Interested in learning more about security?

SANS Institute InfoSec Reading Room

Skype: A Practical Security Analysis

The purpose of this paper is to suggest best practices and recommendations when running Skype. Although Skype is available for myriad different hardware platforms, this document will focus on the Mac, Windows, and Linux environments.

Skype: A Practical Security Analysis

GSEC Gold Certification

Author: Bert Hayes, bhayes@infosec.utexas.edu

Adviser: Dominicus Adriyanto

Accepted: October 9 2008

Table of Contents

1. Introduction

Skype is communications software that allows users to communicate with each other in real time using VOIP (Voice Over IP), video chat, or more traditional text chat. It is unique among other IM (Instant Messaging) applications in that Skype runs over a decentralized P2P (Peer to Peer) network rather than routing all communications packets through a central server. Skype is designed to work out of the box on modern networks, and has no problems working behind a NAT (Network Address Translation) device or other firewalls. Because of its decentralized architecture, Skype makes extensive use of strong encryption, making casual eavesdropping or impersonation all but impossible.

Many network and systems administrators take a dim view of Skype because historical use has shown that it can be a bandwidth hog. Other administrators fear that Skype's inherent ability to traverse firewalls is a security risk. And some administrators feel the combination of Skype's encryption and its binary only, closed-source nature make it a black box, or complete unknown that has no place being on a well-maintained network. While these are all valid concerns, they should be considered in the context of local network policies and weighed against the benefits that Skype can provide. In many cases running Skype in a well-managed environment can mitigate these risks.

The purpose of this paper is to suggest best practices and recommendations when running Skype. Although Skype is available for myriad different hardware platforms, this document will focus on the Mac, Windows, and Linux environments. Unfortunately, many of the management features available to systems administrators are available only for Skype running on Windows.

2. Definitions

Skype Client

The Skype client is the collective term for the system and the software being used to run the user-facing Skype application.

Super Node

A Super Node is a Skype client that has a public IP address and enough spare CPU cycles, RAM, and bandwidth to take on additional duties for the Skype P2P network. Super Nodes hold a portion (up to several hundred users) of the distributed Skype directory.

Although they accept and reply to directory queries from other Skype users, super nodes do not actually relay communications content (voice, video, chat, etc.) for other users. According to the Skype Guide for Network Administrators (Skype, 2006) super nodes are restricted from using more than 5KBps (kilobytes per second) of bandwidth.

Relay Host

A relay host is a Skype client that has a public IP address and enough spare CPU cycles, RAM, and bandwidth to relay Skype content for other Skype users who are behind restrictive firewalls or are otherwise unable to communicate with each other directly. Although it is theoretically possible for a relay host to relay communications traffic for more than one session at a time, Skype claims that this is not commonly seen in practice. The Skype application places limits on how much bandwidth can be consumed by a relay host (Skype, 2006):

File Transfer:3KBps

Voice Call: 4KBps

Video Call: 10KBps

Skype Login Server

This is the only centralized piece of the Skype network. This collection of servers

handle user authentication and manages the creation of Skype usernames.

3. Major Points

Skype has the potential to be a bandwidth hog, but high bandwidth consumption while the user is not actively using Skype can be mitigated through changes in configuration or deployment. Provided that mitigating steps are taken, bandwidth consumption while the user is not actively using Skype is low.

Although the core technologies used in Skype have not had a known security vulnerability in years, Skype still presents an attack vector for spam, phishing, or the transfer of malicious code by way of traditional social engineering. However, these attack vectors already exist on user's systems in e-mail clients, web browsers, or other IM applications. Skype's functionality makes it no more or less inherently secure than other network communication applications.

Skype has a publicly available API and third parties can write applications to take advantage of Skype's functionality at the user level. Arguably, the most noteworthy examples of these applications are the viruses and worms that occasionally make their rounds on the

Skype network. The overwhelming majority of these can be defeated with simple security measures such as user education (don't click the link that comes to you unexpectedly, especially if it's from someone you don't know) and up to date anti-virus software. Additionally, users of Skype on Windows can edit a registry key to disable third party applications from using the Skype API. See Appendix A for a summary of Skype security bulletins.

Skype's encryption of communications is secure enough to prevent casual eavesdropping and it provides a measure of non-repudiation in that unless a user's credentials (username and password) have been compromised, it is nearly impossible to impersonate another user.

Although the network-based security threats that Skype can present can be mitigated through secure deployment and configuration, it still presents the very real threat of data leakage and information disclosure. In network environments that are subject to strict communication regulations, Skype presents a simple to use, highly encrypted channel for sensitive data exfiltration that can be difficult to detect and hard to block. Administrators who manage systems and networks that are subject to legal and administrative communication regulations may want to prohibit Skype to reduce the risk of unauthorized communications.

In less restrictive network environments, such as higher education networks, Skype is an attractive alternative to using traditional telephone service for costly long distance calls, particularly when collaborating with colleagues in other countries.

Skype should not be relied on for strong anonymity. Although it uses encryption to protect its network traffic, if this traffic is captured, it is trivial for the certificate owners (Skype and its parent company EBay) to decrypt the traffic. Additionally, Skype takes no measures to hide its presence on the system it's running on. It is easy for a forensic analyst to discover the presence of Skype and to enumerate a user's contact list among other details. Unless configured to use a proxy (a feature that is native only to the Windows versions of Skype at the time of this writing), the direct peer-to-peer nature of Skype communications traffic indicates the IP address of the sender or receiver; in many cases this could lead to identification of either party.

Skype is designed to easily traverse firewalls and works fine behind a NAT firewall. This feature makes Skype extremely difficult to block with a traditional perimeter firewall. Even in very restrictive network environments, if either HTTP or HTTPS traffic is allowed, Skype will use port 80 or 443 for its traffic.

Although Skype excels at getting around restrictive firewalls, it does not modify the

firewalls or their rules themselves in any way. With the exception of a host-based firewall, it does not require specific ports or port ranges to be opened on the perimeter/NAT firewall. Skype does use an arbitrarily designated port to listen for connections on. This can be specified during installation, otherwise Skype will select one at random.

Although Skype's encryption makes it impossible to detect the contents of a user's communications on the network, and its firewall traversal abilities make it extremely difficult to filter at a border firewall, use of Skype can be detected on a network and it can be blocked by an Intrusion Prevention System (IPS) or other reactive Intrusion Detection System (IDS), or an ambitious administrator. An examination of Skype's network behavior follows later in this paper.

Skype does an excellent job of getting around restrictive firewalls and obfuscating the contents of its communications, but it does not represent a secure computing platform, nor is it a secure storage platform (Skype, 2600). Text based chat sessions are logged by default, and information such as contact list entries, IP addresses, and Skype cookie information are all kept on the client systems with little, if any, obfuscation. On Windows systems, this data is kept in C:\Documents and Settings\username\Application Data\Skype and in /Users/username/Library/Application Support/Skype on Mac OS X systems. In Linux, this data is kept in ~user/.Skype/

The Skype application is very much an opaque black box. The code itself is not open source; it is distributed as a binary only, which uses packing and other obfuscation methods to defeat reverse engineering. Skype will detect when certain debuggers are running in the operating system and cease to function in an attempt to protect itself from prying eyes. It can be very difficult to know exactly what Skype is doing on your system, or what data about your system is being transmitted to super nodes and login servers. Users with extreme paranoia to satisfy may want to run Skype on a dedicated system.

4. Observations

Network Utilization

Skype's P2P network architecture means that where possible, users will be sending data streams directly to and from each other. This is easy to imagine where each host has a publicly facing IP address and communications are unfettered by restrictive firewalls. In cases where users are behind a firewall or are otherwise using Network Address Translation (NAT), direct communication is still frequently possible even if inbound UDP packets are restricted to existing “sessions” initiated by the internal host. If both users are behind firewalls that prevent all outgoing UDP traffic, Skype will send its conversations using TCP through a third host that is publicly addressable. Such hosts are known as "Relay Hosts".

It's the use of such relay hosts that causes such consternation among network and systems administrators as this functionality means that Skype is consuming network resources even when the end-user is not using the Skype application. Although Skype has self-imposed limits on how much bandwidth in can consume, many administrators feel that this is inappropriate and fear that unchecked use of Skype will lead to inordinate bandwidth use. This behavior can be mitigated (or stopped altogether) during deployment or subsequent configuration of Skype.

Skype's EULA (End User License Agreement) addresses this relay host functionality in the following clause:

> "Utilization of Your Computer. Skype Software may utilize the processor and bandwidth of the computer (or other applicable device) you are utilizing, for the limited purpose of facilitating the communication between Skype Software users. Skype Software will use its commercially reasonable efforts to protect the privacy and integrity of the computer resources (or other applicable device) you are utilizing and of your communication, however Skype cannot give any warranties in this respect."

Some institutions prohibit Skype specifically because of this clause. Some administrators and legal counselors will advise prohibiting Skype because this functionality means that a third-party, for profit company is using University resources for the express purpose of furthering their commercial enterprise.

It is the informal of some legal counselors that this behavior is not fundamentally different from most other network applications that are developed and supported by commercial interests. For example, consider a user surfing the web and being subject to advertisements on a web site. Or someone who uses Yahoo mail and has an advertisement automatically appended to his or her email. Many other IM applications (and indeed many other network applications) are supported by the use of in-line advertisements.

Some protest that when agreeing to the EULA, the end-user agrees to let Skype use resources that he or she has absolutely no authority over, and that this alone is reason to prohibit its use. Informal legal opinion tells us that this should not be a concern: since the user has no authority over the resource he or she is granting access to, the authorization is not binding or relevant. It would be comparable to this author granting you, the reader, unrestricted access to my trans-oceanic canal in Panama.

For those wishing to eliminate the possibility of a Skype client becoming a super node or relay host, the simplest solution is to place the host behind a NAT firewall or otherwise restrict its ability to be publicly addressed. If other Skype users cannot see the host directly, the super node and relay host functions will simply not work and the host will remain a regular Skype client with no additional functionality.

For Windows systems, Skype's functionality can be managed at a number of levels. Skype configuration and policy settings are maintained in the following hierarchy:

```
HKEY_LOCAL_MACHINE Registry Keys
HKEY_LOCAL_USER Registry Keys
XML config files in
C:\Documents and Settings\user\Application Data\Skype\
Skype client user preferences and default settings
```

As of the 3.0 version, Skype can use Group Policies so that administrators can make system management changes to sets of enterprise users. The Skype Administrative Template can be found at http://www.skype.com/security/Skype-v1.5.adm

See Appendix B for a complete list of registry entries and configuration parameters that can be employed to help secure the Skype client when run on Windows systems.

Skype's Network Behavior

Although the contents of Skype communications are nearly impossible to identify due to its strong encryption, it is possible to identify Skype traffic on a network, and optionally block it. Automatic detection of Skype requires an Intrusion Detection System (IDS) and automated blocking of Skype requires a reactive IDS or Intrusion Prevention System (IPS). In this paper the open source IDS Snort is used in example IDS rules.

When Skype starts up, it exhibits some predictable network behavior. When the Skype

client is first installed, it checks in with ui.skype.com to let it be known that the installation was a success. This check is done over HTTP; the payload from a sample packet of this traffic looks like this (this check is from a Linux system):

```
GET http://ui.skype.com/ui/2/2.0.0.72/en/installed HTTP/1.1
Connection: Keep-Alive
Host: ui.skype.com
```

And the server's response is as follows:

```
HTTP/1.1 200 OK
Date: Fri, 15 Aug 2008 21:55:24 GMT
Server: Apache
Cache-control: no-cache, must revalidate
Pragma: no-cache
Expires: 0
Set-Cookie: SC=CC=:CCY=:LC=en-
us:TM=1218627324:TS=1212437324:TZ=:VER=2/2.0.0.72/0; expires=Sat, 15-
Aug-09 21:55:24 GMT; path=/; domain=.skype.com;
Content-Length: 0
Connection: close
Content-Type: text/html; charset=utf-8
Content-Language: en
```

Because the text in the GET request always includes the word "installed" and it is always directed at ui.skype.com, this traffic can be detected on the network using the following Snort rule:

alert tcp $HOME_NET 1024: -> $EXTERNAL_NET 80 (msg:"Skype Client has been installed"; flow:to_server,established; uricontent:"/ui/"; uricontent:"/installed"; reference:url,www1.cs.columbia.edu/~library/TR-repository/reports/reports-2004/cucs-039-

04.pdf; sid:1000006; rev:3;)

This won't alert you if a user has installed Skype on a personal system at home, and then plugged it into your network, so to detect Skype in use, additional IDS rules are needed.

If Skype has already been installed, it will check with ui.skype.com to make sure it's running the latest version. This check occurs every time Skype is started:

```
     GET
/ui/3/2.7.0.257/en/getlatestversion?format=plist&uhash=1fa9aeaf181295
287bdad5e032dffd0d8&machine=%0dMacBookPro2%2c2&osversion=1052
HTTP/1.1
     User-Agent: Skype for Macintosh
     Accept: */*
     Accept-Language: en-us
     Accept-Encoding: gzip, deflate
     Cookie:
SC=CC=:CCY=:LC=en:TM=1207596355:TS=1207595145:TZ=:VER=3/2.7.0.257/0
     Connection: keep-alive
     Host: ui.skype.com
```

Older versions of Skype will send out a slightly different GET request, looking for the "latestversion" instead of the "newestversion":

```
     GET
/ui/0/3.2.0.163/en/getlatestversion?ver=3.2.0.163&uhash=1fa538b0b938e
7b0450db64bc06a5b5d4 HTTP/1.1
     User-Agent: Skype. 3.2
```

```
Host: ui.skype.com
Cache-Control: no-cache
```

(Note: Although this version is 3.2.0 and would appear to be more recent than 2.7.0 in the example above, this packet is from a Windows system. At the time of this writing, 2.7 is the latest version for the Mac, and 3.8 is the latest version for Windows.)

This activity can be detected with the following Snort rules:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"Skype client
get newest version"; flow:to_server,established;
uricontent:"/ui/"; uricontent:"/getnewestversion"; content:"Host|3A|
ui.skype.com";
reference:url,www1.cs.columbia.edu/~library/TR-repository/reports/reports-2004/cucs-
039-04.pdf;
classtype:policy-violation; sid:1000007; rev:1; )
```

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"Skype client
get latest version attempt"; flow:to_server,established;
uricontent:"/ui/"; uricontent:"/getlatestversion?ver="; content:"Host|3A|
ui.skype.com";
reference:url,www1.cs.columbia.edu/~library/TR-repository/reports/reports-2004/cucs-
```

039-04.pdf;
classtype:policy-violation; sid:1000008; rev:1;)

Although using these two Snort rules together would seem like an ideal way to watch for users running Skype, this automatic check could be sent through an HTTP proxy or simply disabled in the client options. Additional IDS rules are required to reliably detect Skype on a network.

In addition to looking for the latest version of itself, Skype starts contacting super nodes as soon as it starts up. Like all of Skype's signaling traffic, this is encrypted, but there are some observable patterns.

Skype initiates contact with several super nodes by sending them UDP packets from the listening port specified by the Skype configuration. When Skype is first installed, it has no knowledge of super node status, and must resort to a list of "Bootstrap Nodes" that are distributed with the application (Baset & Schulzrinne, 2006). The initial packets sent to the super nodes usually have a relatively small payload, although the size is variable (between 18 and 36 bytes). What is noteworthy about these packets is that regardless of size, the third byte is always 02. This helps in detection, but since the source port is random and the payload size is variable, detection remains difficult.

When a super node is online and receives one of these packets, it will respond to the Skype client to its listening port with a similar UDP packet of variable payload size. The third byte of this payload will also always be 02. These packets tell the Skype client that the super node is either available or unavailable (if the port is closed, an ICMP unreachable packet will be sent, provided that the network allows it).

If a super node is available and willing to act as the super node for the client that sent it the initial UDP packet, it will reply with a UDP packet whose payload is exactly 18 bytes, and whose 3rd byte is always 02. Eureka! This would seem to be reliable way to detect super node activity on Skype. Unfortunately, this traffic is occasionally sent by the Skype client (whether or not it's acting as a super node) as well as the super node itself. Additionally, this traffic is seen frequently while Skype is running. Although an IDS rule to catch this activity may reliably indicate the presence of a Skype super node, it can be a very noisy rule and will alert often. Using an IPS to automatically block end points of this exchange seems unwise as the UDP traffic could easily be spoofed.

The first super node to reply to the Skype client with this 18-byte packet will become the chosen super node for this Skype session. The Skype client will immediately initiate a TCP connection to this super node; this connection will not be made from the dedicated listening port configured in Skype, it will be initiated from a random ephemeral port. This

activity is relatively easy to spot when looking at a packet capture after the fact, but is difficult to identify using automated methods such as an IDS because the source and destination ports are random and the payload is of varying size and is encrypted. The TCP session between the Skype client and its super node will last for the entirety of the session, being torn down only when the Skype client exits. If the Skype client cannot establish a TCP session with a super node, a login failure will be reported and the Skype client will not be usable (Baset & Schulzrinne, 2004).

After the Skype client has established a session with a super node, it will contact the login server. Part of the encrypted conversation between super node and client is the current IP address and listening port of the available login server. Communication with the login server is always made over TCP, but the destination port on the server is variable. There is, however, a common pattern in login traffic that can be used to reliably detect Skype. Furthermore, blocking remote hosts based on the presence of this common pattern will result in blocking Skype login activity rendering the Skype client unusable.

Once the TCP session is established with the login server, the first packet that the client sends to the login server has the PSH and ACK flags set and is 5 bytes long with the first 3 bytes equal to 16 03 01. The server will respond with an ACK packet, then it will send a TCP packet with a 5-byte payload and the PSH and ACK flags set. The first three bytes of

this payload will be 17 03 01. Blocking this traffic should effectively disable the Skype client.

A Snort rule to watch for this traffic from the client would be:

alert tcp $HOME_NET 1024: -> $EXTERNAL_NET 1024: (msg:"Skype client login -- from client"; flags:AP,SUFR12; flow:to_server,established; dsize:5; content:"|16 03 01|"; depth:3; flowbits:set,skype.login; sid:1000009; rev:2;)

And a corresponding rule to watch for the reply packet from the server:

alert tcp $EXTERNAL_NET 1024: -> $HOME_NET 1024: (msg:"Skype client login -- reply from server"; flags:AP,SUFR12; flow:to_client,established; dsize:5; content:"|17 03 01 00|"; depth:4; sid:1000010; rev:2;)

Since the server reply only happens after the client initiates login, it is possible to use the "flowbits" syntax of Snort to only alert on server replies to the client packet. This would avoid false positive alerts should some non-Skype system happen to send a 5 byte PSH-ACK packet with 17 03 01 as the first three bytes (this is unlikely but certainly possible). This event could then trigger an IPS response to block the remote host. Once the rules are in production, it's optional to alert on the client traffic. In the following, the rules listed above have been modified to use flowbits and to generate alerts only on login server replies and not

on the client login packet.

alert tcp $HOME_NET 1024: -> $EXTERNAL_NET 1024: (msg:"Skype client login -- from client"; flags:AP,SUFR12; flow:to_server,established; flowbits:set,skype_client_login,noalert; dsize:5; content:"|16 03 01|"; depth:3; sid:1000011; rev:2;)

alert tcp $EXTERNAL_NET 1024: -> $HOME_NET 1024: (msg:"Skype client login -- reply from server"; flags:AP,SUFR12; flow:to_client,established; flowbits:isset,skype_client_login; dsize:5; content:"|17 03 01|"; depth:3; sid:1000012; rev:2;)

This exchange of login packets is sometimes seen on 33033/TCP on the login server. The Skype client application has the ability to remember your username and password and to start automatically. Experimental observations indicate that the TCP/33033 login is limited to logins that are manual and not automatic. These logins can be captured with the following Snort rule:

alert tcp $HOME_NET 1024: -> $EXTERNAL_NET 33033 (msg:"Skype client manual login -- TCP/33033"; flow:to_server,established; flags:AP,SUFR12; content:"|17 03 01|"; depth:3; sid:1000400; rev:2;)

Since the Skype client application does not have the username and password stored in it when it is first installed, a manual login is required. Accordingly, the above rule can also be used to detect first-time Skype use.

After login occurs, super nodes are updated with the user's public-facing IP address, Skype listening port, and the user's presence (Online, Not Available, Do Not Disturb, etc.) In fact, a substantial number of packets are exchanged between the Skype client and its super node regardless of whether the user is engaging in a Skype conversation. Based on this activity, the Skype global index, the distributed "address book" that contains presence and location information for all Skype users is updated and now any user on the Skype network can initiate a conversation with this user.

When it comes time for a Skype conversation to occur, the caller will search the global index to find the callee's public IP address and Skype listening port. With this information, the caller can initiate a conversation.

If both Skype clients have publicly facing IP addresses, call initiation or "ringing" will occur directly between each client over TCP (Baset & Schulzrinne, 2006). Once a call is accepted, the conversation occurs over UDP directly to and from each client on their configured listening ports in true "peer to peer" fashion.

This may be the simplest scenario, but is unlikely to be the most common. Arguably, the most common modern scenario occurs when both clients are behind NAT firewalls or routers and do not have publicly facing IP addresses. Even when both firewalls are NAT firewalls and are configured to drop incoming TCP and UDP packets that do not have an existing, internally initiated session associated with them, Skype works flawlessly, and NAT-ed clients will exchange UDP conversation streams directly. How?

In this case, Skype uses a version of the STUN (Simple Tunneling of UDP over NATs) protocol (Schmidt, 2006). In a nutshell, it works like this: Skype user A, lets call him Alan, wants to contact Skype user B, lets call her Betty. Both Alan and Betty are behind NAT firewalls and are using RFC 1918 private IP space. Alan's IP is 192.168.1.1, and his Skype client is configured to listen on port 54321/UDP; Betty's IP is 10.0.0.2 and her client is configured to listen on port 12345/UDP.

Alan's Skype client already has an established TCP session with its super node; it tells the super node that Alan wants to talk to Betty. The super node already knows Alan's public IP address and listening Skype port; it communicates with other super nodes and finds Betty's public IP address, listening port, and Betty's associated super node. Alan's Skype client sends a call request message to Betty's super node asking it to forward to Betty's Skype client.

Betty's Skype client receives the message request from her associated super node and then sends a UDP packet to Alan:

10.0.0.2:12345/UDP -> 192.168.1.1:54321/UDP

Alan's firewall will drop this packet, assuming that it is not associated with a previous internally initiated UDP stream. But it ultimately doesn't matter, because now Betty's firewall thinks that there is an outbound, established UDP "session" to Alan.

Now when Alan's Skype client sends a UDP packet to Betty:

192.168.1.1:54321/UDP -> 10.0.0.2:12345/UDP

Betty's firewall thinks that this packet is already part of an internally initiated UDP session and lets it through. Now both Skype clients are able to exchange UDP traffic directly to and from each other and Skype conversations flow smoothly. This is all made possible by the extensive P2P network that Skype runs on, handling TCP signaling traffic and coordinating communications of otherwise unreachable hosts.

(Note: Strictly speaking, UDP is a stateless, connectionless protocol. It does not have any of the receipt verification overhead built into the TCP protocol, and does not care if packets are dropped or received out of order. However, most stateful NAT firewalls treat

UDP traffic as sessions and will allow incoming UDP packets as long as the source IP address and source port match those of a packet that was sent from an internal host before the session timeout expires on the firewall.)

In cases where one Skype user is behind a NAT firewall, but the other is on a public IP, it is easy to imagine how simple variations of the above behavior, e.g. direction of call initiation traffic, would yield similar results.

When both Skype users are behind firewalls that completely restrict the sending of outbound UDP packets, the Skype clients will send TCP packets a third host to relay their traffic. These hosts accept Skype communications on their designated listening port, but also on port 80 and 443. Since nearly all network environments allow outbound TCP traffic to these ports for web surfing, Skype is able to smoothly traverse restrictive firewalls.

5. Practical Advice and Real World Recommendations

To more securely use Skype requires at the minimum a few configuration changes from the default settings. Unfortunately, many of these settings can only be modified on Windows systems via the registry.

The most basic configuration change that can be made is to limit communications to

only those people in a user's contact list. This will stop unsolicited communication and will stop most spam and phishing attempts made over Skype. This configuration change can be made regardless of the platform on which Skype is running. It is highly recommended. Note that this change can be over-ridden when a user sets his or her presence to “SkypeMe”. In this mode, invitations to chat can be accepted from anyone on the Skype network.

Disable the Skype API -- If third party applications are not allowed to use Skype, then viruses and worms will be prevented from using it as a transmission and attack vector. This configuration change can be made in the Windows registry.

Disable File Transfer -- This setting makes sense for most managed deployments and will reduce the risk of data exfiltration. This setting can also be changed in the Windows registry. When file transfer is enabled, Skype users can transmit files of up to 2GB in size directly to and from each other.

Disable HTTP Ports -- This will stop Skype from listening on TCP ports 80 and 443 and will assist in keeping bandwidth consumption by your Skype client low. Note: when Skype is run by a user without administrative (root) privileges on OS X or Linux, the Skype client will not listen on these ports as non-root users cannot open listening ports below 1024. Not listening on these ports will make it less likely that your Skype client becomes a super node or

relay host.

Disable Super node -- If it's not possible to put the Skype client behind a NAT firewall, you can still stop Skype from becoming a Super node by making this registry change. Super nodes don't consume as much bandwidth as relay hosts, but they still handle a significant chunk of Skype's P2P signaling traffic.

Disable TCP Listen – This appears to be the relatively undocumented silver bullet that will without a doubt prevent your Skype client from becoming a bandwidth-devouring relay host. If the client is unable to accept incoming TCP sessions (those that are not associated with an outgoing TCP connection), then it will be unable to route other Skype users' traffic at all. It will still make outgoing TCP connections, and will still maintain a TCP session with its designated super node, but it will not route communications content for anyone other than the end-user sitting at that system. This is another change that is available only via the Windows registry. Linux and OS X users can use their operating system's firewall to achieve the same results by blocking inbound TCP connections to the listening port designated in Skype's configuration files.

The Skype client is more than just the user-facing GUI application. It is a P2P application that will continue to operate on the P2P network long after the end-user has

closed the application. In Windows, this is most evident by the Skype icon sitting in the Systray. Users should be educated so they know that unless they fully quit the application, Skype will continue to consume computing and network resources.

6. Conclusions

Skype stands alone among VOIP applications due to its Peer-To-Peer network architecture and its extensive use of strong encryption of not only communications content but signaling traffic as well. The application itself and its network communications are extremely resistant to reverse engineering, making Skype activity difficult to detect and its communications impossible to decipher. For networks that are subject to strict legal or administrative regulations, Skype should be banned to prevent unauthorized communications. For more open networks, Skype can be a boon for end-users who wish to communicate with colleagues on another side of our rapidly shrinking world. Since Skype is a communication application, users will always be subject to unsolicited messages and end-user education is recommended to ensure that bogus links sent by unknown correspondents don't result in system compromises.

7. Links and Additional Information

http://skype.com/security/

http://skype.com/business/security/

http://www.skype.com/security/Skype-v1.5.adm

http://www.ietf.org/rfc/rfc3489.txt

8. Appendix A: A Brief History of Skype's Security Vulnerabilities

SKYPE-SB/2008-003: Skype File URI Security Bypass Code Execution Vulnerability
2008-06-04 14:00:00 +0000
Affects Skype for Windows
No CVE reference

When a user is sent a URL via Skype, a handler checks the URL against a blacklist to make sure that it contains no known bad extensions (.exe, etc.) This handler can be bypassed as it was case sensitive, and it was not inclusive of all executable file types.

SKYPE-SB/2008-001: Skype Cross Zone Scripting Vulnerability
Initial report: 2008-01-23 9:00:00 +0000
Updated: 2008-01-31 11:00:00 +0000
Updated: 2008-02-05 14:00:00 +0000
Affects Skype for Windows
No CVE reference
Additional reference: http://seclists.org/fulldisclosure/2008/Jan/0328.html

Skype for Windows uses Internet Explorer to render HTML. Part of this functionality enables users to "add video to mood", allowing a short video to be played when other Skype users view a user's mood, an extension of their online presence. Videos for this feature are available at a number of third party content providers. If an attacker placed a malicious script

at the content provider's site via a code injection vulnerability at that site, and a user finds that malicious video at the site, the script could be run in the "local zone" of Internet Explorer allowing for execution of arbitrary code.

SKYPE-SB/2008-002: Skypefind Cross Zone Scripting Vulnerability
2008-01-31 11:00:00 +0000
Affects Skype for Windows
No CVE reference
Additional reference:
http://aviv.raffon.net/2008/01/17/SkypeCrosszoneScriptingVulnerability.aspx

SkypeFind is a feature whereby Skype users can recommend local businesses to other Skype users. If an attacker were to submit a specially crafted business contact and a user were to navigate directly to that contact, or if the user searches for a business and is presented with a specially crafted contact, the Skype client could execute arbitrary code without the user's consent. A mitigating factor to this vulnerability is that the user must receive a contact request authorization from the attacker's Skype account. This vulnerability is exploitable because of SKYPE-SB/2008-001 security zone elevation vulnerability, which allows sripts to be run in the local zone context of Internet Explorer.

SKYPE-SB/2007-003: Improper handling of URI arguments
2007-12-12 11:00:00 +0000
Affects Skype for Windows
CVE reference: CVE-2007-5989
Additional reference: http://www.zerodayinitiative.com/advisories/ZDI-07-070.html

If an attacker sends a Skype user a specially crafted URI, the users's Skype client can crash and allow for the execution of arbitrary code. In order for this vulnerability to be exploited, the user must visit a maliciously crafted web page.

SKYPE-SB/2007-001: Improper handling of URI arguments
2007-12-12 11:00:00 +0000
Affects Skype for Windows running on XP and Server 2003 using IE 7
CVE reference: CVE-2007-3896

If a user clicks on a maliciously crafted link in a chat or mood message, or visit a maliciously crafted web page, the Skype client can be made to execute arbitrary code. This vulnerability was introduced by a flaw in URL handling by Shell32.dll in Windows XP and Server 2003 with Internet Explorer 7 installed.

SKYPE-SB/2006-002: Improper handling of URI arguments
2006-10-03 15:00:00 +0000

Affects Skype for Mac
CVE reference: CVE-2006-5084
Additional reference: http://www.securityfocus.com/bid/20218

If an attacker sends a Skype user a specially crafted URI, and the user clicks on the link, the Skype client application may crash and possibly execute arbitrary code. This is due to a format string vulnerability in the URI handler whereby the format parameter can be specified by the URI string itself.

SKYPE-SB/2006-001: Improper handling of URI arguments
2006-05-19 08:00:00 +0000
Affects Skype for Windows
CVE reference: CVE-2006-2312

If an attacker sends a Skype user a specially crafted URL, and the user clicks on that link, the attacker can initiate the transfer of a single file from the user provided that the user has already authorized file transfers from the attacker. The file transfer would still generate an alert dialogue box allowing the user to cancel the transfer by clicking "cancel".

SKYPE-SB/2005-003: Heap overflow in networking routine
2005-10-25 08:00:00 +0000
Affects Skype client on *ALL* platforms
CVE reference: CVE-2005-3267
Additional reference: http://seclists.org/fulldisclosure/2005/Oct/0533.html

The Skype protocol was vulnerable to a heap overflow, which could be generated by a single remotely, generated UDP packet; this overflow could lead to the execution of arbitrary code.

9. Appendix B: Registry Entries and Configuration Settings for Skype Clients on Windows

The following is a list of configurable policies for the Windows Skype Client that can be managed via Group Policy Objects (Skype, 2008)

DisableFileTransferPolicy -- Disable file transfer to prevent the user from sending and receiving files using Skype

DisableContactImportPolicy -- Disable import contacts

DisablePersonalisePolicy -- Disable personalization to prevent the user from changing sounds

DisableLanguageEditPolicy -- Disable language edit to prevent the user from editing language strings

WebStatusPolicy -- When enabled, always publishes the user's status on the web as Skype buttons, when disabled, prevents the user from publishing status on the web

DisableApiPolicy -- Disable the Skype Public API to prevent third-party applications from accessing Skype functionality

DisableVersionCheckPolicy -- Disable new version checking by preventing Skype from detecting new versions and updates

MemoryOnlyPolicy -- Run in memory-only mode so Skype does not store any data on the local disk

ListePortPolicy -- Set the listening port where Skype listens for incoming connections
ListenPort -- Listening port number

ListenHTTPPortsPolicy -- When enabled, listen on HTTP (port 80) and HTTPS (port 443) ports; when disabled, don't listen on HTTP/HTTPS ports; when not configured, let

the user decide

DisableTCPListenPolicy -- Disable listening for TCP connections to prevent the Skype client from receiving incoming TCP connections

DisableUDPPolicy -- Disable UDP communications to prevent the Skype client from using UDP to communicate with the network

DisableSupernodePolicy -- Prevent the Skype client from becoming a super node or relay host

ProxyPolicy -- Establish the proxy policy

ProxyType -- Establish the proxy type

ProxyUnset -- Unset

ProxyAutomatic -- Automatic

ProxyDisabled -- Disabled

ProxyUnset -- Unset

ProxyHTTPS -- HTTPS

ProxySOCKS5 -- SOCKS5

ProxyAddress -- Proxy address (host:port)

ProxyUsername -- Username

ProxyPassword -- Password

The following is a list of configurable registry entries that apply to the Windows Skype Client as taken from the Skype Guide for Network Administrators (HKLM is short for HKEY_LOCAL_MACHINE) (Skype, 2008):

```
HKLM\Software\Policies\Skype\Phone, DisableApi, REG_DWORD = {0,1}
HKLM\Software\Policies\Skype\Phone, DisableFileTransfer, REG_DWORD = {0,1}
HKLM\Software\Policies\Skype\Phone, MemoryOnly, REG_DWORD = {0,1}
HKLM\Software\Policies\Skype\Phone, DisableContactImport, REG_DWORD = {0,1}
HKLM\Software\Policies\Skype\Phone, DisableVersionCheck, REG_DWORD = {0,1}
HKLM\Software\Policies\Skype\Phone, DisablePersonalise, REG_DWORD = {0,1}
HKLM\Software\Policies\Skype\Phone, DisableLanguageEdit, REG_DWORD = {0,1}
HKLM\Software\Policies\Skype\Phone, ListenPort, REG_DWORD = {0,1}
HKLM\Software\Policies\Skype\Phone, ListenHTTPPorts, REG_DWORD = {0,1}
HKLM\Software\Policies\Skype\Phone, DisableTCPListen, REG_DWORD = {0,1}
HKLM\Software\Policies\Skype\Phone, DisableUDP, REG_DWORD = {0,1}
HKLM\Software\Policies\Skype\Phone, DisableSupernode, REG_DWORD = {0,1}
HKLM\Software\Policies\Skype\Phone, ProxySettings, REG_SZ = {string}
HKLM\Software\Policies\Skype\Phone, ProxyAddress, REG_SZ = {string}
HKLM\Software\Policies\Skype\Phone, ProxyUsername, REG_SZ = {string}
HKLM\Software\Policies\Skype\Phone, ProxyPassword, REG_SZ = {string}
HKLM\Software\Policies\Skype\Phone, WebStatus, REG_DWORD = {0,1}
```

These same registry settings are available for the current user at HKEY_CURRENT_USER\Software\Policies\Skype\Phone but the HKEY_LOCAL_MACHINE

entries take precedence.

10. References

Baset, S. A., & Schulzrinne, H. (2008, September 15). An Analysis of the Skype Peer-to-Peer Internet Telephony . Retrieved June 13, 2008, from http://www1.cs.columbia.edu/~library/TR-repository/reports/reports-2004/cucs-039-04.pdf

Biondi, P., & Desclaux, F. (2006, March 2). *Silver Needle in the Skype*. Retrieved June 6, 2008, from http://www.secdev.org/conf/skype_BHEU06.handout.pdf

Saikat, G., Daswani, N., & Jain, R. (06, February). An Experimental Study of the Skype Peer-to-Peer VoIP System. Retrieved June 13, 2008, from http://www.guha.cc/saikat/pub/iptps06-skype.pdf

Schmidt, J. (2006, December 15). The hole trick - How Skype & Co. get round firewalls.

Retrieved August 16, 2008, from http://www.heise-online.co.uk/security/How-Skype-Co-get-round-firewalls--/features/82481/0

United States Patent Office. (07/12/07). System and method for detection of data traffic on a network (US 2007/0159979). Washington, DC: U.S. Government Printing Office.

Max, H., & Ray, T. (2006). Skype: The definitive guide. Indianapolis: Que Corporation.

Skype Network Administrator's Guide (2006, October 31). Retrieved June 13, 2008, from www.skypeclub.ru/files/ guide-for-network-admins-30beta.pdf

Network Admin Guide version 2.2 (2008, February 5). Retrieved June 13, 2008, from http://www.skype.com/security/network-admin-guide-version2.2.pdf

Berson, T. (2005, October 18). Skype Security Evaluation. Retrieved June 7, 2008, from http://www.skype.com/security/files/2005-031%20security%20evaluation.pdf

Last Updated: May 23rd, 2017

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Madrid 2017	Madrid, ES	May 29, 2017 - Jun 03, 2017	Live Event
SANS Atlanta 2017	Atlanta, GAUS	May 30, 2017 - Jun 04, 2017	Live Event
SANS San Francisco Summer 2017	San Francisco, CAUS	Jun 05, 2017 - Jun 10, 2017	Live Event
Security Operations Center Summit & Training	Washington, DCUS	Jun 05, 2017 - Jun 12, 2017	Live Event
SANS Houston 2017	Houston, TXUS	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS Thailand 2017	Bangkok, TH	Jun 12, 2017 - Jun 30, 2017	Live Event
SANS Milan 2017	Milan, IT	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Charlotte 2017	Charlotte, NCUS	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Secure Europe 2017	Amsterdam, NL	Jun 12, 2017 - Jun 20, 2017	Live Event
SEC555: SIEM-Tactical Analytics	San Diego, CAUS	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Rocky Mountain 2017	Denver, COUS	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Minneapolis 2017	Minneapolis, MNUS	Jun 19, 2017 - Jun 24, 2017	Live Event
DFIR Summit & Training 2017	Austin, TXUS	Jun 22, 2017 - Jun 29, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MDUS	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Cyber Defence Canberra 2017	Canberra, AU	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS Paris 2017	Paris, FR	Jun 26, 2017 - Jul 01, 2017	Live Event
SEC564:Red Team Ops	San Diego, CAUS	Jun 29, 2017 - Jun 30, 2017	Live Event
SANS London July 2017	London, GB	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, JP	Jul 05, 2017 - Jul 15, 2017	Live Event
SANS Los Angeles - Long Beach 2017	Long Beach, CAUS	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, SG	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS ICS & Energy-Houston 2017	Houston, TXUS	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Munich Summer 2017	Munich, DE	Jul 10, 2017 - Jul 15, 2017	Live Event
SANSFIRE 2017	Washington, DCUS	Jul 22, 2017 - Jul 29, 2017	Live Event
Security Awareness Summit & Training 2017	Nashville, TNUS	Jul 31, 2017 - Aug 09, 2017	Live Event
SANS San Antonio 2017	San Antonio, TXUS	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Prague 2017	Prague, CZ	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Hyderabad 2017	Hyderabad, IN	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Boston 2017	Boston, MAUS	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UTUS	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS New York City 2017	New York City, NYUS	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VAUS	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Stockholm 2017	OnlineSE	May 29, 2017 - Jun 03, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced