Real time interception and filtering of instant messaging protocols

Warren Harrop[1]
Centre for Advanced Internet Architectures. Technical Report 030919A
Swinburne University of Technology
Melbourne, Australia
104304@swin.edu.au

***Abstract*- Internet engineers must actively develop tools and technologies to support safe, secure and trustworthy methods for Lawful Interception. This paper describes the development of MsgSifter, a multi-protocol, instant message interception utility based on the existing open source software package 'msgsnarf'. Logging and certain protocol filtering changes were made to the msgsnarf code. In addition, a Java based GUI was created to place an easy to use control and filtering layer over the command line msgsnarf program.**

Keywords- Instant Messaging, Network Sniffing, Security, Lawful Interception, MsgSifter, Dsniff

I. Introduction

Dr. Philip Branch's paper, 'Lawful Interception of the Internet' [1] discusses the present day issues surrounding Lawful Interception of Internet data. The paper argues that without input on behalf of research and Internet standards bodies, Lawful Interception methods could prove to be a threat to both security and privacy. It is further argued that Internet engineers are to be actively involved in the creation of Lawful Interception tools and utilities to prevent Lawful Interception becoming a liability for everyone.

With the aforementioned ideas in mind a program 'MsgSifter' was created, initially striving to be a proof of concept utility. The selective interception and real time processing of data is performed on messages sent by the most popular instant messaging protocols. MsgSifter provides the ability to watch, filter and log multiple instant message conversations in real time. In a Lawful Interception situation, more advanced programs could preform this collection and filtering on multiple protocols, but as a starting point, a set of instant messaging protocols were chosen because of their current wide usage. MSN, AIM, Yahoo and ICQ are supported.

The Centre for Advanced Internet Architectures (CAIA) uses FreeBSD as its standard Unix-like operating system, thus, freeBSD 4.8 was the platform used for the software development of MsgSifter.

II. Related Work

A number of options presented themselves for the development path of MsgSifter during research into related work. Coding the entire solution from scratch was not done due to the limited time and large amount of coding work involved. Four pieces of software were considered to be possible starting points.

Pkthisto [2] is a packet traffic analysis tool for the study of on line game traffic, and is being used within CAIA's Game ENvironment Internet Utilisation Study (GENIUS) project. It has a real time packet sniffing ability that could have been modified to recreate TCP sessions, and the MsgSifter code could have been built from this point. Two other programs, msn666 [3] and aimsniff [4], already do part of the desired task, sniffing their respective protocols, msn and aim. Each of these 3 programs are quite good in their own right, but combining or extending upon their respective source code to produce the desired results of this project would not have been a trivial matter in the time given.

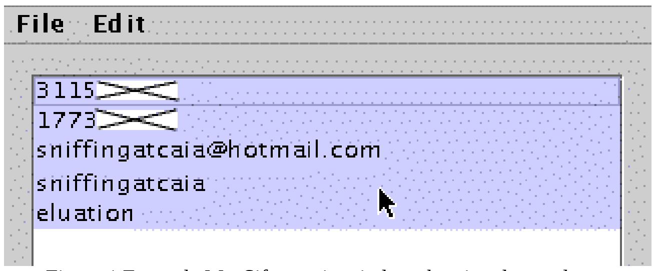
Finally, the code that was expanded upon is 'msgsnarf', part of the dsniff [5] package of programs. Starting with the msgsnarf program provided code to passively watch network activity, reconstruct TCP sessions and decode the most popular instant messaging protocols. Msgsnarf however has no GUI, no method for logging and no method for discerning between conversation flows - these features were added by this project.

III. Interception Architecture

The MsgSifter package builds on the existing msgsnarf code in two ways. The first is to add simple logging code to the existing msgsnarf source code, making it 'msgsnarf+'. The second is the creation of a Java based GUI application able to keep track of specific instant messaging conversations and display these in individual windows.

A. Features Of Implementation

MsgSifter has the ability to connect directly to msgsnarf+ (or an unmodified version of msgsnarf) and display conversations in real time. It also has the ability to scan msgsnarf+ logs, (or the stored standard output of a msgsnarf session) and run faster than real time giving 'play back' of conversations. While running, the main



3115[redacted]
1773[redacted]
sniffingatcaia@hotmail.com
sniffingatcaia
eluation

Figure 1 Example MsgSifter main window showing detected users

1 The author is currently a final year Engineering student at Swinburne University

window of MsgSifter is propagated with instant messenger user names as they are detected. As user to user conversations begin, a new message window is spawned for each conversation. This window will then continue to mirror the conversation as each user sends new messages. The ability to save a conversation window to a log at anytime from within the GUI is also provided.

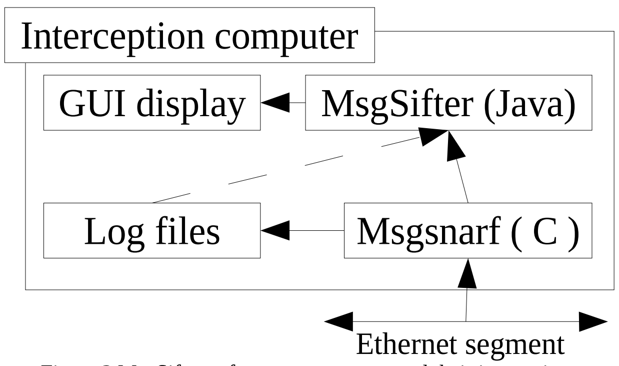


Figure 2 MsgSifter software components and their intercation

B. *Limitations Of Implementation*

There are a number of factors that limit the practical ability of the MsgSifter program to intercept instant messaging conversations. For example, msgsnarf must be active when the TCP session between instant messaging server and client is created. If this is not the case msgsnarf is unable to detect any message data.

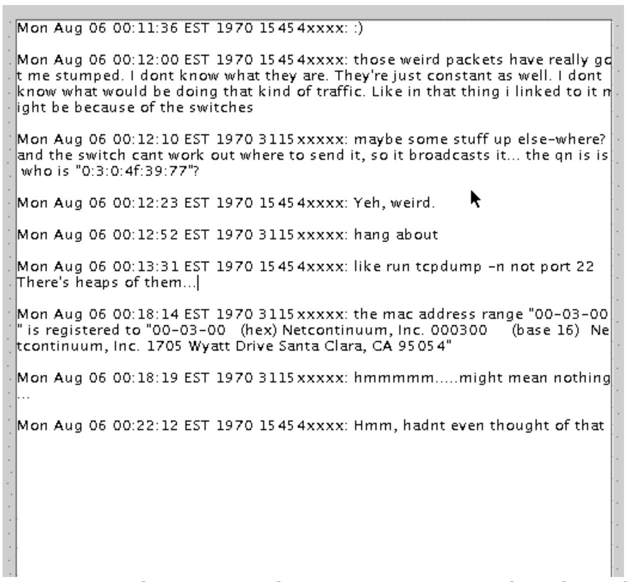
Furthermore there are certain problems with the msgsnarf ICQ/AIM parsing code that causes the start of messages to be clipped by a number of characters. This error is alluded to in the msgsnarf source code and will be resolved during future development of msgsnarf.

C. Software development

The msgsnarf program was changed by adding an additional feature. 'Msgsnarf+' appends all detected messages to a log to a file, 'msgsnarf.log'. MsgSifter can then be used to parse this log file at a later date to view individual conversations.

MsgSifter can be run in two different ways, these options are both available under the 'File' menu of the GUI. The first option 'Sniff now...' launches msgsnarf or msgsnarf+ in a new thread and reads message lines from its output. Each time a line is read from msgsnarf, a Java event is fired. When these occur the program adds newly found users to the list in the main window and places the received messages in the appropriate conversation window. To control what user's conversations are ignored, MsgSifter users can select or deselect user screen names in the main window. If their name is unselected when they initiate new conversations with other users, their conversations are ignored.

The alternative method for running MsgSifter is to use the log parsing feature, 'Parse Existing Logs...'. This reads in a msgsnarf+ 'msgsnarf.log' file or the saved standard output of a msgsnarf session. In operation it works just as if live sniffing were taking place. The only difference is that the log file is parsed for users first, then pauses. This allows the MsgSifter user to make decisions on what instant messaging users are to be ignored. After selection of another menu item the program then runs faster than real time creating and propagating conversation windows.



```
Mon Aug 06 00:11:36 EST 1970 15454xxxx: :)

Mon Aug 06 00:12:00 EST 1970 15454xxxx: those weird packets have really go
t me stumped. I dont know what they are. They're just constant as well. I dont
know what would be doing that kind of traffic. Like in that thing i linked to it m
ight be because of the switches

Mon Aug 06 00:12:10 EST 1970 3115xxxxx: maybe some stuff up else-where?
and the switch cant work out where to send it, so it broadcasts it... the qn is is
 who is "0:3:0:4f:39:77"?

Mon Aug 06 00:12:23 EST 1970 15454xxxx: Yeh, weird.

Mon Aug 06 00:12:52 EST 1970 3115xxxxx: hang about

Mon Aug 06 00:13:31 EST 1970 15454xxxx: like run tcpdump -n not port 22
There's heaps of them...

Mon Aug 06 00:18:14 EST 1970 3115xxxxx: the mac address range "00-03-00
" is registered to "00-03-00   (hex) Netcontinuum, Inc. 000300    (base 16)  Ne
tcontinuum, Inc. 1705 Wyatt Drive Santa Clara, CA 95054"

Mon Aug 06 00:18:19 EST 1970 3115xxxxx: hmmmmm.....might mean nothing
...

Mon Aug 06 00:22:12 EST 1970 15454xxxx: Hmm, hadnt even thought of that
```

Figure 3 A sample conversation beween two users, icq numbers obscured

IV. Limitations of real time interception

In a broader sense, but still valid to MsgSifter execution are the general limitations to real time interception. The underlying architecture of the Internet prevents simple access to any user's data from any point on the network, even if Lawful Interception is the reason for data collection. Currently the main limitation is physical access to the data to be intercepted. Three scenarios for data interception exist in current networks:

A common medium: If the interception computer and target computer are on the same shared medium, such as Ethernet running over thin coax or across a twisted pair medium connected by simple hub, data can be collected. In these situations all network data is available to all network nodes. This scenario is becoming very uncommon, but allows interception to occur simply.

Switched environment: In most situations now, Ethernet is deployed using layer 2 switched devices. Only data destined for a specific device is placed onto its medium. This increases network throughput but also stops simple network sniffing from occurring. A simple interception device is unable to work in this situation.

A number of methods exist for sniffing in a switched environment, some of these methods are included as utilities in the dsniff package. The main two methods include using a program like macof to send large amounts of 'random' frames onto the network in the hope of overloading the lookup tables of switching layer 2 network devices. This forces certain devices 'open', making them to act as hubs. Thus, the network segment then acts as a shared medium. Alternatively, using the programs arpspoof or dnsspoof to convince the network to redirect data to a delegated intercepting machine is possible. These solutions, although effective, are what would be used in a illicit 'cracking' situation and would not be generally considered Lawful Interception techniques.

The third option for data interception is using a bridge situation. Hardware can be installed at a privileged network position intercepting and passing on all movement of data. Hardware installation like this requires special access, but this is not a problem in a Lawful Interception situation as proper channels and authorities are consulted before hardware is installed. It is most likely that this form of data interception would be deployed in a Lawful Interception situation. Figure 4 below gives a simple example situation of such bridged interception. In the Figure 4, example MsgSifter is used with remote monitoring provided by an SSH tunneled X11 session.

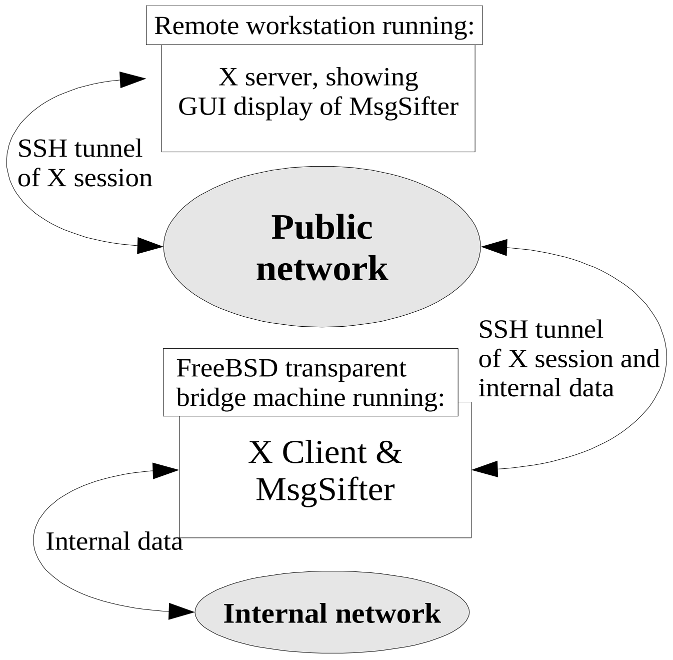

Remote workstation running:
X server, showing GUI display of MsgSifter
SSH tunnel of X session
Public network
SSH tunnel of X session and internal data
FreeBSD transparent bridge machine running:
X Client & MsgSifter
Internal data
Internal network

Figure 4 A simple implementaion of MsgSifter in a bridged interception situation with remote control provided via an SSH tunneled X11 windows session

V. Conclusion

MsgSifter and msgsnarf is not intended for use in a Lawful Interception situation 'as is'. Rather, this project is a prototype for further research and further software development. Our goal is to move the industry away from the idea that Lawful Interception of Internet data must be performed indiscriminately and close to the physical layer. Selective interception of data is possible by parties such as ISPs and only data requested by Law Enforcement Authorities (LEAs), as set out in warrant, need be handed over for use. It is the data discrimination code and not the GUI or raw packet sniffing code of MsgSifter that is the most pertinent to Lawful Interception and thus, for further investigation and development.

A number of possible extensions to MsgSifter have already been considered.

Ideally we should be able to decouple the interception of traffic and the review of intercepted traffic, both in time and space. However, msgsnarf and MsgSifter currently do not cleanly support remote monitoring. It is possible to indirectly decouple the interception and review locations by tunneling the java GUI's X11 session back over an ssh connection between interception host and an operator's desktop machine. Unfortunately this is clumsy and uses network resources inefficiently.

A major addition to the MsgSifter program will be the ability to verify log integrity. This will become a considerable issue with the Lawful Interception of digital Internet data. How can data be intercepted, stored and independently verified as true and correct at a later time? The inclusion of a trusted, secure, third system to collect time stamped, digitally signed hashes of log components from multiple interception devices should be investigated. This methodology would go a long way to protect against log fraud, an easily accomplished feat in the digital age and something that must be protected against.

Acknowledgments

This project was undertaken with the valued input of Associate Professor Grenville Armitage. The project attempts to tackle poignant issues raised by Dr Philip Branch and relies heavily on the code of msgsnarf by Doug Song.

References

[1] P. Branch, "LIFE - Lawful Interception for Everyone," http://caia.swin.edu.au/life, June 6th 2003

[2] "Game Environments Internet Utilisation Study", http://caia.swin.edu.au/genius/ (as of July 31st 2003)

[3] "MSN666 - msn sniffer", http://underground.or.kr/project/msn666/ (as of July 31st 2003)

[4] "AIM Sniff", http://sourceforge.net/projects/aimsniff/ (as of July 31st 2003)

[5] D. Song, "dsniff", http://www.monkey.org/~dugsong/dsniff/ (as of July 31st 2003)