SSE - Passive SSH Key Compromise via Lattices - Ferrara Justin

Signatures digitales

Dans ce document, je traiterais uniquement des signatures RSA PKCS#1 V1.5. Ces signatures qui sont pourtant vielles et dépréciées (Duc Alexandre, 2024) sont encore relativement répandue. Selon l'article, cela représente encore environ un tiers des signatures dans le cadre de la collecte d'informations effectuée (Keegan Ryan; Kaiwen He; George Arnold Sullivan; Nadia Heninger, 2023).

Erreurs dans les signatures digitales

Si une erreur de calcul apparait dans un signature digitale cela peut avoir des conséquences graves sur la sécurité de la construction, allant notamment jusqu'à la divulgation de la clé privée.

Les erreurs pouvant arriver lors du calcul d'une signature découler de plusieurs facteurs. Une mauvaise implémentation, une erreur de calcul provoquée par une side chanel attack, etc. Selon l'article, des routeurs ou firewall des marques comme Zyxel, SSHD, Mocana ou Cisco peuvent générer des signatures avec des erreurs (Keegan Ryan; Kaiwen He; George Arnold Sullivan; Nadia Heninger, 2023). Cependant, la plupart des erreurs on été corrigées après mise à jour du software.

Prérequis pour que l'attaque réussisse

- Posséder une signature invalide d'un message (posséder le message n'est pas un nécessaire)
- Posséder la clé publique correspondante
- L'algorithe étudié ici concerne les signature RSA PKCS#1 V1.5
- Les signatures sont réalisées en utilisant le théorème des restes chinois
- L'erreur dans la signature provient d'une des opérations faite dans le monde des tuples (théorème des restes chinois)

> Pour faire cette attaque sur TLS, il faut faire une écoute passive sur TLS 1.2 et une attaque active sur TLS 1.3. ??? changer position

SSH

Authentification et échange de clés

Authentification du serveur

Les serveurs SSH sont identifiés par leurs clés publiques. L'échange commence par une négociation des différents algorithmes utilisés par la suite suivie d'un échange de clés Diffie-Hellman. Le serveur s'authentifie en signant le session identifier avec sa clé privée, vérifié ensuite par le client. Le session identifier contient le D-H, les messages échangés pour les algorithmes, l'ID du client et l'ID du serveur.

Authentification du client

Elle se déroule après l'établissement du canal chiffré. Il y a deux méthodes possibles, par mot de passe ou par clé publique. Pour l'authentification par mot de passe, le mot de passe est envoyé en clair dans le canal chiffré. Pour l'authentification par clé publique, le client signe un identifiant de session avec sa clé privée.

Algorithmes cryptographiques

Pour l'échange des clés, différents algorithme sont à choix notamment Diffie-Hellman, Elliptic Curve Diffie-Hellman (ECDH), RSA.
Pour les signatures, différents algorithme sont à choix notamment DSA, RSA, ECDSA, Ed25519.

> OpenSSH 8.8 (septembre 2021) désactive par défaut `ssh-rsa` (SHA-1) mais supporte `rsa-sha2-256` et `rsa-sha2-512`. ?????

Sécurité et attaques possibles

Compromission des clés de signatures

Une compromission des clés de signatures ne permet pas de déchiffrer les connexions de manière passives mais peut permettre une attaque active (Man-in-the-middle). Elle peut permettre également à un attaquant d'usurper l'identité du serveur et établir une connexion chiffrée avec le client pour obtenir son mot de passe par exemple.

Exploitation des méthodes d'authentification

Lors d'une authentification par mot de passe, un attaquant peut effectuer un Man-in-the-middle en usurpant l'identité du serveur pour intercepter et relayer les identifiants et mot de passe du client. Lors d'une authentification par clé publique, cela n'est pas possible. L'attaquant peut cependant connaitre les commandes que le client aurait envoyé au serveur.

SSH Agent Forwarding ??? todo remove ???

- Permet de transmettre l'authentification via un agent SSH distant.
- Peut être exploité par un attaquant pour se connecter à d'autres serveurs avec les clés de l'agent.
- OpenSSH 8.9 (février 2022) introduit des restrictions pour limiter ce risque, mais leur adoption reste limitée.

IPsec

IPsec est un ensemble de protocoles (RFC 2408, 2409, 7296) visant à garantir la confidentialité, l'intégrité des données et l'authentification des sources des paquets IP. Il est beaucoup utilisé par les VPN et repose sur le protocole Internet Key Exchange (IKE) pour négocier les algorithmes utilisés, définir comment dériver les clés, etc.

IKE : Versions et Fonctionnement

IKE existe en deux versions : IKEv1 et IKEv2. IKE permet l'établissement d'une Security Association (SA) et l'authentification mutuelle. Une SA permet le choix des algorithmes et l'échange initial de clés Diffie-Hellman. L'authentification mutuelle est effectuée entre l'initiateur et le répondeur via différentes méthodes (signatures, clés pré-partagées, etc.).

IKEv1

IKEv1 prend en charge trois modes d'authentification, les signatures numériques, le chiffrement par clé publique ou des clés pré-partagée (PSK). Il y a également deux modes de communication, le Main Mode qui est le plus sécurisé, toutes les communications sont chiffrées après l'échange initial et le mode agressif, où l'échange initial réduit mais moins sécurisé (les signatures sont envoyées en clair).

Sécurité et attaques possibles

Avec le Main Mode, une attaque passive seule n'est pas possible car elle ne permet pas de collecter des signatures du fait qu'elles sont transmises de manières chiffrées. Pour recevoir des signatures, un adversaire doit se connecter de manière active au serveur.

Avec le Mode Agressif, un attaquant peut capturer les signatures envoyées en clair par le client et le serveur en écoutant passivement la communication.

À noter que un format non standard de signature RSA est utilisé. RFC 2409 impose une variation du format PKCS#1 v1.5, supprimant l'OID du hachage car il est inclus dans la SA, ce qui diminue la partie connue de la signature et augmente de manière négligeable la complexité de l'attaque.

En cas de compromission des clés de signature, un attaquant obtenant la clé privée de signature pourrait usurper cette identité. Pour pouvoir faire une attaque Man-in-the-middle, il faut compromettre la clé privée de signature des deux parties à cause du D-H qui est signé.

IKEv2

IKEv2 n'est pas compatible avec IKEv1 et introduis certaines modifications notamment le fait que toutes les signatures sont chiffrées et transmises dans l'AUTH payload après l'établissement de la SA. L'extensible Authentication Protocol (EAP) permet d'obtenir de manière active une signature de l'host distant sans authentifier l'initiator de la connection. L'authentification par clés cryptographique peut ne concerner qu'un participant en fonction des modes utilisés.

Sécurité et attaques possibles

En cas d'une compromission d'une clés de signature, un attaquant peut usurper cette identité. Pour pouvoir faire une attaque Man-in-the-middle, il faut compromettre la clé privée des deux parties.

En cas d'utilisation de clé pré-partagée (PSK) pour l'authentification, un attaquant peut, si la PSK est faible (mot de passe faible), mener une attaque par dictionnaire hors ligne. Et par la suite, avec la clé privé de signature de l'autre partie, faire un Man-in-the-middle complet.

Certains modes EAP (comme **EAP-MS-CHAPv2**) permettent une attaque Man-in-the-middle complète, en attaquant le hash du mot de passe trouvé par une attaque hors ligne. ??? attention différent du dessus ???

PKCS#1 V1.5 padding pour signature RSA

> Note : toutes les formules mathématiques suivantes sont dérivées de l'article Keegan Ryan, Kaiwen He, George Arnold Sullivan, Nadia Heninger, 2023.

Le padding PKCS#1 V1.5 utilisé par les signatures est donné par :

$$00||01||FF...FF||ASN.1||Hash(m)$$

Où :

- `ASN.1` est un identifiant pour définir la fonction de hachage utilisée
- `Hash(m)` est le hash du message `m`

Expression du padding

Pour le padding d'un message `m` inconnu, le padding ISO/IEC 9796-2 peut être exprimé de la manière suivante :

$$a + b * x + c * y$$

Avec `a`, `b` et `c` connu et `x` et `y` inconnu. Si une erreur de calcul survient `mod q` et mais que `mod p` est effectué correctement, nous pouvons en déduire l'équation suivante :

$$s'^e = a + b * x + c * x \mod p$$

Pour le padding PKCS#1 V1.5 d'un message, l'équation originale peut être simplifiée de la manière suivante :

$$a + x$$

$$x < 2^{hash\_len}$$

Avec `a` la partie connue du padding et `x` representant la partie inconnue du message, bornée par la taille de sortie de la fonction de hachage. En posant cette équation, il est possible de retrouver la clé privée de signature si `hash_len <= log(N) / 4`.

RSA Signatures avec padding PKCS#1 V1.5

La signature `s` RSA d'un message `m` sans padding est donnée par :

$$s = f(m)^d \mod N$$

La vérification de cette signature est donnée par :

$$f(m') = s^e \mod N$$

$$f(m) =? f(m')$$

Avec `f(m)` la fonction de de padding du message.

> À noter que dans ce cas la signature doit être faite en utilisant le théorème des restes chinois.

PACD (Partial Approximate Common Divisors)

PACD est une généralisation du PGDC :

$$N_i = p \cdot q_i$$

Avec `p` un facteur inconnu de longueur `log p`.

Pour pouvoir résoudre le problème PACD, il faut que les diviseurs communs approximatifs soient suffisamment proches pour que l'algorithme LLL puisse les trouver. Nous pouvons donc essayer de poser les équations suivantes :

$$N_0 = p_0 \cdot q_0 + r_0$$

$$N_1 = p_1 \cdot q_1 + r_1$$

Avec `r_1` une erreur de calcul et `r` l'espace de la fonction de hachage.

$$|r_1| < 2^{log_2(r)}$$

Comme nous n'avons pas `N1`, nous pouvons essayer de le trouver en utilisant les équations qui définissent la padding PKCS#1 V1.5.
???

$$s'^e = m' = a + x \mod p$$
$$s'^e \neq a + x \mod q$$
$$s'^e = k * p + a + x \mod N$$
$$(s'^e - a \mod N) = k * p + x \mod N$$
$$x < 2^r$$

Nous pouvons donc déduire les équations suivantes :

$$N_0 = p \cdot q_0$$
$$N_1 = (s'^e - a \mod N) = k \cdot p + h$$

Avec `h` une valeur inconnue et `a` la partie connue du padding PKCS#1 V1.5.
???? pourquoi pas mod n aussi sur s^e ????? todo

Lattice

Nous pouvons ensuite remplir la matrice suivante :

$$B = \begin{bmatrix} -2^{2\log r} & 2^{\log r} N_1 & 0 \\ 0 & -2^{\log r} & N_1 \\ 0 & 0 & N_0 \end{bmatrix}$$

Avec `r` l'espace de la fonction de hachage.

LLL

Pour réduire la matrice précédente, nous pouvons utiliser l'algorithme LLL pour essayer de trouver des vecteurs avec une base plus courte et plus proche de l'orthogonalité que la base d'entrée:

- appliquer la réduction de Gram-Schmidt pour décomposer chaque vecteur
- modifier un peu les vecteurs en les ajustant, en remplaçant certains vecteurs par des combinaisons linéaires des autres vecteurs
- vérifie la condition de Lovász pour chaque vecteur

À la fin de l'algorithme, on obtient une base plus courte et plus proche de l'orthogonalité que la base d'entrée.

$$reduced\_matrix = LLL(B)$$
$$reduced\_matrix = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix}$$

Si la réduction a fonctionné, nous pouvons retrouver un vecteur `v` qui peut être interprété comme les coefficients du polynôme suivant :

$$\vec{\mathrm{v}}$$
$$g(2^{\log r} x)$$
$$g(x) = a_1 * \frac{x^2}{2^{2^{log(r)}}} + b_1 * \frac{x}{2^{log(r)}} + c_1$$

Si les coefficients sont assez petits, on peut poser les équations suivantes : ???

$$|g(y)| < p^k$$
$$|y| \leq 2^{\log r}$$
$$g(y) = 0$$
$$y = r_1$$

Avec `r_1` une petite racine mod p.

Comme nous connaissons `r_1`, nous pouvons donc déduire `p` :

$$p = \gcd(N_0, N_1 - r_1)$$

> Note : LLL trouvera une solutions seulement si `hash_len * 4 <= log(N)`.

Digression de l'attaque

L'attaque est possible si dans le message représentant le message original, il y a une partie connue et une partie inconnue représentant le message original avec une erreur dans la signature. ???

Je peux donc déjà exclure les algorithmes suivants:

- Tous les algorithmes de signatures basé sur les courbes elliptiques
- RSA PSS

Mesures de protections

- **Validation des signatures** : Il faut vérifier que la signature effectuée ne contient pas d'erreur avant de l'envoyer et dans le cas contraire, en refaire une nouvelle.
- Ne pas utiliser un padding déterministe et donc utiliser un algorithme de signature sûr, par exemple RSA-PSS.
- **RSA dans SSH** : Éviter d'utiliser des versions vulnérables ou faibles du padding PKCS#1 V1.5 avec RSA (SHA-1).
- **Design du protocol** :
  - Chiffrer la communication le plus tôt possible dès que les clés cryptographiques sont disponibles pour protéger les metadata
  - Se baser sur TLS 1.3 pour plus de détail

Bibliography

- **Keegan Ryan, Kaiwen He, George Arnold Sullivan, Nadia Heninger**. *Passive SSH Key Compromise via Lattices*. Cryptology ePrint Archive, Paper 2023/1711, 2023. DOI: 10.1145/3576915.3616629, URL: https://eprint.iacr.org/2023/1711.
- Duc, Alexandre, 2024. *Asymmetric Cryptography Standards* [PDF]. Support de cours : Cryptographie avancée appliquée, HEIG-VD, 2024.