

# Monitoring and Controlling Suspicious Activity in Real-time With IP-Watcher

Michael Neuman  
*En Garde Systems*  
St. Louis, MO USA

## Abstract

*When a network intruder is suspected, the ability to monitor and track his actions in real-time is critically important. While programs like Tcpcdump have provided the ability to do this for some time, their output is generally too complex and extensive to extract useful information from, especially in anywhere near real-time. In addition, they only allow the administrator to watch as an intruder moves freely about the network. The IP-Watcher program was written specifically to allow the security administrator to monitor network log-ons in real-time, record only the relevant streams of data as evidence, and more importantly, control the intruder by terminating his connection, or even taking it over. This paper describes the IP-Watcher program, its implementation, uses (and dangers!), and future directions.*

## 1. Introduction

Once suspicious activity is detected, the ability to monitor the responsible user is extremely important. Primarily, it allows the administrator to establish the intent of the user. Even after it is determined the user is an intruder, continued monitoring of him may be more valuable than immediately cutting him off. Rarely will an intruder have access to only one account or have simply stolen a user's password. More commonly, an intruder will exploit a security vulnerability which doesn't require a password, and he will also install back-doors to allow easier access to the system later. By continued monitoring, these security vulnerabilities can be discovered as well as many of the back-doors. Knowledge of these allow the administrator to clean up the affected systems easier, and secure other systems on his network from future attacks. Aside from discovering how the intruder entered the system, continued monitoring may help assess damage. From this, the administrator can learn how many systems were penetrated, if root access was gained on these machines, what files were accessed or destroyed, and if other sites are being attacked. Continued monitoring also has the benefit of allowing the administrator to collect evidence against the intruder. While network traffic logs

probably have never been tested as legal evidence, they may prove to be a valuable piece of evidence either in determining guilt or damages.

Because continued monitoring's primary purpose is to determine the extent of damage and to collect evidence, it should be done on a network-wide basis, rather than on a per-machine basis. In large networks, security administrators rarely have access to every machine on the network, and even if they do, determining which machine is being targeted, and then logging onto that machine without being noticed is difficult at best. A distributed tool which monitors each machine and allows the administrator a shell on that machine is one possible solution, however such a system would require a great deal of maintenance, and possibly open even more security holes. Therefore, monitoring at the network level is the easiest and most efficient solution.

Unfortunately, with either a distributed or a network based tool, there often is far more risk in allowing an intruder continued access than benefit. He may turn malicious and begin destroying files or disrupting networks. This could result in lost work and destroyed files, or worse, he may steal or alter proprietary information. The value of this information, once stolen, is gone forever and can never be retrieved, and altered files may be impossible to find across even small networks. All this could happen while the security administrator is held responsible for not shutting off the intruder's access immediately. While shutting off all network access for a couple weeks as machines are cleaned up and further secured may be the safest way of responding to an intruder, if business depends on the network connections, significant money could be lost, and the intruder will probably be back anyway, either through a back-door that wasn't found, or the security hole of the week. And, if the intruder does come back, most likely he'll be angry he was caught and try to "get even".

There is a way to minimize the risks associated with allowing an intruder continued access. A program called IP-Watcher was created specifically for the purpose of monitoring and controlling network intruders. IP-Watcher allows the administrator to see exactly what the intruder sees and what he is typing. This data is extracted from

network traffic and presented to the administrator in real-time. The administrator can then see what the intruder is doing, as he does it. The administrator can also record the relevant data streams either as text for written transcripts, or as raw data for later replay through IP-Watcher.

IP-Watcher monitors all log-ons at the network level and therefore, no access to other machines is required. If an intruder starts a new connection, that connection can be selected from IP-Watcher, rather than forcing the administrator to log-on to the machine the intruder is on. Network monitoring also has the benefit of being passive and undetectable, so the intruder won't know he's being watched.

Most importantly, IP-Watcher minimizes the risk of granting further access to the intruder by allowing the administrator control over the intruder. The intruder's connection can be instantly terminated when he begins to look at or do something he shouldn't. In an emergency, makeshift firewall filtering can be setup without access to the router, allowing the administrator some extra time to investigate the intruder's activities, or allowing him to protect the network when he doesn't have immediate access to the firewall. In addition to denial of service, the administrator can even take over the intruder's connection. Taking over connections can be useful for the administrator to abort dangerous commands or investigate remote accounts the intruder has been copying files to. Terminated connections, taken over connections, and the traffic filtering appear as network interruptions to the intruder, so it is not an obvious sign he has been discovered. All of these allow the administrator to monitor connections while minimizing the risk to his network and data.

There are other tools which monitor networks such as Tcpcdump [1], Etherfind, Netlog [2], and SNIF [3]. None are specifically for log-on connections, so a great deal of data must be sorted through by hand to determine which connections exist, and what data is traveling over them. Only SNIF extracts the data in any meaningful way in real-time, although none allow the administrator to control the actions of the intruder. Finally, none of the tools allow the administrator to dynamically filter out the relevant data streams so, if evidence collection is desired, hundreds of megabytes of data must be sorted through.

## 2. Overview of the IP-Watcher Program

This section provides an overview of the IP-Watcher program. First, a look at the features of the program is provided. Next, some of the limitations of its network traffic monitoring approach is described.

### 2.1 Features

Fundamentally, IP-Watcher is a network monitor

(sniffer) with a graphical or text interface. It can be installed without any additional hardware or software on any SUN with root access and can monitor any traffic on the network segment the monitoring machine is installed on (see "Sniffing Limitations" below). A user interface allows IP-Watcher to provide a great deal of functionality not available in any non-interactive sniffers. Those sniffers require the user to have a deep understanding of TCP/IP and a lot of time to decode the extensive output. The user must remove protocol overhead, eliminate retransmissions, correlate data by connection, and finally, translate the result into meaningful data. Even if the user knows beforehand which specific connection he's interested in and sets the network sniffer accordingly, the filter can't be changed dynamically if a new connection is started, and even if it could, because of the significant amount of time and effort needed to decode the recorded traffic, the user would only know after the fact that he should have switched connections. IP-Watcher, on the other hand, handles the time consuming and complex operations for the user. No understanding of TCP/IP is required, data is automatically correlated by connection for the user, extraneous protocol information is filtered out, and the user can monitor any log-on or mail connection at any time.

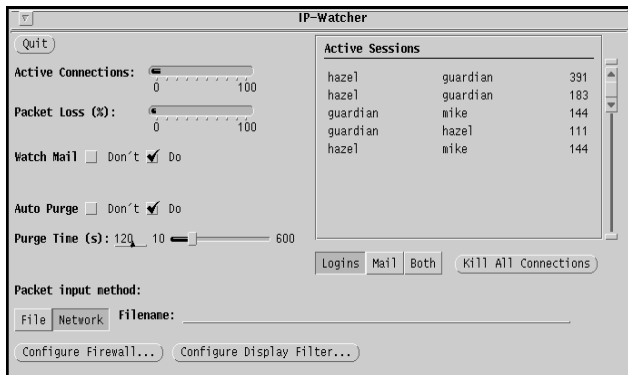
In addition to its passive monitoring capabilities, IP-Watcher provides a variety of active countermeasures. These countermeasures can be performed on any connection at any time by the user and are implemented using a technique called IP spoofing. IP spoofing is not the same as sequence number prediction as described by Bellovin [4] and Morris [5] and was allegedly used by Kevin Mitnick against Tsutomu Shimomura's computers. Because the countermeasures affect preexisting connections, no sequence number prediction is necessary. Through IP spoofing, IP-Watcher can immediately terminate connections, provide simple firewall filtering of a subnet, and even take over connections. This allows the administrator control over intruders by giving him complete control over each connection.

IP-Watcher has two user interfaces: a graphical X-Windows interface and a text interface. Both provide similar monitoring and active countermeasure features.

### 2.2 Graphical Interface

The graphical interface to IP-Watcher (see figure 1) presents the user with a list of active connections on the network segment as well as a variety of configuration options. At the top right of the window, a list of every active session monitored by IP-Watcher is displayed along with the number of bytes which have traveled over that connection since it was last selected. In this example, there are several connections to guardian from both hazel.EnGarde.com and mike.EnGarde.com, and the first

connection on the list has communicated 4460 bytes since it was last chosen by the user.



**Figure 1. The IP-Watcher Main Window**

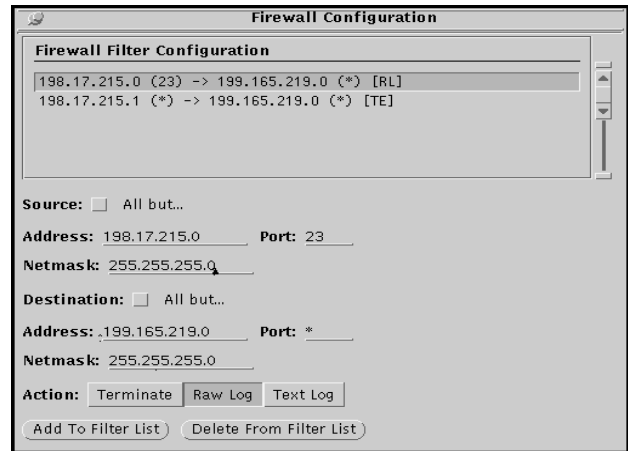
Beneath the Active Sessions list is a three button toggle between “Logins”, “Mail” and “Both”. These control the type of network traffic displayed on the Active Sessions List. The “Logins” setting shows each connection to the telnet and rlogin/rsh ports, “Mail” shows the list of email messages which have been recorded, and “Both” shows both the Login and Mail sessions. If the user isn’t interested in Mail, monitoring and recording of this type of data can be turned off by the “Watch Mail” checkbox.

All forms of traffic are removed from the Active Sessions list if they are inactive, or, in the case of Mail, older than the number of seconds specified by the “Purge Time” slider. If expiring old or inactive connections isn’t desired, auto-purge can be turned off by a checkbox.

Near the bottom of the window, the user can specify the packet input method. By default, IP-Watcher reads its data directly from the network; however, if the user has recorded a connection to disk using the raw log feature of IP-Watcher, that log can be played back by entering its filename. The playback looks identical to normal input from the network, with recorded connections listed in the Active Sessions list which are selectable by the user.

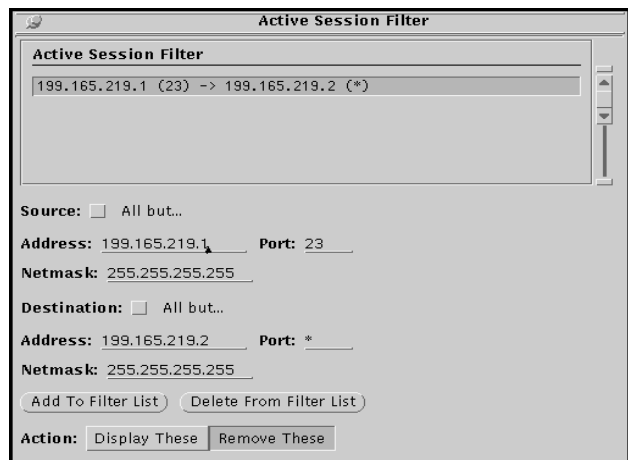
There are also active countermeasures available from the main window. The button beneath the Active Sessions list, “Kill All Connections”, immediately terminates every active login connection displayed on the Active Sessions list. The second active countermeasure, “Configure Firewall”, allows the user to setup makeshift firewall filtering and logging. The Firewall Configuration window (see figure 2) allows the user to specify source and destination address masks and ports for which all TCP connections will be instantly terminated or logged. Automatically logging connections is useful for unattended operation, whereas automatically terminating connections

is useful if the user needs to keep an intruder out while he gets a real filter entry put into the firewall, or if he needs some time to investigate something an intruder did.



**Figure 2. The Firewall Configuration Window**

Also available from the main window is the Display Filter configuration (see figure 3). On busy networks, hundreds of connections can be simultaneously active. To help reduce the amount of extraneous data presented to the user, IP-Watcher allows the user to configure Display Filters. These filters limit the number of connections presented to the user by including or excluding only those patterns which the user specifies.



**Figure 3. The Display Filter Window**

When the user chooses a connection, the IP-Watcher connection window is displayed (see Figure 4). All data transmitted from the server to the client is displayed in the main text window. This window does VT100 emulation, and consequently, screen oriented programs, such as vi or emacs, will also be displayed correctly, exactly as the user would see it. A single line underneath the main window

displays all data sent from the client to the server. This includes passwords and other text which normally is not echoed back by the server.

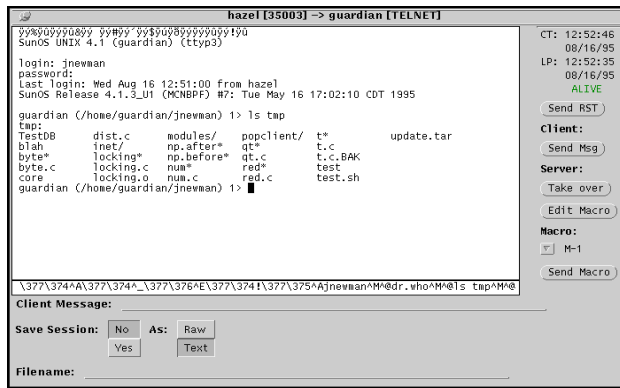


Figure 4. The Watcher Connection Window

To the right of the main text window are two clocks, a connection status indication, and the active countermeasure buttons relevant to this particular connection. The clocks display the current time (CT), and the time the last packet was transmitted for this connection (LP). Beneath the clocks is a display indicating the connection is still "Alive". The first active countermeasure, "Send RST" allows the user to instantly terminate the current connection. The second countermeasure, "Send Msg", and the "Client Message" text input area beneath the main window, allow the user to send a message to the client side of the connection. "Take Over" allows the user to hijack the current connection to the server. In other words, the user becomes the client and can type directly to the shell, or any other program that was running on the server at the time. "Edit Macro" and "Send Macro" allow the user to specify, hijack, and automatically send prearranged key sequences. Because taking over a connection is not extremely stable as the protocol is being spoofed, sending a predefined message is a good solution to ensuring the entire message is received. Except for "Send Msg", neither the client nor the server will notice anything out of the ordinary from the use of the active countermeasures. To the client and the server, terminating the connection looks like the network went down (telnet will report: "Connection Closed by Foreign Host" and rlogin will report: "Connection reset by peer"). Taking over a connection appears to the client as though the network is simply slow—no data is received back from the server, and to the server, it appears as though the original client is still connected.

At the bottom of the connection window are several controls to allow the user to log each connection. By default, connections are not logged, however, individual connections can be logged to separate text files, or multiple

connections can be logged to a single raw log file. All data sent from server to client is recorded in the text files, and all data sent in either direction along with time stamps is saved in the raw log file. Raw logs can be played back by IP-Watcher which replays the data as it happened. In other words, all transactions are displayed exactly as they would have appeared had the user been watching the live connection. Playback can be sped up or paused, and the individual connections in a playback can be further logged into a text file.

### 2.3 Text Interface

IP-Watcher also has a text interface (see figure 5) which is useful in monitoring a network when X-Windows isn't available or is filtered out between the user and the target network. The text interface is similar to the X-Windows interface, and is valuable in situations where only limited bandwidth or tty connections are available to the user.

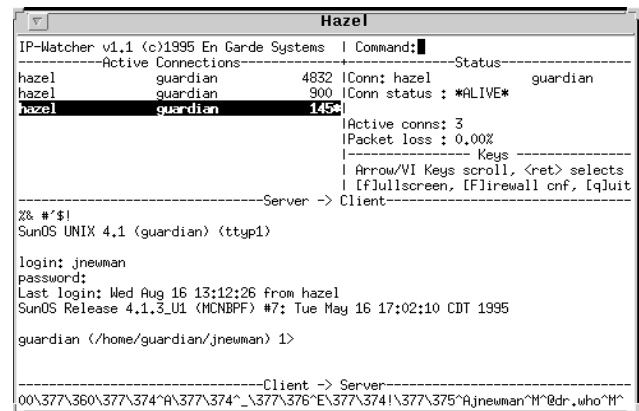


Figure 5. The IP-Watcher Text Window

The active sessions list, in the upper left corner, displays all of the active login sessions on the current subnet along with the number of bytes which have been transmitted over each connection since that connection was last selected by the user. At the top right are statistics about the current connection and the IP-Watcher program, including the number of active connections, packet loss, and whether or not the current connection is alive or dead. The remainder of the screen is taken up with the server to client and client to server data windows. In the example above, the user logged in as "jnewman", and the server prompted for a password which the user entered as "dr.who".

### 2.4 Sniffing Limitations

All of the features IP-Watcher offers are dependant on its ability to monitor network traffic. However, network sniffing does have several limitations. First, the data must be in a readable form such as plaintext ASCII. If traffic is encrypted, IP-Watcher has no way to decode it and present

it in any meaningful way. Second, the data must be a part of a stateless or at least a guessable state protocol. This is because the initiation of a connection is often not observed and therefore any initialization data is lost. Fortunately, TCP has a guessable state because there really is no context—if there is data, there is a connection. Higher level protocols are more problematic. With telnet, rlogin, SMTP, ftp, etc., meaningful data can be derived even if the transaction is joined in the middle. X-Windows, on the other hand, is extremely difficult to extract useful information from without the context of windows and fonts.

The most severe limitation of network monitoring is the limited domain of the data. With a broadcast Ethernet network, sniffing is limited to the traffic traveling over the subnet where the monitor is being run. Figure 6 shows a hypothetical network configuration with several Ethernet subnets. Machine W, which runs the IP-Watcher program, can only monitor traffic traveling over Subnet 2. In this limited example, the only traffic IP-Watcher would not be able to monitor is a connection between machines A and B. Since their traffic is limited to Subnet 1, IP-Watcher will not be aware of the connection. Ethernet switches and hubs often eliminate broadcasting by only transmitting packets to their specific destination. If this is the case, a network sniffer could only monitor traffic between itself and another machine.

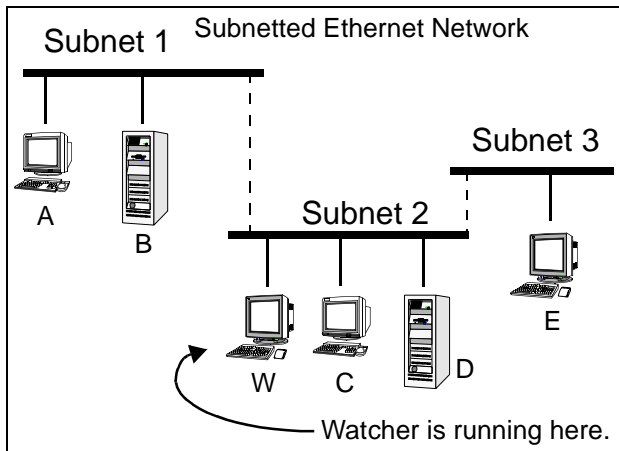


Figure 6. Example Ethernet Network

As a result of these limitations, the placement of IP-Watcher needs to be carefully considered. If the primary threat to a network is from external attackers, IP-Watcher needs to be placed on the network nearest the router to the outside world. If the primary threat is to a specific service subnet from either internal or external attackers, IP-Watcher should be placed on that particular subnet.

### 3. Implementation of Active Countermeasures

This section provides the technical details of how active countermeasures are implemented in IP-Watcher.

#### 3.1 IP Spoofing

IP spoofing takes advantage of the reliance on IP addresses, TCP ports, and sequence numbers for authentication of the remote system. Data is assumed to be from the remote host if it presents the correct identifying information to the server. This is not a correct assumption as a network monitor can easily record the identifying information and reproduce it.

When a TCP connection starts, both sides randomly create a sequence number and exchange them with the other side. Each side uses its sequence number to identify its place in the data stream to the other side. The other side uses this sequence number as an acknowledgment (ACK) number. A common exchange is as follows: the sender transmits a packet with its sequence number to the receiver. If the receiver gets the packet correctly, it acknowledges the packet by sending a reply with the receiver's sequence number, an acknowledgment flag, and the ACK number it's acknowledging. In this way, if there's a network fault, the receiving side can discover missing data, and specifically request it from the sender. To accommodate this, both sides must be ready to deal with old or new sequence and old ACK numbers. Figure 7 shows the two possible exchanges where duplicate (or old) sequence and ACK numbers are sent.

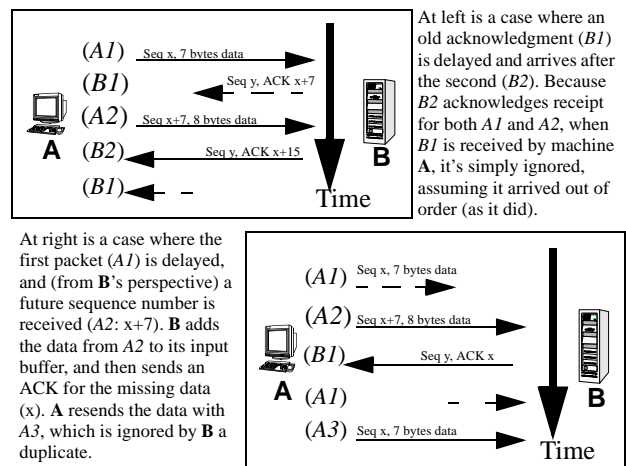
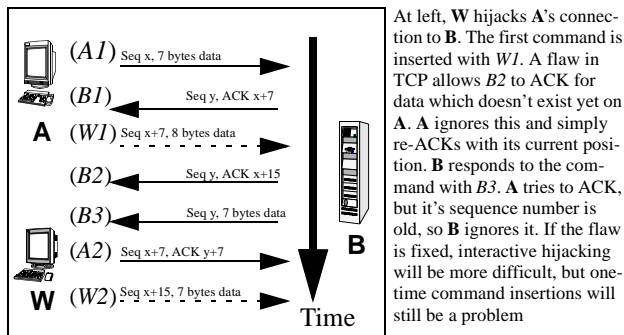


Figure 7. Valid Protocol Scenarios

The two scenarios in figure 7 allow the real endpoints of the connection to ignore each other once the connection has been hijacked. Each side assumes it has missed packets or is getting retransmissions, and will send acknowledgments for the missing data or ignore the repeated data. Figure 8

shows how these two methods can be combined to hijack a TCP connection.



**Figure 8. Hijacking an IP Connection**

The server accepts retransmitted data from **A** as a result of an unstable network. The TCP specification states that retransmitted data should be ACK'd for and dropped. Therefore, **B** ignores any data from **A**. **A**, on the other hand, notices the ACK from **B** for data which doesn't exist yet. In response, **A** ACKs, hoping to reset the stream to **A**'s position and drops the packet. Consequently, **A** ignores any data from **B** because the ACK number is no longer correct. Unfortunately, the results of both sides ACKing each other, attempting to reposition the stream, is an ACK "war" whereby each side ACKs in response to the other's ACK. This problem can't be easily solved without removing **A** from the network. Techniques for remotely denying service for an entire machine are widely published and are beyond the scope of this paper.

If hijacking isn't necessary, a one-time packet insertion can be used to terminate the connection. The TCP protocol allows for connections to be "reset" without acknowledgment, and therefore, resetting a connection is a unilateral decision. By inserting a single packet into the network with the correct sequence and ACK numbers, and connection can be reset, and therefore killed. By applying some intelligence to resetting connections, simple firewalls can be setup to filter based upon address or even data content.

### 3.2 How Sequence Number Prediction Differs From IP Spoofing

When a connection is first started, the two sides exchange sequence numbers. This uniquely identifies the connection, and makes it impractical for a third party outside of the path between the two hosts to insert packets since he has no way of knowing the proper sequence numbers. If the sequence number of the remote host can be guessed, however, a connection can be started by faking the handshaking and subsequent protocol exchanges without ever getting any data back. In this way, an attacker could

send packets to a server pretending to be from a trusted peer. Normally, the server would respond to the trusted peer with its sequence number. If the attacker is outside of the trusted network, he has no way of sniffing the response sequence number, and therefore, can't properly reply to the handshaking. In addition, if the trusted peer is still on-line when the server attempts handshaking, the trusted host will immediately reset the server's handshaking packet. As a result, in order to perform TCP sequence number prediction, the attacker must first take the trusted host off-line, then he must correctly guess at the next TCP sequence number on the server. If he can do these two things, he can send any data he wishes, and it will appear to have come from a valid connection from the trusted host.

Taking a trusted host off-line can be accomplished in a variety of ways. The technique allegedly used by Kevin Mitnick was to flood the host with unacknowledged connection requests. Eventually the TCP connection queue was filled to it's limit and the system was no longer able to respond to any further packets. Next, a number of connection requests were sent to determine the pattern of the sequence number generation on the server. Once the pattern was determined, a new connection was started, and when the server responded with the (predictable) sequence number, the attacker could ACK it properly to initialize a connection. At this point the attacker sends a command which would allow him a normal connection to the server.

There are several ways to defend against sequence number prediction. The first is to setup a filtering router to not accept packets from the outside which have addresses from the inside. The second is to not allow any explicit trust between hosts. Had the server not trusted the client, sequence number prediction would fail as the connection wouldn't have any more privileges than a non-trusted connection.

Whereas sequence number prediction relies on a variety of assumptions (host trust, routers not filtering obviously incorrect source addresses, and a predictable sequence number generator on the server), IP spoofing has no such assumptions or reliance on software bugs. Instead, IP spoofing is an abuse of an otherwise sound and properly configured protocol.

### 3.3 Related Work

Recently, a paper [6] devoted to the IP Spoofing attack was published. A slightly more powerful, but less practical technique for IP-hijacking is described, in which the attacker mediates the remainder of the hijacked connection. This, theoretically, allows the real client to continue his connection without realizing he's under attack, and gives the attacker complete control over every aspect of the connection.

## 4. Defensive Uses for IP-Watcher

IP-Watcher has a variety of security and non-security applications. It can be useful in any situation where monitoring network users or specific connection types is important, such as debugging network problems, understanding protocol exchanges, helping users with questions or problems, as well as auditing usage, and monitoring intruders.

Systems administrators often receive calls from users who are receiving consistent error messages, or who don't understand why something they're doing doesn't work. With IP-Watcher, the administrator can bring up and display that user's connection on his screen and watch him reproduce his problem. The administrator no longer needs to log in (or even have access) to the machine in question, nor does he need to distribute and use tools such as TAP or Advise which, if improperly distributed or installed, could be used maliciously. As a result, large networks which have dedicated user help centers will not need root access to every machine on the network to successfully diagnose and help a user with their problem.

IP-Watcher can also help in debugging network service problems. Probably the most often misconfigured service is SMTP such that connections hang or are ended prematurely due to newer features, a specifically formatted "HELO" command is not given, messages may not be forwarded correctly, or the SMTP daemon is not configured to accept messages to a certain address. IP-Watcher helps the administrator diagnose these problems by presenting him with an exact network transcript of the SMTP transaction, viewed in the context of all other SMTP connections on that network. In this way, incorrectly forwarded messages, rejected addresses, and incorrect protocol exchanges can be caught.

Bolstering network security was the primary reason IP-Watcher was written. It has been used to perform random Fraud, Waste, and Abuse audits, and in particular has saved several organizations from the embarrassment of hosting illegal software and pornographic picture archives. Its primary benefit, however, is to the system administrator who needs to monitor and control suspicious behavior. IP-Watcher allows the entire network to be monitored from a single machine. This reduces the administrative and technical overhead of tracking intruders by not requiring the administrator to log-on to each individual machine. The administrator also can record detailed evidence of exactly what the intruder did and when he did it. Finally, the risk of allowing the intruder continued access to the network is greatly reduced because IP-Watcher allows the administrator to control intruders by terminating their connections interactively or automatically.

## 5. Offensive Uses for IP-Watcher

Like most security tools, IP-Watcher can be subverted into an extremely dangerous hacking tool. Aside from the ability to monitor any connection in a graphical way, denial of service attacks are available at the push of a button, and connection take over allows firewall filtering to be bypassed and any one time password scheme to be defeated.

In the past, network sniffing was a threat because passwords often passed over the network in cleartext. An attacker could set up a network sniffer on a network backbone machine and login later to collect the passwords that were recorded [7]. The solution to this problem was the use of expensive smartcards, challenge/response schemes, prearranged password sequences, or encrypted password exchanges. Programs like S/Key, SRA telnet and FTP, and the default unencrypted Kerberos were all designed to eliminate the threat of password sniffing. Tens (hundreds?) of thousands of smartcards were sold for the same purpose. IP-Watcher demonstrates, through session take over, that password sniffing is no longer the biggest threat. An attacker could, for example, wait for a user to authenticate himself on a machine, and then hijack that connection. No one-time authentication system can prevent the theft of pre-established credentials. A program like IP-Watcher could even automate connection take over so it doesn't require human intervention. This, combined with the fact IP-Watcher requires no more access to the network than a simple password sniffer does, makes IP-Watcher, and programs like it, a potentially dangerous tool.

The only way to prevent connection hijacking is to completely encrypt, or cryptographically sign the entire connection. Depending on the cryptography scheme used, a one time password scheme may also be required. For example, a simple cryptography scheme which is weak and exportable is sufficient to prevent connection hijacking, as long as it takes longer than the lifetime of the connection to break the cryptographic key (as little as a few hours may be enough). If weak cryptography is desired for export purposes, it must be combined with a one-time authentication system to prevent an attacker from decrypting the data exchanged and extracting the real password. On the other hand, if a strong encryption system is used, reusable passwords are still viable as they can't be decrypted within the password's lifetime.

Currently, Kerberos allows for a completely encrypted connection, but only if the user can *kinit* from whatever remote host he's on, and specifically tells the telnet or klogin software to use encryption. Remotely running *kinit* is often not possible as the program may not exist, its realm configuration probably won't include the user's, and the user's firewall may filter out Kerberos requests. In addition,

the encryption needs to be specifically turned on, when it should instead be the default.

Several other public domain packages have recently become available which perform the required encryption. Stel [8], SSH [9], and a modified SRA telnet all are publicly available, exportable, and sufficiently protect against IP hijacking attacks.

## 6. Future IP-Watcher Features

IP-Watcher has many areas in which improvements can be made. Aside from cleaning up the quirks inherent in the Xview interface and finishing a port to Motif, a variety of active countermeasures could be added to IP-Watcher. These countermeasures would allow less conspicuous session take over, allow the administrator to put the intruder into a guaranteed-safe “cage”, perform spoofing queries for more information about the intruder, and control network conditions to give the administrator more time to investigate the intruder’s actions.

When a connection is taken over, the client is put into a state where the network looks extremely slow, and never comes back, even though other connections may be running at normal speed. Repetitively taking over connections will eventually make the intruder suspicious, and cause him to become more careful in hiding his tracks or he may turn malicious and remove files before disappearing. While controlling intruders is valuable, the point is to give the administrator time to collect evidence and find back-doors, not have the intruder become suspicious and alter his activities. Consequently, the ability to return connections after they are taken over would be useful. To the client it would look as if the network was slow for a while, but came back after a short period of time, at which point the intruder could continue their activities without suspicion. Aside from the security application, an administrator could take over a connection, fix the problem a user has gotten himself into, and then give it back. Giving back connections would allow the administrator greater flexibility in the types of investigations he could perform.

There are two ways in which returning connections could be implemented. IP-Watcher could continuously translate the sequence and ACK numbers, or the client and server could be spoofed until their sequence numbers match. The first technique would require IP-Watcher to be continuously run while the session is active, and force an unstable protocol into an even more unstable state. The second is the better of the two as it only requires a one time interaction between IP-Watcher and the two endpoints. Once the sequence numbers match, IP-Watcher no longer needs to do any more IP spoofing. The problem with this technique is that the client’s sequence number is probably several hundred bytes behind what the server is ACKing.

Therefore, the client must be tricked into sending that missing data while IP-Watcher spoofs ACKs for it. This can be accomplished in a variety of ways: VT100 escapes often force the terminal to transmit a cut and paste buffer or an escape querying the position of the cursor could be sent, or spoofed TELNET protocol negotiations would force data to be sent by the client without the intruder’s knowledge.

TELNET negotiations also allow IP-Watcher to retrieve a variety of information about the client. Newer telnet clients allow most environment variables to be queried including USER, HOME, and PWD. These can be used to help identify the remote user. Further, the DISPLAY environment variable is often propagated from host to host so it can be used to identify the originating host of the intruder. At the very least, if DISPLAY isn’t propagated, it usually contains the name of the host the intruder logged in from.

Another potential countermeasure could further reduce the risk of allowing an intruder continued access to the network. IP-Watcher could be configured to hijack the client side of the connection and connect it to a program running on the IP-Watcher host. In other words, a controlled Unix simulation could be running as part of IP-Watcher, and the client could be hijacked and connected to it. Any action performed by the client would affect the Unix simulation only. The administrator could then control the simulation by inserting “interesting” files, logs of the intruder, or other traps to determine what the intruder does when he runs across specific types of information. If the intruder immediately e-mails the interesting file somewhere, the destination address can be recorded and further investigated. Or, if the intruder immediately deletes any log files containing incriminating evidence, it is probable that he’s likely to have destroyed other traces of himself. Determining the types of evidence the intruder considers incriminating is another sample application. For example, a file could be constructed containing a potential lead to finding the intruder—if the intruder deletes the file, it is probably a valid lead. The administrator could interact with this simulation by creating scenarios or even talking to the intruder, posing as whomever the administrator wishes.

When monitoring an intruder, it is also helpful to slow down his actions to give the administrator enough time to react and approve them. A feature easily added to IP-Watcher is the ability to slow down network traffic to any given speed. This would cause a lot of additional traffic to be generated, but it would allow the administrator time to react and investigate the intruder’s actions.

## 7. Summary

IP-Watcher is a powerful tool for the system administrator. It allows real-time monitoring, evidence

collection, and control of network intruders in a graphical, user-friendly way. This allows network monitoring which was never possible before, and helps to increase the overall security of the network.

Aside from simply monitoring and logging network traffic, IP-Watcher provides a variety of active countermeasures to control the intruder. The controls include instantly terminating connections, setting up makeshift firewall filtering, or even taking over the intruder's connection. IP-Watcher performs all of its active countermeasures through the use of IP spoofing, which is a potentially dangerous technique of assuming control over existing connections. Through the malicious use of IP spoofing, firewall filters can be bypassed, and one-time password systems are worthless. The only protection against IP spoofing is completely encrypted connections, as the methods used to spoof packets do not rely on any bugs, so they can't be easily fixed.

IP-Watcher does have the potential for malicious use. However, its announcement and release is warranted as there are available protections. By carefully releasing IP-Watcher's more dangerous features, in an attempt to limit its use to security defense, as well as announcing the vulnerabilities inherent in IP, hopefully by the time malicious hackers have implemented their own versions, secure encrypted communications will be commonplace.

IP-Watcher was originally created as both a demonstration of network security problems as well as for defending a network against intruders. It has been a success in both roles.

## 8. IP-Watcher Availability

IP-Watcher is a commercially available software package.

It currently runs on Suns under Solaris or SunOS 4.x. The code is generally very portable and is limited only by the availability of a good packet filter. Currently, IP-Watcher uses the Berkeley Packet Filter (BPF), Sun's Network Interface Tap (NIT), and Solaris' Data Link Provider Interface (DLPI). There are text (curses), and Xview interfaces, and Motif interface is complete except for a terminal emulator widget.

For more information about IP-Watcher, please contact us at:

En Garde Systems  
525 Clara Avenue, Suite 202  
St. Louis, MO 63112  
(314) 367-6402  
(314) 367-3555 FAX

Or, see IP-Watcher's URL:  
<http://nad.infostructure.com/watcher.html>

## 9. References

- [1] S. McCanne, V. Jacobson, "The BSD Packet Filter: A New Architecture for User-level Packet Capture," *Proceedings of the 1993 Winter USENIX Conference*, January 1993.
- [2] D. Safford, D. Schales, D. Hess, "The TAMU Security Package: An Ongoing Response to Internet Intruders in an Academic Environment," *Proceedings of the Fourth UNIX Security Symposium*, October 1993.
- [3] J. Alves-Foss, "An Overview of SNIF: A Tool for Surveying Network Information Flow," *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, February 1995.
- [4] S. Bellovin, "Security Problems in the TCP/IP Protocol Suite," *Computer Communications Review* 19:2, April 1989, pp. 32-48.
- [5] R. Morris, "A Weakness in the 4.2BSD UNIX TCP/IP Software," Bell Labs Computing Science Technical Report #117, February 1985.
- [6] L. Joncheray, "A Simple Active Attack Against TCP," *Proceedings of the Fifth UNIX Security Symposium*, June 1995.
- [7] CERT, "Ongoing Network Monitoring Attacks", CERT Advisory CA-94:01, February 1994.
- [8] D. Vincenzetti, S. Taino, F. Bolognesi, "STEL: Secure TELnet", *Proceedings of the Fifth UNIX Security Symposium*, June 1995.
- [9] T. Ylonen, Helsinki University of Technology. Software freely available through the Internet.