Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
PREMIER MINISTRE

S . G . D . S . N
Agence nationale de la sécurité des systèmes d'information
CERT-FR

Paris, le 03 novembre 2014
N° CERTFR-2014-ACT-044

Affaire suivie par :
CERT-FR

BULLETIN D'ACTUALITÉ

Objet : Bulletin d'actualité CERTFR-2014-ACT-044

1 - Publication du guide d'achat de produits de sécurité et de services de confiance qualifiés

Le 09 octobre 2014, l'ANSSI publiait un « Guide d'achat de produits de sécurité et de services de confiance qualifiés ». Son objectif est de faciliter la mise en conformité des administrations avec le Référentiel Général de Sécurité. Il a été élaboré en concertation avec le service des achats de l'état (SAE) et la direction des affaires juridiques (DAJ) des ministères économiques et financiers.

Le guide présente, au regard du code des marchés publics, la méthodologie relative à l'achat de produits de sécurité (chiffreurs, cartes à puces, pare-feu, infrastructures de gestion de clés...) et de services de confiance (prestataires de service de certification électronique, prestataires d'audit, en particulier) ayant fait l'objet d'une qualification.

Le CERT-FR encourage toutes les entités à s'approprier au plus tôt les recommandations présentées dans ce guide.

Documentation

– http://www.ssi.gouv.fr/fr/guides-et-bonnes-pratiques/recommandations-et-guides/choix-des-produits-de-securite/achat-de-produits-de-securite-et-de-services-de-confiance-qualifies.html

2 - Vulnérabilité dans la suite binutils

Le 24 octobre 2014, Michal Zalewski, un ingénieur en sécurité de Google, a publié une vulnérabilité sur la bibliothèque `libbfd` (CVE-2014-8485), découverte grâce à un outil de type "fuzzer".

Cette bibliothèque, présente dans la suite d'outils binutils, est notamment utilisée par l'outil `strings` permettant d'afficher les chaînes de caractères présentes dans un binaire.

Le chercheur a publié un fichier corrompu qui déclenche une erreur de segmentation de l'outil strings lors d'une tentative de déréférencement du registre `EDX`.

```
Program received signal SIGSEGV, Segmentation fault.
bfd_section_from_shdr (abfd=0x8124c90, shindex=2) at elf.c:1942
1942      && (s = idx->shdr->bfd_section) != NULL
(gdb) x/i $eip
=> 0x8078e50 <bfd_section_from_shdr+1568>: mov edx,DWORD PTR [edx+0x2c]
(gdb) info registers edx
edx      0x41414141 1094795585
```

Cette erreur de segmentation est déclenchée par une vulnérabilité présente dans la fonction `setup_group`. Cette dernière omet d'invalider une zone de mémoire tampon lorsqu'une erreur de lecture a été détectée (ex: la longueur des éléments lus n'est pas celle spécifiée par la taille de la section). La fonction se contente de retourner la valeur FALSE alors que la zone de mémoire tampon qui a été modifiée n'est pas réinitialisée.

```
memset (shdr->contents, 0, amt);

if (bfd_seek (abfd, shdr->sh_offset, SEEK_SET) != 0
    || (bfd_bread (shdr->contents, shdr->sh_size, abfd)
        != shdr->sh_size))
  return FALSE;
```

En analysant le code assembleur correspondant à cette fonction, il est possible de voir que les données du binaire ont été copiées vers la zone de mémoire tampon. Cette copie s'effectue lors de l'appel à la fonction `bfd_bread`, située à l'adresse `0x807852b`, et avant le saut vers la fin de la fonction à l'adresse `0x8078536`.

```
Breakpoint 1, 0x0807852b in setup_group at elf.c:619
619         || (bfd_bread (shdr->contents, shdr->sh_size, abfd)
(gdb) x/3i $eip
=> 0x807852b <_bfd_elf_make_section_from_shdr+2379>: call 0x804cfc0 <bfd_bread>
   0x8078530 <_bfd_elf_make_section_from_shdr+2384>: add esp,0x10
   0x8078533 <_bfd_elf_make_section_from_shdr+2387>: cmp eax,DWORD PTR [esi+0x18]
   0x8078536 <_bfd_elf_make_section_from_shdr+2390>: jne 0x8077ca7
(gdb) print shdr.contents
 $1 = (unsigned char *) 0xb6d64010 ""
(gdb) x/50wx 0xb6d64010
 0xb6d64010: 0x00000000 0x00000000 0x00000000 0x00000000
 0xb6d64020: 0x00000000 0x00000000 0x00000000 0x00000000
 0xb6d64030: 0x00000000 0x00000000 0x00000000 0x00000000
 0xb6d64040: 0x00000000 0x00000000 0x00000000 0x00000000
 0xb6d64050: 0x00000000 0x00000000 0x00000000 0x00000000
 0xb6d64060: 0x00000000 0x00000000 0x00000000 0x00000000
 0xb6d64070: 0x00000000 0x00000000 0x00000000 0x00000000
 0xb6d64080: 0x00000000 0x00000000 0x00000000 0x00000000
 0xb6d64090: 0x00000000 0x00000000 0x00000000 0x00000000
 0xb6d640a0: 0x00000000 0x00000000 0x00000000 0x00000000
 0xb6d640b0: 0x00000000 0x00000000 0x00000000 0x00000000
 0xb6d640c0: 0x00000000 0x00000000 0x00000000 0x00000000
 0xb6d640d0: 0x00000000 0x00000000
(gdb) ni
 0x08078530 619       || (bfd_bread (shdr->contents, shdr->sh_size, abfd)
(gdb) x/50wx 0xb6d64010
 0xb6d64010: 0x6c6c6568 0x6f77206f 0x00646c72 0x68732e00
 0xb6d64020: 0x74727473 0x2e006261 0x74786574 0x61642e00
 0xb6d64030: 0x00006174 0x00000000 0x00000000 0x00000000
 0xb6d64040: 0x00000000 0x00000000 0x00000000 0x00000000
 0xb6d64050: 0x00000000 0x00000000 0x00000000 0x0000000b
 0xb6d64060: 0x60000001 0x000002e8 0x08048074 0x00000074
 0xb6d64070: 0x0000000c 0x00000000 0x00000000 0x00000004
 0xb6d64080: 0x00000000 0x0000001b 0x00000011 0x00000003
 0xb6d64090: 0x08049080 0x00000080 0x00efff0c 0x00000000
 0xb6d640a0: 0x00000000 0x00000004 0x00000004 0x00000001
 0xb6d640b0: 0x00000003 0x00000000 0x00010000 0x0000008c
 0xb6d640c0: 0x00000097 0x00000000 0xffde0000 0x19000000
 0xb6d640d0: 0x41414141 0x00000000
(gdb) ni
 0x08078533 619       || (bfd_bread (shdr->contents, shdr->sh_size, abfd)
(gdb) i r eax
```

```
 eax        0xc4 196
(gdb) x/x &shdr.sh_size
 0x8125738: 0x00efff0c
```

Le positionnement d'un point d'arrêt mémoire sur cette zone de mémoire tampon permet de repérer les prochains accès en lecture ou écriture :

```
(gdb) awatch *0xb6d64010
 Hardware access (read/write) watchpoint 2: *0xb6d64010
(gdb) continue
 Continuing.
 Hardware access (read/write) watchpoint 2: *0xb6d64010

 Value = 1819043176
 0x08078e2d in bfd_section_from_shdr (abfd=0x8124c90, shindex=2) at elf.c:1931
 1931           if (idx->flags & GRP_COMDAT)
(gdb) print idx
 $2 = (Elf_Internal_Group *) 0xb6d64010
```

L'extrait ci-dessus met en avant l'utilisation de la zone de mémoire tampon précédemment initialisée dans la fonction `bfd_section_from_shdr` comme un pointeur de structure de type `Elf_Internal_Group`.

Dans le code C suivant, ce pointeur va être incrémenté d'une valeur dépendante de la taille de la section, puis décrémenté tant que le compteur `n_elt` (dépendant de cette même valeur) est positif. Dans le cas où `idx->shdr` est non nul, la structure `bfd_section` de la structure `shdr` est récupérée. C'est lors de ce déréférencement que se produit l'erreur de segmentation vue précédemment.

```
  case SHT_GROUP:
    [...]
    if (hdr->contents != NULL)
    {
      Elf_Internal_Group *idx = (Elf_Internal_Group *) hdr->contents;
      unsigned int n_elt = hdr->sh_size / GRP_ENTRY_SIZE;
      asection *s;

      if (idx->flags & GRP_COMDAT)
        hdr->bfd_section->flags
             |= SEC_LINK_ONCE | SEC_LINK_DUPLICATES_DISCARD;

      /* We try to keep the same section order as it comes in. */
      idx += n_elt;
      while (--n_elt != 0)
      {
        --idx;

        if (idx->shdr != NULL
            && (s = idx->shdr->bfd_section) != NULL
            && elf_next_in_group (s) != NULL)
        [...]
      }
    }
    break;
```

La traduction assembleur de la décrémentation du pointeur `idx` et du test de nullité est la suivante :

```
 0x8078e40 <bfd_section_from_shdr+1552>: sub  eax,0x1
 0x8078e43 <bfd_section_from_shdr+1555>: je   0x80788f0 <bfd_section_from_shdr+192>
 0x8078e49 <bfd_section_from_shdr+1561>: mov  edx,DWORD PTR [esi+eax*4]
```

```
0x8078e4c <bfd_section_from_shdr+1564>: test edx,edx
0x8078e4e <bfd_section_from_shdr+1566>: je   0x8078e40 <bfd_section_from_shdr+1552>
0x8078e50 <bfd_section_from_shdr+1568>: mov  edx,DWORD PTR [edx+0x2c]
```

Lors de l'entrée dans la boucle, les registres `ESI` et `EAX` ont pour valeurs respectives `0xb6d64010` et `0x3bffc3`. Le code va déterminer si l'adresse `0xb7c63f1c` est nulle : si c'est le cas, le registre `EAX` va être décrémenté lors de l'itération suivante et l'adresse vérifiée sera alors `0xb7c63f18`.

Comme il s'agit de la zone mémoire allouée précédemment par la fonction `setup_group`, l'ensemble des valeurs de cette dernière sont à `NULL`, exceptées les valeurs provenant du binaire. C'est ainsi que pour une valeur de `EAX` égale à `0x30`, le registre `EDX` prendra pour valeur `0x41414141` et provoquera l'erreur de segmentation.

```
0xb6d64010: 0x6c6c6568 0x6f77206f 0x00646c72 0x68732e00
0xb6d64020: 0x74727473 0x2e006261 0x74786574 0x61642e00
0xb6d64030: 0x00006174 0x00000000 0x00000000 0x00000000
0xb6d64040: 0x00000000 0x00000000 0x00000000 0x00000000
0xb6d64050: 0x00000000 0x00000000 0x00000000 0x0000000b
0xb6d64060: 0x60000001 0x000002e8 0x08048074 0x00000074
0xb6d64070: 0x0000000c 0x00000000 0x00000000 0x00000004
0xb6d64080: 0x00000000 0x0000001b 0x00000011 0x00000003
0xb6d64090: 0x08049080 0x00000080 0x00efff0c 0x00000000
0xb6d640a0: 0x00000000 0x00000004 0x00000004 0x00000001
0xb6d640b0: 0x00000003 0x00000000 0x00010000 0x0000008c
0xb6d640c0: 0x00000097 0x00000000 0xffde0000 0x19000000
0xb6d640d0: 0x41414141 0x00000000 0x00000000 0x00000000
0xb6d640e0: 0x00000000 0x00000000 0x00000000 0x00000000
[...]
0xb7c63f20: 0x00000000 0x00000000 0x00000000 0x00000000
```

Cette vulnérabilité permet donc à un attaquant de réaliser une écriture de mémoire arbitraire.

Le 27 octobre 2014, Nick Clifton, mainteneur du projet `binutils`, a publié un correctif visant cette vulnérabilité sur le dépôt du projet. En complément des vérifications sur la cohérence des sections de type groupe, ce correctif permet, lorsque la lecture d'une section ne s'est pas faite correctement, de réinitialiser la mémoire allouée à `0`.

```
@@ -618,8 +619,17 @@ setup_group (bfd *abfd, Elf_Internal_Shdr *hdr, asection *newsect)
      if (bfd_seek (abfd, shdr->sh_offset, SEEK_SET) != 0
          || (bfd_bread (shdr->contents, shdr->sh_size, abfd)
              != shdr->sh_size))
-       return FALSE;
-
+       {
+         _bfd_error_handler
+             (_("%B: invalid size field in group section header: 0x%lx"),
+             abfd, shdr->sh_size);
+         bfd_set_error (bfd_error_bad_value);
+         -- num_group;
+         /* PR 17510: If the group contents are even partially
+            corrupt, do not allow any of the contents to be used. */
+         memset (shdr->contents, 0, amt);
+         continue;
+       }


        /* Translate raw contents, a flag word followed by an
           array of elf section indices all in target byte order,
           to the flag word followed by an array of elf section
```

Après application de ce patch à la librairie `libbfd`, l'utilitaire `strings` a le comportement suivant sur le fichier corrompu :

```
$ strings /tmp/strings-bfd-badptr
 BFD: /tmp/strings-bfd-badptr: invalid size field in group section header: 0xefff0c
 BFD: /tmp/strings-bfd-badptr: no valid group sections found
 BFD: /tmp/strings-bfd-badptr: no group info for section .text
 BFD: /tmp/strings-bfd-badptr: warning: sh_link not set for section `.text'
 hello world
 .shstrtab
 .text
 .data
 AAAA
```

Une méthode de contournement du problème pour l'outil `strings` n'utilisant pas une version corrigée de la librairie `libbfd` est d'activer l'option `-a`. Cette dernière spécifie à l'outil que l'utilisateur souhaite extraire les chaînes de caractère de l'ensemble du fichier. Par conséquent, lorsque l'outil se retrouve face à un binaire de type ELF, il ne tente pas d'interpréter les zones `.text` de ce dernier. La librairie vulnérable n'est donc pas utilisée et l'erreur de segmentation n'est pas déclenchée :

```
$ strings -a /tmp/strings-bfd-badptr
 hello world
 .shstrtab
 .text
 .data
 AAAA
```

Cependant ce contournement ne peut pas s'appliquer à d'autres outils de la suite comme par exemple `objdump`, qui est lui aussi vulnérable.

Cet exemple montre que les outils utilisés au quotidien par les équipes d'analyse forensique peuvent présenter des vulnérabilités.

Lors de l'analyse de binaires provenant d'une source inconnue, voire suspicieuse, il est préférable de réaliser les manipulations sur un poste déconnecté (afin d'éviter, dans le cas d'une infection locale, une étendue vers l'ensemble du parc) au sein d'un environnement virtualisé (afin de pouvoir, dans le cas d'une contamination, retrouver un environnement sain).

Documentation

- Entrée de blog :
  http://lcamtuf.blogspot.ro/2014/10/psa-dont-run-strings-on-untrusted-files.html
- Binaire posant problème :
  http://lcamtuf.coredump.cx/strings-bfd-badptr
- Ticket du bug :
  https://sourceware.org/bugzilla/show_bug.cgi?id=17510
- Patch :
  https://sourceware.org/git/gitweb.cgi?p=binutils-gdb.git;a=commitdiff;h=493a33860c71cac998f1a56d6d87d6faa801fbaa

3 - Rappel des avis émis

Dans la période du 27 octobre au 02 novembre 2014, le CERT-FR a émis les publications suivantes :

- CERTFR-2014-AVI-442 : Multiples vulnérabilités dans DokuWiki
- CERTFR-2014-AVI-443 : Multiples vulnérabilités dans TYPO3
- CERTFR-2014-AVI-444 : Multiples vulnérabilités dans les produits Huawei
- CERTFR-2014-AVI-445 : Vulnérabilité dans le noyau Linux de Red Hat
- CERTFR-2014-AVI-446 : Vulnérabilité dans EMC NetWorker Module

- CERTFR-2014-AVI-447 : Vulnérabilité dans EMC Avamar Data Store et Virtual Edition
- CERTFR-2014-AVI-448 : Multiples vulnérabilités dans MariaDB
- CERTFR-2014-AVI-449 : Multiples vulnérabilités dans IBM Tivoli Management Framework
- CERTFR-2014-AVI-450 : Multiples vulnérabilités dans Oracle Linux
- CERTFR-2014-AVI-451 : Vulnérabilité dans Nginx
- CERTFR-2014-AVI-452 : Multiples vulnérabilités dans Qemu
- CERTFR-2014-AVI-453 : Vulnérabilité dans GNU Wget
- CERTFR-2014-AVI-454 : Vulnérabilité dans les produits Cisco
- CERTFR-2014-AVI-455 : Multiples vulnérabilités dans le noyau linux de Red Hat
- CERTFR-2014-AVI-456 : Multiples vulnérabilités dans le noyau Linux Ubuntu
- CERTFR-2014-AVI-457 : Multiples vulnérabilités dans Cisco Unified Communications Manager
- CERTFR-2014-AVI-458 : Multiples vulnérabilités dans Aruba Networks ClearPass

Gestion détaillée du document

03 novembre 2014 version initiale.