failing would also be a black mark for **IPCop**.

Ensure /proc/sys/net/ipv4/tcp_syncookies on the **firewall** is set to To do this, simply bring up a console and type **cat /proc/sys/net/ipv4/tcp_syncookies**

Mitigation:

Does **IPCop** make a note of addresses that have been used to initiate an attack? Does it block such traffic from such addresses? Have Ethereal logging while performing the last test above. If, after a few moments, the server stops receiving traffic, even though hping2 is still blasting, then the **firewall** has passed this aspect of the test. Otherwise, it has failed.

Testing if **IPCop** successfully blocks all traffic from private/unallocated/internal/loopback address space is covered under checklist item "Audit Static Ingress Filtering"

Test Nature	Objective
Evidence	
Findings	

12) Ensure Up to Date Patch State

Reference	[4]{4:8}
Risk	IPCop could be circumvented/subverted due to Out of Date Patch State
Explanation	All patches should be applied. If not, some easily exploitable vulnerability may exist on the firewall .
Testing Procedure	From an internal machine, go to IPCop 's IP Address https://192.168.0.1:445 to access IPCop 's web based interface. Click System->updates. You will see a list of all available updates. At the bottom of the page will be a list of uninstalled updates. If there are any uninstalled updates in this list, then the firewall has failed this part of the test.
Test Nature	Objective
Evidence	
Findings	

13) Audit Patch Update Policy

Reference	[3]{Linux:2}
Risk	IPCop could be circumvented/subverted due to Out of Date Patch

	State
Explanation	A plan should be in place to ensure the patch state stays up to date.
Testing Procedure	See checklist item "Ensure Up to Date Patch State" on accessing the IPCop patch update interface. The auditor should interview the system administrator, and ensure that frequent (at least weekly) update checks exist in his working schedule.
Test Nature	Subjective
Evidence	
Findings	

14) Ensure Firewall Logging and IDS are Enabled

Reference	[4]{4:44-51} [3]{Linux:10,23} [8]{7}
Risk	Inadequate Environment Detection
Explanation	We want to insure that if we come under attack, a simple check of the logs will reveal this to us.
Testing Procedure	From an internal machine, go to IPCop 's IP Address https://192.168.0.1:445 to access IPCop 's web based interface. Firewall Logging is enabled by default, but the IDS isn't. To enable it, click System->intrusion detection system. Ensure the option labeled "Snort" is checked. To check the firewall logs, click Logs-> firewall . To check the intrusion detection logs, click log->intrusion detection system. To ensure that the logging is actually enabled, do a NeWT or NMapWin scan (described in checkpoints 1 and 3), and check these logs. If new items show up in the firewall logs from the scanning host, then the firewall has passed this aspect of the test. If the IDS logs make not of some suspicious activity, then the firewall has passed this aspect of the test.
Test Nature	Objective
Evidence	
Findings	

15) Audit Firewall/IDS Log Checking Policy

Reference	[4]{4:49} [3]{Linux:10} [8]{25}
Risk	Inadequate Environment Detection
Explanation	We want to insure that if we come under attack, a simple check of the logs will reveal this to us. A plan should also be in effect to insure these logs are checked regularly
Testing Procedure	See "Ensure Firewall Logging and IDS are Enabled" on accessing the firewall/IDS logs. The auditor should interview the system administrator, and ensure that frequent (daily) log checks exist in his working schedule.

Test Nature	Subjective
Evidence	
Findings	

16) Ensure Access Logging and File Verification are Enabled

Reference	[4]{4:46} [3]{Linux:5,10} [6]{11} [8]{13}
Risk	Inadequate State Detection
Explanation	We want to ensure that if the system is compromised, that a quick check will reveal this to us.
Testing Procedure	<p>Ensure that /var/log/secure has a record of all login attempts. Do this by SSHing into IPCop and typing in the console: Less /var/log/secure. If this file is empty, then the firewall has failed this aspect of the test.</p> <p>Ensure that Tripwire is installed. To do this, type tripwire -m c If Tripwire is installed, this will do a full integrity check (a compare against the last known good state). Side Note: This is a good time to check the integrity report in /var/db/tripwire with the twprint utility. This isn't necessarily on the checklist, but it may reveal a compromised system!</p> <p>You should also ensure Tripwire is set up correctly by inspecting the /usr/local/etc/twpol.txt file. In this file, make sure that all files in /etc, /bin, /usr/bin, and /usr/local/bin are checked, and flagged to generate a warning if changed.</p>
Test Nature	Objective
Evidence	
Findings	

17) Audit Access/File Verification Log Checking Policy

Reference	[4]{4:46} [3]{Linux:5,10} [6]{11} [8]{13,25}
Risk	Inadequate State Detection
Explanation	A plan should also be in effect to ensure /var/log/secure and the FV Database are checked regularly
Testing Procedure	<p>The auditor should interview the system administrator, and ensure that frequent (daily) log checks exist in his working schedule. He should do a full Tripwire integrity check almost as often. Ensure that one copy of a known good Tripwire database is copied onto write protected medium. This copy should be compared against during the integrity checks.</p>
Test Nature	Subjective
Evidence	
Findings	

18) Ensure NTP is in use


Reference	[6]{9} [7]{9.2}
Risk	Inadequate Environment/State Detection
Explanation	We need to ensure the system clock is set appropriately, so that reviewing the logs makes sense.
Testing Procedure	To check NTP settings: From an internal machine, go to IPCop's IP Address https://192.168.0.1:445 to access IPCop's web based interface. Click System->time. Ensure "Enabled" is checked Ensure IPCop is set to a Primary/Secondary NTP server that you or the system admin know and trust. Ensure IPCop is set to update reasonably frequently (I like it about 12 hours). To further ensure that ntp is updating, you can check the logs by clicking Logs->other. Next to "Section", choose IPCop .
Test Nature	Objective
Evidence	
Findings	

19) Ensure Secure Administration

Reference	[3]{Linux:3} [7]{2.8}
Risk	IPCop is subverted due to Administrative Misconfiguration or Improper IPCop Implementation
Explanation	Since I am the only user of the machine, there should be only an administrative(root) login. The only methods of login should be secure (SSH or SSL, require a password). It should be confirmed that no known vulnerabilities exist for the version of OpenSSH on the firewall . We are not as concerned about the OpenSSL (Web GUI) interface, as it is not accessible from the external interface .
Testing Procedure	On the console, type less /etc/passwd, and ensure that the root account is the only account that can bring up a console. To do this, inspect every line of the passwd file, examining the last field. If an account besides root contains a viable shell (such as /bin/bash) in this last field, then the firewall has failed this aspect of the test Inspect the external services, and ensure that insecure remote access services are not enabled. See the checklist item "Audit the Firewall Rulebase" on accessing IPCop's "external services" interface. Use NeWT to ensure that no SSH vulnerabilities exist.
Test Nature	Objective
Evidence	
Findings	

Section 3: Audit Testing, Evidence, and Findings


1) Perform a Vulnerability Scan

Reference	[3]{Linux:6} [6]{6} [8]{27}
Risk	Inadequate State/Environment Detection
Explanation	This lets us know immediately if there any well known vulnerabilities could affect our system.
Testing Procedure	I prefer to use NeWT, a Windows Port of Nessus. It can be downloaded for free from http://www.tenablesecurity.com/newt.html Once installed on a machine on the external interface (redbox), start it up, choose New Scan Task->Input the IPCop IP External Address->Choose Enable All Plugins (Since downtime isn't an issue)->Choose Scan Now.
Test Nature	Objective
Evidence	<div>This is the output from Newt<div>Tenable NeWT Security Reports<div><div>Start</div><div>Wed Apr 21 19:04:31</div><div>Finish</div><div>Wed Apr 21 19:38:37</div><div>Time:</div><div>2004</div><div>Time:</div><div>2004</div></div><div>192.168.2.1</div><div> 192.168.2.1 3 Open Ports, 15 Notes, 5 Infos, 3 Holes.</div><div>192.168.2.1<div><div><div>Port is open</div><div>Plugin ID : 11219</div></div><div><div>An FTP server is running on this port.</div><div>Here is its banner :</div><div>220 Microsoft FTP Service</div><div>Plugin ID : 10330</div></div><div><div>Remote FTP server banner :</div><div>220 Microsoft FTP Service</div><div>Plugin ID : 10092</div></div><div><div>This port was detected as being open by a port scanner but</div></div></div></div></div></div>

is now closed.

This service might have been crashed by a port scanner or by a plugin

Plugin ID : [10919](#)

 You are running a version of OpenSSH which is older than 3.4

There is a flaw in this version that can be exploited remotely to give an attacker a shell on this host.

Note that several distribution patched this hole without changing the version number of OpenSSH. Since Nessus solely relied on the banner of the remote SSH server to perform this check, this might be a false positive.

If you are running a RedHat host, make sure that the command :

ssh (22/tcp) rpm -q openssh-server

Returns :
openssh-server-3.1p1-6


Solution : Upgrade to OpenSSH 3.4 or contact your vendor for a patch

Risk factor : High

CVE : CVE-2002-0639, CVE-2002-0640, CAN-2002-0639, CAN-2002-0640

BID : 5093

Plugin ID : [11031](#)

 You are running a version of OpenSSH older than OpenSSH 3.2.1

A buffer overflow exists in the daemon if AFS is enabled on

your system, or if the options KerberosTgtPassing or AFSTokenPassing are enabled. Even in this scenario, the vulnerability may be avoided by enabling UsePrivilegeSeparation.

Versions prior to 2.9.9 are vulnerable to a remote root exploit. Versions prior to 3.2.1 are vulnerable to a local root exploit.

Solution :

Upgrade to the latest version of OpenSSH

Risk factor : High

CVE : CVE-2002-0575

BID : 4560

Plugin ID : [10954](#)

✗ You are running a version of OpenSSH which is older than 3.7.1

Versions older than 3.7.1 are vulnerable to a flaw in the buffer management functions which might allow an attacker to execute arbitrary commands on this host.

An exploit for this issue is rumored to exist.

Note that several distribution patched this hole without changing the version number of OpenSSH. Since Nessus solely relied on the banner of the remote SSH server to perform this check, this might be a false positive.

If you are running a RedHat host, make sure that the command :

`rpm -q openssh-server`

Returns :

`openssh-server-3.1p1-13 (RedHat 7.x)`

`openssh-server-3.4p1-7 (RedHat 8.0)`

openssh-server-3.5p1-11 (RedHat 9)

Solution : Upgrade to OpenSSH 3.7.1

See also : <http://marc.theaimsgroup.com/?l=openbsd-misc&m=106375452423794&w=2>
<http://marc.theaimsgroup.com/?l=openbsd-misc&m=106375456923804&w=2>


Risk factor : High

CVE : CAN-2003-0682, CAN-2003-0693, CAN-2003-0695

BID : 8628

**Other references : RHSA:RHSA-2003:279-02,
SuSE:SUSE-SA:2003:039**

Plugin ID : [11837](#)

 The remote SSH daemon supports connections made using the version 1.33 and/or 1.5 of the SSH protocol.

These protocols are not completely cryptographically safe so they should not be used.

Solution :

If you use OpenSSH, set the option 'Protocol' to '2'

If you use SSH.com's set the option 'Ssh1Compatibility' to 'no'

Risk factor : Low

Plugin ID : [10882](#)

 You are running OpenSSH-portable 3.6.1p1 or older.

If PAM support is enabled, an attacker may use a flaw in this version

to determine the existence of a given login name by comparing the times

the remote sshd daemon takes to refuse a bad password for a non-existent

login compared to the time it takes to refuse a bad password for a

valid login.

An attacker may use this flaw to set up a brute force attack

against
the remote host.

*Nessus did not check whether the remote SSH daemon is
actually
using PAM or not, so this might be a false positive*

**Solution : Upgrade to OpenSSH-portable 3.6.1p2 or
newer**

Risk Factor : Low

CVE : CAN-2003-0190

BID : 7482, 7467, 7342

Other references : RHSA:RHSA-2003:222-01

Plugin ID : [11574](#)

 You are running OpenSSH-portable 3.6.1 or older.

There is a flaw in this version which may allow an attacker
to
bypass the access controls set by the administrator of this
server.

OpenSSH features a mechanism which can restrict the list
of
hosts a given user can log from by specifying a pattern
in the user key file (ie: *.mynetwork.com would let a user
connect only from the local network).

However there is a flaw in the way OpenSSH does reverse
DNS lookups.

If an attacker configures his DNS server to send a numeric
IP address
when a reverse lookup is performed, he may be able to
circumvent
this mechanism.


Solution : Upgrade to OpenSSH 3.6.2 when it comes out


Risk Factor : Low


CVE : CAN-2003-0386


BID : 7831

Plugin ID : [11712](#)

 Port is open
Plugin ID : [11219](#)

 An ssh server is running on this port
Plugin ID : [10330](#)


 Remote SSH version : SSH-1.99-OpenSSH_3.1p1
Plugin ID : [10267](#)


 The remote SSH daemon supports the following versions of the SSH protocol :

- . 1.33
- . 1.5
- . 1.99
- . 2.0

Plugin ID : [10881](#)


 Port is open
Plugin ID : [11219](#)

 A web server is running on this port
Plugin ID : [10330](#)

 The following directories were discovered:
http /_private, /_vti_bin, /_vti_log, /cfdocs, /cfide, /images
(80/tcp)

While this is not, in and of itself, a bug, you should manually inspect these directories to ensure that they are in compliance with company security standards

The following directories require authentication:
/printers
Plugin ID : [11032](#)

 Nessus was not able to exactly identify this server. It might be:
Microsoft-IIS/5.0 with .NET on Win2000 SP4
The fingerprint differs from these known signatures on 6 point(s)

If you know what this server is and if you are using an up to date
version

of this script, please send this signature to www-signatures@nessus.org :

xxx:200:---

:200:200:xxx:400:400:400:400:400:200:400:400:400:405:-

--:---:---:200:---:---:200::Microsoft-IIS/5.1

Including these headers:

X-Powered-By: ASP.NET

ETag: "b019b8879e22c41:950"

Try to provide as much information as you can: software &
operating

system release, sub-version, patch numbers, and specific
configuration

options, if any.


Plugin ID : [11919](#)

 The remote web server type is :

Microsoft-IIS/5.1

**Solution : You can use urlscan to change reported server for
IIS.**


Plugin ID : [10107](#)

 This port was detected as being open by a port scanner but is now
closed.


This service might have been crashed by a port scanner or by a
plugin

Plugin ID : [10919](#)

general/udp

 For your information, here is the traceroute to 192.168.2.1 :
192.168.2.7
192.168.2.1

Plugin ID : [10287](#)

 The remote host does not discard TCP SYN packets which have the FIN flag set.

Depending on the kind of [firewall](#) you are using, an attacker may use this flaw to bypass its rules.


general/tcp
See also :
<http://archives.neohapsis.com/archives/bugtraq/2002-10/0266.html>
<http://www.kb.cert.org/vuls/id/464113>

Solution : Contact your vendor for a patch

Risk factor : Medium

BID : 7487

Plugin ID : [11618](#)

 The remote host answers to an ICMP timestamp request. This allows an attacker to know the date which is set on your machine.

general/icmp
This may help him to defeat all your time based authentication protocols.

Solution : filter out the ICMP timestamp requests (13), and the outgoing ICMP timestamp replies (14).

Risk factor : Low

CVE : CAN-1999-0524

Plugin ID : [10114](#)

Findings	We see there are in fact some vulnerabilities(Fail). Of particular interest are the OpenSSH vulnerabilities, which definitely warrant further investigation.
----------	--

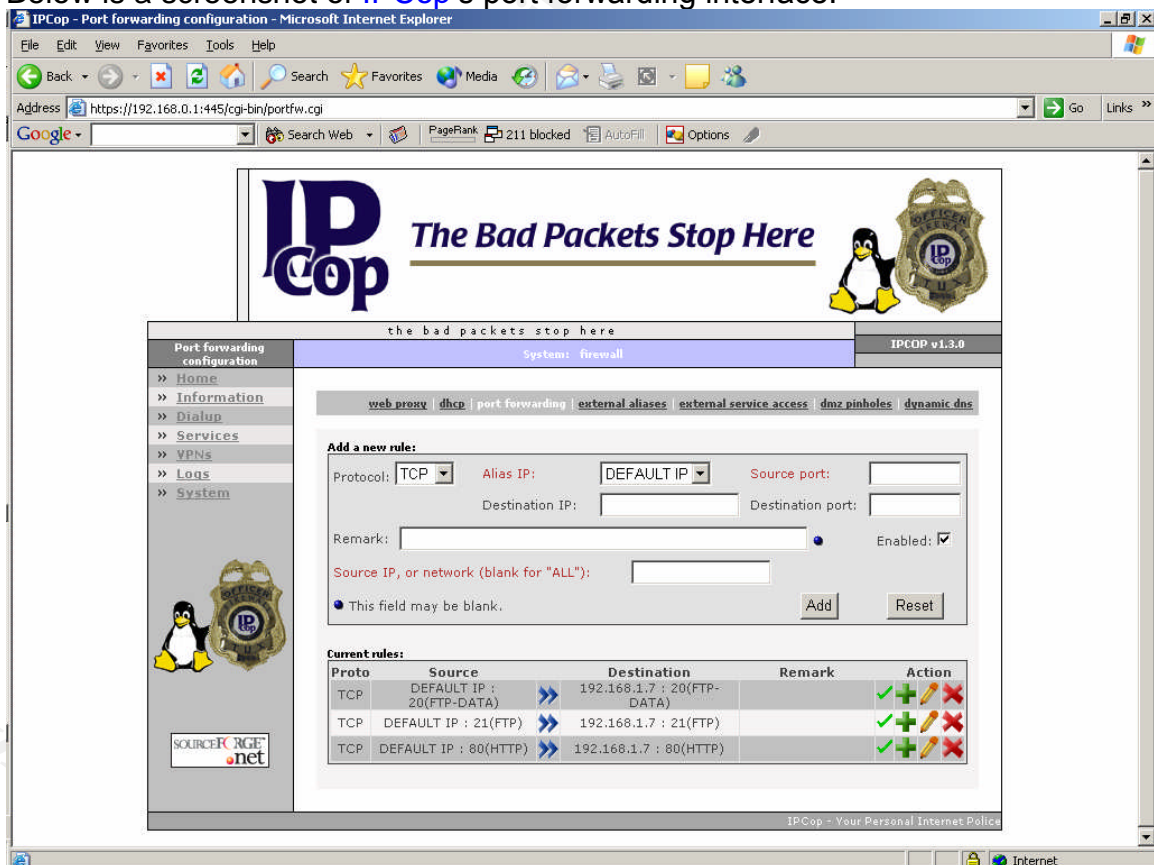
3) Audit Firewall Rulebase

Reference	[4]{4:17-23} [6]{5}
-----------	---------------------

Risk	IPCop could be circumvented due to Administrative Misconfiguration
------	--

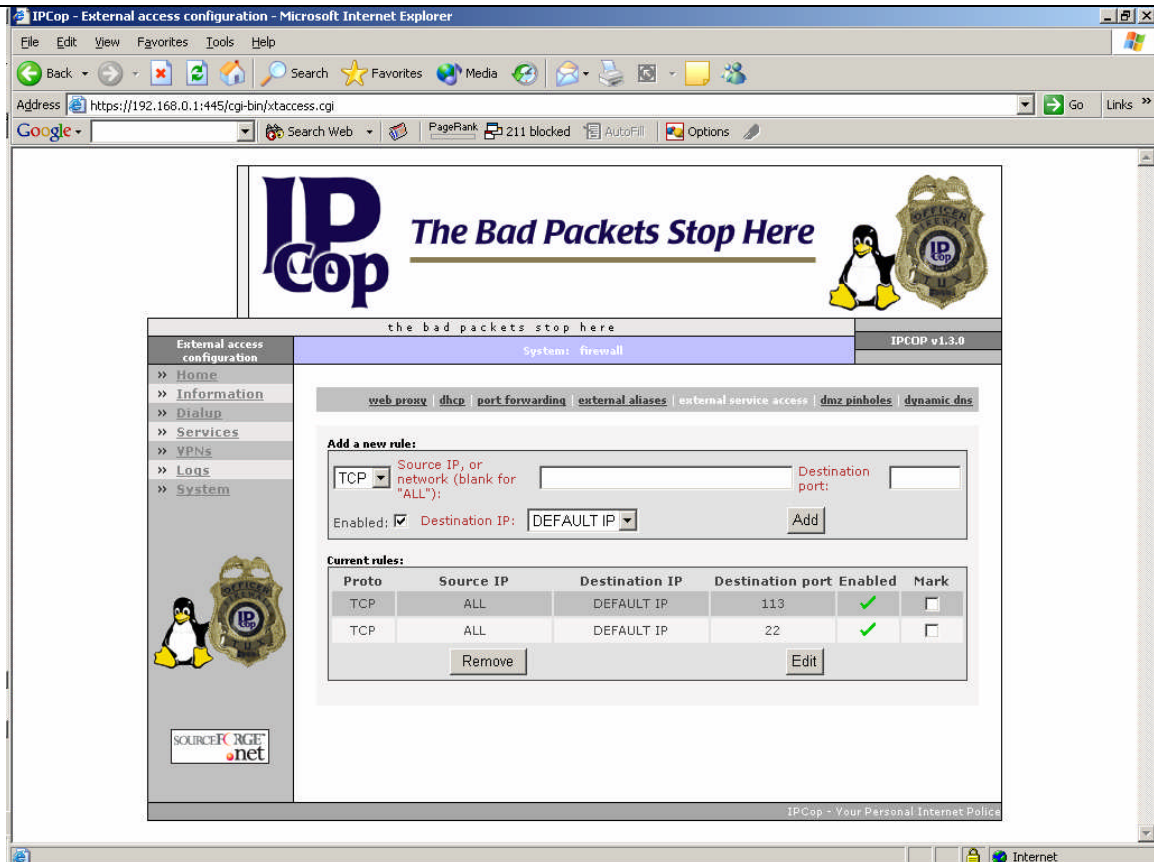
Explanation	We should confirm that all unstealthed/forwarded ports are currently in use. This
-------------	---

	is only a check of the firewall 's configuration. Ensuring that the behavior of the firewall coincides with the configuration is accomplished by other checklist items.
Testing Procedure	<p>The first step is to interview the system administrator to determine which ports should be open/forwarded. With his assistance, evaluate every rule in the IPCop interface as follows:</p> <p>From an internal machine, go to IPCop's IP Address https://192.168.0.1:445 to access IPCop's web based interface.</p> <p>Click Services->port forwarding. Ensure that every forwarded port is currently necessary. Eliminate any that are not.</p> <p>Now click on "external service access". Here you can see services that are directly available on the firewall. Ensure that these available services match up with what is expected. Eliminate any that do not.</p>
Test Nature	Determining what services should be available is Subjective. Making sure the rules match up to this determination is Objective.
Evidence	<p>I know that the only services being forwarded should be to the http/ftp server. Below is a screenshot of IPCop's port forwarding interface.</p>



We see that only ports 20/21(ftp) and 80(http) are being forward.

Next, we take a look at the external service access interface.



Here, we see that the only ports available are 113 and 22. 22 is necessary for external access using SSH. 113 is necessary to prevent my internal IRC clients from hanging when connecting to some servers. Though it can be a security risk if open, it should be closed.

Findings	The rulebase checks out with only necessary services being available(PASS) Keep in mind, just because the rules are correct, doesn't mean the firewall is behaving correctly. We will confirm this with the next tests.
----------	--

6) Scan IPCop for UnStealthed Ports

Reference [4]{4:6,7,24-33} [8]{1}

Risk **IPCop** could be circumvented/subverted due to Administrative Misconfiguration or Improper **IPCop** Implementation

Explanation An available service running on **IPCop**, or one forwarded to another machine, could be an avenue of attack.

Testing Procedure Note: This test will have already been accomplished if the NeWT scan from 1) has been performed. I'm offering these instructions in the instance a user may just want to do a quick scan for this one vulnerability.

With NMapWin installed on a machine connected to an **external interface**, do a SYN Stealth Scan against **IPCop**'s external interface. (Simply select SYN Stealth under mode, and put in the external IP Address of **IPCop** under Host).

IPCop should only respond to the SYN packets on ports which have

	been defined as available in the IPCop rulebase. Of course, the results of this test should be correlated with the settings in the rulebase to confirm this.																		
Test Nature	Objective																		
Evidence	<p>Here are the results from the NMapWin Scan:</p> <p>Starting nmap V. 3.00 (www.insecure.org/nmap)</p> <p>Interesting ports on (192.168.2.1):</p> <p>(The 1596 ports scanned but not shown below are in state: filtered)</p> <table><tr><th>Port</th><th>State</th><th>Service</th></tr><tr><td>20/tcp</td><td>closed</td><td>ftp-data</td></tr><tr><td>21/tcp</td><td>open</td><td>ftp</td></tr><tr><td>22/tcp</td><td>open</td><td>ssh</td></tr><tr><td>80/tcp</td><td>open</td><td>http</td></tr><tr><td>113/tcp</td><td>closed</td><td>auth</td></tr></table> <p>No exact OS matches for host (If you know what OS is running on it, see http://www.insecure.org/cgi-bin/nmap-submit.cgi).</p> <p>TCP/IP fingerprint:</p> <p>SInfo(V=3.00%P=i686-pc-windows-windows%D=4/17%Time=4081DDCC%O=21%C=20)</p> <p>TSeq(Class=RI%gcd=1%SI=46CE%IPID=I%TS=0)</p> <p>TSeq(Class=RI%gcd=1%SI=3E8F%IPID=I%TS=0)</p> <p>TSeq(Class=RI%gcd=1%SI=32B4%IPID=I%TS=0)</p> <p>T1(Resp=Y%DF=Y%W=FAF0%ACK=S+++%Flags=A%Ops=NNT)</p> <p>T1(Resp=Y%DF=Y%W=FAF0%ACK=S+++%Flags=AS%Ops=MNWNNT)</p> <p>T2(Resp=N)</p> <p>T3(Resp=Y%DF=Y%W=FAF0%ACK=S+++%Flags=AS%Ops=MNWNNT)</p> <p>T4(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)</p> <p>T5(Resp=Y%DF=N%W=0%ACK=S+++%Flags=AR%Ops=)</p> <p>T6(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)</p> <p>T7(Resp=Y%D</p> <p>F=N%W=0%ACK=S+++%Flags=AR%Ops=)</p> <p>PU(Resp=N)</p> <p>Nmap run completed -- 1 IP address (1 host up) scanned in 281 seconds</p>	Port	State	Service	20/tcp	closed	ftp-data	21/tcp	open	ftp	22/tcp	open	ssh	80/tcp	open	http	113/tcp	closed	auth
Port	State	Service																	
20/tcp	closed	ftp-data																	
21/tcp	open	ftp																	
22/tcp	open	ssh																	
80/tcp	open	http																	
113/tcp	closed	auth																	
Findings	<p>Notice that only the ports expected to be hosting services are open (21,22,80), and the other 2 shown ports are closed. All other ports are stealthed. (PASS)</p>																		

7) Audit Static Ingress Filtering

Reference	[4]{1:10-13,16 3:69} [6]{8} [8]{2,3,4,6}
Risk	IPCop could be circumvented due to Administrative Misconfiguration or Improper IPCop Implementation
Explanation	<p>The following packets need to be blocked:</p> <p>SourceIP from internal address space: Prevents spoofing</p> <p>SourceIP from Loopback: Prevents some DoS attacks</p>

	<p>SourceIP from private/unallocated address space: Prevents spoofing.</p> <p>Source routed Packets: Makes spoofing difficult</p> <p>ICMP broadcasts</p> <p>ICMP redirects</p> <p>ICMP Timestamp requests</p>
Testing Procedure	<p>For each of these tests, we will use hping2 running on a Linux box connected to the external interface to send inappropriate traffic. Hping2 can be downloaded from http://wiki.hping.org/.</p> <p>We will also have ethereal running on a machine connected to the internal interface. Before each test, we start ethereal capturing by clicking capture->start, selecting the appropriate interface, and clicking OK. After each test, we click stop, and look for any traffic that matches what we just sent with hping. If at any point ethereal logs traffic that we have sent with hping, then the firewall has failed that aspect of the test.</p> <p>It may also be useful to check the logs on IPCop. From an internal machine, go to IPCop's IP Address https://192.168.0.1:445 to access IPCop's web based interface. Click Logs->firewall. If at any point the IPCop logs show traffic being redirected to an internal address, then the firewall has failed that aspect of the test. This is true even if the address the traffic reaches is invalid, since this could result in a vulnerability if IPCop is used with some different network configuration. It can also be informative to check the IDS logs (Logs->intrusion detection system), to see what IPCop reports as suspicious.</p> <p>Note that we would want this traffic blocked even on ports that have been forwarded to another machine from the firewall. For the duration of the test, we will configure a temporary port forward on some unused port to our machine running ethereal. All appropriate hping tests will use this port. See checklist item "Audit the Firewall Rulebase" on accessing IPCop's port forwarding interface. As an addendum to the hping/ethereal tests, I am providing some alternative tests. These tests simply involve checking iptables and /proc/sys/net/ipv4 settings. These tests do not provide the level of certainty as the hping ones, but require only access to the firewall console to perform. They also outline exactly how the machine should be configured, in case the auditor needs to advise the system administrator.</p> <p>Use the following hping commands to perform the test.</p> <p>Key:</p> <p><fp> is the port on the firewall that has been forwarded to our box running ethereal.</p> <p><ia> is an internal IP address from the internal network's IP range.</p> <p><ea> is the external IP address of the firewall</p> <p><ua> is an address from unallocated address space. For address allocation, check www.iana.org/assignments/ipv4-address-space</p> <p><eba> is the broadcast address for the subnet of which the external address is a member. A broadcast address is one in which the host portion of the IP address consists either of all 255's.</p> <p><hpa> the IP address of the machine sending spoofed packets with hping.</p>

SourceIP from internal address space

Hping2 -S -p <fp> -a <ia> <ea>

Note: the -S results in Hping2 sending a SYN packet, so if IPCop is misconfigured, we may get a SYN/ACK back.

Check the current iptables ruleset (by typing iptables -L in the console on the firewall). Look for a rule similar to this one under the FORWARD chain:

REJECT tcp -- 192.168.0.0/24 anywhere reject-with icmp-port-unreachable

SourceIP from Loopback

Hping2 -S -p <fp> -a 127.0.0.1 <ea>

SourceIP from private/unallocated address space

Hping2 -S -p <fp> -a <ua> <ea>

Source routed Packets

Hping2 -S -p <fp> -lsrr <hpa> <ea>

Ensure /proc/sys/net/ipv4/conf/all/accept_source_route = 0

Do this by bringing up the console, and typing cat <the filename above>

These last three tests will target the firewall directly. As such, simply checking ethereal logs on the internal client will tell us nothings. Instead, we will simply have to inspect the firewall behavior.

ICMP broadcasts

Hping2 -C 8 <eba>

Note: be careful to only do this test when the network is not in heavy use. This is a dangerous test in a large environment. If IPCop responds, the test is a failure, as IPCop could become a participant in a SMURF attack.

Ensure /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts = 1

ICMP redirects

Hping2 -C 5 --icmp-gw <hpa> <ea>

To see if this has an effect, bring up a console on the firewall, and type route -C. Then run the above command from the machine on the external interface.

Finally, type route -C, and see if there are any new entries. If there are, the test is a failure.

Ensure /proc/sys/net/ipv4/conf/all/accept_redirects = 0

ICMP Timestamp requests

Hping2 -C 13 <ea>

If IPCop replies, the test is a failure.

Check the current iptables ruleset (by typing iptables -L in the console on the firewall). Look for a rule similar to this one under the INPUT chain:

REJECT icmp -- anywhere anywhere icmp timestamp-request reject-with icmp-port-unreachable

Test Nature Objective

Evidence From **redbox**, I sent the following commands:

Source IP from internal address:

hping2 -S -p 80 -a 192.168.0.7 192.168.2.1

Source IP from Loopback:

hping2 -S -p 80 -a 127.0.0.1 192.168.2.1

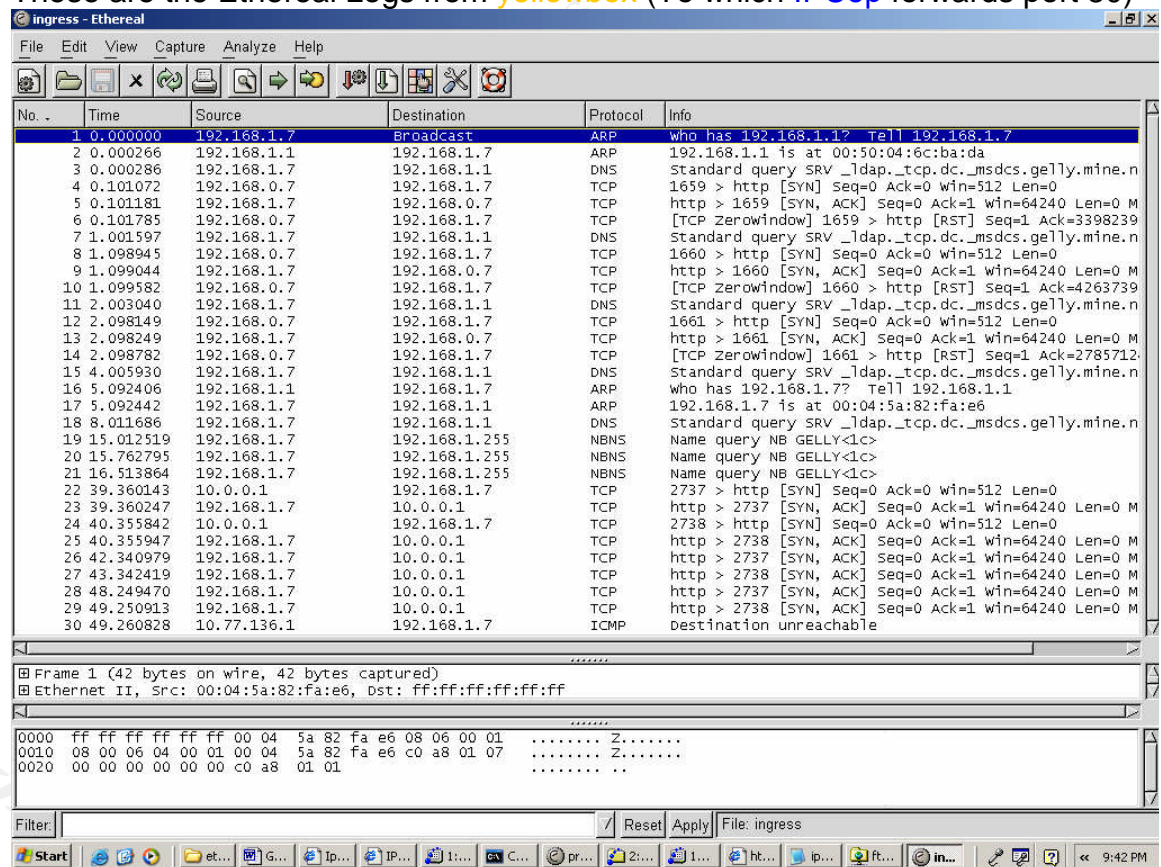
Source IP from Unallocated/Private:

hping2 -S -p 80 -a 10.0.0.1 192.168.2.1

Source Routed Packets:

hping2 -S -p 80 --lsrr 192.168.2.7 192.168.2.1

These are the Ethereal Logs from **yellowbox** (To which **IPCop** forwards port 80)



We see the packets from **192.168.0.7**(internal) (FAIL)

We do not see the packets from **127.0.0.1** (Loopback) (PASS)

We see the packets from **10.0.0.1** (unallocated/private) (FAIL)

We do not see packets from **192.168.2.7** (source routed) (PASS)

I then sent the icmp broadcast test:

hping2 -C 8 192.168.2.255

to which I received:

```
HPING 192.168.2.255 (eth0 192.168.2.255): icmp mode set, 28 headers + 0
data bytes
len=28 ip=192.168.2.7 ttl=64 id=17988 icmp_seq=0 rtt=0.3 ms
DUP! len=46 ip=192.168.2.2 ttl=64 id=48690 icmp_seq=0 rtt=11.5 ms
DUP! len=46 ip=192.168.2.245 ttl=64 id=18246 icmp_seq=0 rtt=11.7 ms
len=28 ip=192.168.2.7 ttl=64 id=17989 icmp_seq=1 rtt=0.2 ms
DUP! len=46 ip=192.168.2.2 ttl=64 id=48691 icmp_seq=1 rtt=1.0 ms
DUP! len=46 ip=192.168.2.245 ttl=64 id=18247 icmp_seq=1 rtt=2.0 ms
Notice that 192.168.2.1 didn't respond (PASS)
```

I then performed the ICMP redirect test:

On the [firewall](#), I typed:

`route -C`

and got:

Kernel IP routing cache

Source	Destination	Gateway	Flags	Metric	Ref	Use	Iface
--------	-------------	---------	-------	--------	-----	-----	-------

On [redbox](#) I typed:

`Hping2 -C 5 --icmp_gw 192.168.2.7 192.168.2.1`

On the [firewall](#), I again typed:

`route -C`

and got:

Kernel IP routing cache

Source	Destination	Gateway	Flags	Metric	Ref	Use	Iface
--------	-------------	---------	-------	--------	-----	-----	-------

Notice there are no new routing entries(PASS).

I then sent the ICMP timestamp test:

`hping2 -C 13 192.168.2.1`

To which I received the following

```
HPING 192.168.2.1 (eth0 192.168.2.1): icmp mode set, 28 headers + 0
data bytes
len=46 ip=192.168.2.1 ttl=64 id=38504 icmp_seq=0 rtt=2.8 ms
ICMP timestamp: Originate=84148628 Receive=84253820 Transmit=84253820
ICMP timestamp RTT tsrtt=3

len=46 ip=192.168.2.1 ttl=64 id=38505 icmp_seq=1 rtt=0.6 ms
ICMP timestamp: Originate=84149624 Receive=84254816 Transmit=84254816
ICMP timestamp RTT tsrtt=1
```

Notice that [IPCop](#) responded (FAIL)

Findings	IPCop does in fact block packets with a Loopback Address source, source routed packets, icmp redirects, and broadcasts. (PASS). Unfortunately, it doesn't block ICMP timestamps, or addresses from internal, unallocated, or private address space (FAIL).
----------	--

11) Ensure the Firewall can Withstand a DoS attack

Reference	[6]{7}
Risk	IPCop is inoperable due to a DoS attack
Explanation	<p>We want to ensure that IPCop is resistant to, and continues operation during a Dos attack</p> <p>There are basically 2 classes of DoS attacks one must be prepared for. The first type exploits some vulnerability to somehow render the firewall (and any routing functionality) inoperative. The best way to ensure your firewall will withstand such an attack is to use a vulnerability scanner to test for most of the known vulnerabilities.</p> <p>The second type of DoS attack can occur if a server is simply flooded with more traffic than it can handle. In a typical environment, there are 3 points of failure:</p> <ol style="list-style-type: none"> 1)The bandwidth available to the site is used up, preventing any legitimate traffic 2)The firewall receives so much traffic it can no longer filter or route traffic 3)Servers behind the firewall receive so much traffic they can no longer service legitimate traffic. <p>The ISP is not the target of the audit, so 1) isn't our concern. We do, however, want to ensure that the firewall can handle the maximum amount of traffic that may come to it. We would also like to know if the firewall provides any mitigation for an attack on a server behind the perimeter.</p> <p>Such attacks can take the form of a simple TCP or UDP flood, though such attacks aren't the most effective. We are more concerned with either a TCP SYN attack (a flood of TCP SYN packets which fills up the new connection queue on a server) or an ICMP flood (either a simple flood of pings, which will tie up a server trying to respond, or a SMURF attack, which sends a flood of pings to some broadcast address, with a source address of the target, resulting in the server getting flooded with echo-replies).</p> <p>The TCP SYN attack will only affect a machine with open TCP ports, so this will most likely target a server behind the firewall, though it could target the firewall directly if remote administration on the firewall is enabled (the traffic could be directed at whichever port the OpenSSH server is awaiting a connection on).</p> <p>That being said, we need to ensure the following:</p> <p>The firewall should continue operation during an icmp flood (both a simple ping flood and a SMURF attack).</p> <p>The firewall should continue operation during a TCP SYN attack</p>

on the [OpenSSH](#) port (if one is available).

The [firewall](#) should continue operation during a TCP SYN attack on a port forwarded to some [internal server](#).

The [firewall](#) should provide some mitigation to help internal servers cope with a TCP SYN attack.

A note on this last point: what can a [firewall](#) do to mitigate such an attack? If the attacker doesn't spoof the source address of the attack, then the attack will consistently originate from some set of addresses (if the attack is a DDos attack, there could be thousands of addresses in this set). The [firewall](#), ideally, would automatically block all traffic from these addresses after a few moments. If the attacker does spoof the source address, they will often cycle through some set of (in)valid addresses, to prevent the above method of mitigation from being effective. The [firewall](#), however, can still help the situation by blocking all traffic from unallocated, private, and internal address space. Traffic with a Loopback source address should obviously be blocked as well.

Testing
Procedure

To perform the vulnerability scan, I prefer to use NeWT, a Windows Port of Nessus.

It can be downloaded for free from

<http://www.tenablesecurity.com/newt.html>

Once installed on a machine on the external interface ([redbox](#)), start it up, choose New Scan Task->Input the [IPCop](#) external IP address->Choose "Define my own set of plugins"->under Families, check "Denial of Service", and uncheck "Do not use dangerous plugins even if they are selected" at the bottom of the window->click Scan Now.

If any applicable vulnerabilities are discovered by the scan, the [firewall](#) has failed the test.

During each of the following tests, we will test [firewall](#)/routing functionality in both directions, by transferring files from an external machine to an internal machine, and back again.

To do this, we need 3 machines connected to the [firewall](#). One machine should be attached to a hub/switch connected to [IPCop](#)'s external interface. This machine will need hping2, which we will use to simulate the various attacks. This machine should have a fairly fast processor, as we want it to send traffic more quickly than [IPCop](#) can process it. Another machine connected to this same hub/switch on the external interface should be serving a large file(100MB+) in some way it can be easily downloaded (http is generally easy to set up and route through a [firewall](#)). Finally, a third machine connected to [IPCop](#)'s internal interface should be running Ethereal and similarly hosting some large file. The appropriate port (in the case of http, port 80) should be forwarded

from the [firewall](#) to this machine. See checklist item “Audit Static Ingress Filtering” on setting up hping2 and ethereal. See checklist item “Audit the [Firewall](#) Rulebase” on accessing [IPCop](#)’s port forwarding interface.

For an http server, one can simply use IIS if it is included with your version of Windows, or use one of the many freeware http servers.

Before performing any of the DoS attacks, copy the file once in each direction, making a note of how long this takes in each direction. This will be used to determine what performance hit, if any, our attacks have on [IPCop](#) and our internal server.

If during any of the tests below, it is impossible to make a connection in either direction, or if the file copies very slowly in either direction, then [IPCop](#) has failed that portion of the test.

Key:

<ea> is the external IP address of the [firewall](#)

<sshp> port on the [firewall](#) used for remote administration (OpenSSH)

<isp> port on the [firewall](#) forwarded to the internal server (probably port 80 if using http)

<spfda> spoofed address. This is used in most DoS attacks to prevent reply traffic from reaching the attacking machine, allowing the attacking machine to spend more resources pumping out more bogus traffic.

Simulating an ICMP flood:

Hping2 -C 8 -i u1 -a <spfda> <ea>

Simulating a TCP SYN flood on OpenSSH:

Hping2 -S -i u1 -p <sshp> -a <spfda> <ea>

Simulating a TCP SYN flood on the internal server:

Hping2 -S -i u1 -p <isp> -a <spfda> <ea>

Note: if this test fails, it is likely directly due to the internal server rather than [IPCop](#). However, since we are interested in [IPCop](#)’s flood mitigation, the internal server failing would also be a black mark for [IPCop](#).

Ensure /proc/sys/net/ipv4/tcp_syncookies on the [firewall](#) is set to To do this, simply bring up a console and type [cat /proc/sys/net/ipv4/tcp_syncookies](#)

Mitigation:

Does **IPCop** make a note of addresses that have been used to initiate an attack? Does it block such traffic from such addresses? Have Ethereal logging while performing the last test above. If, after a few moments, the server stops receiving traffic, even though hping2 is still blasting, then the **firewall** has passed this aspect of the test. Otherwise, it has failed.

Testing if **IPCop** successfully blocks all traffic from private/unallocated/internal/loopback address space is covered under checklist item "Audit Static Ingress Filtering"

Test Nature	Objective
-------------	-----------

Evidence	The NeWT Scan conducted in 1) includes a plethora of DoS attacks. IPCop maintained functionality through this test (PASS)
----------	--

I then connected an ftp client from an **extra machine** on the external interface to port 21 on the **firewall** (which is forwarded to **yellowbox**). While sending specific attacks with **redbox**, I timed how long a file copy took.

No Attack: File copy = 2 minutes both ways

ICMP flood:

Hping2 -i u1 -C 8 -a 192.168.2.201 192.168.2.1

File copy: 2minutes both ways

Easily established a new ssh connection to **192.168.2.1** from the **extra machine**.

Easily established a new ftp connection to **192.168.2.1** from the **extra machine**.

TCP SYN flood on OpenSSH:

Hping2 -i u1 -S -p 22 -a 192.168.2.201 192.168.2.1

File copy: 2 minutes both ways

Could not establish a new ssh connection to **192.168.2.1** from neither the **extra machine** nor **greenbox**.

Easily established a new ftp connection to **192.168.2.1** from the **extra machine**.

TCP SYN flood on internal server: 2 minutes both ways

Hping2 -i u1 -S -p 80 -a 192.168.2.201 192.168.2.1

File copy: 2 minutes both ways

Easily established a new ssh connection to **192.168.2.1** from the **extra machine**.

Could not establish a new ftp connection to **192.168.2.1** from neither the **extra machine** nor **greenbox**.

Typing in

`cat /proc/sys/net/ipv4/tcp_syncookies`
on the **IPCop** console returned a value of 0.

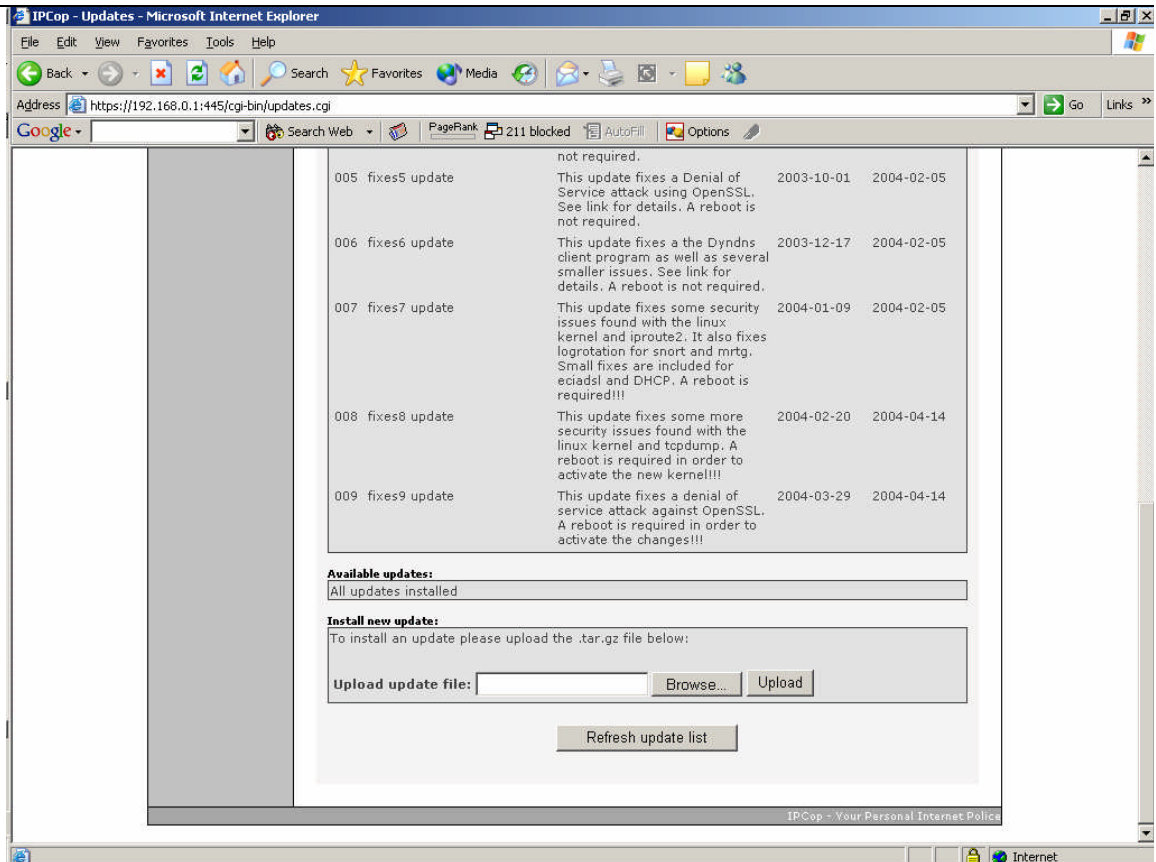
No traffic from **redbox** was ever blocked by **IPCop** during the tests above, even after sending repeated attacks from the same (spoofed) IP address.

We have already determined from the evidence for checklist item “Audit Ingress Filtering” that **IPCop** does not block traffic from unallocated/private/internal addresses, but does block traffic from the loopback address.

Findings	NeWT DoS attacks have little effect on IPCop (PASS). As you can see, the ability of IPCop to route data was not affected by any of these attacks (PASS). However, during the TCP SYN flood on OpenSSH , it was impossible to establish a new connection to OpenSSH (FAIL). The same was true for the FTP server during the TCP SYN flood on FTP (FAIL). We also found that <code>tcp_syncookies</code> was disabled (FAIL). Enabling this would render IPCop immune to a SYN flood on OpenSSH , but would do nothing to help the FTP server . At no point did IPCop start blocking traffic by IP Address (FAIL). IPCop also did not block traffic from unallocated, private, or internal addresses (FAIL), though it did block traffic from the loopback address (PASS).
----------	--

12) Ensure Up to Date Patch State

Reference	[4]{4:8}
Risk	IPCop could be circumvented/subverted due to Out of Date Patch State
Explanation	All patches should be applied. If not, some easily exploitable vulnerability may exist on the firewall .
Testing Procedure	From an internal machine, go to IPCop ’s IP Address https://192.168.0.1:445 to access IPCop ’s web based interface. Click System->updates. You will see a list of all available updates. At the bottom of the page will be a list of uninstalled updates. If there are any uninstalled updates in this list, then the firewall has failed this part of the test.
Test Nature	Objective
Evidence	Below is a screenshot of the bottom half of IPCop ’s updates interface.



Findings Note that all updates are installed (PASS).

14) Ensure Firewall Logging and IDS are Enabled

Reference [4]{4:44-51} [3]{Linux:10,23} [8]{7}

Risk Inadequate Environment Detection

Explanation We want to insure that if we come under attack, a simple check of the logs will reveal this to us.

Testing Procedure From an internal machine, go to IPCop's IP Address <https://192.168.0.1:445> to access IPCop's web based interface.

Firewall Logging is enabled by default, but the IDS isn't. To enable it, click System->intrusion detection system. Ensure the option labeled "Snort" is checked.

To check the firewall logs, click Logs->firewall.

To check the intrusion detection logs, click log->intrusion detection system.

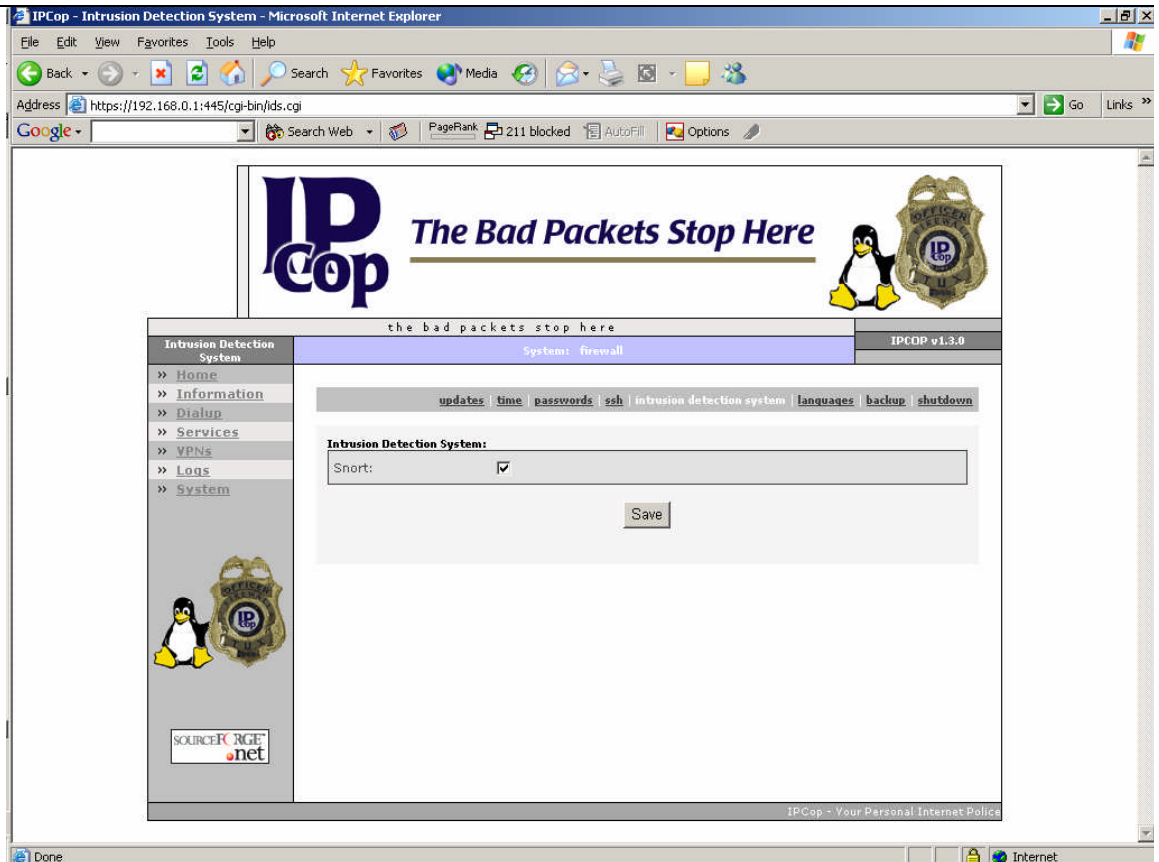
To ensure that the logging is actually enabled, do a NeWT or NMapWin scan (described in checkpoints 1 and 3), and check these logs.

If new items show up in the firewall logs from the scanning host, then the firewall has passed this aspect of the test.

If the IDS logs make not of some suspicious activity, then the firewall has passed this aspect of the test.

Test Nature Objective

Evidence Here is a screenshot of the interface to enable Snort



Here is some sample output of the [firewall](#) logs after an NMapWin Scan.

Time	Chain	Iface	Proto	Source	Src Port	Destination	Dst Port
19:31:07	INPUT	eth1	UDP	<input type="checkbox"/> 192.168.2.7	1269	<input type="checkbox"/> 192.168.2.1	514(SYSLOG)
19:31:09	INPUT	eth1	UDP	<input type="checkbox"/> 192.168.2.7	1269	<input type="checkbox"/> 192.168.2.1	514(SYSLOG)
19:31:18	INPUT	eth1	TCP	<input type="checkbox"/> 192.168.2.7	1271	<input type="checkbox"/> 192.168.2.1	17990
19:31:21	INPUT	eth1	TCP	<input type="checkbox"/> 192.168.2.7	1271	<input type="checkbox"/> 192.168.2.1	17990
19:31:27	INPUT	eth1	TCP	<input type="checkbox"/> 192.168.2.7	1271	<input type="checkbox"/> 192.168.2.1	17990
19:31:37	INPUT	eth1	TCP	<input type="checkbox"/> 192.168.2.7	1272	<input type="checkbox"/> 192.168.2.1	6004
19:31:41	INPUT	eth1	TCP	<input type="checkbox"/> 192.168.2.7	1273	<input type="checkbox"/> 192.168.2.1	14238
19:31:47	INPUT	eth1	TCP	<input type="checkbox"/> 192.168.2.7	1273	<input type="checkbox"/> 192.168.2.1	14238
19:31:51	INPUT	eth1	TCP	<input type="checkbox"/> 192.168.2.7	1274	<input type="checkbox"/> 192.168.2.1	4662
19:31:57	INPUT	eth1	TCP	<input type="checkbox"/> 192.168.2.7	1274	<input type="checkbox"/> 192.168.2.1	4662
19:32:04	INPUT	eth1	2	<input type="checkbox"/> 192.168.2.2	2	<input type="checkbox"/> 224.0.0.1	2
19:32:09	INPUT	eth1	UDP	<input type="checkbox"/> 192.168.2.3	138(NETBIOS-DGM)	<input type="checkbox"/> 192.168.2.255	138(NETBIOS-DGM)
19:32:30	INPUT	eth1	TCP	<input type="checkbox"/> 192.168.2.7	1299	<input type="checkbox"/> 192.168.2.1	8181

19:32:33	INPUT	eth1	TCP	<input type="checkbox"/>	192.168.2.7	1299	<input type="checkbox"/>	192.168.2.1	8181
19:32:39	INPUT	eth1	TCP	<input type="checkbox"/>	192.168.2.7	1299	<input type="checkbox"/>	192.168.2.1	8181
19:32:40	INPUT	eth1	TCP	<input type="checkbox"/>	192.168.2.7	1300	<input type="checkbox"/>	192.168.2.1	8383
19:32:43	INPUT	eth1	TCP	<input type="checkbox"/>	192.168.2.7	1300	<input type="checkbox"/>	192.168.2.1	8383
19:32:49	INPUT	eth1	TCP	<input type="checkbox"/>	192.168.2.7	1300	<input type="checkbox"/>	192.168.2.1	8383
19:33:35	INPUT	eth1	UDP	<input type="checkbox"/>	192.168.2.7	1346	<input type="checkbox"/>	192.168.2.1	500(ISAKMP)
19:33:36	INPUT	eth1	UDP	<input type="checkbox"/>	192.168.2.7	1346	<input type="checkbox"/>	192.168.2.1	500(ISAKMP)
19:33:37	INPUT	eth1	UDP	<input type="checkbox"/>	192.168.2.7	1346	<input type="checkbox"/>	192.168.2.1	500(ISAKMP)
19:33:38	INPUT	eth1	UDP	<input type="checkbox"/>	192.168.2.7	1346	<input type="checkbox"/>	192.168.2.1	500(ISAKMP)
19:33:39	INPUT	eth1	UDP	<input type="checkbox"/>	192.168.2.7	1346	<input type="checkbox"/>	192.168.2.1	500(ISAKMP)

Here is some sample output from the Snort Logs after an NMapWin Scan

Total of number of Intrusion rules activated for April 21: 5119

Date:	04/21 19:39:07	Name:	SCAN nmap TCP
Priority:	2	Type:	Attempted Information Leak
IP info:	192.168.2.7:41284 -> 192.168.2.1:22		
References:	none found	SID:	628

Date:	04/21 19:39:07	Name:	SCAN nmap TCP
Priority:	2	Type:	Attempted Information Leak
IP info:	192.168.2.7:41348 -> 192.168.2.1:22		
References:	none found	SID:	628

Date:	04/21 19:39:07	Name:	SCAN nmap TCP
Priority:	2	Type:	Attempted Information Leak
IP info:	192.168.2.7:41404 -> 192.168.2.1:22		
References:	none found	SID:	628

Date:	04/21 19:39:07	Name:	SCAN nmap TCP
Priority:	2	Type:	Attempted Information Leak
IP info:	192.168.2.7:41464 -> 192.168.2.1:22		
References:	none found	SID:	628

Date:	04/21 19:39:07	Name:	SCAN nmap TCP
Priority:	2	Type:	Attempted Information Leak
IP info:	192.168.2.7:41520 -> 192.168.2.1:22		
References:	none found	SID:	628

Date:	04/21 19:39:07	Name:	SCAN nmap TCP
Priority:	2	Type:	Attempted Information Leak
IP info:	192.168.2.7:41572 -> 192.168.2.1:22		
References:	none found	SID:	628

Date:	04/21 19:39:07	Name:	SCAN nmap TCP
Priority:	2	Type:	Attempted Information Leak

IP info:	192.168.2.7 :41624 -> 192.168.2.1 :22		
References:	none found	SID:	628
Date:	04/21 19:39:07	Name:	SCAN nmap TCP
Priority:	2	Type:	Attempted Information Leak
IP info:	192.168.2.7 :41676 -> 192.168.2.1 :22		
References:	none found	SID:	628
Date:	04/21 19:39:07	Name:	SCAN nmap TCP
Priority:	2	Type:	Attempted Information Leak
IP info:	192.168.2.7 :41735 -> 192.168.2.1 :22		
References:	none found	SID:	628
Date:	04/21 19:39:07	Name:	SCAN nmap TCP
Priority:	2	Type:	Attempted Information Leak
IP info:	192.168.2.7 :41796 -> 192.168.2.1 :22		
References:	none found	SID:	628
Date:	04/21 19:39:07	Name:	SCAN nmap TCP
Priority:	2	Type:	Attempted Information Leak
IP info:	192.168.2.7 :41859 -> 192.168.2.1 :22		
References:	none found	SID:	628
Date:	04/21 19:39:07	Name:	SCAN nmap TCP
Priority:	2	Type:	Attempted Information Leak
IP info:	192.168.2.7 :41912 -> 192.168.2.1 :22		
References:	none found	SID:	628
Date:	04/21 19:39:07	Name:	SCAN nmap TCP
Priority:	2	Type:	Attempted Information Leak
IP info:	192.168.2.7 :41972 -> 192.168.2.1 :22		
References:	none found	SID:	628
Date:	04/21 19:39:07	Name:	SCAN nmap TCP
Priority:	2	Type:	Attempted Information Leak
IP info:	192.168.2.7 :42029 -> 192.168.2.1 :22		
References:	none found	SID:	628
Date:	04/21 19:39:07	Name:	SCAN nmap TCP
Priority:	2	Type:	Attempted Information Leak
IP info:	192.168.2.7 :42080 -> 192.168.2.1 :22		

Findings **Firewall** Logging and IDS are both enabled, and shown working. (PASS).

16) Ensure Access Logging and File Verification are Enabled

Reference [\[4\]{4:46}](#) [\[3\]{Linux:5,10}](#) [\[6\]{11}](#) [\[8\]{13}](#)

Risk Inadequate State Detection

Explanation We want to ensure that if the system is compromised, that a quick check will reveal this to us.

Testing Ensure that /var/log/secure has a record of all login attempts.

Procedure Do this by SSHing into **IPCop** and typing in the console:
Less /var/log/secure.
If this file is empty, then the **firewall** has failed this aspect of the test.

Ensure that Tripwire is installed.

To do this, type `tripwire -m c`

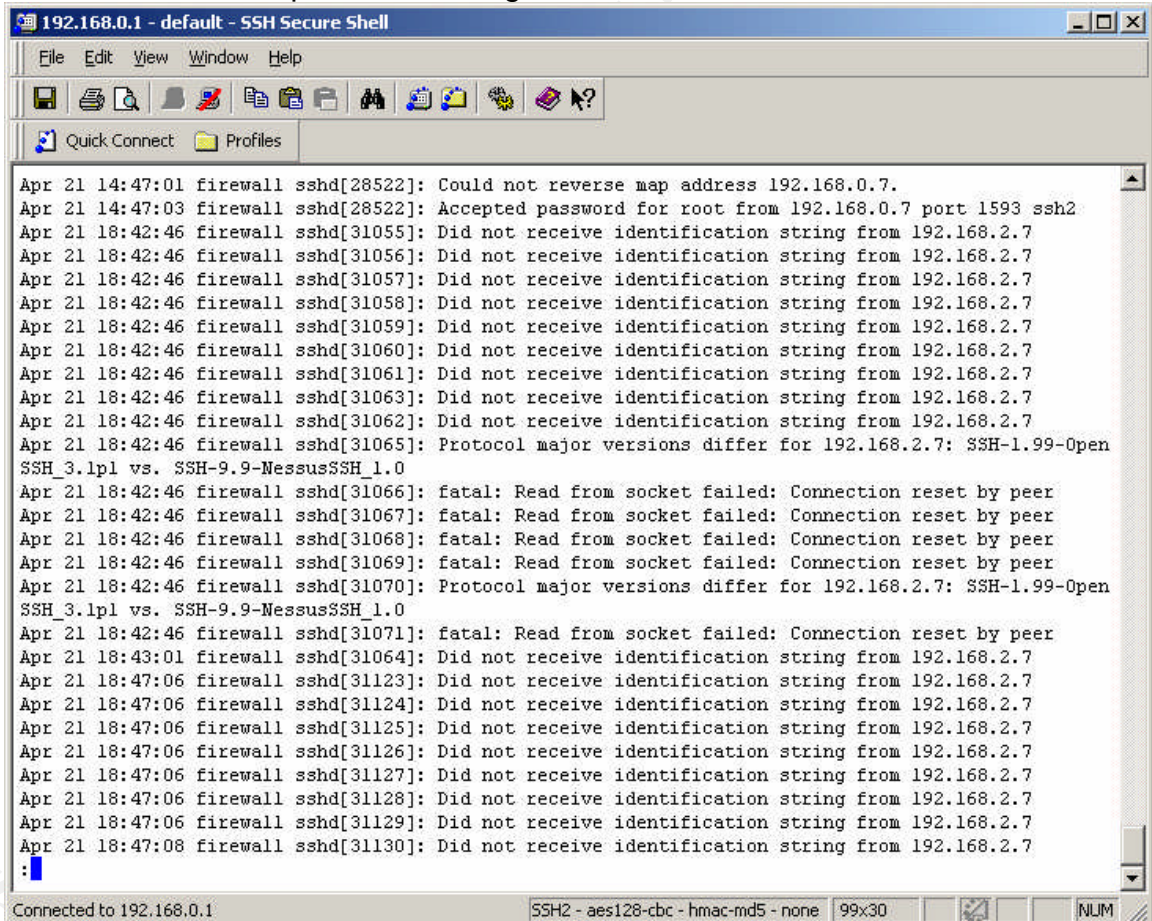
If Tripwire is installed, this will do a full integrity check (a compare against the last known good state).

Side Note: This is a good time to check the integrity report in `/var/db/tripwire` with the `twprint` utility. This isn't necessarily on the checklist, but it may reveal a compromised system!

You should also ensure Tripwire is set up correctly by inspecting the `/usr/local/etc/twpol.txt` file. In this file, make sure that all files in `/etc`, `/bin`, `/usr/bin`, and `/usr/local/bin` are checked, and flagged to generate a warning if changed.

Test Nature Objective

Evidence Here we see the output from `/var/log/secure`



There are a lot of errors thanks to the recent NeWT scan.

Unfortunately, TripWire is not installed on [IPCop](#).

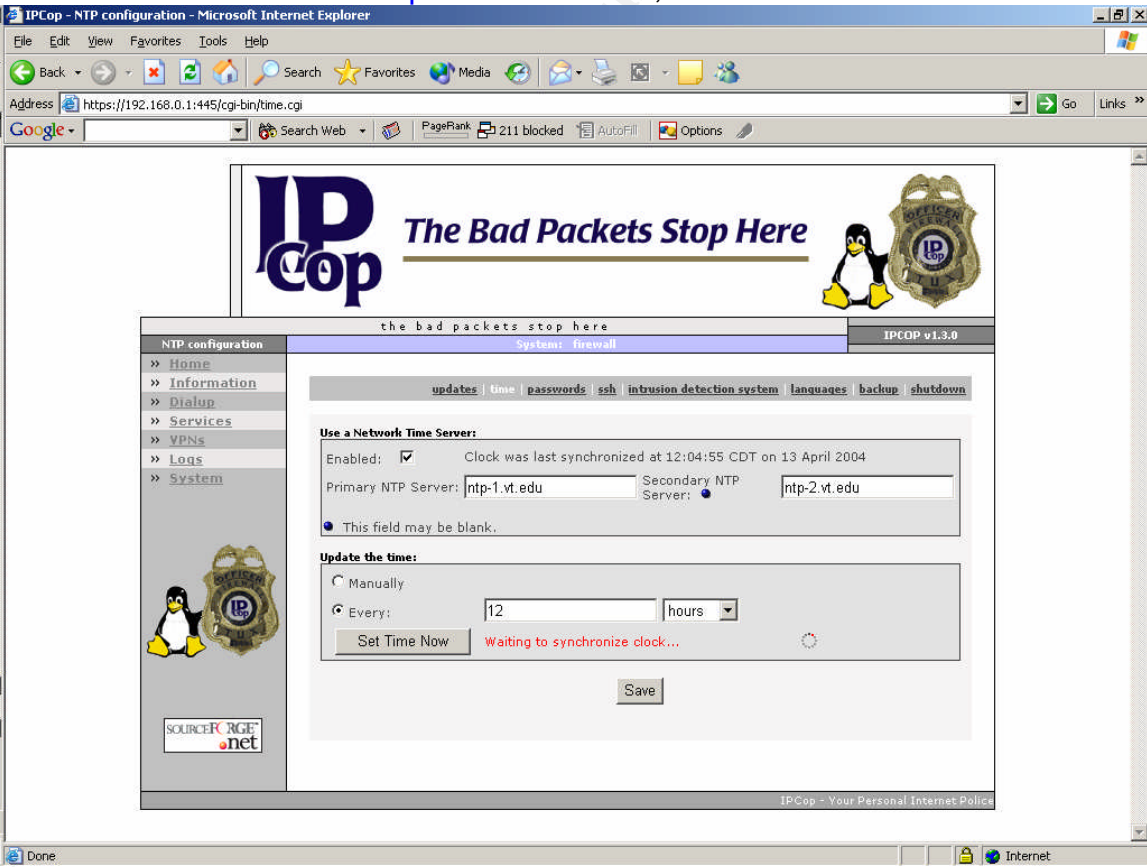
Findings `/var/log/secure` shows that login failures are being logged, but the lack of a TripWire database may make detection of subversion difficult (FAIL).

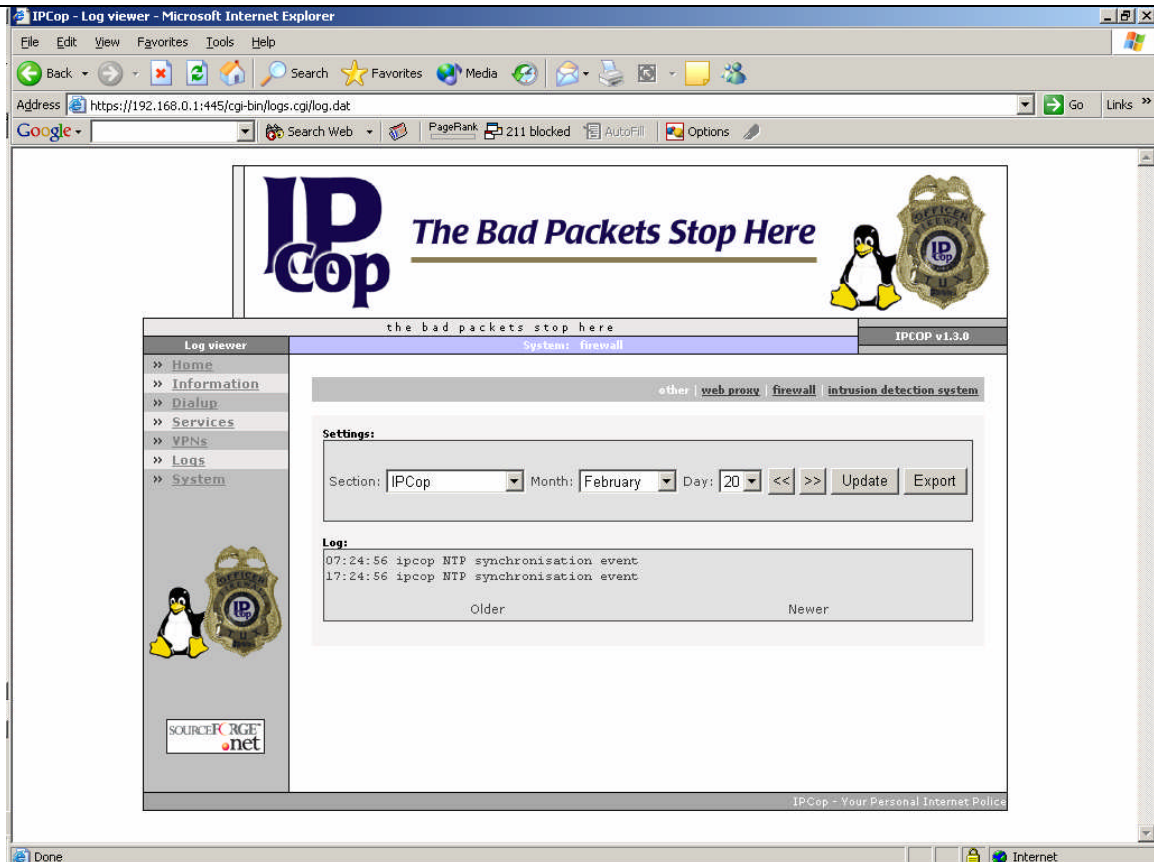
18) Ensure NTP is in use

Reference [6]{9} [7]{9.2}

Risk Inadequate Environment/State Detection

Explanation We need to ensure the system clock is set appropriately, so that reviewing the

	logs makes sense.
Testing Procedure	<p>To check NTP settings:</p> <p>From an internal machine, go to IPCop's IP Address https://192.168.0.1:445 to access IPCop's web based interface.</p> <p>Click System->time.</p> <p>Ensure "Enabled" is checked</p> <p>Ensure IPCop is set to a Primary/Secondary NTP server that you or the system admin know and trust.</p> <p>Ensure IPCop is set to update reasonably frequently (I like it about 12 hours).</p> <p>To further ensure that ntp is updating, you can check the logs by clicking Logs->other. Next to "Section", choose IPCop.</p>
Test Nature	Objective
Evidence	<p>Here is a screenshot of IPCop's NTP interface, with the time servers set.</p>  <p>Here is a screenshot of the logs, showing the update occurring.</p>



Findings NTP was configured and working (PASS).

19) Ensure Secure Administration

Reference [3]{Linux:3} [7]{2.8}

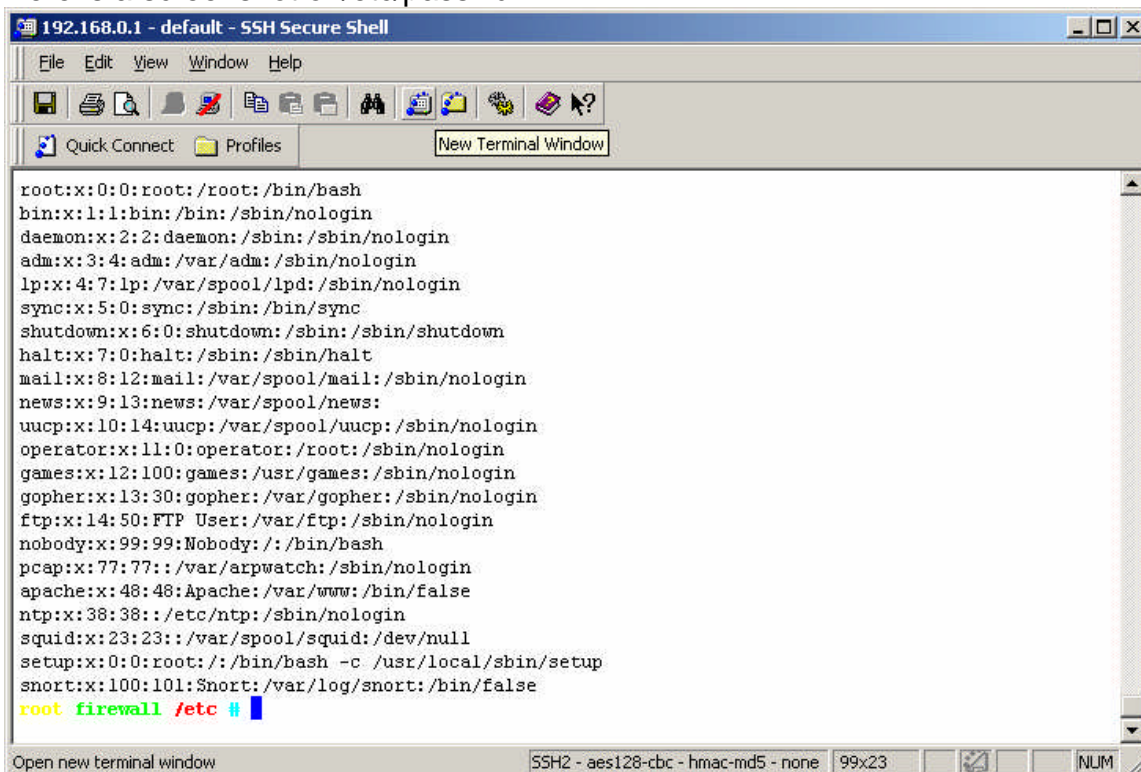
Risk [IPCop](#) is subverted due to Administrative Misconfiguration or Improper [IPCop](#) Implementation

Explanation Since I am the only user of the machine, there should be only an administrative(root) login. The only methods of login should be secure (SSH or SSL, require a password).
It should be confirmed that no known vulnerabilities exist for the version of OpenSSH on the [firewall](#).
We are not as concerned about the OpenSSL (Web GUI) interface, as it is not accessible from the **external interface**.

Testing Procedure On the console, type `less /etc/passwd`, and ensure that the root account is the only account that can bring up a console. To do this, inspect every line of the `passwd` file, examining the last field. If an account besides root contains a viable shell (such as `/bin/bash`) in this last field, then the [firewall](#) has failed this aspect of the test
Inspect the external services, and ensure that insecure remote access services are not enabled. See the checklist item "Audit the [Firewall](#) Rulebase" on accessing [IPCop](#)'s "external services" interface.
Use NeWT to ensure that no SSH vulnerabilities exist.

Test Nature Objective

Evidence Here is a screenshot of /etc/passwd



We see that root is the only login that goes to /bin/bash

We know from our NeWT(1) and NMapWin(3) Scans that port 22 is the only open external port that hosts a service on the [firewall](#), so it is the only method of remote login.

Since internal users aren't considered a threat, we will only be concerned with login methods from the external interface.

Unfortunately, our NeWT scan turned up some possible vulnerabilities from it's SSH specific scans (Look back at evidence #1). These vulnerabilities warrant closer inspection

Findings SSH is the only method of login, root is the only user, but there are vulnerabilities in the version of OpenSSH running in [IPCop](#) that warrant further inspection (FAIL).

Section 4.1 Executive Summary

In its current (near default) state, [IPCop](#) is extremely effective at blocking inappropriate traffic directed at my internal clients, with only a few, relatively harmless exceptions. It is, unfortunately, at relatively high risk of subversion. Due to an out of date and vulnerable remote administration service, a skilled attacker could feasibly gain control of the system. To make matters worse, the lack of any file verification system means such an attacker could easily cover his tracks, delaying his discovery indefinitely. This fact completely negates the protection [IPCop](#) offers. Once the [firewall](#) is subverted, the attacker could not only disable the protection it provides, but even use the [firewall](#) itself to stage

attacks on the internal clients. This could result in a loss of internet access, site availability, and data.

Section 4.2 Audit Findings

One of the most effective methods of determining overall system vulnerability is the TBS, or Time Based Security Method. The general idea is that we want to ensure that our Protection, or the time it takes an attacker to successfully infiltrate the system, is greater than our Detection and Reaction time added together ($P > D + R$). In plain English, we want to ensure that once an attack begins, we know about it and have stopped the attacker before the task is complete.

Unfortunately, our Protection time is very low due to the SSH vulnerability, and Detection could be postponed indefinitely without a file verification system to help us catch the culprit. This leaves a very definite possibility of an attacker coming in and subverting our system before we would even know about it.

There are also 2 less critical issues:

- 1) There are in fact some variations of inappropriate traffic which can get through or get a response from the firewall. Examples include traffic with a source address from the internal network and icmp timestamp requests.
- 2) Very little is done by the firewall to mitigate a DoS attack.

Section 4.3 Audit Recommendations

Thankfully, with a few fixes, we can turn this situation around.

My primary recommendations for further securing [IPCop](#) include:

Install Tripwire, as outlined in checklist item "Ensure Access Logging and File Verification are Enabled"

Update OpenSSH to the newest version. Until this is done, remote access to port 22 should be disabled.

Once a week:

- Update NeWT and run a scan against the [firewall](#).

- Do a Tripwire compare against a known good initialization database.

- Check for [IPCop](#) Updates

Once a day:

- Check Firewall logs

- Check IDS logs

- Check /var/log/secure

The above will insure that any attack on my system will be promptly noticed.

Finally, I recommend creating the following startup script in /etc/rc.d

```
#!/bin/sh
```

```
echo 0 > /proc/sys/net/ipv4/tcp_syncookies
```

#helps combat DoS attacks

iptables -I FORWARD 1 -i eth1 -p tcp -s 192.168.0.0/255.255.0.0 -j REJECT

#blocks internal addresses coming through the external interface

#there should be a rule similar to the one above for every

#unallocated/private address range.

iptables -I INPUT 1 -p icmp --icmp-type timestamp-request -j REJECT

#blocks timestamp replies

These improvements greatly improve our TBS situation.

Looking at our TBS Formula for **IPCop** subversion:

P=until the next SSH vulnerability is discovered. SSH is the only form of remote access to the system. If it is secured (either by patching to the newest version or simply closing the port), it will become very difficult for an attacker to find his way into our system.

D= No longer than a week (when a Tripwire compare is performed), probably much sooner (daily checks of /var/log/secure and IDS/Firewall logs would provide clues that something was wrong).

R= One day. The instant I read the logs (at least once a day) I can take action.

This scenario is much more tolerable. In this state, IPCop is nearly impossible to subvert or circumvent.

References:

- 1) Goldschmitt, Harry. "IPCop v1.3.0 Installation Manual". 2003.
<http://www.ipcop.org/1.3.0/en/install/html/> (April 18, 2004).
- 2) Brice, James et. al. "Administrative Guide". 2003.
<http://www.ipcop.org/1.3.0/en/admin/html/> (April 18, 2004)
- 3) SANS Institute Track 7.1, Auditing Principles and Concepts. 2003. Chapter 4, Audit Checklist: Linux
- 4) SANS Institute Track 7.2, Auditing the Perimeter. 2003. Chapters 1-4.
- 5) SANS Institute Track 7.4, Network Auditing Essentials.
- 6) Horne, Jeff. "Auditing a Symantec VelociRaptor Firewall: An Independent Auditor's Perspective". November, 2003.
http://www.giac.org/practical/GSNA/Jeff_Horne_GSNA.pdf (April 18, 2004)
- 7) Madison, Elaine. "Linux Firewall Audit".
http://www.giac.org/practical/GCUX/Elaine_Madison_GCUX.pdf (April 18, 2004)
- 8) Sweetser, Frank. "Auditing Perimeter Defenses in a Home Office Environment with an OpenBSD Firewall/VPN Branch Tunnel Gateway – An Administrator's Perspective". February 19, 2004.
http://www.giac.org/practical/GSNA/Frank_Sweetser_GSNA.pdf (April 18, 2004).