Approfondimenti sulla Sicurezza del Trasporto Chiavi RSA

Trascrizione da Presentazione

1 Trasporto Chiavi RSA e Attacchi (CCA)

Il meccanismo base del trasporto chiavi RSA (usato in TLS fino a TLS 1.2) prevede che il Client crittografi un segreto (il *Pre-Master Secret*) usando la chiave pubblica RSA del Server.

Il server riceve il ciphertext C e usa la sua chiave privata per decrittarlo.

$$C = M^e \pmod{n}$$

L'obiettivo dell'attaccante è trovare M (il segreto) partendo da C.

$$M = C^d \pmod{n}$$

1.1 Attacco con Oracolo di Decrittazione (Chosen Ciphertext Attack)

Assumiamo che l'attaccante abbia accesso a un **oracolo di decrittazione**: un sistema (il server stesso) che decritta dati per lui. Ovviamente, l'oracolo non accetterà di decrittare il ciphertext originale C, altrimenti l'attacco sarebbe banale.

Attacco (Malleabilità di RSA):

1. L'attaccante sceglie un valore casuale r.
2. Costruisce un NUOVO ciphertext X "mascherando" C:
$$X = (r^e \cdot C) \pmod{n}$$
3. L'attaccante invia X all'oracolo, che appare come un messaggio innocuo.
4. L'oracolo (il server) calcola la decrittazione di X:
$$X^d = (r^e \cdot C)^d = (r^e)^d \cdot C^d = r^{ed} \cdot C^d = r \cdot C^d \pmod{n}$$
(Dato che $ed = 1 \pmod{\phi(n)}$, $r^{ed} = r$) L'oracolo restituisce $r \cdot M$.
5. L'attaccante, che conosce r, riceve $r \cdot M$, calcola l'inverso moltiplicativo $r^{-1} \pmod{n}$ e annulla r per trovare M:
$$(r \cdot M) \cdot r^{-1} = M \pmod{n}$$

Questo dimostra che l'RSA "da manuale" (textbook RSA) non è robusto contro Attacchi Basati su Testi Cifrati Scelti (Chosen Ciphertext Attacks - CCA). Il problema è la **malleabilità** di RSA.

Spiegazione Didattica

Malleabilità e Omomorfismo Moltiplicativo
Il problema evidenziato è una proprietà intrinseca dell'RSA "da manuale", noto come **omomorfismo moltiplicativo**.
Questa proprietà afferma che:

$$Enc(M_1) \cdot Enc(M_2) = (M_1^e) \cdot (M_2^e) = (M_1 \cdot M_2)^e = Enc(M_1 \cdot M_2)$$

L'attacco sfrutta esattamente questo. L'attaccante calcola r^e (che è $Enc(r)$) e lo moltiplica per C (che è $Enc(M)$), ottenendo $Enc(r \cdot M)$.
Quando l'oracolo decritta questo nuovo ciphertext, restituisce $r \cdot M$. L'attaccante ha così ottenuto un messaggio (M) modificato in un modo che lui può prevedere e invertire.
Per mitigare questo problema, si introduce il **padding**.

2 RSA Padding: PKCS #1 v1.5

Per risolvere il problema della malleabilità, non si crittografa mai M direttamente, ma un blocco formattato (padded). Lo standard storico è il PKCS #1 v1.5 (Public-Key Cryptography Standards).

Struttura del padding (per crittografia): Il messaggio M viene formattato in un blocco EB (Encryption Block) della stessa lunghezza k del modulo RSA n.

$EB = 00 \mid 02 \mid PS \mid 00 \mid D$

- **00 02:** Due byte fissi. Il '02' indica il modo di padding per la crittografia.
- **PS (Padding String):** Una stringa di almeno 8 byte **casuali** e **non-zero** (diversi da '00'). Questo serve a rendere la crittografia non deterministica (evitando attacchi CPA).
- **00:** Un byte separatore fisso.
- **D (Data):** I dati effettivi (il Pre-Master Secret).

Se k è la dimensione del modulo in byte (es. $k = 128$ per RSA-1024), la dimensione massima dei dati D è $k - 11$ byte (2 byte per '00 02', 8 byte minimi per 'PS', 1 byte per '00'). Per RSA-1024 (128 byte), si possono crittografare al massimo $128 - 11 = 117$ byte.

3 L'Oracolo di Padding (Bleichenbacher)

Come veniva gestito questo padding in SSL v3.0?

1. Il Client invia 'RSA-PKCS1-v1.5(Pre-Master secret)'.
2. Il Server riceve, decritta con la sua chiave privata.
3. Il Server controlla il padding:
   - Il blocco decifrato inizia con '00 02'?
   - C'è un '00' che separa 'PS' e 'D'?
4. **Se il padding NON è valido:** Il server invia un messaggio di 'abort'.
5. **Se il padding È valido:** Il server prosegue con il setup della sessione.

OOOOPPPPSSS!!! Questo comportamento crea un **Oracolo di Padding**!

Spiegazione Didattica

Cos'è un Oracolo di Padding?
Un Oracolo di Padding è un tipo di vulnerabilità (un "side channel") che si verifica quando un server rivela un'informazione, apparentemente innocua, sul fatto che il padding di un messaggio decifrato sia valido o meno.
L'attaccante non ottiene il plaintext decifrato, ma solo una risposta binaria: "Padding OK" (il server prosegue) o "Padding Errato" (il server invia un 'abort').
Questo singolo bit di informazione (Sì/No), se interrogato un numero sufficiente di volte con ciphertext appositamente costruiti (manipolati), è sufficiente per decifrare completamente il messaggio originale.

3.1 L'Attacco di Bleichenbacher (1998)

Daniel Bleichenbacher scoprì (1998) questo attacco CCA adattivo contro RSA PKCS #1 v1.5, che colpiva SSL v3.0.

Come funziona (concettualmente):

1. L'attaccante ha il ciphertext originale $C = M^e$.
2. Sceglie un valore r e costruisce $C' = C \cdot r^e \pmod{n}$ (come nell'attacco di malleabilità).
3. Invia C' al server SSLv3.0.
4. Il server decritta C' e ottiene $M' = M \cdot r \pmod{n}$.
5. L'oracolo (il server) risponde 'abort' o 'prosegui', dicendo all'attaccante se $M \cdot r \pmod{n}$ **inizia con i byte 00 02** o no.

Sapere che $M \cdot r$ ricade nell'intervallo dei messaggi che iniziano con '00 02' (circa $1/2^{16}$ dello spazio totale) **fornisce informazioni su M!**

Bleichenbacher ha dimostrato che, scegliendo r in modo adattivo (basandosi sui risultati precedenti) e con un numero sufficiente di tentativi (circa 2^{20} per RSA-1024), è possibile restringere l'intervallo dei possibili valori di M fino a trovarlo.

Spiegazione Didattica

Come un Oracolo Binario può Rivelare Tutto
Immaginiamo un oracolo giocattolo molto più semplice: l'oracolo ci dice solo se il plaintext decifrato M' è **pari o dispari** (un oracolo di parità).
Se inviamo $C' = C \cdot 2^e$ (cioè $Enc(M \cdot 2)$), l'oracolo decrifterà $M \cdot 2$ (che è sempre pari) e ci dirà "Pari". Questo non è utile.
Ma se l'attaccante conduce una **ricerca binaria** (binary search), può restringere l'intervallo di M. L'attacco di Bleichenbacher è concettualmente simile, ma matematicamente molto più complesso perché l'intervallo "valido" (quello che inizia con '00 02') non è semplice come "pari" o "dispari".
Il punto chiave è: **appena un singolo bit di informazione sul plaintext viene rivelato** (anche solo la validità del padding), l'intera sicurezza del messaggio è a rischio.

4 La Saga di Bleichenbacher: "In the Wild"

L'attacco fu scoperto nel 1998 e corretto in TLS 1.0. Ma la vulnerabilità non è scomparsa. È una "saga" che ha generato innumerevoli varianti (la "Pandora's Box" dei side channels):

- Klima et al. (CHES 2003)
- XML Encryption (ESORICS 2012)
- Degabriele et al. (CT-RSA 2012)
- Bardou et al. (CRYPTO 2012)
- Attacchi Side-Channel Cross-Tenant in PaaS Clouds (CCS 2014)
- Attacchi Side Channel a implementazioni TLS (USENIX 2014)
- Attacco a QUIC (CCS 2015)
- **DROWN (USENIX 2016)**
- **ROBOT (USENIX 2018)**
- Attacchi a IPsec IKE (USENIX 2018)
- **CAT (Cache Attacks) (IEEE S&P 2019)**

Nel 2019, Adi Shamir (l'inventore di RSA) è arrivato a raccomandare la deprecazione di RSA in TLS.

4.1 Esempi Recenti

DROWN (2016): Un attacco devastante che utilizzava un oracolo Bleichenbacher su server che supportavano ancora **SSLv2** (un protocollo obsoleto e rotto). Se un server usava la stessa chiave privata RSA sia per TLS (sicuro) che per SSLv2 (vulnerabile), un attaccante poteva usare l'oracolo SSLv2 per attaccare e decifrare connessioni TLS moderne. Nel 2016, questo affliggeva circa 1/3 dei siti HTTPS, inclusi i principali.

ROBOT (2018): "Return Of Bleichenbacher's Oracle Threat". Una scansione su larga scala ha rivelato che, a causa di implementazioni errate e complesse, moltissimi server (Facebook, Cisco, IBM, Palo Alto Networks, etc.) erano ancora vulnerabili. La vulnerabilità non risiedeva nel protocollo TLS, ma nelle *implementazioni* che rispondevano in modo diverso (es. inviando alert TLS diversi, o con tempistiche diverse) a messaggi con padding valido e non valido, ricreando di fatto l'oracolo.

IPSEC IKE (2018): Vulnerabilità simili trovate in implementazioni VPN (Cisco, Huawei, ZyXEL), combinando l'attacco con un downgrade a IKEv1.0.

5 Contromisure e Lezioni Apprese

5.1 Contromisure

1. **Usare un padding sicuro (es. RSA-OAEP):** Questa è la soluzione crittografica corretta (OAEP è provabilmente sicuro contro CCA). Tuttavia, fu considerata un cambiamento troppo radicale per essere implementato come "patch".
2. **Implementazione attenta (Soluzione "Patch"):** La correzione in TLS 1.0 prevede che, se il padding è malformato, il server **debba comportarsi ESATTAMENTE come se fosse valido**. Genera un PMS casuale falso, completa l'handshake e fallisce solo in seguito (al momento del controllo del MAC).

   Problema: Questo è *incredibilmente difficile* da implementare correttamente. Come ha dimostrato l'attacco ROBOT, minime differenze nel comportamento (side channels) ricreano l'oracolo.
3. **Sbarazzarsi di RSA Key Transport (Soluzione Definitiva):** Questo è ciò che ha fatto **TLS 1.3**. TLS 1.3 *elimina* completamente la modalità "RSA Key Transport".

Spiegazione Didattica

TLS 1.3 e Forward Secrecy
TLS 1.3 non usa più RSA per *crittografare* un segreto, ma solo per *autenticare* (firmare).

- **RSA Key Transport (Obsoleto):** Il Client crea un segreto (PMS) e lo *crittografa* con la chiave pubblica RSA del server. *Svantaggio:* Se un attaccante registra tutto il traffico oggi e ruba la chiave privata del server tra 5 anni, può decrittare tutte le sessioni passate. (Nessuna *Forward Secrecy*).
- **(EC)DHE (Moderno, TLS 1.3):** Client e Server negoziano un segreto usando un protocollo di scambio effimero (Diffie-Hellman). Usano RSA solo per *firmare* i messaggi di questo scambio, per provare la loro identità. *Vantaggio:* Il segreto di sessione non viene mai trasmesso e non dipende dalla chiave privata RSA del server. Rubare la chiave privata del server non permette di decrittare sessioni passate. (Garantisce *Forward Secrecy*).

Eliminando il RSA Key Transport, TLS 1.3 ha eliminato alla radice l'intera classe di attacchi Bleichenbacher.

5.2 Lezioni Apprese (Take-home messages)

- L'implementazione della crittografia è estremamente complessa e subdola.
- Gli errori crittografici (come il padding in SSL) sono difficili da "patchare" in modo incrementale e tendono a riemergere per decenni. (Ricorda la saga "MAC-then-Encrypt", anch'essa eliminata in TLS 1.3).
- Spesso, nemmeno gli esperti di sicurezza comprendono appieno le implicazioni degli attacchi crittografici. La slide finale ne è un esempio ironico:

"Ehi, il vostro server è vulnerabile all'attacco di Bleichenbacher!"

"Nessun problema. Usiamo una crittografia di livello militare."