EFFECTIVE BLOCKING OF THE SKYPE PROTOCOL WITH CISCO IOS NATIVE FEATURES

Kartvelishvili Mikheil, Davitashvili Nicolas
Green Networks Ltd., Tbilisi, Georgia,
O. Kartvelishvili - GTU, Georgia

Abstract

The given work illustrates an effective method of blocking Skype protocol using a single Cisco IOS based router tested on different client platforms. This task becomes more and more relevant as popularity of Skype application increases throughout Internet and control of these types of applications in network with minimal hardware/software tools becomes a real issue.

Keywords: Skype. Protocol. Block. FPM, CBAC, IOS, Filtering.

1. INTRODUCTION

Skype has appeared in the Internet in mid-2003 and has rapidly gained popularity as a low-cost telephony and messaging application. It became widely deployed in SOHO, enterprise environments, and educational institutions.

As a result of growth of its popularity, a requirement to limit and control this protocol has unfolded in different network environments. This task appeared to be quite complex, due to Skype's proprietary nature and its use of advanced techniques for obfuscation and firewall traversal. Consequently Skype protocol analysis and blocking methodology has become the topic of huge number of discussions and publications in Internet throughout recent several years. [1][2][3][4][5]

The present work is an attempt to formulate a new, effective method of blocking current and former Skype protocol versions on different client platforms, based solely on native Cisco IOS features without any other additional intermediate network nodes.

2. TEST NETWORK TOPOLOGY

The testbed network topology is quite simple as shown in the Figure 1. It consists of a single Cisco 1841 router performing standard NAPT functionality with outside interface (FastEthernet0/1) connected to the Internet and inside interface (FastEthernet0/0) connected to LAN with a single client running the Skype application. The testing was performed on the following OSs:

- MS Windows XP SP3
- Ubuntu Linux 9.04
- Mac OS Leopard 10.5.8

The following Skype versions were tested:

- Skype for Windows v4.1
- Skype for Windows v3.8
- Skype for Windows v2.6
- Skype for Linux v2.0.0.72
- Skype for MacOS v2.8

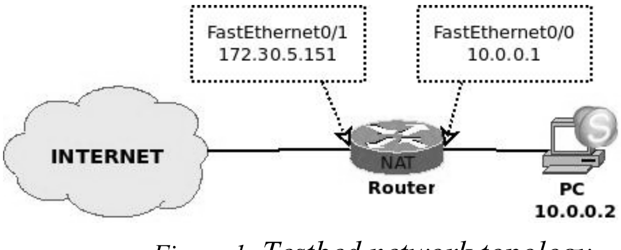


Figure 1. Testbed network topology

The minimum IOS version supporting all features mentioned in this document is 12.4(4)T.

3. SKYPE PROTOCOL BEHAVIOR AND PATTERN ANALYSIS

Blocking the Skype protocol can be broken up in 3 phases.

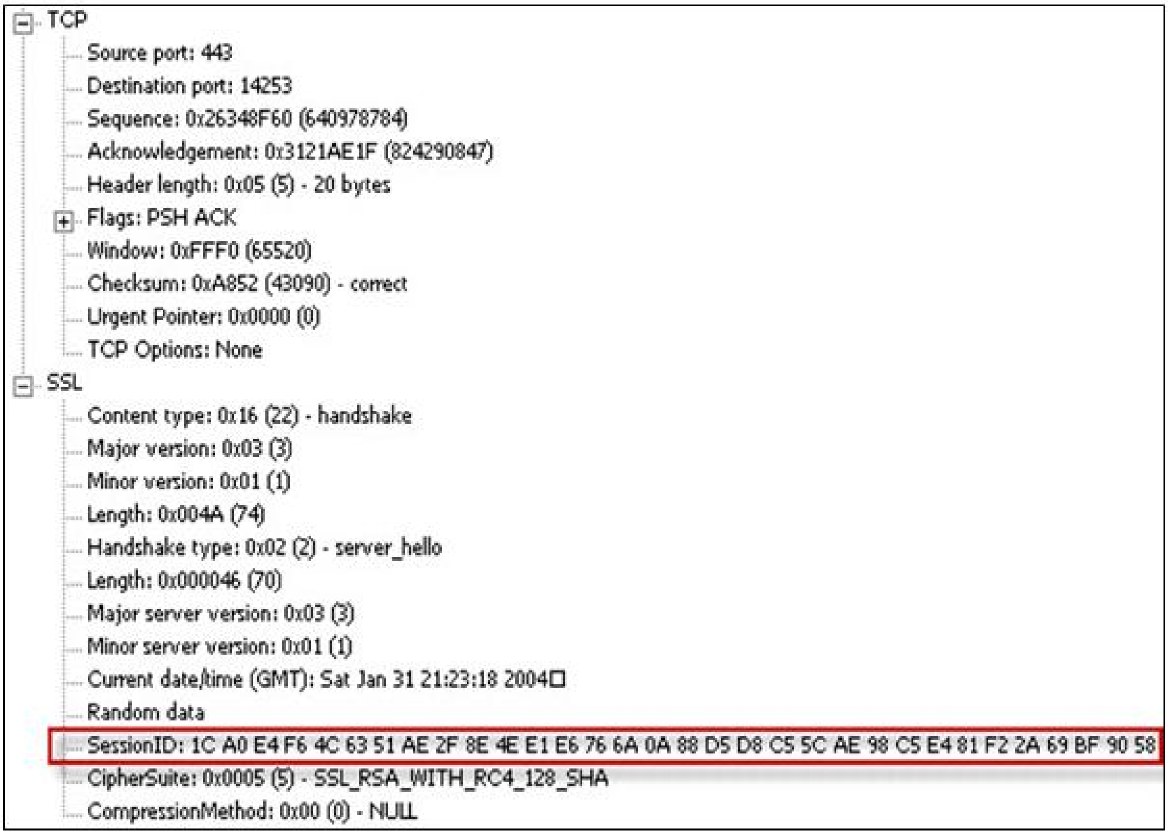
1. Blocking the major bulk of high numbered ports
2. Filtering HTTPS protocol
3. Filtering HTTP protocol

With no filters implemented in the path, Skype is using random TCP/UDP ports above 1024 to communicate.

After blocking all TCP ports except 443 (HTTPS) and 80 (HTTP), Skype negotiates using SSL over port 443 and that is when we are able to filter it using deep packet inspection.

It was observed, that, during SSL negotiation all versions of Skype on all operating systems we have tested are using the same Session ID (1C A0 E4 F6 4C….) in Server Hello packets, which we use to differentiate Skype SSL sessions from other HTTPS traffic.

In case of Windows OS the Session ID field is located in TCP segment's payload with an offset of 44 bytes. But the tricky part is that this is not always the case with other OS's. The sample Server Hello packet captured on the client node is shown in Figure 2.



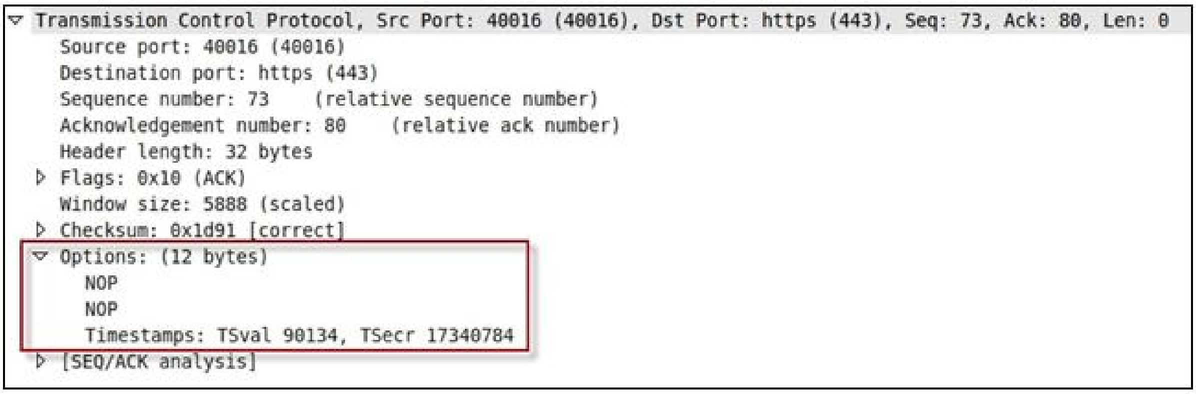
```
⊟ TCP
   Source port: 443
   Destination port: 14253
   Sequence: 0x26348F60 (640978784)
   Acknowledgement: 0x3121AE1F (824290847)
   Header length: 0x05 (5) - 20 bytes
 ⊞ Flags: PSH ACK
   Window: 0xFFF0 (65520)
   Checksum: 0xA852 (43090) - correct
   Urgent Pointer: 0x0000 (0)
   TCP Options: None
⊟ SSL
   Content type: 0x16 (22) - handshake
   Major version: 0x03 (3)
   Minor version: 0x01 (1)
   Length: 0x004A (74)
   Handshake type: 0x02 (2) - server_hello
   Length: 0x000046 (70)
   Major server version: 0x03 (3)
   Minor server version: 0x01 (1)
   Current date/time (GMT): Sat Jan 31 21:23:18 2004□
   Random data
   SessionID: 1C A0 E4 F6 4C 63 51 AE 2F 8E 4E E1 E6 76 6A 0A 88 D5 D8 C5 5C AE 98 C5 E4 81 F2 2A 69 BF 90 58
   CipherSuite: 0x0005 (5) - SSL_RSA_WITH_RC4_128_SHA
   CompressionMethod: 0x00 (0) - NULL
```

.

Figure 2. SessionID that can be used to capture and drop the Server Hello.

This packet is captured on Windows machine, thus TCP Options field is empty, which may not be the case for UNIX/Darwin-based systems.

In Linux and MacOS things get complicated as Skype may or may not inject some variable length options in TCP header (12 bytes in case of Linux). This makes it a bit harder to find Session ID inside the payload. This does not happen in Windows due to recent API limitations concerning raw socket creation, which are not an issue for Linux and MacOS systems. The Options field insertion into TCP header is illustrated in Figure 3.



```
▽ Transmission Control Protocol, Src Port: 40016 (40016), Dst Port: https (443), Seq: 73, Ack: 80, Len: 0
    Source port: 40016 (40016)
    Destination port: https (443)
    Sequence number: 73    (relative sequence number)
    Acknowledgement number: 80    (relative ack number)
    Header length: 32 bytes
  ▷ Flags: 0x10 (ACK)
    Window size: 5888 (scaled)
  ▷ Checksum: 0x1d91 [correct]
  ▽ Options: (12 bytes)
      NOP
      NOP
      Timestamps: TSval 90134, TSecr 17340784
  ▷ [SEQ/ACK analysis]
```

Figure 3. TCP Option field in non-Windows originated Server Hello message
This packet is captured on Linux machine, thus TCP Options field is present and is 12 bytes long

After blocking SSL Server Hellos with the mentioned Session ID in the incoming path, Skype tries to exchange data (probably keys for encryption) over HTTP (tcp/80), but it is not using standards based HTTP protocol syntax, so it can be filtered using strict HTTP inspection and dropping all non-conforming datagrams.

4. CONFIGURING CISCO IOS FOR SKYPE BLOCKING

Based on the previous discussion we decided to use Cisco Flexible Packet Matching (FPM) feature to detect and drop Session ID patterns in SSL packets and Application Firewall feature in CBAC for preventing consequent Skype data tunneling over HTTP.

First of all we blocked all the high port traffic allowing a range of well-known ports (1-1023) through the router.

```
ip access-list extended block
 permit udp any any range 1 1023
 permit tcp any any range 1 1023
 permit icmp any any
!interface FastEthernet0/0
 ip access-group block in
```

This step has forced Skype to try to use its HTTP/HTTPS protocol tunneling feature and made it vulnerable to detection.

We continued with configuring FPM feature. The first step is to load PHDF files for IP and TCP protocols into the router. These files (ip.phdf and tcp.phdf in this case) can be obtained from http://www.cisco.com/cgi-bin/tablebuild.pl/fpm[1]. These files are to be uploaded to router’s flash memory (usually by means of TFTP protocol) and made available to IOS using commands:

[1] Valid CCO account is required.

```
load protocol flash:ip.phdf

load protocol flash:tcp.phdf
```

Definitions were created based on the analysis performed in the previous section.

```
class-map type stack match-all ip_tcp
 match field IP protocol eq 6 next TCP
class-map type access-control match-any skype
 match start TCP payload-start offset 44 size 4  eq 0x1CA0E4F6
 match start TCP payload-start offset 48 size 4 eq 0x1CA0E4F6
 match start TCP payload-start offset 52 size 4 eq 0x1CA0E4F6
 match start TCP payload-start offset 56 size 4 eq 0x1CA0E4F6
 match start TCP payload-start offset 60 size 4 eq 0x1CA0E4F6
!
!
policy-map type access-control skype-policy
 class skype
   drop
policy-map type access-control fpm-policy
 class ip_tcp
  service-policy skype-policy
```

As Cisco IOS FPM feature does not support variable length fields such as TCP Options and assumes TCP header to be strictly 20 bytes long, we are trying to catch the Session ID signature sequentially on several offsets with 4 byte steps. 4-byte steps are derived from the fact that actual TCP header is always aligned to 32-bit boundary and as its fixed part is 20 bytes long, the TCP options should also conform to this alignment. This makes the range of possible offsets quite limited. We decided to include first 5 possible offsets into the sample configuration (this assumption appeared to be sufficient for all the tested platforms and software versions).

```
interface FastEthernet0/0
   service-policy type access-control input
    fpm-policy
```

This newly created policy-map was applied to the outside interface in ingress direction, since Server Hello messages are always inbound to the client.

The next task is to prevent Skype from exchanging data over tcp/80 port. This is achieved using CBAC Application Firewall feature, which can easily identify traffic flow not matching HTTP specification and terminate it.

```
ip inspect name NO_SKYPE appfw WEB
ip inspect name NO_SKYPE http
!
appfw policy-name WEB
  application http
    strict-http action reset alarm
```

This configuration was applied to the ingress traffic on the inside interface.

```
interface FastEthernet0/0
```

```
 ip inspect NO_SKYPE in
```

Although this configuration prevents Skype client from logging in to the server, it does not break any existing Skype sessions.

5. CONCLUSION

The present work showed that blocking Skype protocol is not as difficult as it may seem without appropriate analysis. Although Cisco IOS was used to block Skype in this illustration, this method is also applicable to any type of firewall supporting payload pattern matching and HTTP traffic inspection features. Even though the presented test was successful, upcoming Skype versions may change the protocol behavior and patterns significantly, making the method described here completely or partially ineffective. In addition several other patterns were identified during SSL exchange, but we decided not to use them due to higher probability of false positives.

REFERENCES:

1. G Salman A. Baset and Henning Schulzrinne, “An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol”, Sept. 2004.
2. Dario Bonfiglio et al. “Revealing Skype Traffic: When Randomness Plays with You,” ACM SIGCOMM Computer Communication Review, Volume 37:4 (SIGCOMM 2007), p. 37-48
3. P. Biondi and F. Desclaux, "Silver Needle in the Skype", BlackHat Europe, Mar. 2006
4. Fabrice Desclaux, Kostya Kortchinsky. "Vanilla Skype part 1". RECON2006, Jun. 2006.
5. Fabrice Desclaux, Kostya Kortchinsky. "Vanilla Skype part 2". RECON2006, Jun. 2006.
6. “Flexible Packet Matching”, Data Sheet, Cisco Systems, Jul. 2007

SKYPE ᲞᲠᲝᲢᲝᲙᲝᲚᲘᲡ ᲔᲤᲔᲥᲢᲣᲠᲘ ᲑᲚᲝᲙᲘᲠᲔᲑᲐ CISCO IOS-ᲘᲡ ᲩᲐᲨᲔᲜᲔᲑᲣᲚᲘ ᲡᲐᲨᲣᲐᲚᲔᲑᲘᲗ

მიხეილ ქართველიშვილი, ნიკოლოზ დავითაშვილი (Green Networks Ltd.,თბილისი)
ოთარ ქართველიშვილი – საქართველოს ტექნიკური უნივერსიტეტი

რეზიუმე

მოცემულ ნაშრომში ნაჩვენებია Skype პროტოკოლის ბლოკირების ეფექტური მეთოდი Cisco IOS ოპერაციულ სისტემაზე ბაზირებული მარშრუტიზატორის გამოყენებით და განხორციელებულია მისი ტესტირება სხვადასხვა კლიენტურ პლატფორმებზე.

ЭФФЕКТИВНОЕ БЛОКИРОВАНИЕ ПРОТОКОЛА SKYPE СРЕДСТВАМИ ВСТРОЕННОЙ CISCO IOS

Картвелишвили М., Давиташвили Н. - Green Networks Ltd. Тбилиси
Картвелишвили О. - Грузинский Технический Университет

Резюме

В представленной работе показан эффективный метод блокирования протокола Skype с использованием маршрутизатора, базированном на операционной системе Cisco IOS, и выполнено его тестирование на различных клиентских платформах.