SecureWorks

The Information Security Experts

Loaded Dice: SSH Key Exchange & the Debian OpenSSL PRNG Vulnerability

Ben Feinstein, CISSP GCFA
SecureWorks Counter Threat Unit℠

ToorCon X
September 27, 2008

The Information Security Experts

What's In This Talk?

- Basics of Key Exchange
- Intro to Ephemeral Diffie-Hellman
- Intro to Debian OpenSSL PRNG Vulnerability
  - CVE-2008-0166
- SSH2 Diffie-Hellman Group Key Exchange (KEXDH GEX)
- Live Demo – Brute Force a "Weak" SSH Session

What's NOT In This Talk?

- An efficient algorithm for solving the Discrete Logarithm problem!
  - (You really think I'd drop it here?!?)

The Basics of Key Exchange

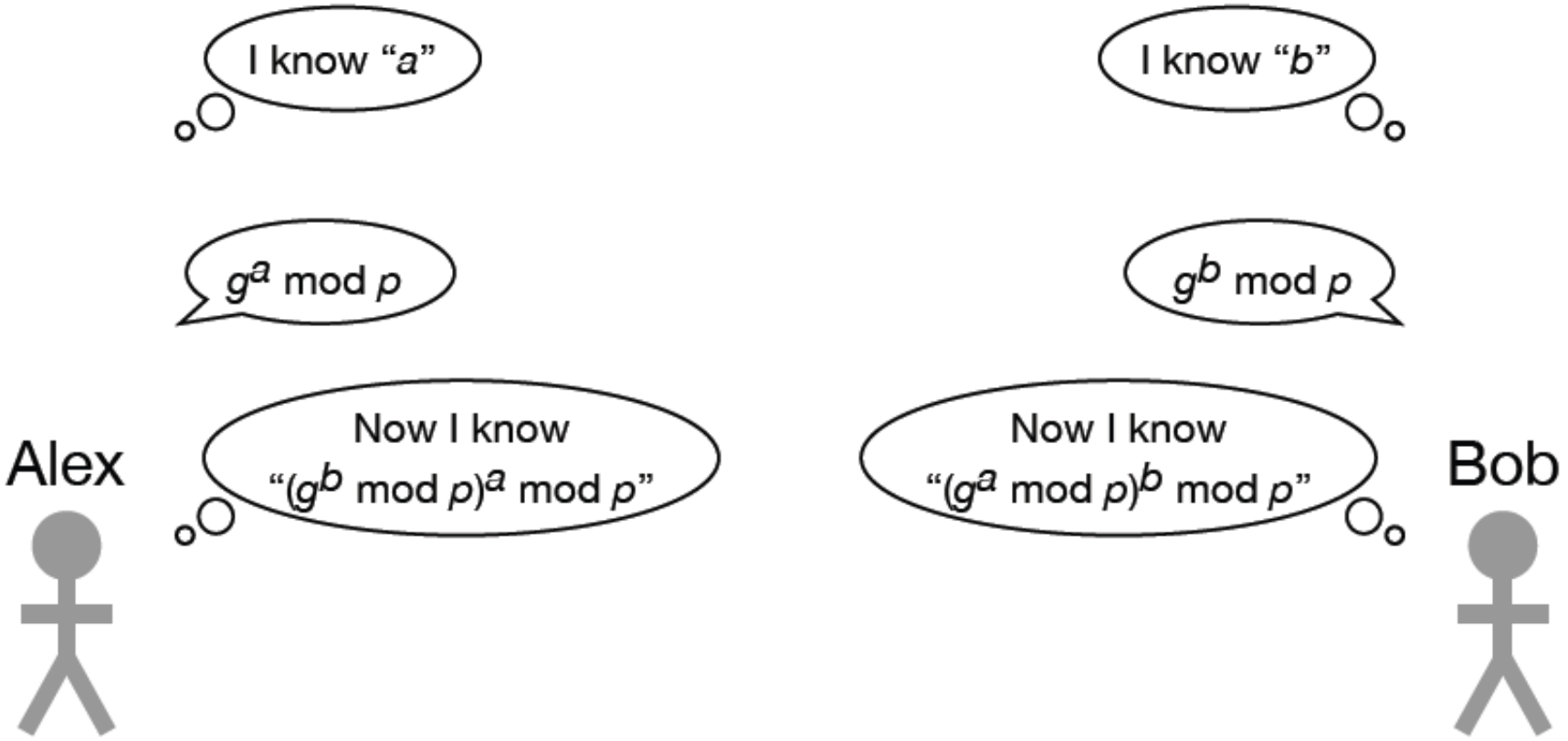
- Symmetric Key Crypto
  - Shared Secret / Pre-Shared Key (PSK)
- Public-Key Crypto
  - RSA
  - DSA
- Key Management / Key Exchange
  - Common point of attack on cryptosystems
  - Often times the point of least resistance

Brief Intro to Ephemeral Diffie-Hellman KEX

- Invented by Whitfield Diffie & Martin Hellman in 1976
  - 1st practical alg. for agreeing on shared secret over insecure channel w/o any prior knowledge
  - Based on Ralph Merkle's work on public key distribution

Merkle, Hellman, Diffie (L to R)

$$(g^b \bmod p)^a \bmod p = (g^a \bmod p)^b \bmod p$$



Source: Stephen Cristol, DC404

Debian OpenSSL Predictable PRNG Vuln

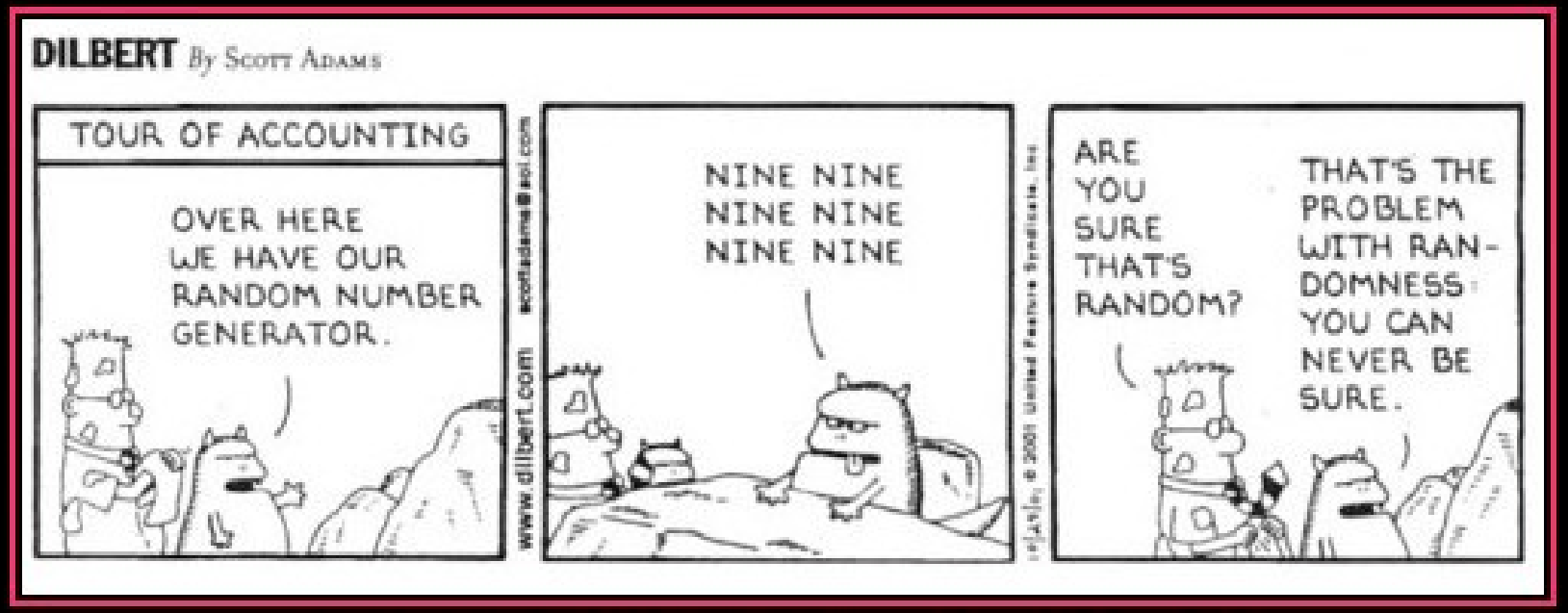
CVE-2008-0166

- Lack of sufficient entropy in PRNG delivered by Debian's OpenSSL package
- Discovered by Luciano Bello
  - Great talk at DEFCON 16 w/ Maximiliano Bertacchini
- One of the coolest vulns of 2008!
  - Pwnie for Most Epic FAIL!

- Keys generated since 2006-09-17
- Keys generated with Debian Etch, Lenny or Sid
  - Downstream distros such as Ubuntu also vulnerable

Debian OpenSSL Predictable PRNG Vuln

Dilbert (source: H D Moore, metasploit.com)

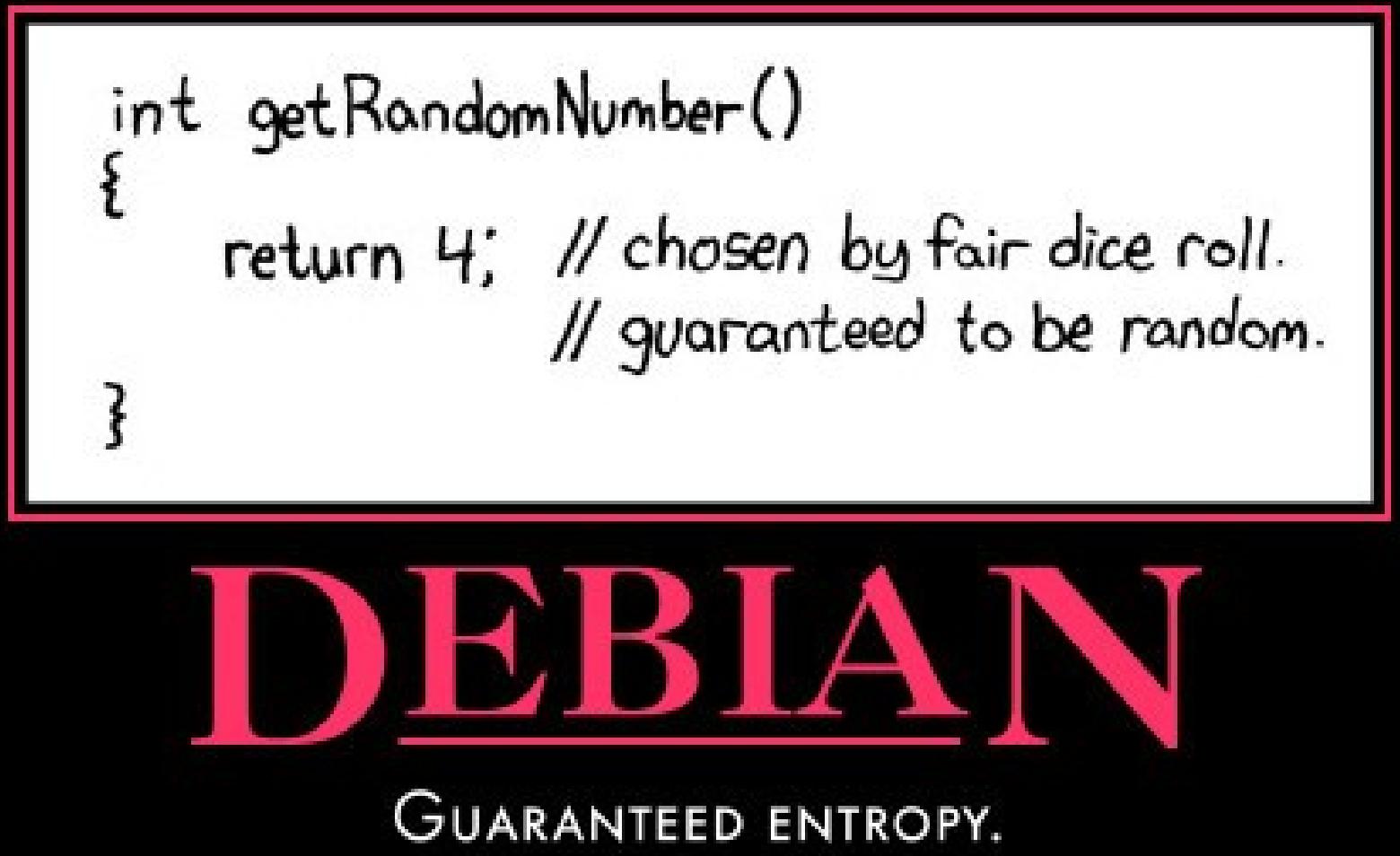


DEBIAN

YOU CAN NEVER BE SURE.

Debian OpenSSL Predictable PRNG Vuln

XKCD (source: H D Moore, metasploit.com)



Debian OpenSSL Predictable PRNG Vuln

It's Bad!

- From the Debian Wiki (http://wiki.debian.org/SSLkeys):

- "… any DSA key must be considered compromised if it has been used on a machine with a 'bad' OpenSSL. Simply using a 'strong' DSA key (i.e., generated with a 'good' OpenSSL) to make a connection from such a machine may have compromised it. This is due to an 'attack' on DSA that allows the secret key to be found it the nonce used in the signature is known or reused."

- H D Moore was all over this one with a quickness!
  - Metasploit hosting lists of brute-forced 'weak' keys

Debian OpenSSL Predictable PRNG Vuln

Detection & Mitigation

- You scanned your assets for SSH / SSL servers using the blacklisted keys, right? (Tenable Nessus, others?)
- You scanned all user home dirs for blacklisted SSH keys?
  - Debian ssh-vulnkey tool
  - Why isn't this tool being shipped w/ other distros???
- You scanned all user homedirs, Windows Protected Storage, and browser profiles for blacklisted SSL certs, right?
- But what about connections to external servers that use the vulnerable Debian OpenSSL?

Snort Dynamic Preprocessor

SSH Diffie-Hellman Group Key Exchange

- Initial Goal: Detect SSH Diffie-Hellman Key Exchange (KEX) where client and/or server are OpenSSH linked against vulnerable Debian OpenSSL
- Just that detective capability is valuable. Why?
- Even w/ great technical controls in place, you're likely missing:
  - Users connecting to external servers using bad OpenSSL
  - Connections to/from external hosts that use bad OpenSSL

Snort Dynamic Preprocessor

SSH Diffie-Hellman Group Key Exchange (2)

- Initial code released at DEFCON 16
- Currently released version supports detection of both vulnerable client and server
  - http://www.secureworks.com/research/tools/snort-plugir
- Version w/ more capabilities will be released soon
- Current limitations
  - Performance – I have some ideas on this
  - Requires file of pre-calculated "weak" random values

Snort Dynamic Preprocessor

Coming Enhancements

- Have preprocessor(s) "normalize" traffic by brute-forcing the DH key exchange, decoding both sides of session on-the-fly.
  - Snort rule matching engine and other preprocessors can then inspect unencrypted session
  - Unencrypted sessions can be logged (Unified or PCAP)
- Better performance
  - Use Raphaël Rigo's approach w/ special shared libs?

Giving Credit Where Credit Is Due!

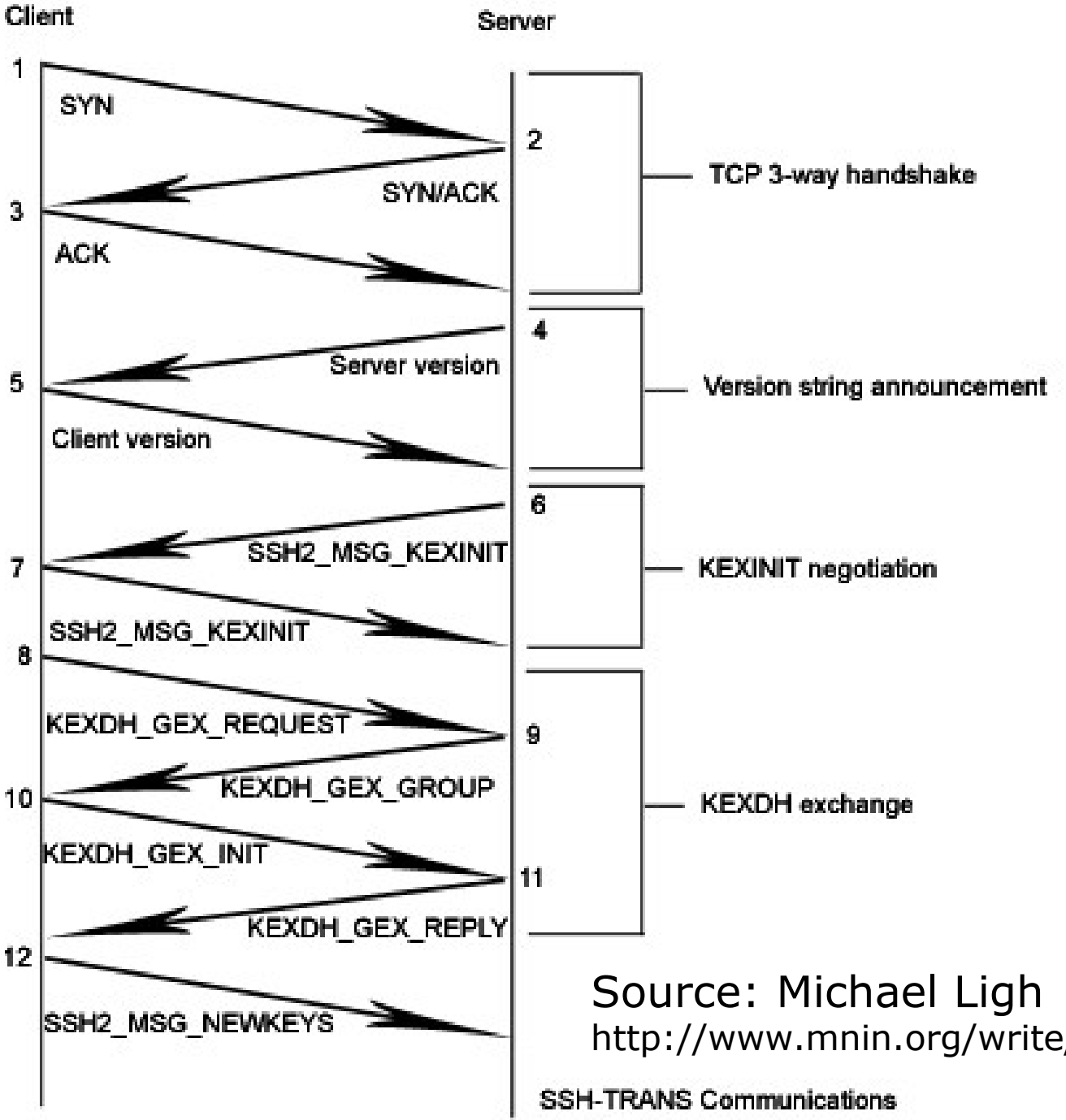
- Raphaël Rigo & Yoann Guillot
  - New work on SSH and Debian OpenSSL PRNG Vuln
  - Unknown to me until hearing about it at DEFCON
  - http://www.cr0.org/progs/sshfun/
- Alexander Klink
  - http://seclists.org/fulldisclosure/2008/May/0592.html
  - http://www.cynops.de/download/check_weak_dh_ssh.pl.bz2
- Paolo Abeni, Luciano Bello & Maximiliano Bertacchini
  - Wireshark patch to break PFS in SSL/TLS
  - https://bugs.wireshark.org/bugzilla/show_bug.cgi?id=2725

Diffie-Hellman Key Exchange for SSH

The Specifications

- RFC 4253 – The Secure Shell (SSH) Transport Layer Protocol
  - Defines commonly used key exchange groups
    - diffie-hellman-group1-sha1
    - diffie-hellman-group14-sha1

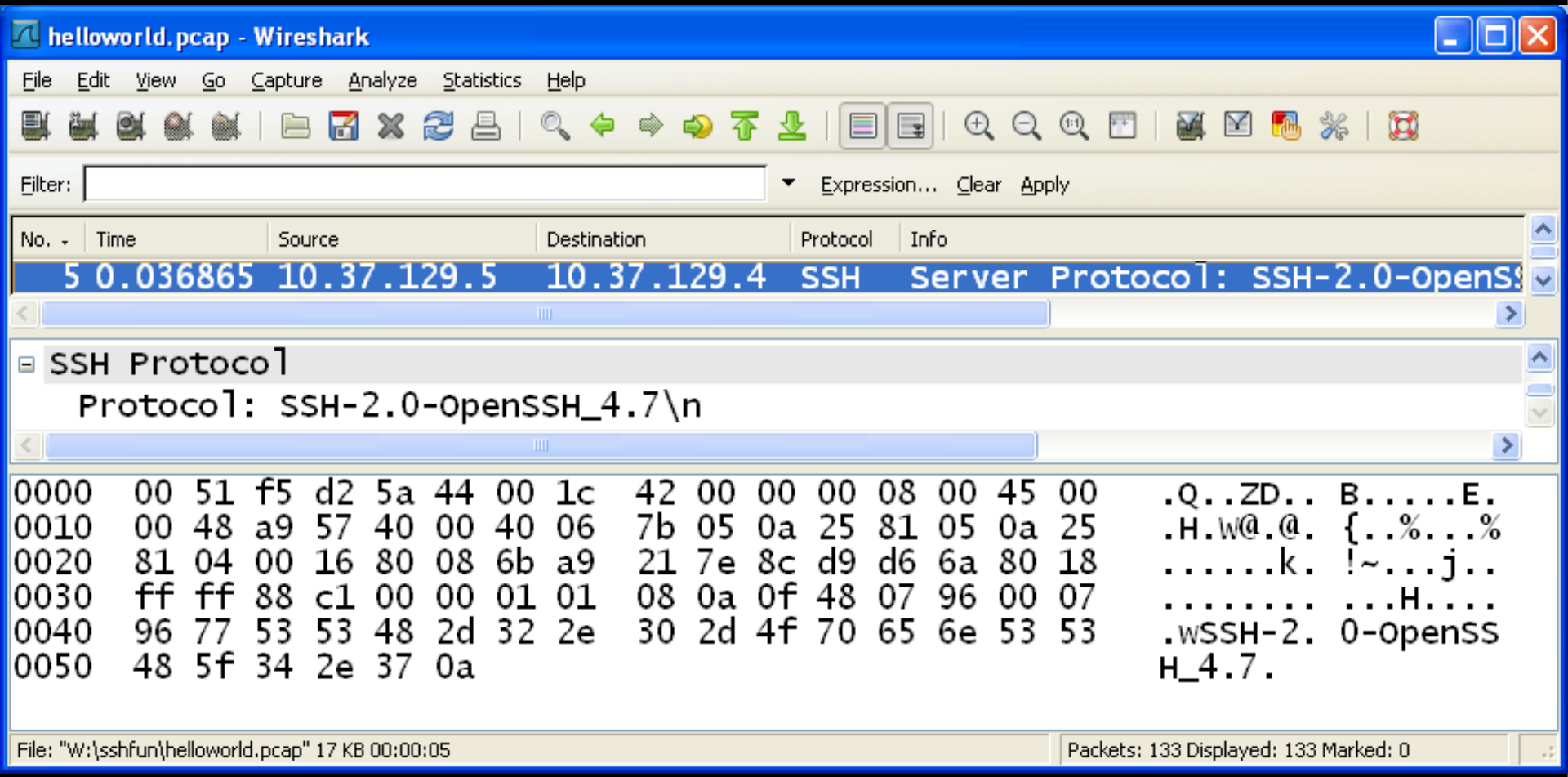
- RFC 4419 – Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol
  - Defines the guts of how SSH performs DH GEX
  - SSH payload formats



Source: Michael Ligh
http://www.mnin.org/write/2006_sshcrypto.html

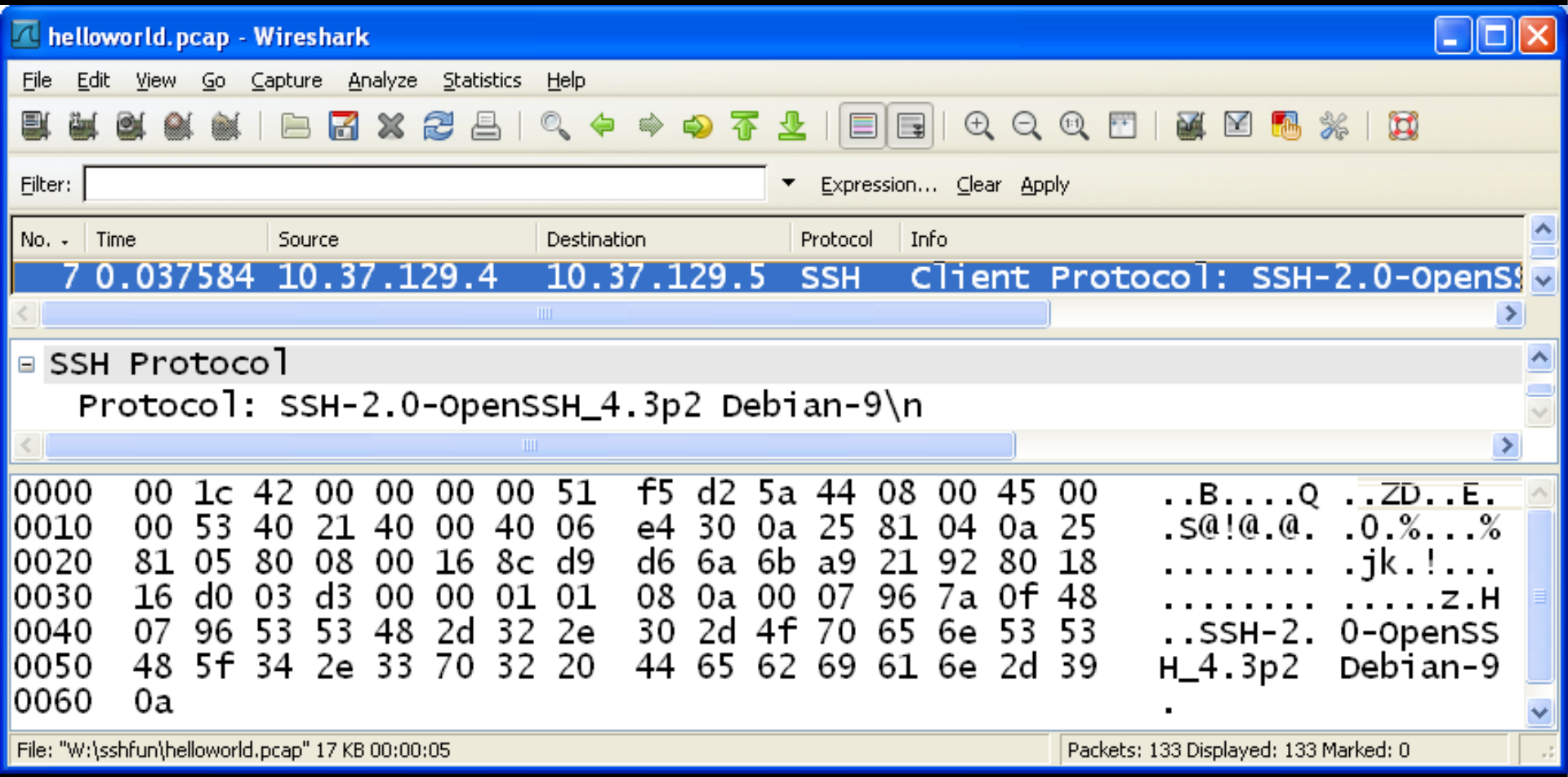
Breakout of Server Version String Announcement

- Sent from Server to Client
- Payload must start w/ 4-bytes "SSH-"
- SSH server version string
- e.g. "SSH-2.0-OpenSSH_4.7"


helloworld.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Info
5 0.036865 10.37.129.5 10.37.129.4 SSH Server Protocol: SSH-2.0-OpenS
SSH Protocol
Protocol: SSH-2.0-OpenSSH_4.7\n
0000 00 51 f5 d2 5a 44 00 1c 42 00 00 00 08 00 45 00 .Q..ZD.. B.....E.
0010 00 48 a9 57 40 00 40 06 7b 05 0a 25 81 05 0a 25 .H.W@.@. {..%...%
0020 81 04 00 16 80 08 6b a9 21 7e 8c d9 d6 6a 80 18 ......k. !~...j..
0030 ff ff 88 c1 00 00 01 01 08 0a 0f 48 07 96 00 07 ....H....
0040 96 77 53 53 48 2d 32 2e 30 2d 4f 70 65 6e 53 53 .wSSH-2. 0-OpenSS
0050 48 5f 34 2e 37 0a H_4.7.
File: "W:\sshfun\helloworld.pcap" 17 KB 00:00:05
Packets: 133 Displayed: 133 Marked: 0

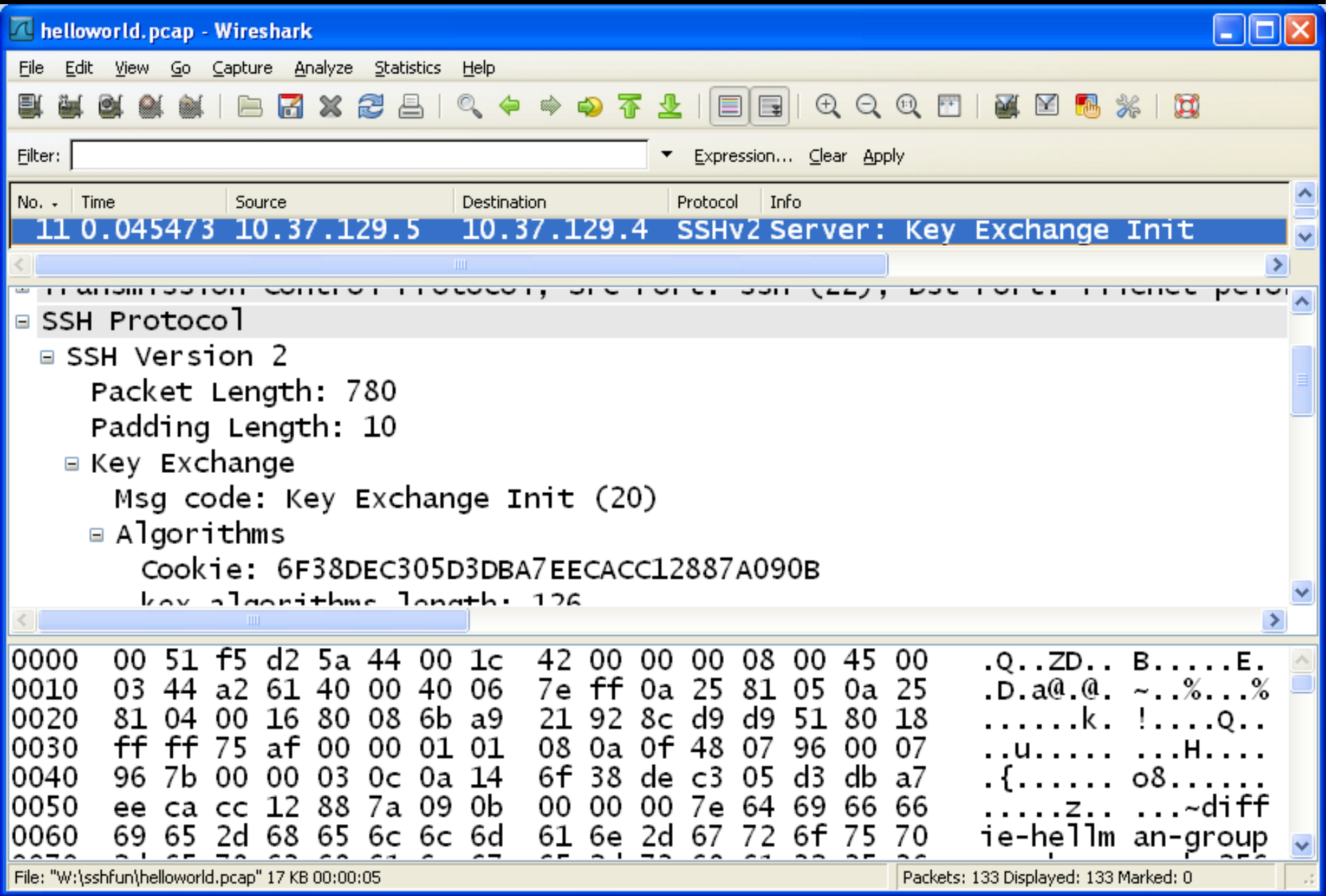
Breakout of Client Version String Announcement

- Sent from Client to Server
- Payload must start w/ 4-bytes "SSH-"
- SSH client version string
- e.g. "SSH-2.0-OpenSSH_4.3p2 Debian-9"


helloworld.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Info
7 0.037584 10.37.129.4 10.37.129.5 SSH Client Protocol: SSH-2.0-OpenSS
SSH Protocol
Protocol: SSH-2.0-OpenSSH_4.3p2 Debian-9\n
0000 00 1c 42 00 00 00 00 51 f5 d2 5a 44 08 00 45 00 ..B....Q ..ZD..E.
0010 00 53 40 21 40 00 40 06 e4 30 0a 25 81 04 0a 25 .S@!@.@. .0.%...%
0020 81 05 80 08 00 16 8c d9 d6 6a 6b a9 21 92 80 18 ..... .jk.!...
0030 16 d0 03 d3 00 00 01 01 08 0a 00 07 96 7a 0f 48 ....z.H
0040 07 96 53 53 48 2d 32 2e 30 2d 4f 70 65 6e 53 53 ..SSH-2. 0-OpenSS
0050 48 5f 34 2e 33 70 32 20 44 65 62 69 61 6e 2d 39 H_4.3p2 Debian-9
0060 0a .
File: "W:\sshfun\helloworld.pcap" 17 KB 00:00:05
Packets: 133 Displayed: 133 Marked: 0

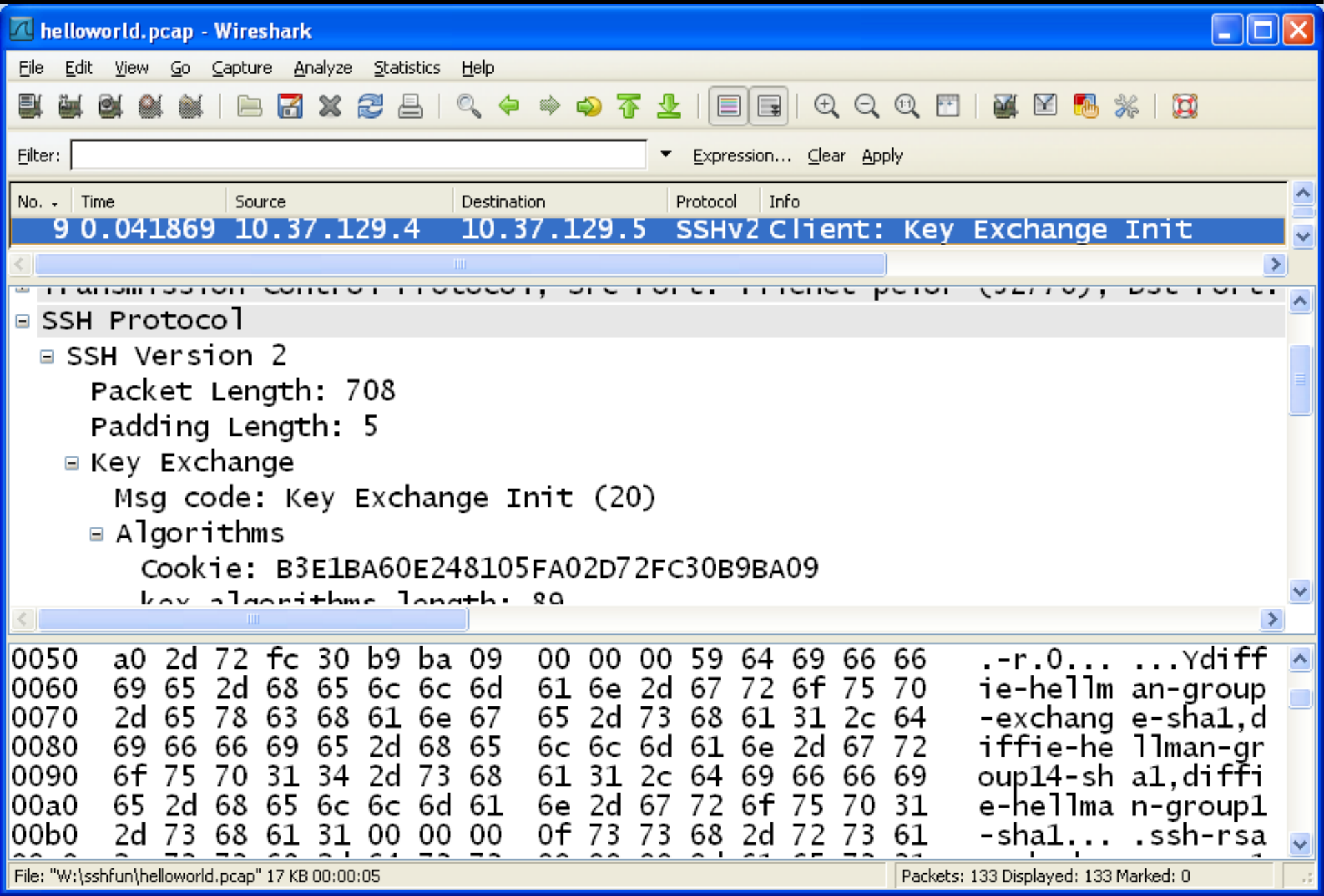
Breakout of SSH Server Key Exchange Init

- SSH2_MSG_KEXINIT (Message Code 20)
- Sent from Server to Client
- Lists algorithms supported by SSH server
  - Key Exchange Algs: diffie-hellman-group-exchange-sha1
  - Server Host Key Algs: ssh-rsa, ssh-dsa
  - Encryption Algs: aes128-cbc, 3des-cbc
  - MAC Algorithms: hmac-md5, hmac-sha1
  - Compression Algs: zlib@openssh.com, zlib
- Random NULL padding at end of payload
  - Part of SSH protocol
  - Intended to thwart traffic analysis


helloworld.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Info
11 0.045473 10.37.129.5 10.37.129.4 SSHv2 Server: Key Exchange Init
SSH Protocol
SSH Version 2
Packet Length: 780
Padding Length: 10
Key Exchange
Msg code: Key Exchange Init (20)
Algorithms
Cookie: 6F38DEC305D3DBA7EECACC12887A090B
0000 00 51 f5 d2 5a 44 00 1c 42 00 00 00 08 00 45 00 .Q..ZD.. B.....E.
0010 03 44 a2 61 40 00 40 06 7e ff 0a 25 81 05 0a 25 .D.a@.@. ~..%...%
0020 81 04 00 16 80 08 6b a9 21 92 8c d9 d9 51 80 18 ......k. !....Q..
0030 ff ff 75 af 00 00 01 01 08 0a 0f 48 07 96 00 07 ..u..... ...H....
0040 96 7b 00 00 03 0c 0a 14 6f 38 de c3 05 d3 db a7 .{...... o8......
0050 ee ca cc 12 88 7a 09 0b 00 00 00 7e 64 69 66 66 .....z.. ...~diff
0060 69 65 2d 68 65 6c 6c 6d 61 6e 2d 67 72 6f 75 70 ie-hellm an-group
File: "W:\sshfun\helloworld.pcap" 17 KB 00:00:05
Packets: 133 Displayed: 133 Marked: 0

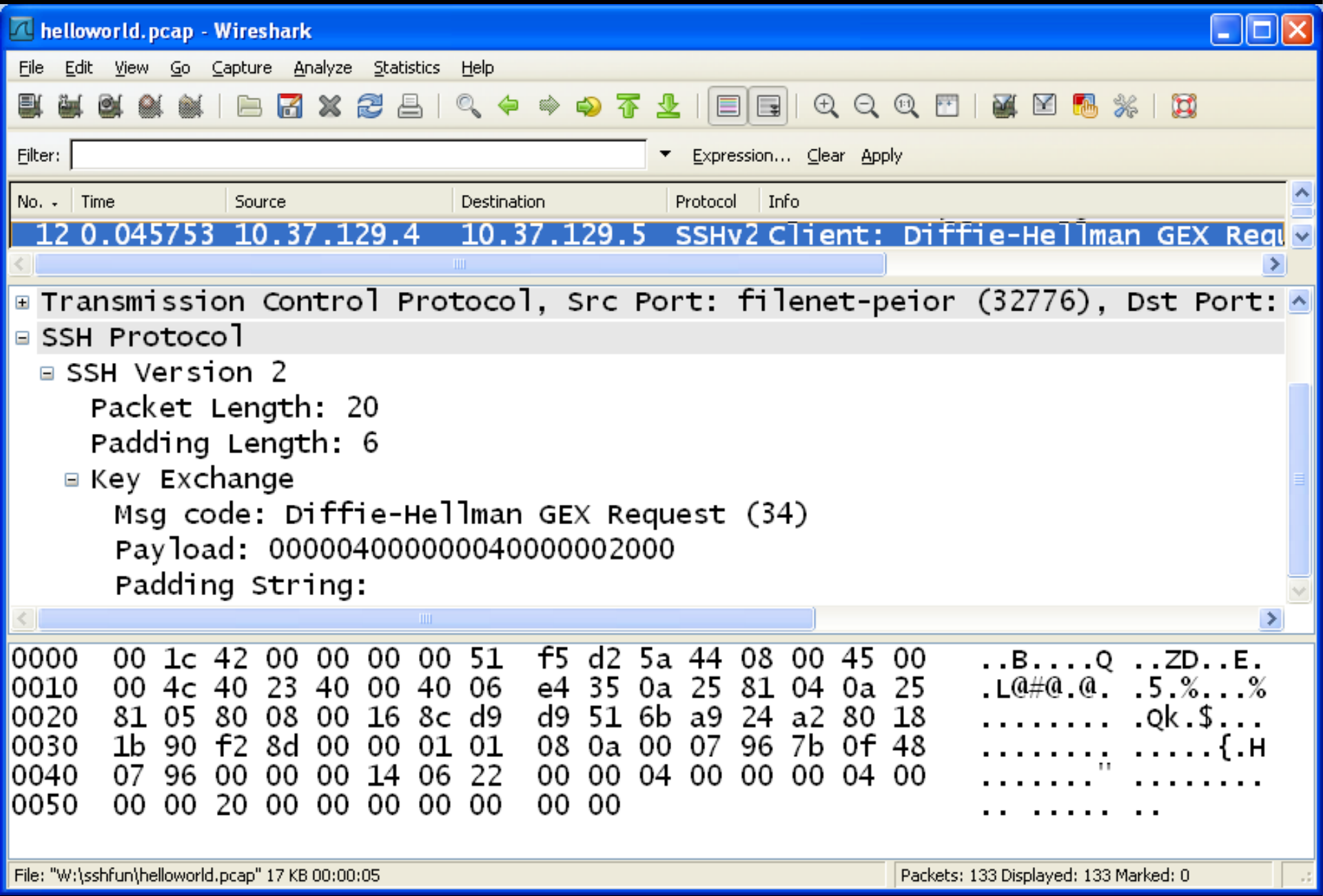
Breakout of SSH Client Key Exchange Init

- SSH2_MSG_KEXINIT (Message Code 20)
- Sent from Client to Server
- Lists algorithms supported by SSH client
  - Key Exchange Algs: diffie-hellman-group-exchange-sha1
  - Server Host Key Algs: ssh-rsa, ssh-dsa
  - Encryption Algs: aes128-cbc, 3des-cbc
  - MAC Algorithms: hmac-md5, hmac-sha1
  - Compression Algs: zlib@openssh.com, zlib
- Random NULL padding at end of payload
  - Part of SSH protocol
  - Intended to thwart traffic analysis


helloworld.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Info
9 0.041869 10.37.129.4 10.37.129.5 SSHv2 Client: Key Exchange Init
SSH Protocol
SSH Version 2
Packet Length: 708
Padding Length: 5
Key Exchange
Msg code: Key Exchange Init (20)
Algorithms
Cookie: B3E1BA60E248105FA02D72FC30B9BA09
0050 a0 2d 72 fc 30 b9 ba 09 00 00 00 59 64 69 66 66 .-r.0... ...Ydiff
0060 69 65 2d 68 65 6c 6c 6d 61 6e 2d 67 72 6f 75 70 ie-hellm an-group
0070 2d 65 78 63 68 61 6e 67 65 2d 73 68 61 31 2c 64 -exchang e-sha1,d
0080 69 66 66 69 65 2d 68 65 6c 6c 6d 61 6e 2d 67 72 iffie-he llman-gr
0090 6f 75 70 31 34 2d 73 68 61 31 2c 64 69 66 66 69 oup14-sh a1,diffi
00a0 65 2d 68 65 6c 6c 6d 61 6e 2d 67 72 6f 75 70 31 e-hellma n-group1
00b0 2d 73 68 61 31 00 00 00 0f 73 73 68 2d 72 73 61 -sha1... .ssh-rsa
File: "W:\sshfun\helloworld.pcap" 17 KB 00:00:05
Packets: 133 Displayed: 133 Marked: 0

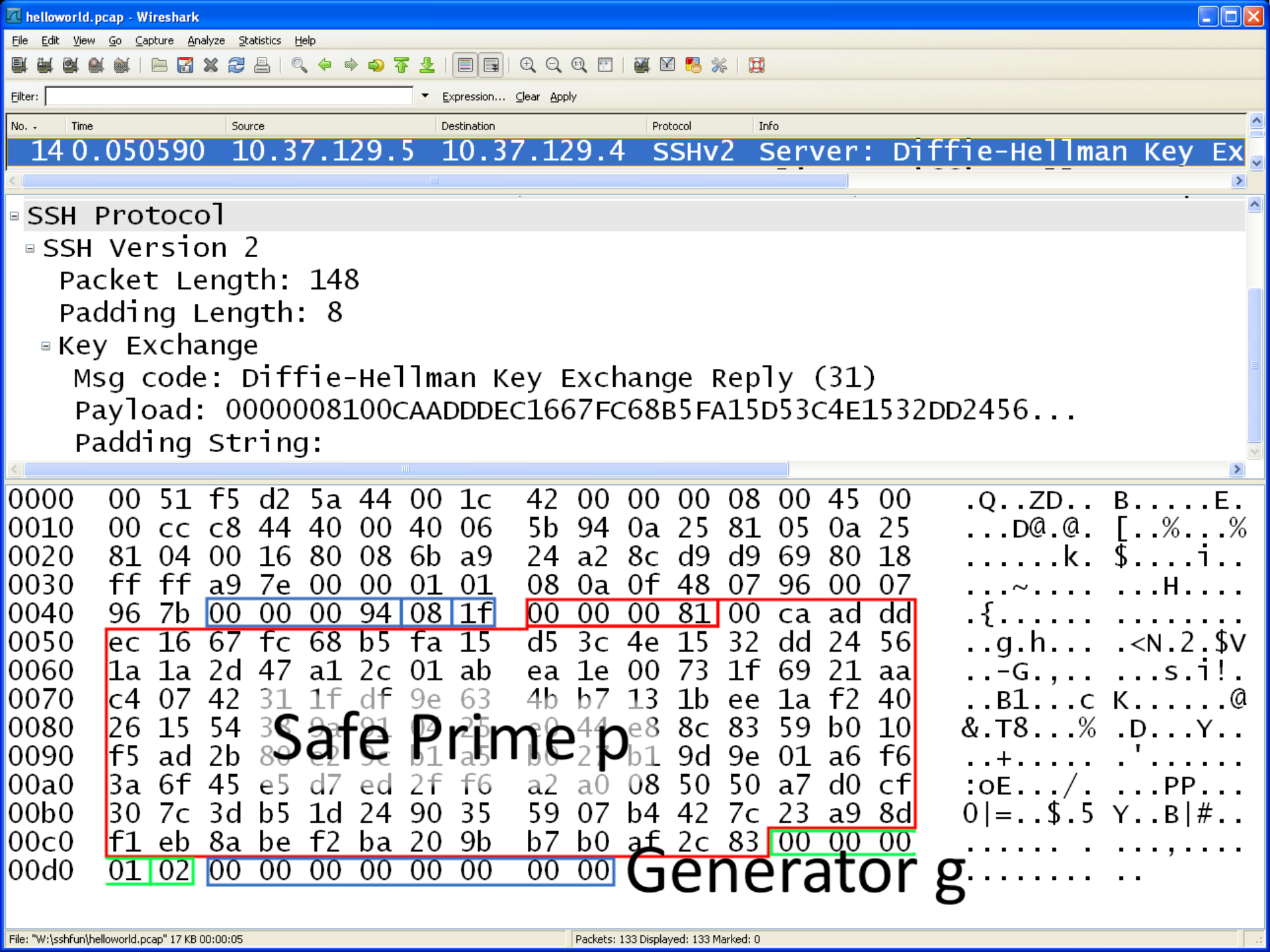
Breakout of SSH Diffie-Hellman Group Key Exchange Request

- SSH2_MSG_KEXDH_GEX_REQUEST (Message Code 34)
- Sent from Client to Server
- Minimum, Preferred, and Maximum Prime Size for the Group
  - Typical values are 1024, 1024, 8192


helloworld.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Info
12 0.045753 10.37.129.4 10.37.129.5 SSHv2 Client: Diffie-Hellman GEX Requ
Transmission Control Protocol, Src Port: filenet-peior (32776), Dst Port:
SSH Protocol
SSH Version 2
Packet Length: 20
Padding Length: 6
Key Exchange
Msg code: Diffie-Hellman GEX Request (34)
Payload: 000004000000040000002000
Padding String:
0000 00 1c 42 00 00 00 00 51 f5 d2 5a 44 08 00 45 00 ..B....Q ..ZD..E.
0010 00 4c 40 23 40 00 40 06 e4 35 0a 25 81 04 0a 25 .L@#@.@. .5.%...%
0020 81 05 80 08 00 16 8c d9 d9 51 6b a9 24 a2 80 18 ..... .Qk.$...
0030 1b 90 f2 8d 00 00 01 01 08 0a 00 07 96 7b 0f 48 ....{.H
0040 07 96 00 00 00 14 06 22 00 00 04 00 00 00 04 00 ..... " .....
0050 00 00 20 00 00 00 00 00 00 00 .. ..
File: "W:\sshfun\helloworld.pcap" 17 KB 00:00:05
Packets: 133 Displayed: 133 Marked: 0

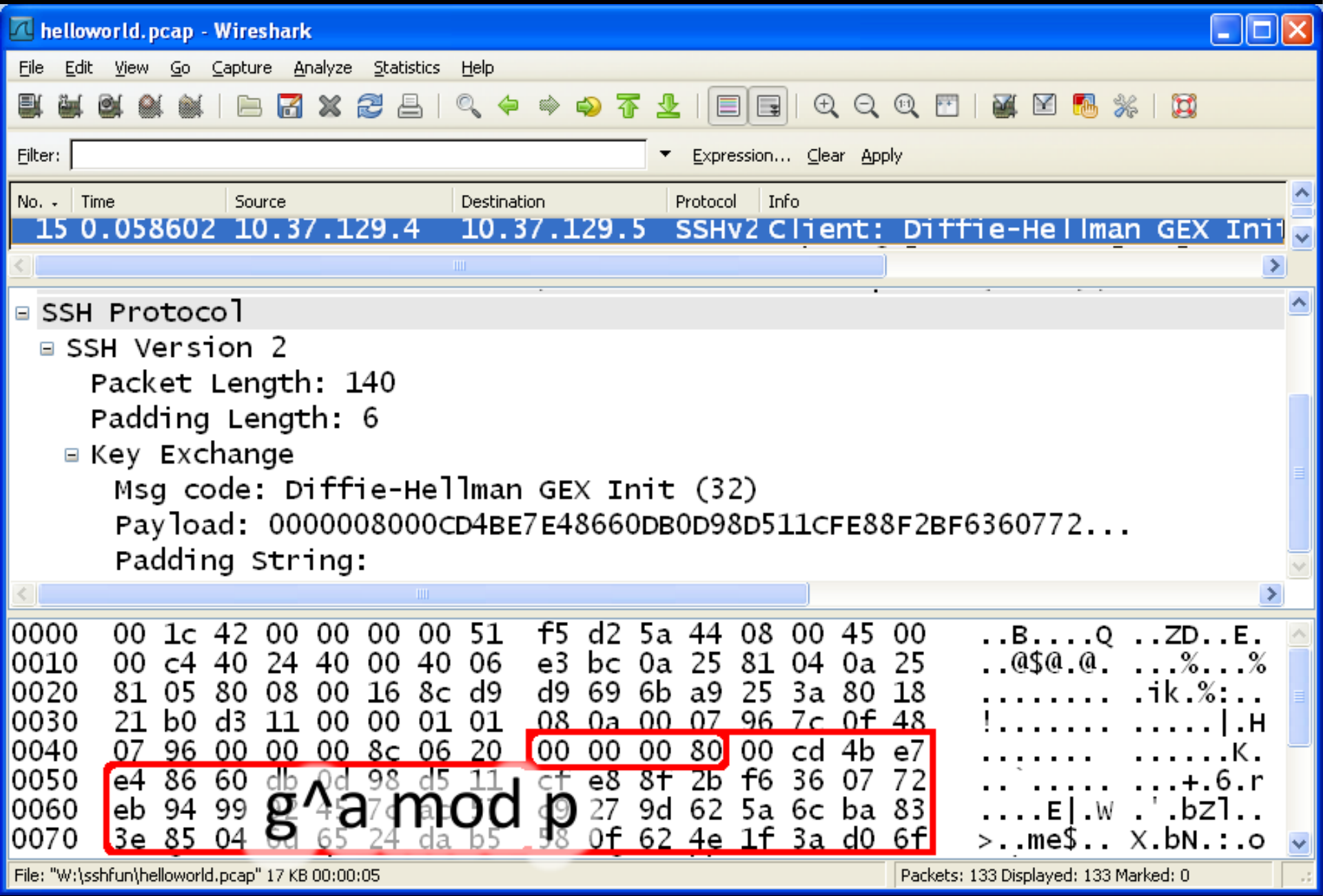
Breakout of SSH Diffie-Hellman Key Exchange Group Message

- SSH2_MSG_KEXDH_GEX_GROUP (Message Code 31)
- Sent from Server to Client
- Contains safe prime *p* and generator *g*
- Uses Multiple Precision Integers (MPI)
  - SSH specific MPI encoding (length, value)
  - Need to pass libgcrypt the right flag
- Random NULL padding at end of payload
  - Part of SSH protocol
  - Intended to thwart traffic analysis


helloworld.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Info
14 0.050590 10.37.129.5 10.37.129.4 SSHv2 Server: Diffie-Hellman Key Ex
SSH Protocol
SSH Version 2
Packet Length: 148
Padding Length: 8
Key Exchange
Msg code: Diffie-Hellman Key Exchange Reply (31)
Payload: 0000008100CAADDDEC1667FC68B5FA15D53C4E1532DD2456...
Padding String:
0000 00 51 f5 d2 5a 44 00 1c 42 00 00 00 08 00 45 00 .Q..ZD.. B.....E.
0010 00 cc c8 44 40 00 40 06 5b 94 0a 25 81 05 0a 25 ...D@.@. [..%...%
0020 81 04 00 16 80 08 6b a9 24 a2 8c d9 d9 69 80 18 ......k. $....i..
0030 ff ff a9 7e 00 00 01 01 08 0a 0f 48 07 96 00 07 ...~.... ...H....
0040 96 7b 00 00 00 94 08 1f 00 00 00 81 00 ca ad dd .{...... ....
0050 ec 16 67 fc 68 b5 fa 15 d5 3c 4e 15 32 dd 24 56 ..g.h... .<N.2.$V
0060 1a 1a 2d 47 a1 2c 01 ab ea 1e 00 73 1f 69 21 aa ..-G.,.. ...s.i!.
0070 c4 07 42 31 1f df 9e 63 4b b7 13 1b ee 1a f2 40 ..B1...c K......@
0080 26 15 54 38 9a 91 04 25 e0 44 e8 8c 83 59 b0 10 &.T8...% .D...Y..
0090 f5 ad 2b 80 e2 9c b1 a5 b0 2f b1 9d 9e 01 a6 f6 ..+..... ./......
00a0 3a 6f 45 e5 d7 ed 2f f6 a2 a0 08 50 50 a7 d0 cf :oE.../. ...PP...
00b0 30 7c 3d b5 1d 24 90 35 59 07 b4 42 7c 23 a9 8d 0|=..$.5 Y..B|#..
00c0 f1 eb 8a be f2 ba 20 9b b7 b0 af 2c 83 00 00 00 ....,....
00d0 01 02 00 00 00 00 00 00 00 00 ....
Safe Prime p
Generator g
File: "W:\sshfun\helloworld.pcap" 17 KB 00:00:05
Packets: 133 Displayed: 133 Marked: 0

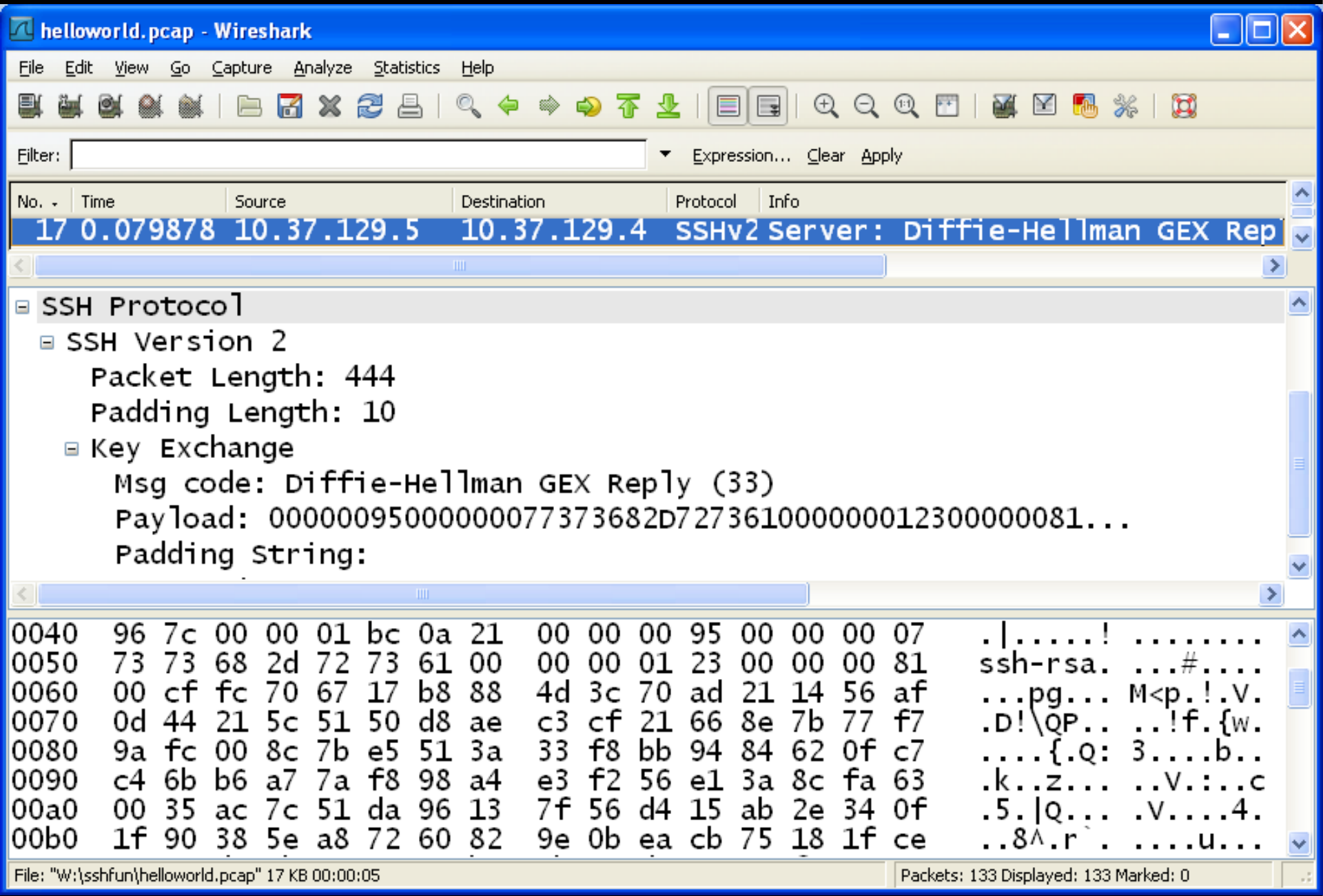
Breakout of SSH Diffie-Hellman Group Key Exchange Init

- SSH2_MSG_KEXDH_GEX_INIT (Message Code 32)
- Sent from Client to Server
- Client generates a random value *a*
- Client then calculates g^a mod p (or just g^a for short)
- Sends g^a value to SSH server in this message


helloworld.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Info
15 0.058602 10.37.129.4 10.37.129.5 SSHv2 Client: Diffie-Hellman GEX Init
SSH Protocol
SSH Version 2
Packet Length: 140
Padding Length: 6
Key Exchange
Msg code: Diffie-Hellman GEX Init (32)
Payload: 0000008000CD4BE7E48660DB0D98D511CFE88F2BF6360772...
Padding String:
0000 00 1c 42 00 00 00 00 51 f5 d2 5a 44 08 00 45 00
0010 00 c4 40 24 40 00 40 06 e3 bc 0a 25 81 04 0a 25
0020 81 05 80 08 00 16 8c d9 d9 69 6b a9 25 3a 80 18
0030 21 b0 d3 11 00 00 01 01 08 0a 00 07 96 7c 0f 48
0040 07 96 00 00 00 8c 06 20 00 00 00 80 00 cd 4b e7
0050 e4 86 60 db 0d 98 d5 11 cf e8 8f 2b f6 36 07 72
0060 eb 94 99 27 9d 62 5a 6c ba 83
0070 3e 85 04 0f 62 4e 1f 3a d0 6f
g^a mod p
File: "W:\sshfun\helloworld.pcap" 17 KB 00:00:05
Packets: 133 Displayed: 133 Marked: 0

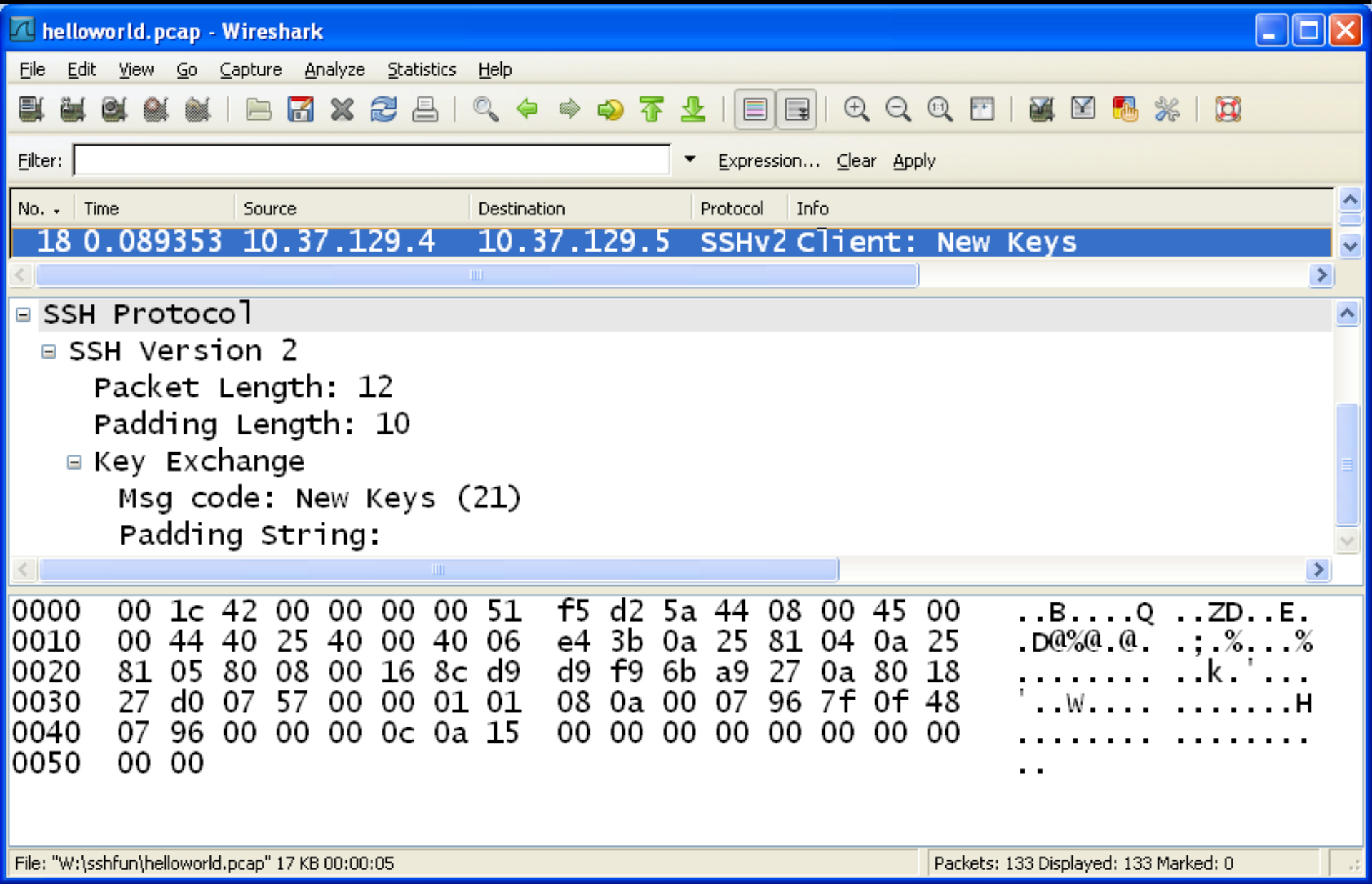
Breakout of SSH Diffie-Hellman Group Key Exchange Reply

- SSH2_MSG_KEXDH_GEX_REPLY (Message Code 33)
- Sent from Server to Client
- Server generates a random value *b*
- Server then calculates g^b mod *p* (or just g^b for short)
- Sends g^b value to SSH client in this message
- Also sends SSH server's public host key (ssh-rsa, ssh-dsa)
  - Sole authentication of server's identity


helloworld.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Info
17 0.079878 10.37.129.5 10.37.129.4 SSHv2 Server: Diffie-Hellman GEX Rep
SSH Protocol
SSH Version 2
Packet Length: 444
Padding Length: 10
Key Exchange
Msg code: Diffie-Hellman GEX Reply (33)
Payload: 000000950000000773736 82D72736100000001230000008 1...
Padding String:
0040 96 7c 00 00 01 bc 0a 21 00 00 00 95 00 00 00 07 .|.....! ....
0050 73 73 68 2d 72 73 61 00 00 00 01 23 00 00 00 81 ssh-rsa. ...#....
0060 00 cf fc 70 67 17 b8 88 4d 3c 70 ad 21 14 56 af ...pg... M<p.!.V.
0070 0d 44 21 5c 51 50 d8 ae c3 cf 21 66 8e 7b 77 f7 .D!\QP.. ..!f.{w.
0080 9a fc 00 8c 7b e5 51 3a 33 f8 bb 94 84 62 0f c7 ....{.Q: 3....b..
0090 c4 6b b6 a7 7a f8 98 a4 e3 f2 56 e1 3a 8c fa 63 .k..z... ..V.:..c
00a0 00 35 ac 7c 51 da 96 13 7f 56 d4 15 ab 2e 34 0f .5.|Q... .V....4.
00b0 1f 90 38 5e a8 72 60 82 9e 0b ea cb 75 18 1f ce ..8^.r`. ....u...
File: "W:\sshfun\helloworld.pcap" 17 KB 00:00:05
Packets: 133 Displayed: 133 Marked: 0

Breakout of SSH Client New Keys Message

- SSH2_MSG_NEWKEYS (Message Code 21)
- Sent from Client to Server
- Indicates the negotiated keys/algorithms should go into effect from now on
- During session, either host can request re-keying w/ this
- RFC recommends re-keying once per GB of data, or once per hour, whichever comes first


helloworld.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Info
18 0.089353 10.37.129.4 10.37.129.5 SSHv2 Client: New Keys
SSH Protocol
SSH Version 2
Packet Length: 12
Padding Length: 10
Key Exchange
Msg code: New Keys (21)
Padding String:
0000 00 1c 42 00 00 00 00 51 f5 d2 5a 44 08 00 45 00 ..B....Q ..ZD..E.
0010 00 44 40 25 40 00 40 06 e4 3b 0a 25 81 04 0a 25 .D@%@.@. .;.%...%
0020 81 05 80 08 00 16 8c d9 d9 f9 6b a9 27 0a 80 18 ....k.'...
0030 27 d0 07 57 00 00 01 01 08 0a 00 07 96 7f 0f 48 '..W.... ....H
0040 07 96 00 00 00 0c 0a 15 00 00 00 00 00 00 00 00 ....
0050 00 00 ..
File: "W:\sshfun\helloworld.pcap" 17 KB 00:00:05
Packets: 133 Displayed: 133 Marked: 0

Diffie-Hellman Key Exchange for SSH

Do the Math!

- A way for two parties to agree on a random shared secret over an insecure channel.

- Server sends to Client
  - p – large "safe" prime number
    - A prime p is "safe" if $p = 2q + 1$, and q is prime
  - g – generator of the field $(Z_p)^*$ (typically 0x02)

Diffie-Hellman Key Exchange for SSH

Do the Math! (2)

- Client generates random number *a*
  - Calculates (g^a mod p)
  - Sends calculated value to server
- Server generates random number *b*
  - Calculates (g^b mod p)
  - Sends calcualted value to client

Diffie-Hellman Key Exchange for SSH

Do the Math! (3)

- DH shared secret is defined as both a function of *a* and of *b*
- Only parties that know *a* or *b* can (feasibly) calculate it

Diffie-Hellman Key Exchange for SSH

Do the Math! (4)

- Client
  - knows g, a and $(g^b \bmod p)$
  - Calculates shared secret as $(g^b \bmod p)^a \bmod p$
- Server
  - knows g, b and $(g^a \bmod p)$
  - Calculates shared secret as $(g^a \bmod p)^b \bmod p$

Diffie-Hellman Key Exchange for SSH

Do the Math! (5)

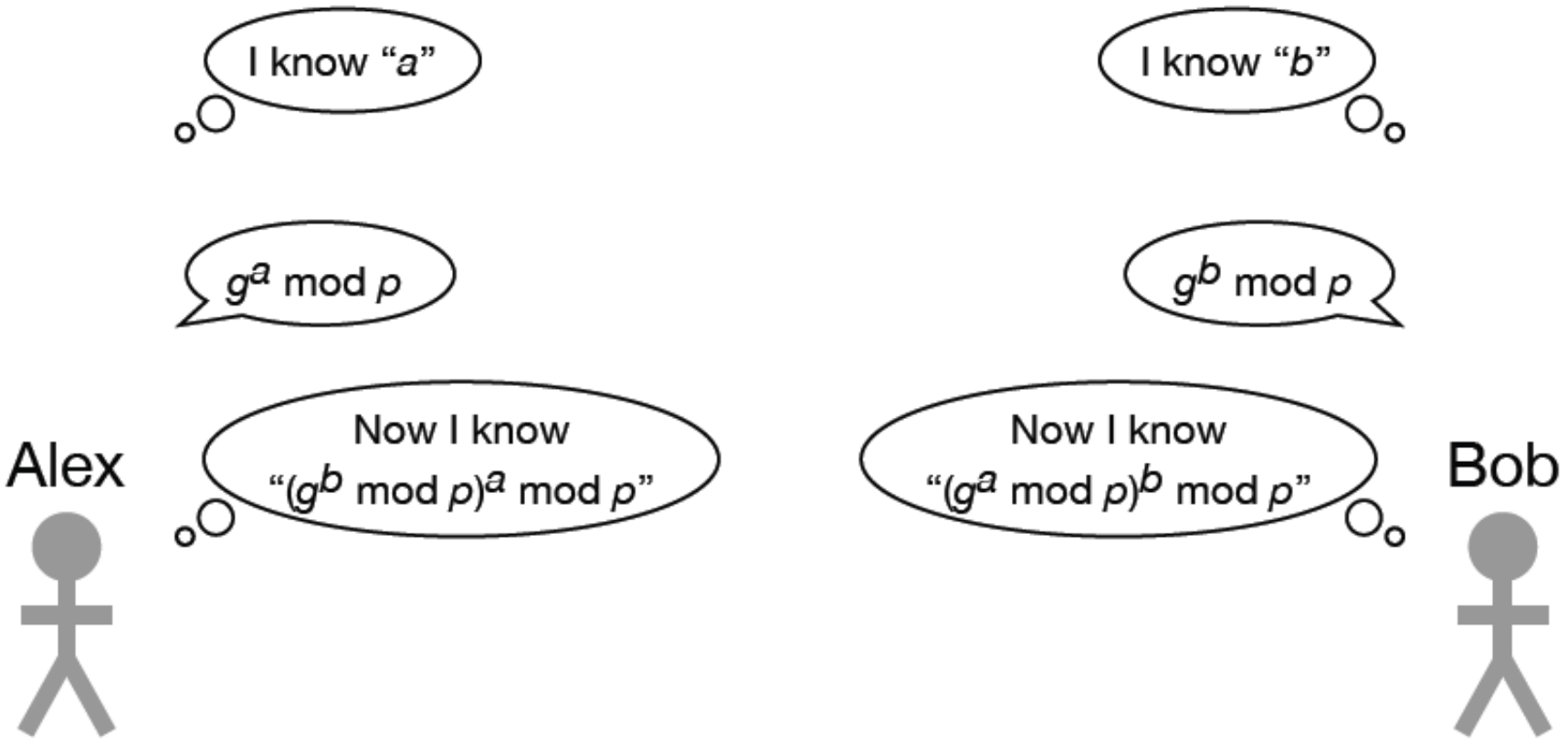
- Eavesdropper knows g, (g^a mod p) and (g^b mod p)
- Computationally infeasible to calculate (g^{ab} mod p) from (g^a mod p) and (g^b mod p)
- Why?

Diffie-Hellman Key Exchange for SSH

Do the Math! (6)

- Must solve the discrete logarithm problem
  - No known (non-quantum) algorithm to solve in polynomial time
  - *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*
  - Peter W. Shor, AT&T Research
  - 30 August 1995, Revised 25 January 1996
  - arXiv:quant-ph/9508027v2

$$(g^b \bmod p)^a \bmod p = (g^a \bmod p)^b \bmod p$$



Source: Stephen Cristol, DC404

Diffie-Hellman Key Exchange for SSH

Do the Math! (7)

- Encryption IVs and Keys generated from DH shared secret
- V_C, V_S – Client / Server's SSH version announce string
- I_C, I_S – Client / Server's SSH2_MSG_KEXINIT message
- K_S – Server's Public Host Key

Diffie-Hellman Key Exchange for SSH

Do the Math! (8)

- H = hash(V_C | V_S | I_C | I_S | K_S | g^a mod p | g^b mod p | g^{ab} mod p)
- Hash function typically negotiated to sha1
- SSH *session_id* = H of initial DH key exchange

Diffie-Hellman Key Exchange for SSH

Do the Math! (9)

- IV client to server: hash(g^{ab} mod p | H | "A" | *session_id*)
- IV server to client: hash(g^{ab} mod p | H | "B" | *session_id*)
- Enc Key client to server: hash(g^{ab} mod p | H | "C" | *session_id*)
- Enc Key server to client: hash(g^{ab} mod p | H | "D" | *session_id*)

OMG! Pwnies!

The Debian OpenSSL PRNG and SSH DH GEX

- If OpenSSH client or server is linked against vulnerable Debian OpenSSL
  - *a* or *b* is completely predictable based on Process ID of OpenSSH
- We can quickly brute force *a* or *b*.
  - Only 32768 possibilites!
- If we know *a* or *b*, we can calculate DH shared secret
- Once we know the DH shared secret, we have everything needed to decrypt the SSH session layer!

The Debian OpenSSL PRNG and SSH DH GEX

The Impact

- Tunneled Clear Text Passwords are compromised
  - …if ***either*** client or server is using vulnerable OpenSSL
  - RSA / DSA public key authentication is not affected
- Files or other data protected by SSH Session layer are compromised
  - …if ***either*** client or server is using vulnerable OpenSSL
- Observers can easily tell if either client or server is using vulnerable OpenSSL
  - …and proceed to decrypt the stream

Brute Force Decryption of Captured SSH Session Generated With Vulnerable Debian OpenSSL

Live Demo

Into the Future

- Fingerprinting version string announcements for potentially vulnerable SSH clients and servers?
  - Only attempt to crack sessions w/ hosts advertizing a potentially vulnerable SSH version string
  - e.g, "SSH-2.0-OpenSSH_4.3p2 Debian-9"
- Processing large pcap archives of SSH traffic
  - Captured at a recent major security conference
- Enhancements to Snort dynamic preprocessor
  - Session normalization/decoding
  - Performance

Thanks to h1kari, Geo, Tim, et al for making ToorCon X a reality!

Questions?

bfeinstein@secureworks.com