SSH – The 'Secure' Shell

Course: CSCI 5931 Web Security

Instructor : Dr.Yang

Presented by

Dr. S. R. Shinde

Secure What.. ?

- ‘*Secure shell is a de facto standard for remote logins and encrypted file transfers*.’ [SSH communications inc.]
- Founded in 1995 by Tatu Ylonen, a researcher at Helsinki University of Technology, Finland
- It provides authentication and encryption for business critical applications to work securely over the internet.
- Originally introduced for UNIX terminals as a replacement for the insecure remote access “Berkeley services" , viz. rsh, rlogin, rcp, telnet, etc.
- It can also be used for port forwarding of arbitrary TCP/IP or X11 connections (interactive terminals)
- It is a layer over TCP/IP and runs on the port 22.

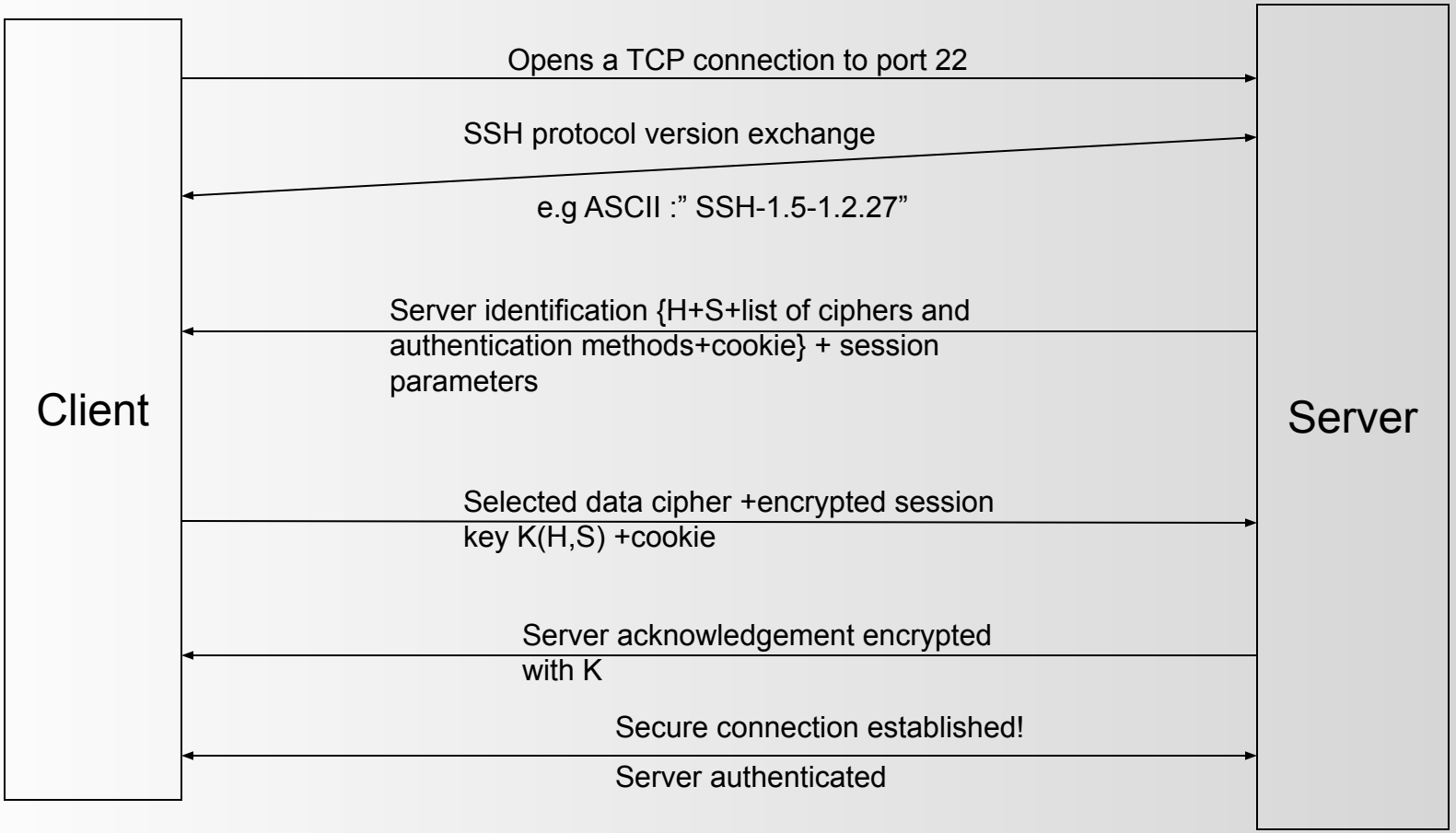
Why SSH?

- The three core security requirements for a remote access technology – confidentiality, integrity and authentication
- Most of the earlier technologies lack confidentiality and integrity. For e.g Telnet and FTP transmit username and passwords in cleartext.
- They are vulnerable to attacks such as IP spoofing, DoS, MITM and eavesdropping.
- Secure shell satisfies all the three requirements by using:
  - Data Encryption to provide confidentiality
  - Host-based and (or) client-based authentication
  - Data integrity using MACs and hashes

Flavors of SSH

- There are two incompatible versions of the SSH protocol: SSH-1 and SSH-2
- SSH-2 was introduced in order to fix several bugs in SSH-1 and also includes several new features
- Both versions have technical glitches and are vulnerable to known attacks (discussed later)
- SSH-1 is more popular nowadays due to licensing issues with SSH-2
- OpenSSH is a free version of the SSH with open source code development. It supports both the versions and mainly used in UNIX environment.

The SSH-1 protocol exchange



H – host key ; S- Server Key ; cookie- sequence of 8 random bytes; K- session key

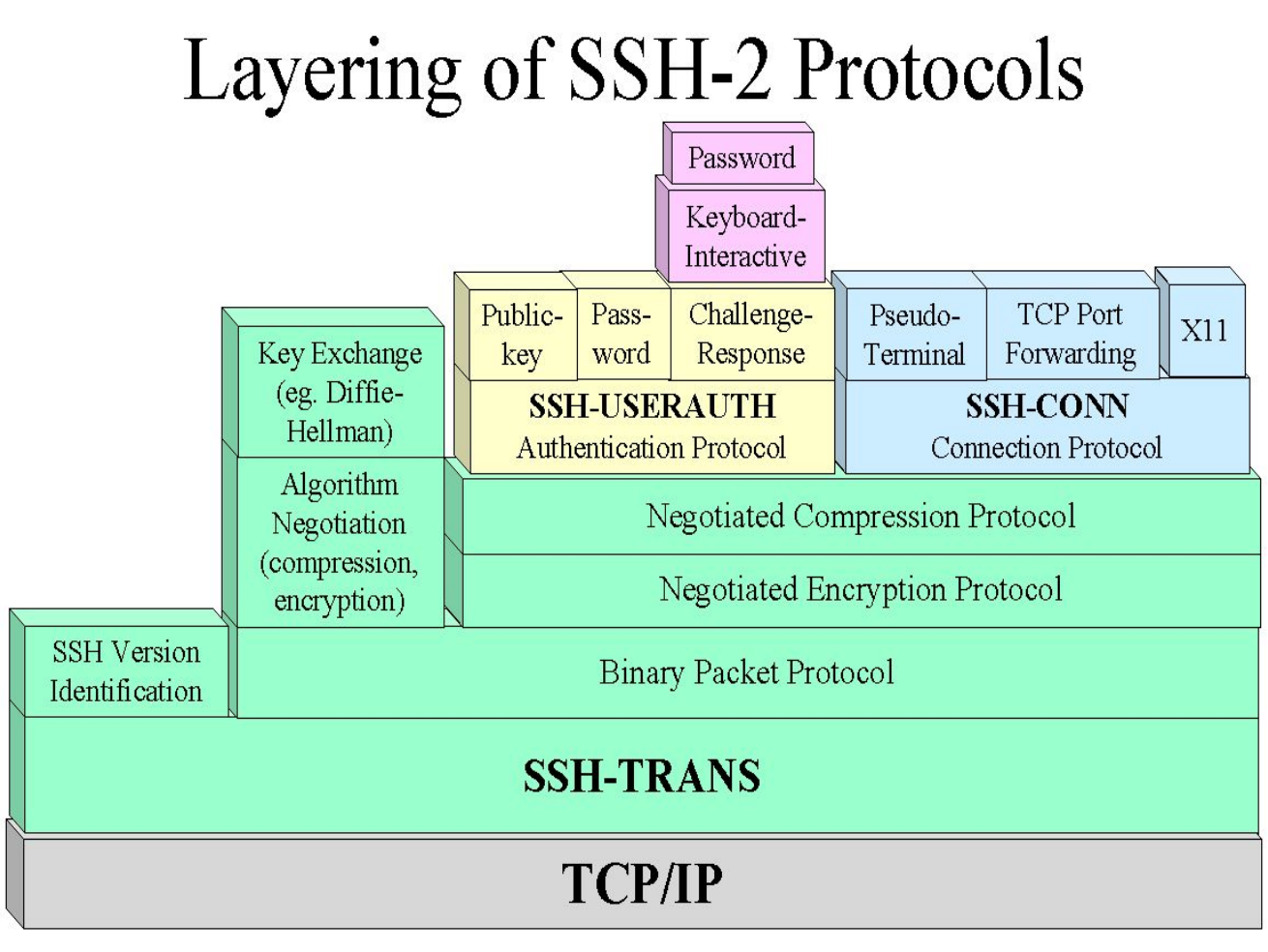
The SSH-1 protocol exchange (contd.)

- Once secure connection is established, client attempts to authenticate itself to server.
- Some of the authentication methods used are:
  - Kerberos
  - RHosts and RHostsRSA
  - Public key
  - Password based authentication (OTP)
- Integrity checking is provided by means of a weak CRC -32
- Compression is provided using the “deflate” algorithm of GNU *gzip* utility. It is beneficial for file transfer utilities such as *ftp*, *scp*, etc.

The SSH-2 protocol

- SSH-1 is monolithic, encompassing multiple functions into a single protocol whereas SSH-2 has been separated into modules and consists of 3 protocols:
  - **SSH Transport layer protocol (SSH-TRANS) :** Provides the initial connection, packet protocol, server authentication and basic encryption and integrity services.
  - **SSH Authentication protocol (SSH-AUTH) :** Verifies the client's identity at the beginning of an SSH-2 session, by three authentication methods: Public key, host based and password.
  - **SSH Connection protocol (SSH-CONN) :** It permits a number of different services to exchange data through the secure channel provided by SSH-TRANS.
- A fourth protocol SSH protocol architecture (SSH-ARCH) describes the overall architecture.
- All the protocols are still in the draft stage.

The SSH-2 protocol (contd.)



[Source: 'Analysis of Secure shell vulnerability' – white paper]

SSH-1 vs SSH-2

SSH-2	SSH-1
Separate transport, authentication and connection protocols	One monolithic protocol
Strong cryptographic integrity check	Weak CRC-32 integrity check
No server keys needed due to Diffie Hellman Key exchange	Server key used for forward secrecy on the session key
Supports public key certificates	N/A
Algorithms used: DSA, DH, SHA-1, MD5, 3DES, Blowfish, Twofish, CAST-128, IDEA, ARCFOUR	RSA, MD5, CRC-32, 3DES, IDEA, ARCFOUR, DES
Periodic replacement of session keys	N/A

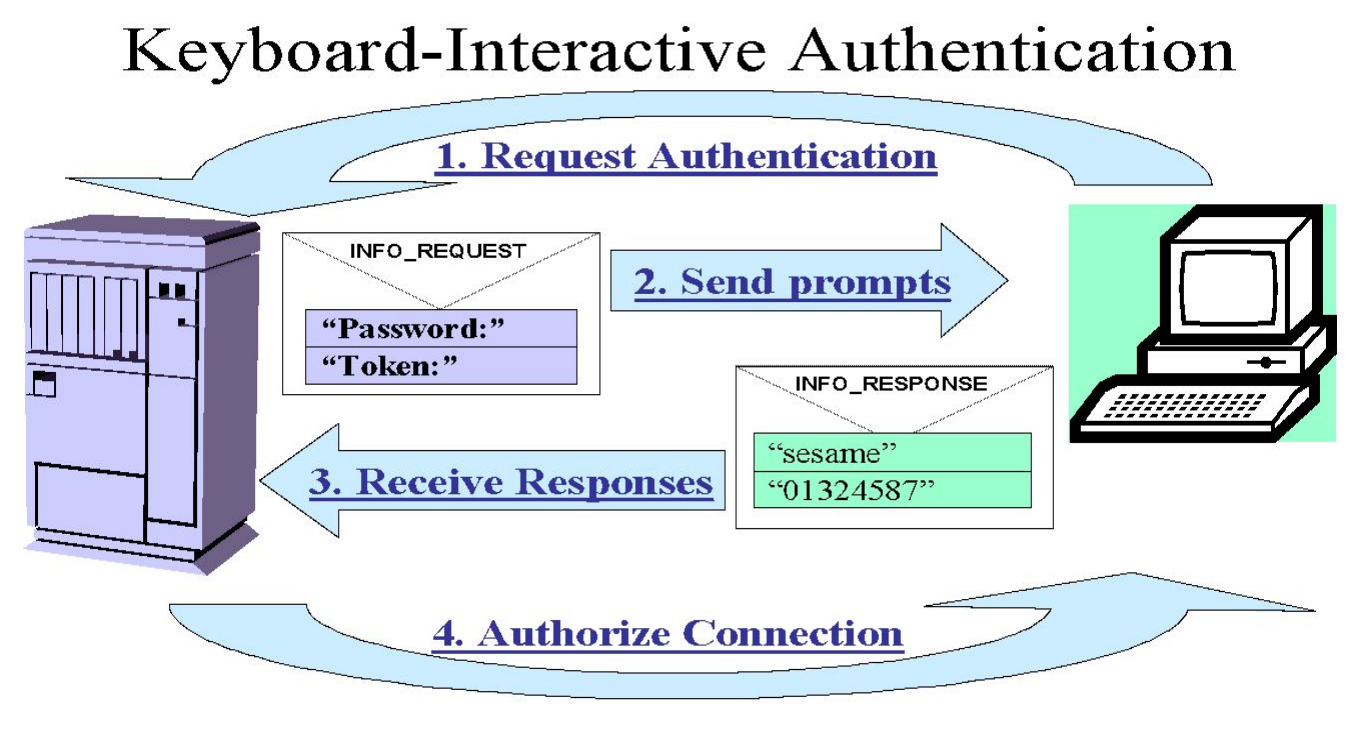
Is SSH really secure?

- Secure shell does counter certain types of attacks such as:
  - Eavesdropping
  - Name service and IP spoofing
  - Connection hijacking
  - MITM
  - Insertion attack
- However it fails against these attacks:
  - Password cracking
  - IP and TCP attacks
    - *SYN flooding*
    - *TCP RST, bogus ICMP*
    - *TCP desynchronization and hijacking*
  - Traffic analysis
  - Covert channels

Some known vulnerabilities in SSH

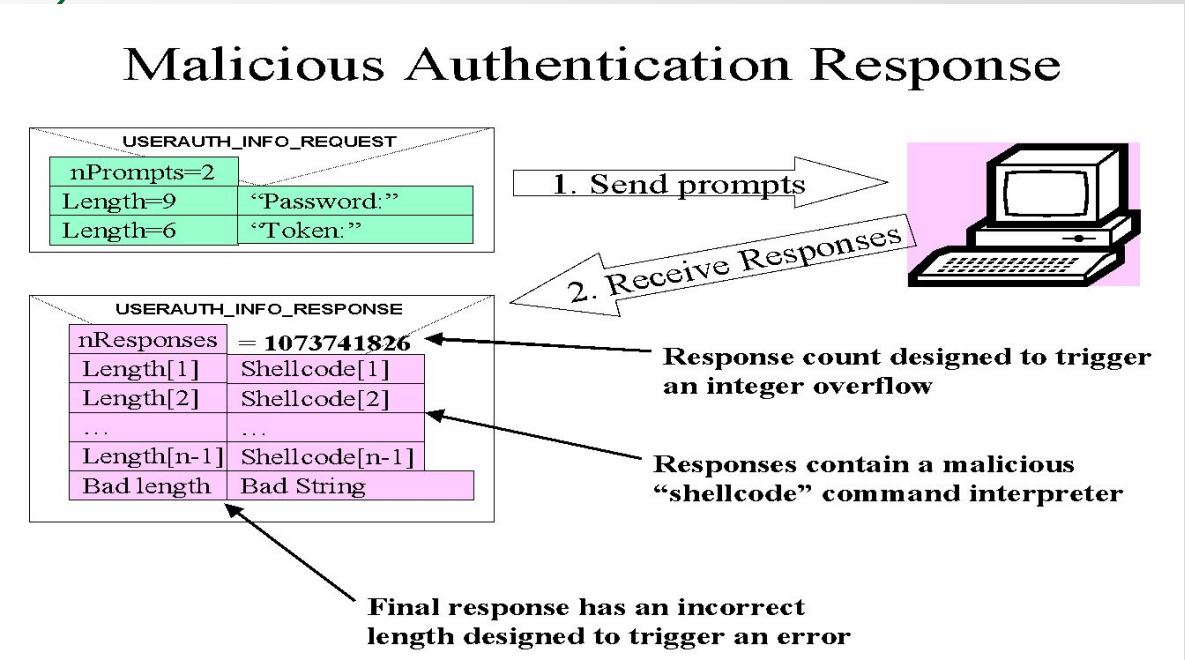
- OpenSSH Challenge-Response Authentication Buffer Overflow: A hostile modification to the SSH client floods the server with authentication responses and causes a buffer overflow. [SSH-2]
- Passive analysis of SSH traffic: It lets the attacker obtain sensitive information by passively monitoring encrypted SSH sessions. The information can later be used to speed up brute-force attacks on passwords, including the initial login password and other passwords appearing in interactive SSH sessions. [SSH-1 & SSH-2]
- Key recovery in SSH protocol 1.5 : This vulnerability may lead to the compromise of the session key. Once the session key is determined, the attacker can proceed to decrypt the stored session using any implementation of the crypto algorithm used. This will reveal all information in an unencrypted form. [SSH-1]
- CRC-32 integrity check vulnerability : It is based on the weakness of the CRC-32 integrity that becomes exploitable due to CBC and CFB feedback modes [SSH-1]

Case study : The 'GOBBLES' exploit



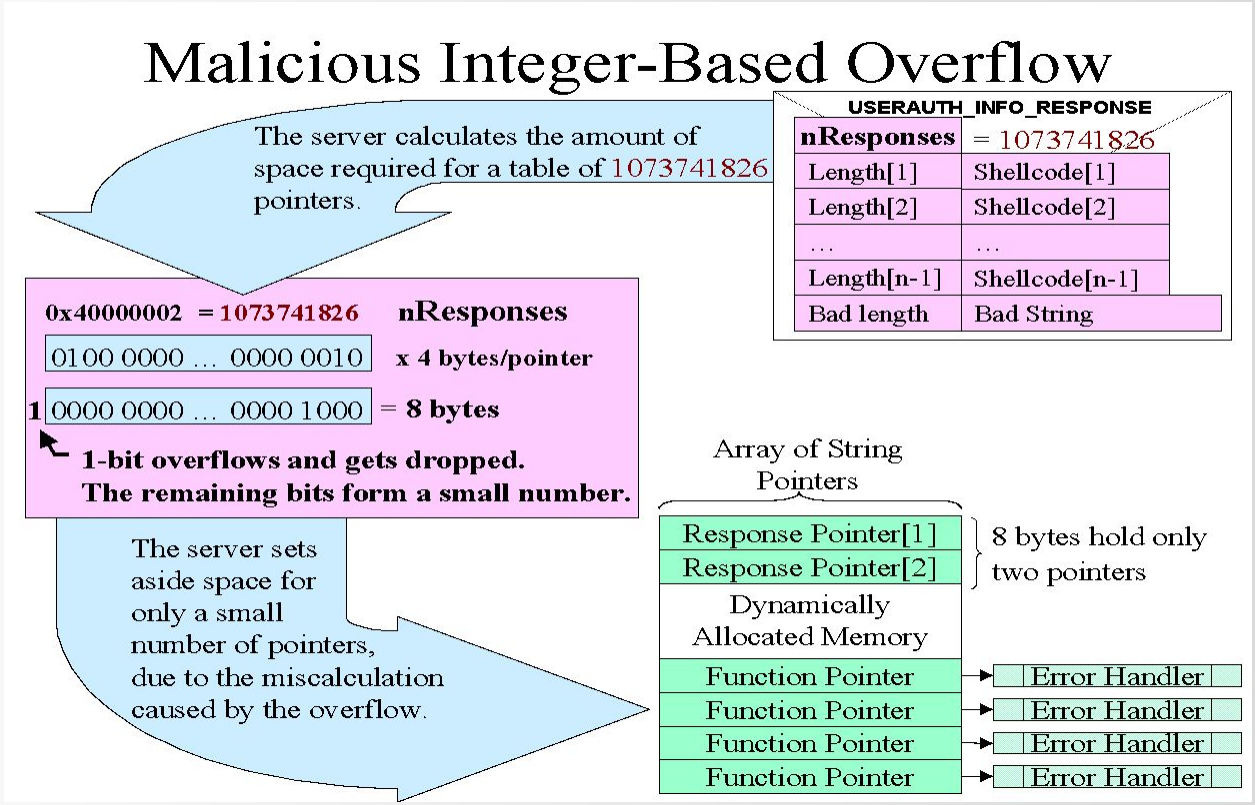
- This exploits the weakness in the challenge-response authentication mechanism of OpenSSH.
- SSH_MSG_USERAUTH_REQUEST, SSH_MSG_USERAUTH_INFO_REQUEST , SSH_MSG_USERAUTH_INFO_RESPONSE

Case study : The 'GOBBLES' exploit (contd.)

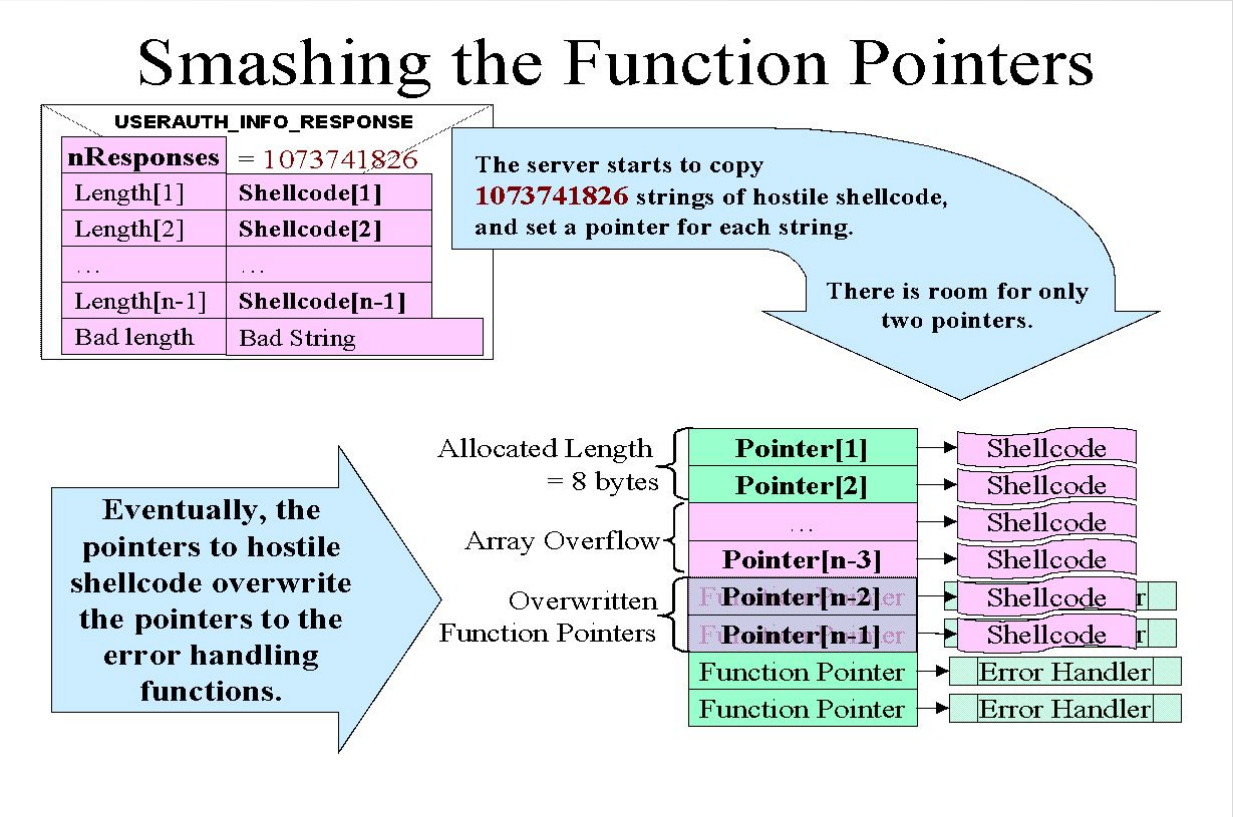


- Instead of sending the requested number of responses, the client sends a large integer
- The server allocates memory for these response strings

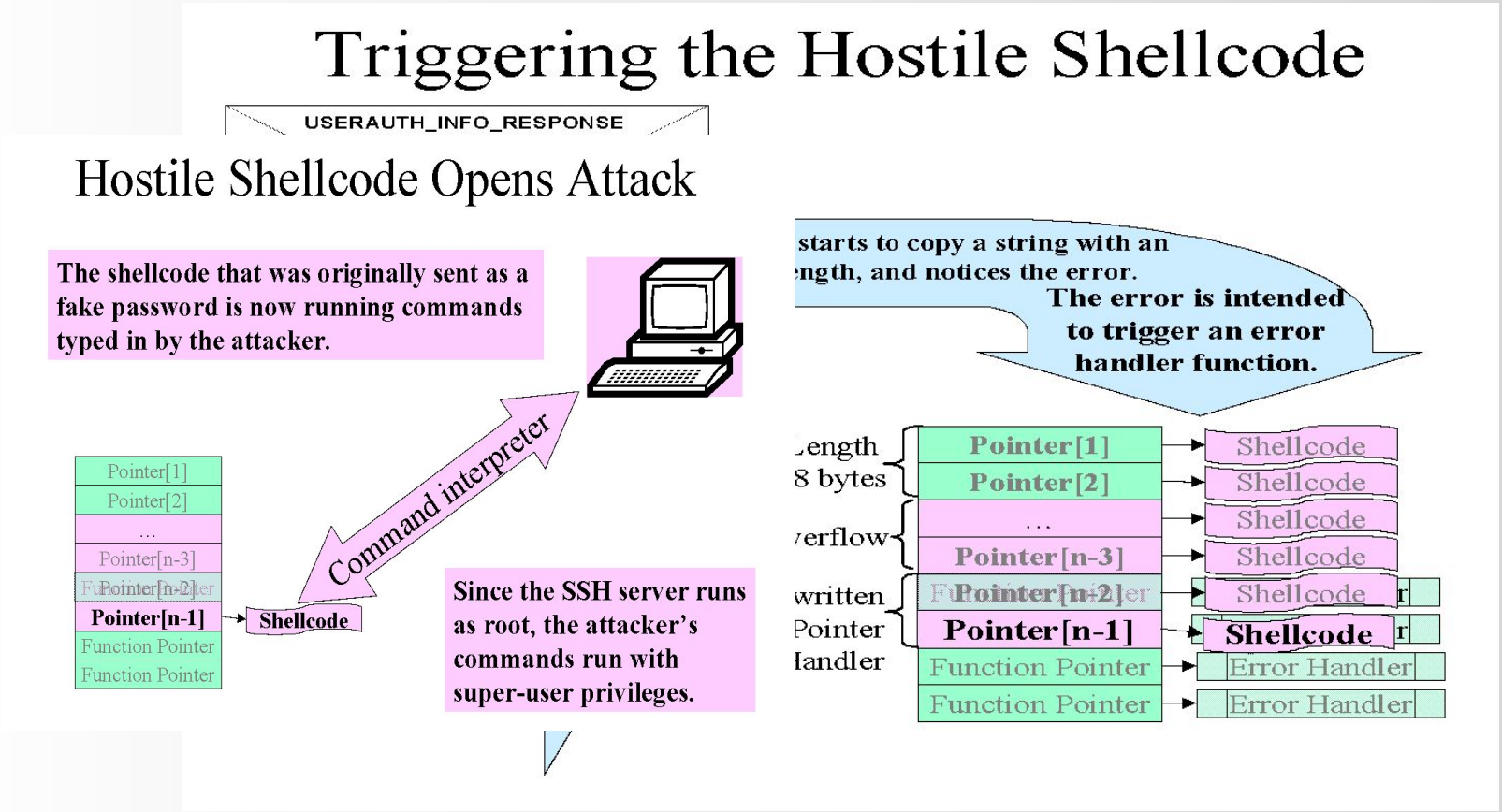
Case study : The 'GOBBLES' exploit (contd.)



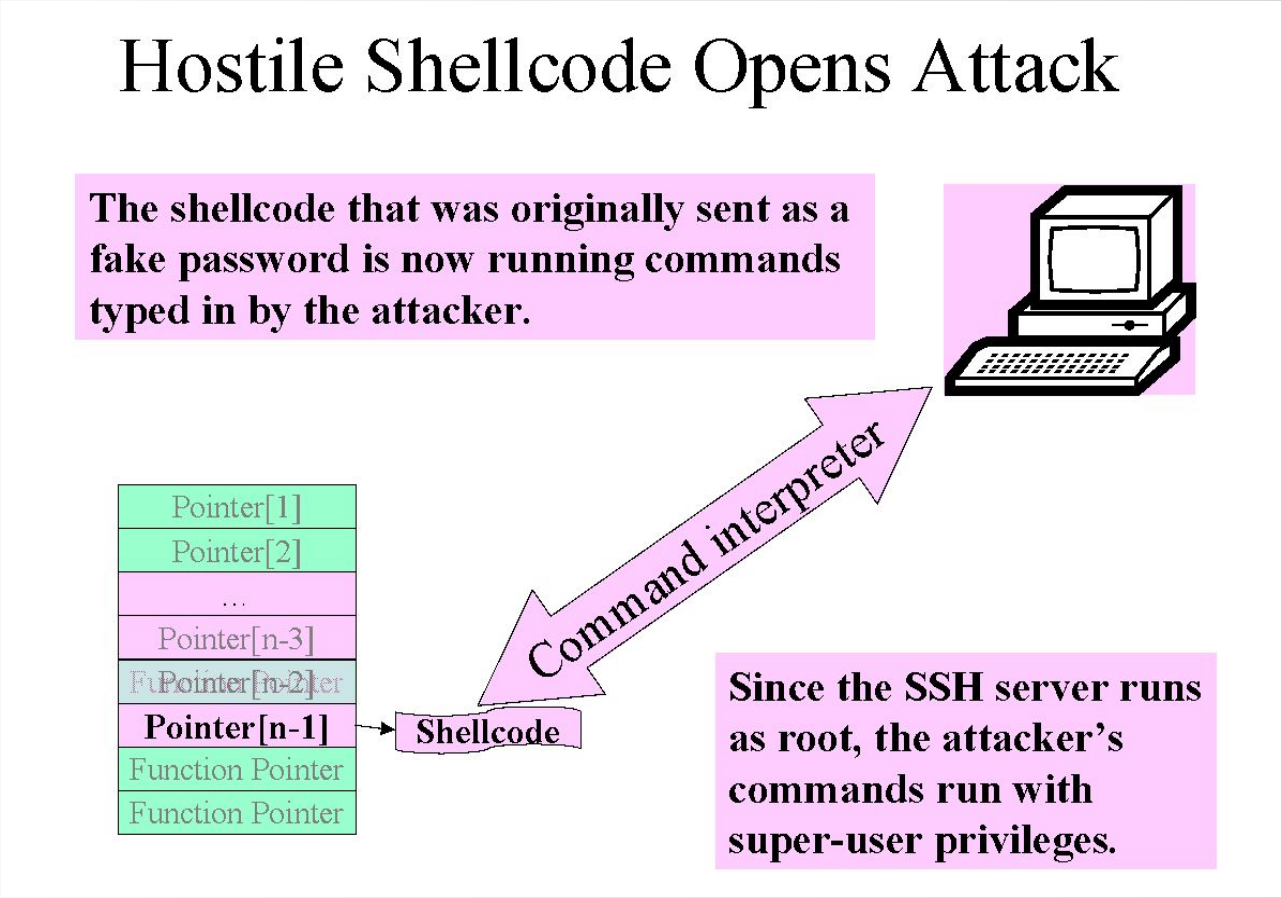
Case study : The 'GOBBLES' exploit (contd.)



Case study : The 'GOBBLES' exploit (contd.)



Case study : The 'GOBBLES' exploit (contd.)



SSH tools

- There are several commercial and freeware SSH tools for UNIX and windows platforms
- Windows –
  - PuTTY
  - TeraTerm
  - SecureCRT
- UNIX –
  - The RedHat Linux package and OpenBSD come alongwith tools such as ssh, sshd(daemon), scp(secure copy), sftp(secure ftp), etc.
  - OpenSSH

Bibiliography

[1] ‘Securing remote connections with SSH’ – a white paper by SSH communications Security corp.

[2] Red Hat tutorials – Chapter 10 SSH : http://www.redhat.com/docs/manuals/linux/RHL-7.3-Manual/ref-guide/ch-ssh.html

[3] SSH tools - http://bach.dynet.com/sshtools/

[4] ’Firewalling a Secure Shell Service’ – a white paper http://www.esat.kuleuven.ac.be/~joclaess/pub/ceci2001.pdf

[5] Cisco Security Advisory: Multiple SSH Vulnerabilities –http://www.cisco.com/warp/public/707/SSH-multiple- pub.html

[6] SSH -The Secure Shell :The definitive guide Daniel Barrett & Richard E. Silvermann (O’Reilly publications)

[7] ‘Analysis of a Secure Shell vulnerability’ - James M. Mike Anthis A white paper