Systems Security

Operating Systems

Baochun Li
Department of Electrical and Computer Engineering
University of Toronto

Real-world protocols using the ideas so far

Message Authentication Code (MAC)

Message Authentication Code

- MAC — Message Authentication Code
  - Designed for checking **integrity** and **authentication**
  - Based on the assumption that a shared secret key has already been established

MAC: the basic idea

HMAC — Hash based MAC

- Using two padding sequences — outer padding is `5C5C5C...` (in hexadecimal) and inner padding is `363636...`

- Use the XOR operation to add padding before applying the secure hash function (SHA)

- Apply the secure hash function twice

key

i_pad

XOR

i key pad

64 Byte

<= 64 Byte

key

o_pad

XOR

o key pad

64 Byte

<= 64 Byte

i key pad

message

SHA1 - 1st pass

hash sum 1

o key pad

hash sum 1

SHA1 - 2nd pass

hash sum 2

64 Byte

20 Byte

HMAC: pseudocode

```
function hmac (key, message)
     if (length(key) > blocksize) then
          key = hash(key) // keys too long are
shortened
     if (length(key) < blocksize) then
          key = key || [0x00 * (blocksize -
length(key))] // keys too short are zero-padded
     o_key_pad = [0x5c * blocksize] XOR key
     i_key_pad = [0x36 * blocksize] XOR key
          return hash(o_key_pad || hash(i_key_pad
|| message))
end function
```

Transport Layer Security (TLS)

Transport Layer Security (TLS)

- The older generations of the protocol are called the **Secure Socket Layer (SSL)**

- The foundation of the secure HTTP protocol, secure electronic mail, and most secure protocols in the Internet — the foundation of e-commerce!

- The most up-to-date is TLS 1.2, used in most modern web browsers

  - Considered secure against all known attacks

The TLS Handshake Protocol

- A client sends **ClientHello**
  - Highest protocol version of TLS supported
  - A random number

The TLS Handshake Protocol

- The server sends **ServerHello**
  - Chosen protocol version
  - A random number

The TLS Handshake Protocol

- The server sends **Certificate**
- The server sends **ServerHelloDone**
- The client sends **ClientKeyExchange**
  - with a **PreMasterSecret**, encrypted using the public key within the server certificate
- Both server and client generate the secret key

TLS Handshake Protocol

- The client sends **ChangeCipherSpec**
  - “Everything I tell you from now on will be authenticated and encrypted.”
- The client sends **Finished,** which is authenticated and encrypted

TLS Handshake Protocol

- The server sends **ChangeCipherSpec**
  - “Everything I tell you from now on will be authenticated and encrypted”
- The server sends **Finished,** which is authenticated and encrypted

The choice of MAC

- After the session is established using the TLS handshake protocol, HMAC with **SHA-256** is used for MAC

User authentication in OS

Authentication with passwords

- Something that a user **knows** — extremely common
- Problems are plenty —
  - More passwords are better, but a user won't like it
  - How easy it is to guess a password?
  - How easy it is to obtain a password without guessing?
    - sniffing and phishing are common techniques

Guessing passwords

LBL> telnet elxsi
ELXSI AT LBL
LOGIN: root
PASSWORD: root
INCORRECT PASSWORD, TRY AGAIN
LOGIN: guest
PASSWORD: guest
INCORRECT PASSWORD, TRY AGAIN
LOGIN: uucp
PASSWORD: uucp
WELCOME TO THE ELXSI COMPUTER AT LBL

Storing passwords

- The system must store passwords in order to perform authentication
- How can passwords be protected?
  - Rely on file protection and store them in protected files
    - compare typed password with stored password
  - Rely on encryption
    - store them encrypted — **in readable files?**
    - use one way function (cryptographic hash)

Example: PHP website

- A website must store **sensitive** information, such as user passwords, in encrypted form

```
function get_password_hash($password)
{
        global $dbc;

        return mysqli_real_escape_string
         ($dbc, hash_hmac('sha256', $password,
          'c#haRl891', true));
}
```

Storing passwords in Unix

- Password file: **/etc/passwd**
  - It's a world readable file!
- **/etc/passwd** entries
  - User name, password (encrypted), user id, group id, home directory, shell preference, name

Dictionary attacks

- If encrypted passwords are stored in world readable files and you see that another user's encrypted password is the same as yours
  - Their password is also the same!
- If the encryption method is well known, attackers can —
  - Encrypt an entire dictionary
  - Compare encrypted dictionary words with encrypted passwords until they find a match

Salting passwords (Morris and Thompson, 1979)

- The salt is a random number combined with the password prior to encryption
  - It changes when the password is changed
  - The salt is stored with the encrypted password
- Different user's with the same password see different encrypted values in **/etc/passwd**
- Dictionary attack requires time-consuming re-encoding of entire dictionary for every salt value

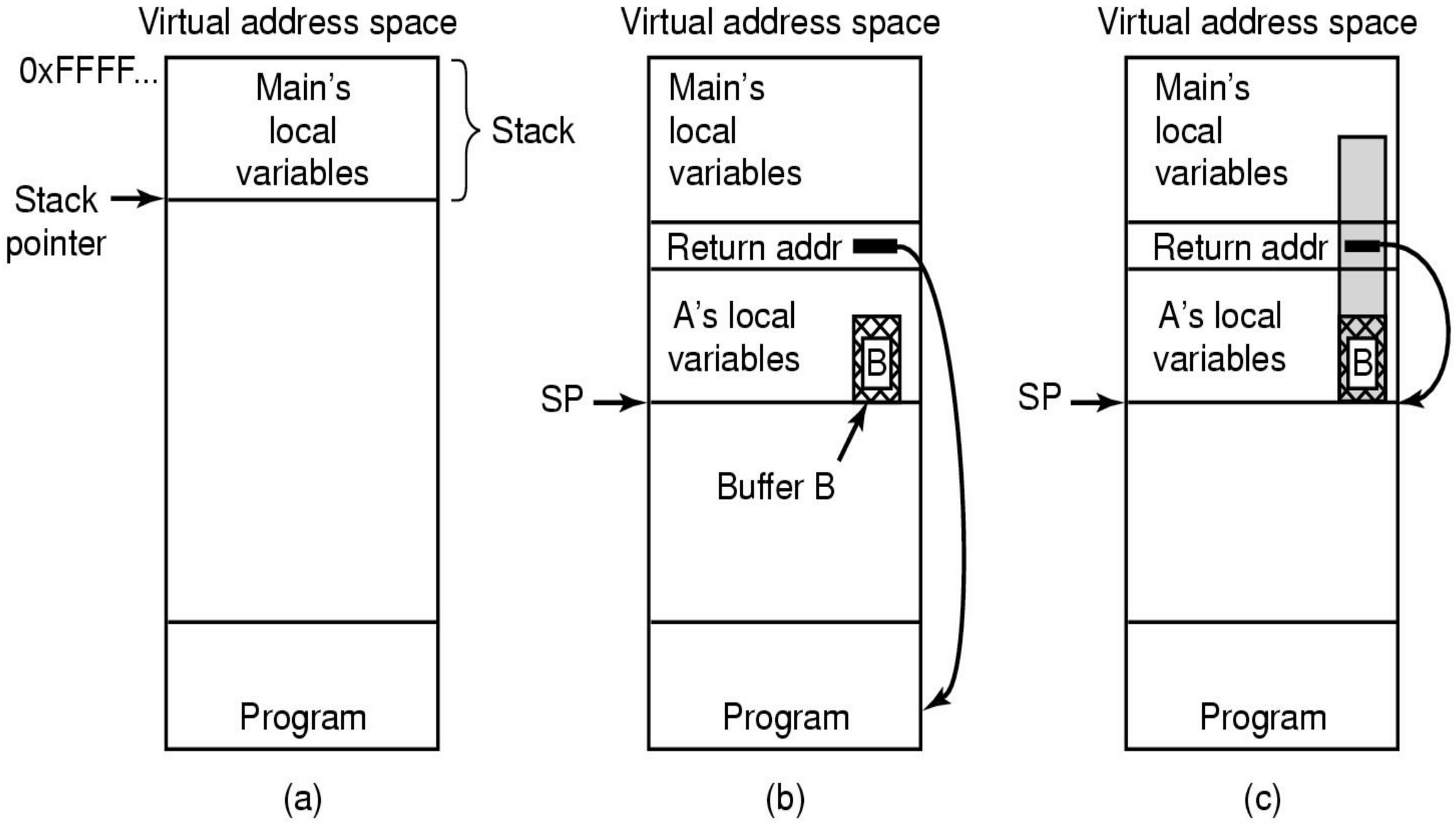
Challenge-Response Authentication

- Simple: Ask a list of questions that only the authentic user knows the answers to
- More complex — used by SSH
  - The user generates a public-private key pair
  - The public key is stored on the remote server
  - The private key is stored in main memory, or generated on-the-fly when the user is prompted a “passphrase”
  - The private key is used to authenticate the user with the public key on the server

Best practice: **two-factor authentication**

Common attacks

Buffer overflow attacks



Example C program

```
int main(int argc, char *argv[])
{
  char buffer[256];
  if (argc < 2) return -1;
  else {
    // correct version is
    // strncpy(buffer, argv[1], 255);
    strcpy(buffer, argv[1]);
    return 0;
  }
}
```

Defeating Buffer Overflow Attacks

- Fixing buffer overflow bugs one at a time
- Hardware solutions
  - CPU includes the ability to mark a page non-executable
    - NX feature in AMD and Intel x86 CPUs
    - Linux and Windows XP SP2 support the feature

That’s it for this course

Thank you