Ingénierie inverse du code exécutable ELF dans l'analyse après intrusion

Marek Janiczek

L'ingénierie inverse, associée souvent à l'enlèvement des protections des programmes, peut être utilisée avec succès dans l'analyse après intrusion. Grâce à cette méthode, nous obtenons la réponse aux questions concernant le rôle, les possibilités et le mécanisme du fonctionnement des fichiers suspects trouvés dans le système.

Il y a un mois, pendant l'analyse après intrusion d'un système compromis, j'ai trouvé dans le répertoire */usr/share/doc/shellutils* le programme appelé *kstatd* laissé par l'intrus. L'analyse après intrusion standard, basée sur l'analyse des informations laissées dans la mémoire et sur le disque dur du système compromis, n'a pas permis de déterminer le rôle joué par le fichier suspect. Je n'ai pas eu la possibilité de reconstruire le code source – soit le programme a été compilé sur la machine compromise, soit il a été introduit sous forme compilée. Dans cette situation, l'unique solution était d'effectuer l'analyse détaillée du code au moyen de l'ingénierie inverse (en anglais *reverse engineering*).

Les actions que j'ai effectuées peuvent être appelées analyse statique – le code analysé n'a jamais été lancé. Les résultats ressortent à partir de l'analyse de la structure et du contenu de l'objet analysé.

Outils nécessaires

Comme plate-forme pour l'analyse, j'ai utilisé le système Mandrake Linux 10.0. Les outils utilisés sont les programmes disponibles par défaut sur les systèmes Linux et la famille BSD et autres disponibles dans Internet. Les outils système standard fournis dans les distributions, c'est avant tout le paquet *binutils* (*GNU binary utilities*). De ce paquet, nous utiliserons les programmes suivants :

- *ar* – pour obtenir les informations sur les objets des bibliothèques (des archives **.a*) et à les extraire,
- *nm* – pour obtenir la liste des liens symboliques (symboles) dans les objets,
- *objdump* – permet d'obtenir des informations détaillées sur l'objet et son contenu,
- *strings* – pour l'extraction des chaînes de caractères imprimables ASCII du fichier.

Cet article explique ...

- comment désassembler le code exécutable ELF,
- comment se servir d'ingénierie inverse pendant l'analyse après intrusion dans Linux.

Ce qu'il faut savoir ...

- les notions de base de programmation en C et en Assembleur.

Format ELF

ELF (*Executable and Linking Format*) est un format typique de trois types d'objets binaires dans Linux : réalloués, exécutables et partagés.

- Les objets réalloués (à extension **.o*) sont les objets qui sont associés pour créer un fichier exécutable ou une bibliothèque partagée – ce sont les fichiers résultants des compilateurs et assembleurs.
- Les objets exécutables sont les fichiers prêts à être lancés, déjà réalloués, avec les liens symboliques résolus (excepté ceux qui sont relatifs aux bibliothèques partagées, résolues pendant le démarrage).
- Les objets partagés (à extension **.so*) sont les objets contenant du code et des données pouvant participer au processus de l'assemblage des objets dans deux contextes. Le premier concerne l'assemblage via un éditeur de liens avec les objets réalloués ou partagés pour créer un autre objet. Par contre le deuxième consiste à assembler avec le code du programme exécutable à travers un programme chargeur système (éditeur de liens/chargeur) pour créer une image du processus dans la mémoire.

L'élément principal des fichiers au format ELF est l'en-tête ELF (cf. la Figure 1). Cet en-tête constitue une sorte de plan du fichier et se trouve toujours à son début. L'en-tête ELF contient les éléments suivants : l'emplacement de l'en-tête du programme et de l'en-tête de la section par rapport au début du fichier, l'adresse (appelée en anglais *entrypoint*) à laquelle sera transmis le contrôle après le lancement du programme et les informations déterminant, indépendamment de la plate-forme matérielle, la façon d'interpréter les données contenues dans le fichier (*ident*).

Pour assurer la performance appropriée du format ELF, deux vues parallèles des objets à ce format ont été conçues : la vue d'assemblage et la vue d'exécution (cf. la Figure 2). Pendant la construction de l'objet, les compilateurs, les assembleurs et les éditeurs de liens considèrent le fichier au format ELF comme un ensemble de sections déterminées dans l'en-tête de la section (c'est-à-dire la vue d'assemblage), avec l'en-tête optionnel du programme (cf. la Figure 3). Par contre, le programme système chargeant les fichiers exécutables (éditeur de liens/chargeur) considère un fichier au format ELF comme un ensemble des segments décrits dans l'en-tête du programme, avec l'en-tête optionnel de la section. La vue d'assemblage n'est pas nécessaire pour lancer et exécuter correctement le code exécutable.

Pour consulter et analyser la structure des fichiers au format ELF et des informations qu'ils contiennent, nous pouvons utiliser le programme *objdump*, l'outil *elfsh* ou le programme *ht* étant en même temps navigateur, éditeur et analyseur.

De plus, nous nous servirons de :

- *ht* – un navigateur, éditeur et analyseur de fichiers exécutables ELF,
- *elfsh* – un outil permettant de consulter de façon interactive les détails du format ELF,
- *ndisasm* – un programme permettant le désassemblage des fichiers binaires pour la plate-forme x86,
- *elfgrep* – un outil de recherche des objets (par exemple des bibliothèques) dans les autres objets ELF.

Outre les outils utilisés pendant l'analyse, il faut mentionner un très bon outil commercial *IDAPro*. C'est un programme fonctionnant sous Windows qui sert à désassembler différents types de fichiers exécutables (y compris ELF), pour différentes familles de processeurs. Il effectue une analyse automatique et permet d'autocommenter. Mais il ne nous servira à rien, vu son caractère commercial et le système d'exploitation envisagé. Nous exploiterons les outils gratuits, soumis à la licence GNU.

Reconnaissance préliminaire de l'objet

Nous commencerons le processus d'analyse en obtenant les informations de base sur l'objet analysé. Les informations obtenues dans cette étape détermineront la direction de nos actions futures. Pour obtenir ces informations, nous utiliserons l'outil système standard *file*.

```
#  file kstatd
kstatd: ELF 32-bit LSB executable,
Intel 80386, version 1 (SYSV),
for GNU/Linux 2.2.5,
statically linked, stripped
```

À la suite de l'exécution de cette commande, nous connaissons le type du fichier – c'est un programme exécutable au format ELF (cf. l'Encadré *Format ELF*) et l'architecture pour laquelle il a été compilé (Intel 80386, 32 bits, LSB – *least significant byte*). Nous savons également que l'objet analysé est compilé en statique (*statically linked*) et qu'il a été soumis au processus de stripping (*stripped*). S'il reste d'autres informations que l'on peut obtenir à l'aide de la commande *file* lancé pour un programme exécutable, ce sont les anomalies éventuelles dans l'en-tête ELF.

Recherche d'une chaîne de caractères

L'étape suivante de la reconnaissance préliminaire consiste à retrouver dans le programme analysé des chaînes de caractères intéressantes (suspectes). Cette recherche nous donnera quelques renseignements sur la plate-forme sur laquelle le programme a été construit et quelles actions il peut exécuter. Il ne faut pas oublier que chaque détail peut nous aider dans notre analyse.

Pour la recherche des chaînes de caractères, nous nous servirons de *strings*, un excellent outil qui affiche les séquences de caractères imprimables (ASCII) dont la longueur est supérieure à 4 caractères (c'est la valeur par défaut qui peut être changée – option `-n`). Si nous utilisons l'outil *strings*, dans le cas des objets ELF, il affiche par défaut les chaînes de caractères uniquement à partir des sections du fichier initialisées et chargées. Pour afficher toutes les