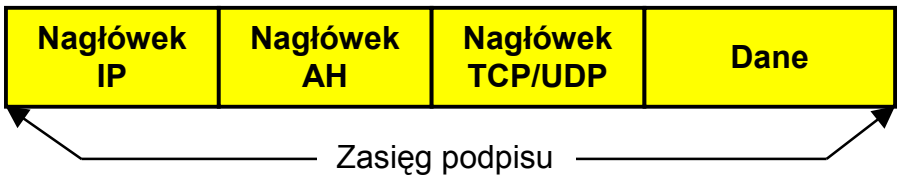
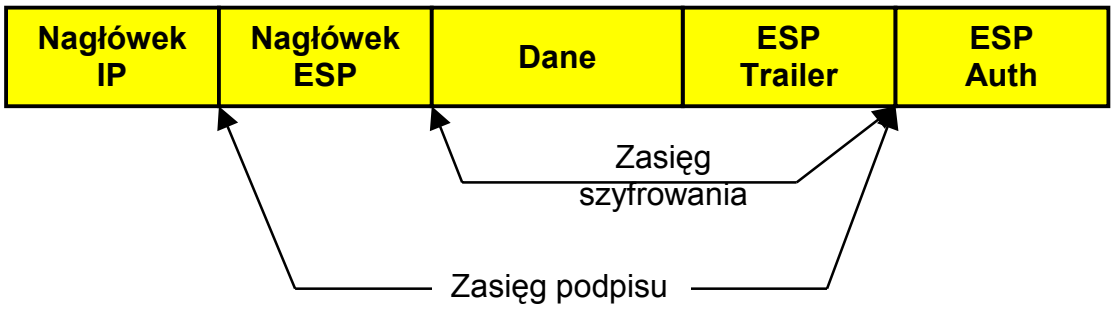
Nagłówki IPSec w trybie transportowym

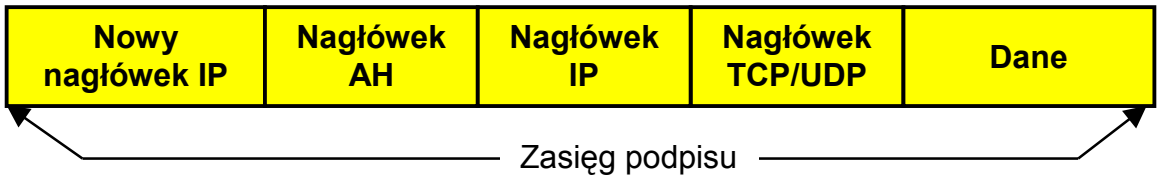


Nagłówek AH: podpisywany jest cały pakiet łącznie z nagłówkiem IP

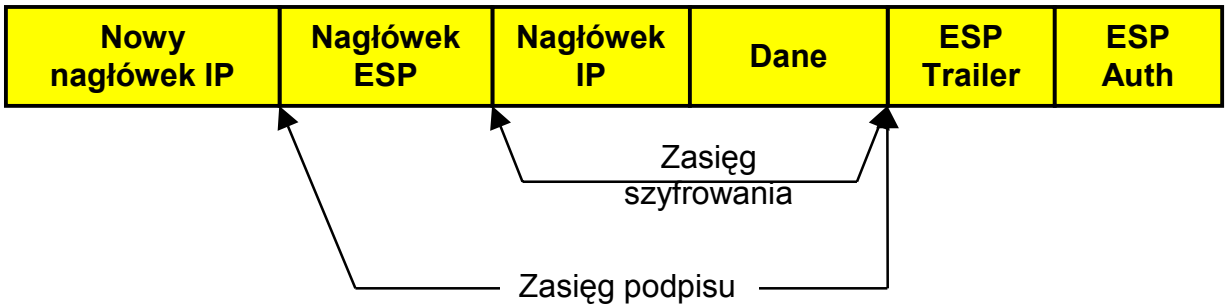


Nagłówek ESP: szyfrowane są tylko dane

Nagłówki IPSec w trybie tunelowym



Nagłówek AH: podpisywany jest cały pakiet łącznie z nowym nagłówkiem IP



Nagłówek ESP: szyfrowane są dane wraz z oryginalnym nagłówkiem IP
Cały oryginalny pakiet jest traktowany jako dane nowego pakietu IP

Protokoł SSL (*Secure Socket Layer*)

SSL *Record Protocol*

Skrót wiadomości (MAC-DATA),
Dane do przesłania (ACTUAL-DATA),
Dane wypełniające (PADDING-DATA).

SSL *Handshake Protocol*

CLIENT		SERVER
CLIENTHELLO	-------->	
		SERVERHELLO
		CERTIFICATE*
		SERVERKEYEXCHANGE*
		CERTIFICATEREQUEST*
	<--------	SERVERHELLODONE
CERTIFICATE*		
CLIENTKEYEXCHANGE		
CERTIFICATEVERIFY*		
[CHANGECIPHERSPEC]		
FINISHED	-------->	
		[CHANGECIPHERSPEC]
	<--------	FINISHED
APPLICATION DATA	<------->	APPLICATION DATA

Biblioteki: SSLRef, Open SSL

Protokół HTTP

Request = Request-Line
***((general-header**
| request-header
| entity-header) CRLF)
CRLF
[message-body]

Response = Status-
***((general-header**
| response-header
| entity-header) CRLF)
CRLF
[message-body]

Request-Line = Method SP Request-URI SP HTTP-Version CRLF
Np.: **GET http://www.w3.org/pub/WWW/TheProject.html HTTP/1.1**

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF
Np.: **HTTP/1.1 200 OK**

Przykłady kodów:
Status-Code =
"100" ; Continue
| "200" ; OK
| "202" ; Accepted
| "400" ; Bad Request
| "403" ; Forbidden
| "404" ; Not Found

Protokół S-SHTTP
(*Secure HyperText Transfer Protocol*)

Linia żądania:

Secure * Secure-HTTP/1.4

Odpowiedź serwera:

Secure-HTTP/1.4 200 OK

Nagłówki

SHTTP-Privacy-Domain
określa standard zapisu zabezpieczanej wiadomości

SHTTP-Certificate-Type
określa akceptowane formaty certyfikatów

SHTTP-Key-Exchange-Algorithm
określa algorytm używamy do wymiany kluczy

SHTTP-Signature-Algorithms
określa algorytm podpisu cyfrowego

SHTTP-Message-Digest-Algorithms
określa algorytm zapewnienia integralności danych – funkcja skrótu

SHTTP-Symmetric-Content-Algorithms
określa algorytm symetrycznego szyfru blokowego,
który zostanie wykorzystany do szyfrowania danych

SHTTP-Symmetric-Header-Algorithms
określa symetryczny algorytm szyfrowania,
który zostanie wykorzystany do szyfrowania nagłówków

SHTTP-Privacy-Enhancements
określa zabezpieczenia związane z wiadomością

Protokół SSH

- ***Transport layer protocol*** [SSH-TRANS]
- ***User authentication protocol*** [SSH-USERAUTH]
- ***Connection protocol*** [SSH-CONN]

Struktura pakietu SSH-TRANS

uint32	**packet_length**
byte	**padding_length**
byte[n1]	**payload;** n1 = packet_length - padding_length - 1
byte[n2]	**random padding;** n2 = padding_length
byte[m]	**mac (message authentication code);** m = mac_length

Struktura pakietu inicjującego negocjacje (SSH-TRANS)

byte	**SSH_MSG_KEXINIT**
byte[16]	**cookie (**random bytes**)**
string	**kex_algorithms**
string	**server_host_key_algorithms**
string	**encryption_algorithms_client_to_server**
string	**encryption_algorithms_server_to_client**
string	**mac_algorithms_client_to_server**
string	**mac_algorithms_server_to_client**
string	**compression_algorithms_client_to_server**
string	**compression_algorithms_server_to_client**
string	**languages_client_to_server**
string	**languages_server_to_client**
boolean	**first_kex_packet_follows**
uint32	**0 (**reserved for future extension**)**

User authentication protocol [SSH-USERAUTH]

Żądanie autentykacji:

byte	SSH_MSG_USERAUTH_REQUEST
string	user name (in ISO-10646 UTF-8 encoding)
string	service name (in US-ASCII)
string	authentication method name (US-ASCII)

reszta pakietu zależy od zaproponowanej metody autentykacji

Odpowiedź na żądanie autentykacji:

negatywna:

byte	SSH_MSG_USERAUTH_FAILURE
string	authentications that can continue
boolean	partial success

pozytywna:

byte	SSH_MSG_USERAUTH_SUCCESS

Banner dla klienta:

byte	SSH_MSG_USERAUTH_BANNER
string	message (ISO-10646 UTF-8)
string	language tag (as defined in RFC 1766)

User authentication protocol [SSH-USERAUTH]

Metoda *publickey*

byte	SSH_MSG_USERAUTH_REQUEST
string	user name
string	service
string	"publickey"
boolean	FALSE
string	public key algorithm name
string	public key blob – tutaj może być certyfikat
string	signature

Pozytywna odpowiedź serwera:

byte	SSH_MSG_USERAUTH_PK_OK
string	public key algorithm name from the request
string	public key blob from the request

Metoda *hostbased*

byte	SSH_MSG_USERAUTH_REQUEST
string	user name
string	service
string	"hostbased"
string	public key algorithm for host key
string	public host key and certificates for client host
string	client host name (FQDN; US-ASCII)
string	client user name on the remote host (ISO-10646 UTF-8)
string	signature

User authentication protocol [SSH-USERAUTH]

Metoda *password*

byte	SSH_MSG_USERAUTH_REQUEST
string	user name
string	service
string	"password"
boolean	FALSE
string	plaintext password (ISO-10646 UTF-8)

żądanie zmiany hasła:

byte	SSH_MSG_USERAUTH_PASSWD_CHANGEREQ
string	prompt (ISO-10646 UTF-8)
string	language tag (as defined in RFC 1766)

komunikat z nowym hasłem:

byte	SSH_MSG_USERAUTH_REQUEST
string	user name
string	service
string	"password"
boolean	TRUE
string	plaintext old password (ISO-10646 UTF-8)
string	plaintext new password (ISO-10646 UTF-8)

Connection protocol [SSH-CONN]

Żądanie otwarcia kanału:

byte	SSH_MSG_CHANNEL_OPEN
string	channel type (restricted to US-ASCII)
uint32	sender channel identifier
uint32	initial window size
uint32	maximum packet size
channel type specific data follows	

Odpowiedź:

pozytywna

byte	SSH_MSG_CHANNEL_OPEN_CONFIRMATION
uint32	recipient channel identifier
uint32	sender channel from request
uint32	initial window size
uint32	maximum packet size
channel type specific data follows	

negatywna:

byte	SSH_MSG_CHANNEL_OPEN_FAILURE
uint32	recipient channel
uint32	reason code
string	additional textual information (ISO-10646 UTF-8[RFC-2044])
string	language tag (as defined in [RFC-1766])

Connection protocol [SSH-CONN]

Zmiana rozmiaru okna:

```
byte      SSH_MSG_CHANNEL_WINDOW_ADJUST
uint32    recipient channel
uint32    bytes to add
```

Przesyłanie danych:

```
byte          SSH_MSG_CHANNEL_DATA
uint32    recipient channel
string    data
```

Zamknięcie kanału:

```
byte      SSH_MSG_CHANNEL_EOF
uint32    recipient_channel
```

i

```
byte      SSH_MSG_CHANNEL_CLOSE
uint32    recipient_channel
```

Literatura:

1) V. Ahuja. *Network & Internet Security*. Academic Press, Inc, 1996. (*tłum.*)
2) D. Atkins. *Internet Security. Professional Reference.* New Riders Publishing, 1997. (*tłum. LT&P*).
3) R. Atkinson. *Security Architecture for Internet Protocol*, RFC 1825, Aug 1995.
4) R. Atkinson. *IP Authentication Header.* RFC 1826, Aug 1995.
5) R. Atkinson. *IP Encapsulation Security Payload (ESP),* RFC 1827, Aug 1995.
6) S. Garfinkel, G. Spafford. *Practical Unix and Internet Security,* O'Reilly&Associates Inc. 1996. (*tłum.*)
7) K.E.B. Hickman, T.Elgamal. *The SSL Protocol.* Internet Draft, 1995.
8) P.Karn, P.Metzger, W.Simpson. *The ESP DES-CBC Transform,* RFC 1829, Aug. 1995.
9) L. Klander. *Hacker Proof.* Jamsa Press, 1997. (*tłum.*)
10) P. Metzger, W. Simpson. *IP Authentication using Keyed MD5,* RFC 1828, Aug. 1995.
11) E.Rescola, A.Schiffman. *The Secure HyperText Transfer Protocol*, Internet Draft, Jul 1995.
12) P. Wayner. *Digital Cash: Commerce on the Net.* AP Professional, 1996
13) T. Ylonen, i inni. *SSH Protocol Architecture.* Network Working Group, Internet Draft, Feb. 1999.
14) T. Ylonen, i inni *SSH Transport Layer Protocol*, Internet Draft, Feb. 1999.
15) T. Ylonen, i inni. *SSH Authentication Protocol*, Internet Draft, Feb. 1999.
16) T. Ylonen, i inni. *SSH Connection Protocol*, Internet Draft, Feb. 1999.

Dodatkowe źródła:

1. Z. Kozak, *Wirtualne sieci prywatne,* Software 2.0, nr 11/2000.
2. M. Szymański, *IPSec i Linux,* Linux Plus, nr 9/2000.
3. P. Wojakowski, R. Kuliga, *Rozwiązania IPSec i VPN*. Software 2.0, nr 09/2000.