SECURE P2P VOICE OVER IP USING DEEP PACKET INSPECTION

[1]Satish N. Gujar, [2]Dr. V.M.Thakare
[1]Information Technology Department,Shri Jagdish Prasad Jhabarmal Tibrewala University, Rajasthan
[2]Computer Science Department, Sant. Gadgebaba Amravati University, Amravati
Email: [1]satish_gujar@yahoo.com , [2]vmthakare@yahoo.com

Abstract—Skype is a Peer-to-Peer (P2P) Voice over IP (VOIP) chat program. It provides its clients with an inexpensive means to communicate worldwide via the internet through wired and wireless networks. In the past this application was limited strictly to computers, yet with continuous advancements in mobile communication, Skype phones and other mobile devices have recently hit the market in an attempt to capitalize on Skype's reliable connection algorithms. However, despite the success of this application, it is important to note that due to Skype's connection algorithm and the nature of P2P, a number of vulnerabilities emerge that threaten both users and their networks. This paper outlines how to block the Skype application through the use of Deep Packet Inspection. This novel approach is completely scalable to networks of any size as a means of blocking one of the largest threats to commercial and government networks today.

I. INTRODUCTION

Skype is an international enterprise with more than 100 million users worldwide [1]. It is available in 28 languages and can connect people in almost every country in the world. It provides chat services as well as video conferencing, and computer-to-telephone communication. It is extremely inexpensive with free domestic and computer-to-computer calls while international calls average $.02 per minute. However, its most notable feature is its ability to form connections on any network with internet access. These features have enabled Skype to build and maintain its large and growing user base. However, due to the P2P nature of this application, a number of vulnerabilities quickly present themselves. Although users may choose to disregard these vulnerabilities when they use Skype on their personal computers, corporations and government entities are not so quick to accept the same risks. The basis of this decision can be understood when reviewing the Skype User Agreement that must be accepted before one can install the application. One small portion of the agreement reads: "You hereby grant permission for the Skype Software to utilize the processor and bandwidth of your computer for the limited purpose of facilitating the communication between Skype Software users." (excerpt from 4.1). In this agreement, users essentially agree to give complete control of their computer and their network to facilitate the needs of Skype. Moreover, due to the nature of its connection, Skype is also prone to a number of common P2P vulnerabilities. These can range from buffer overflow attacks, to denial of service, to diminished network bandwidth. For large corporations and government entities, these types of attacks hinder productivity and efficiency.

However, despite all of these aforementioned issues, the largest vulnerability that Skype presents is the fact that it forms direct connections with unknown clients while also allowing these unknown clients to connect directly to it. As such, these connections can be exploited by malicious users as well as viruses, trojans, and worms. This was seen recently in September 2007 when the Ramex worm was able to easily spread through the Skype network. This worm not only disabled antivirus, malware, and Windows Update software on hundreds of thousands of Skype users, but it also installed a key logger to steal private information from these infected hosts [2]. As a result of these vulnerabilities, corporations and government entities alike actively try to restrict users from running Skype on their networks through firewalls and other olicies. However, due to Skype's complex connection algorithms, these policies are largely ineffective and often easily circumvented. The next section will reveal how this algorithm functions, followed by a section on a proven methodology to block it. Test results and their significance are then presented before concluding.

II. Background

Traditional means of using firewall rules to block unwanted applications have been found to be ineffective when applied to the Skype application [3]. These results are due to the fact that Skype was designed with a four-tier approach aimed specifically at circumventing firewalls and forming connections at all costs. At the foundation of this connection design is the unique application of the P2P model. Most P2P applications Yahoo Messenger, AOL Instant Messenger, etc.) form direct connections with their authentication servers and then connect the two users who wish to communicate. Skype on the other hand, uses its client base to forward traffic between its authentication server and between two hosts wishing to communicate [4]. As a result, Skype clients always form indirect connections between each other and the authentication server. With this in mind, the first tier of the connection design is the maintenance a list of valid IP addresses that Skype can connect through. Older versions of Skype maintained a list of up to 200 IP address that were updated every time a user logged in [4]. However, gaining access to this list of addresses has been made increasingly more difficult with newer releases. Despite this fact, the basic principle remains that firewall rules cannot be applied to block connection attempts because the destination addresses are continuously changing. Having defeated IP address blocking, Skype also conducts port hopping in order to circumvent any port based firewalls. Although its most common port is 33033, the application will alternate through a small range of port numbers during a single session. However, should it fail to establish a connection on its desired ports, Skype will fall back onto ports 80 (HTTP) and 443 (HTTPS) which are required for web traffic [5]. As a result of this second tier, port-based firewall rules are also ineffective at blocking Skype. Tier three of the connection design uses both TCP and UDP traffic. During a normal session, the application will use TCP traffic to establish a connection and then will fall back on to UDP traffic for the remainder of the session. Once the initial connection is made, Skype can then use TCP or UDP traffic interchangeably with respect to the restrictions of the network it operates on [5]. The final tier of Skype's connection design is its encryption. Skype uses 256-bit Advanced Encryption Standard (AES) for its packets. After the initial connection, all packets are encrypted and decrypted with 1536 to 2048-bit RSA private keys [5]. As a result, the likelihood of identifying a packet signature once the connection is established also becomes problematic. Thus, due to the effectiveness of this four-tier connection approach, the only known way to identify Skype traffic is through Deep Packet Inspection (DPI) conducted by Network Intrusion Detection Systems (NIDS). Snort-Inline™, a widely used NIDS, currently uses four rules to identify Skype. The first two focus on web requests and identify the "getnewestversion" and "getlatestverision" requests that Skype generates when it forms a connection [6]. The second two rules are ASCII representations of packets that can be used to identify Skype. They are 16 03 01 00 and 17 03 01 00. Although both of these are effective at identifying Skype, these ASCII representations fall short in two areas. First, they are generic to

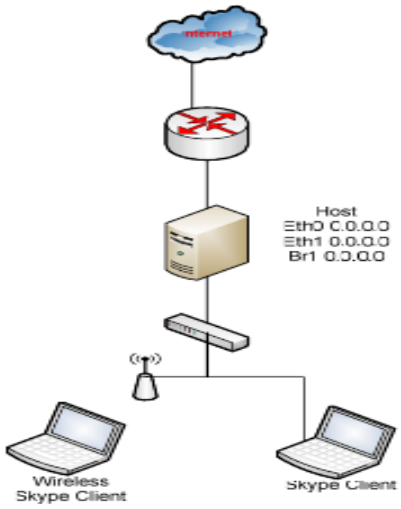
all P2P applications. Therefore, if a user chooses to block packets containing these two strings, s/he will inadvertently block a number of other desired applications. Second, although baring packets with these strings will block other applications, it will not deter Skype. Should Skype not be able to connect using these two strings, it will generate new packets in order to circumvent the NIDS rules. In summary, Skype's connection approach is extremely effective at circumventing all types of firewalls and network policies. It cannot be prevented from connecting through any network connection that has access to the internet.

III. Methodology

In order to block Skype's complex connection algorithms, a new approach must be applied to the traditional means of application blocking. Instead of strictly using firewalls, or DPI methods alone, a hybrid of the two must be utilized. The following subsections will outline the components needed to build such a system, the rule-sets that must be applied to identify Skype traffic, and how all the pieces fit together to form this hybrid system. The methodology will then conclude with a discussion of the testing results.

A. Components

There are four main components required for this system to function properly. The first is a Host computer with two Network Interface Cards (NIC's) and a Ubuntu operating system (a Debian-based version of Linux). The two NIC's are used to allow traffic to pass through the host machine for analysis. The second component is a transparent network bridge that allows traffic to traverse between the two interface cards while the host itself maintains an IP address of 0.0.0.0. This null IP address is a result of the bridge being transparent and allows the host computer to also remain transparent on the network. The Ubuntu operating system was chosen for its compatibility with Snort-Inline™, a widely used freeware Network Intrusion Detection System that conducts DPI. The fourth and final component of this hybrid system is a firewall. For this design, Iptables was chosen as it comes prepackaged in most Linux installations. Additionally, this firewall is most favorable to our design as rules can be added and removed dynamically using simple scripts. Having outlined the four main components, that make up the Skype blocking system, we can then place it on any network between the Skype users and the outside internet. One such example is depicted in Figure 1 below:



B. Rules

In order to actually block Skype, rule-sets must be created in order to identify Skype packets during DPI. To determine how to create these rules, log on attempts were captured using a program called Wireshark which records all of the packets entering and exiting a computer. Wireshark captures were conducted on different computers, at different times, with different Skype accounts, different Skype versions, and different networks to ensure that all possible scenario's were accounted for in our data collection. After collection, the data was then analyzed using a program called Araxis Merge which compared three separate log on attempts at a time for consistencies. The results of this comparison then prompted the creation of rules in three categories: Keywords, Ports, and Content. Keywords. When evaluating the data, it was discovered that the current Snort-Inline rules for identifying Skype update requests were still applicable. Therefore, we created two rules to reject any packets containing get newest version combined with a string. Additionally, we then choose to create a new rule that would reject any packets containing the "Skype" string. This was done to prevent any users operating behind our system from viewing any website containing the word Skype or from being able to download newer versions of the software. However, it is

important to note that this rule is optional as it may prove to be an inconvenience. Ports. The data also revealed a consistency in the ports that it uses. Although we know that Skype can attempt to operate on any port it chooses, the data showed that port 33033 was used extensively for TCP traffic. After checking to ensure no other common applications use this port, two new rules were created to block any TCP traffic coming or going to this port. Content. The most interesting discovery of the data comparison came in the form of content strings. Although the "16 03 01 00" and "17 03 01 00" ASCII strings mentioned above were easy to identify, the data also revealed that an extra "00" was always included with those strings. This is significant because it differentiates Skype packets from all other P2P applications. As such, two rules were created to drop all outgoing packets containing "16 03 01 00 00" and all incoming packets containing "17 03 01 00 00". After implementing these two new rules with the keyword and port rules, we then used Wireshark to collect more data to see how Skype reacted when these packets were dropped. In comparing these data captures, it was discovered that should Skype not be able to use these two content strings, the program tries to send a new packet containing the ACSII strings: "16 03 01 00 cd 41 03 00 09 80 40 04 08 c0 00" and "00 0c 01 17 03 01 00". Thus we created our eighth rule to drop all outgoing packets containing these strings. Upon implementation of this final rule, we once again collected Wireshark captures to see how Skype would adapt to the new restrictions we created. After analyzing the data, we found that Skype could not adapt to our rule-set, as it continued to try and rebroadcast the string that our last rule dropped. As a result, the Skype would-be-user is met with this screen:

C. Application

By denying Skype from obtaining a connection, the majority of the vulnerabilities associated with this application are effectively neutralized. However, since Snort-Inline™ only drops the packets that match pre-established rules, our solution has not yet blocked all connections with the potentially malicious hosts that Skype attempts to connect to. Consequently, we must implement firewall rules to ensure that all communication between the Skype client and the unknown hosts are severed. This is achieved using two Perl scripts. The first script parses the Snort-Inline™ alert logs for the unknown host's IP address. The second script then takes that address and dynamically creates Iptables rules that drop all packets coming from or going to that address. This essentially adds an extra layer of security and ensures that malicious users can not exploit the Skype client.

D. Testing

After implementing this hybrid solution, it was tested to meet three objectives. First, it was tested to ensure that it was completely effective. Our solution was tested multiple times on different computers, with different user accounts, at different times of day, with different versions of Skype, and different networks, both wired and wireless. Despite all of these differences, our solution proved to be effective 100% of the time. Second, our solution was tested for bandwidth degradation. These tests revealed less than 3% loss in upload bandwidth with no measurable loss in download bandwidth. This loss was then deemed acceptable under two considerations. First, although there is a small loss in upload bandwidth, network users running the Skype application will likely have a larger impact on network degradation. Second, and more importantly, this loss in upload bandwidth can be attributed to our use of a freeware NIDS (Snort-Inline™). This consideration was supported by a test that showed a consistent degradation while Snort-Inline™ was run with no rules in place. Although Snort-Inline™ is very efficient, it is by no means an enterprise level solution. Should our solution be applied to an Enterprise NIDS, we are confident that the bandwidth degradation will be significantly less. Finally, this solution was tested to ensure that it did not interfere with any other applications. Although it is impossible to test every application available today, we tested our solution against some of the most popular applications in use. With our solution in place, the following applications operated unimpeded: AKO Chat Client, Aim, Aim Pro, Aim Express, Yahoo Messenger, Mozilla Firefox, Internet Explorer, Pandora, Windows Update, MS Office 2007, Symantec Antivirus and Firewall,

Quicktime, Windows Media Player, VPN Client, iTunes, Ruckus

IV. Contributions

First this approach to application blocking represents a break from the tradition means. Normally firewalls and NIDS are applied independently with no interaction between the two. Although this approach has worked well in the past, it falls short when applied to programs with complex connection algorithms such as Skype. The solution described above represents a valuable contribution for network administrators. Although not necessarily a new concept, this theory of combining two tools already in use (firewalls and NIDS) is greatly advanced by the effectiveness of our solution. Additionally, the combination of the two tools into one complimentary system will reduce potential vulnerabilities for network administrators. Instead of just blocking ports and IP addresses (firewalls) or just blocking individual packets (NIDS), both can be handled simultaneously to enhance the overall security of the networks they protect. Second, this paper outlines a prototype design for blocking Skype. Skype is known for its ability to circumvent firewalls and to cause significant vulnerabilities in the networks that it operates on. Currently Skype can operate unimpeded on any corporate or government network that has access to the internet. As a result, it has been labeled one of the largest security threats operating on these types of networks. The contribution of this prototype is not only that it blocks Skype, but also that it is completely scalable to any size network. For the cost of one computer, this system can be built and installed on any network in a matter of hours with the only network downtime being the time it takes to plug in two wires. Additionally, should network administrators not wish to use this complete solution, they can simply add the rules identified above to their current NIDS in a matter of minutes in order to immediately prevent Skype users from operating on their networks. The return on investment of this approach, the cost of protecting all sensitive information contained on government and commercial networks versus the cost of losing or having it compromised, is enormous.

V. Future work

The prototype described above was designed as a complete solution to the problem of blocking Skype. As such, it was designed to operate with very minimal network administrator involvement. However, future work for this prototype could allow for customization by network administrators so that it can be completely incorporated into the networks they monitor.

Yet, it is important to note that this solution, although complete for an IPv4 network, has not yet been tested against an IPv6 network. As the IP addresses vary between the two, and the current solution has been designed solely using IPv4 addressing, an IPv6 version of this solution would definitely serve to be an area of formidable future work. However, on a different level, aside from the solution itself, a more notable area of future research lies in combining firewalls and NIDS into a hybrid network protection system similar to our prototype. This will allow for increased security as it both stops

immediate threats and allows the network administrator to dynamically block continued threats from the same source, thereby. increasing the overall security of any network.

VI. Conclusion

Skype is one of the largest threats existing on government and commercial networks today. This paper provides a unique and proven solution to that security problem and not only blocks Skype, but completely removes all of the vulnerabilities associated with this application. The return on investment of this solution is significant when considering the price associated with sensitive information being protected as well as the easy scalability and low cost of this solution.

VII. Acknowledgement

We wish to extend special thanks to 2LT Rucheera for research in support of this paper.

VIII. References

[1] Dario Bonfiglio, Marco Mellia, Michela Meo, Dario Rossi, and Paolo Tofanelli, “Revealing Skype Traffic: When Randomness

Plays with You," Presented at SIGCOMM 2007, Koyto, Japan.

[2] InfoWorld: Bubbles the worm adds keylogger. Internet WWW page, at URL: < http://weblog.infoworld.com/ zeroday/archives/2007/09/bubbles_the_wor.html >, September 2007.

[3] Blue Coat, White Paper – Best Practices for Controlling Skype within the Enterprise. Internet WWW page, at URL:<http://www.onixnet.com/Blue%20Coat /ImportMedia /downloads/whitepapers/ BCS_controlling_skype_wp.pdf > February 2006.

[4] Tapio Korpela, "IT Security Evaluation of Skype in Corporate Networks," TKK T-110.5290 Seminar on Network Security, Helsinki University of Technology, December 2006.

[5] Salman A. Basat and Henning Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol," Department of Computer Science, Columbia University, September 2004.

[6] Snort: Vulnerability Research Team. Internet WWW page, at URL: < http://www.snort.org/vrt//>, June 2008.