Compromission d'un environnement VOIP Cisco

Exploitation du Call Manager

SSTIC 2013

Francisco

Plan

- **Introduction**
- Méthodologie
- Exploitation
- Démo
- Conclusion

Contexte

- Environnements VOIP Cisco largement déployés
- Architecture composée de plusieurs éléments
  - Hard phone : Cisco IP Phone
  - Soft phone : Cisco IP Communicator
  - Call manager : Cisco Unified Communications Manager

Fig.: Cisco IP Phone 7945g

Introduction

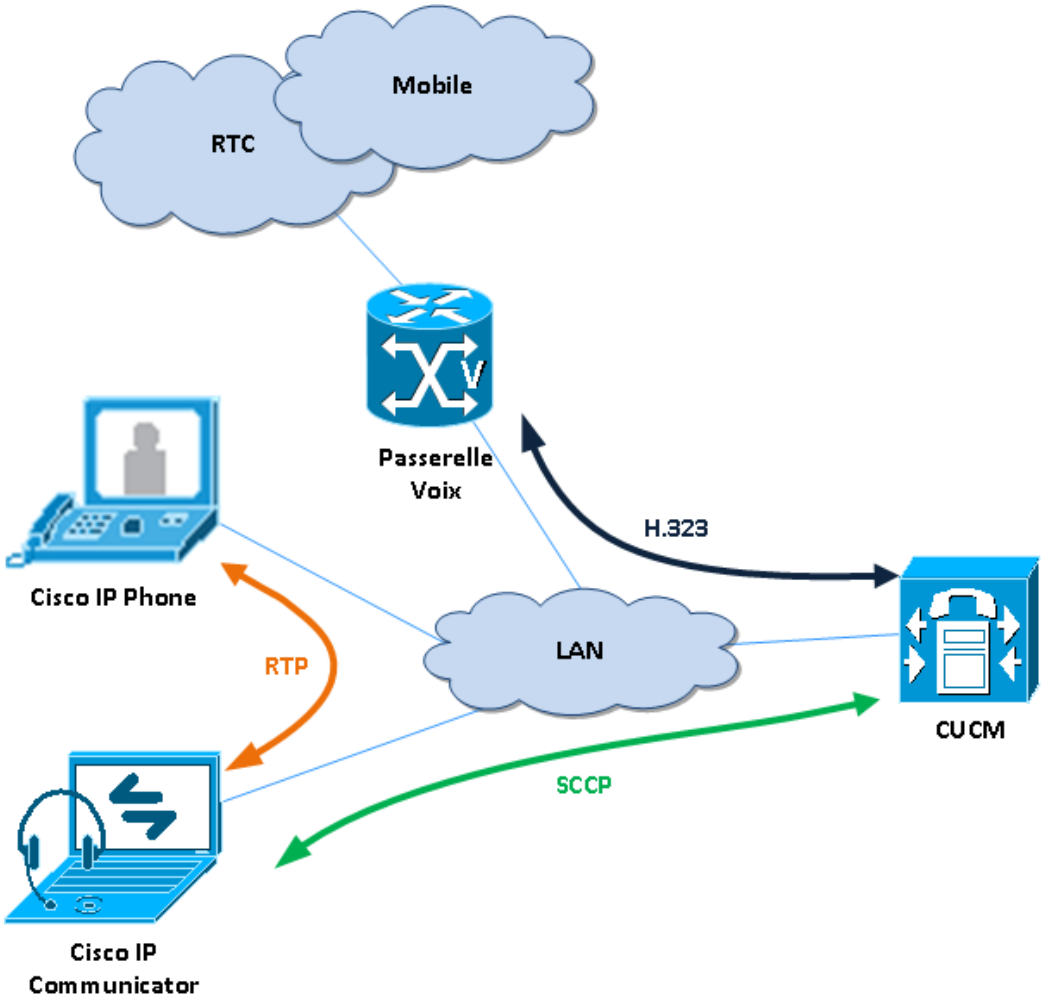


Fig.: Architecture VOIP classique

Introduction

Sécurité

- De plus en plus d'intérêt sur la partie sécurité :
  - *Hack.lu 2007*, Remote Wiretapping on Cisco Phones
  - *Black hat EU 2012*, All Your Calls are Still Belong to Us
  - *29c3 2012*, Hacking Cisco Phones

Et le Call manager ?

- Composant critique de l'architecture
- Permet d'administrer l'ensemble des téléphones
- Gère tous les échanges SCCP effectués
  - Mise en écoute du réseau VOIP en entier si accès *root*
  - Possibilité de cibler une conversation au lieu d'une personne

Plan

- Introduction
- **Méthodologie**
- Exploitation
- Démo
- Conclusion

Méthodologie

Contexte

- Software Appliance livrée avec *Red Hat Enterprise Linux*
- Accès au système de fichier via *vmware-mount*
- Ajout d'un utilisateur SSH et démarrage de l'audit

Plan d'audit

- Un but pour chaque phase…
- Audit *black box* : récupérer les credentials administrateur
- Audit *white box* de l'application : obtenir de l'exécution de code
- Audit du système : obtenir une élévation de privilèges vers *root*

Plan

- Introduction
- Méthodologie
- **Exploitation**
- Démo
- Conclusion

Exploitation

Obtention des credentials

- Recherche d'une injection sql en *black box* :
  - Modification des paramètres réseaux du téléphone
  - Capture des requêtes Cisco Phone <> CUCM
  - Tests de validation de données

- Exploitation de la vulnérabilité :
  - *IBM Informix Dynamic Server 10.00.UC9XF*
  - Impossible d'utiliser la clause *FIRST* sur cette version
  - Exécution de la requête via l'utilisateur *dbadminweb*
  - Récupération des credentials d'un utilisateur admin
  - Credentials chiffrés

Exploitation

Chiffrement des credentials

- Effectué au sein du package *com.cisco.ccm.security*
- La méthode *CCMDecryption.decryptPassword* en dit beaucoup :

```
try
{
  decryptor = JSAFE_SymmetricCipher.getInstance("AES128/CBC/PKCS5Padding", "Java");

  decryptor.setIV(encryptedPassword, 0, 16);
  byte[] temp = decryptor.getIV();

  byte[] encyPassword = new byte[encryptedPassword.length - 16];
  for (int j = 0; j < encryptedPassword.length - 16; j++) {
    encyPassword[j] = encryptedPassword[(16 + j)];
  }
  secretKey = decryptor.getBlankKey();
  secretKey.setSecretKeyData("Clear", keydata, 0, 16);
  decryptor.decryptInit(secretKey);
  recoveredText = new byte[encyPassword.length];
  int partOut = decryptor.decryptUpdate(encyPassword, 0, encyPassword.length, recoveredText, 0);

  int finalOut = decryptor.decryptFinal(recoveredText, partOut);
  totalOut = partOut + finalOut;
}
```

Chiffrement des credentials

- On en conclu les éléments suivants :
  - Chiffrement AES avec une clé de 128 bits
  - Mode d'opération CBC
  - Méthode de padding PKCS5
  - IV stocké dans les 16 premiers octets
  - Ciphertext stocké après les 16 premiers octets

- Où et comment est stockée la clé secrète *keydata* ?
  - Clé hardcodée dans *com.cisco.ccm.security.CCMEncryption*
  - Même valeur pour toutes les instances CUCM

```
static
{
  keydata[0] = 115;  keydata[3] = 116;  keydata[6] = 115;  keydata[9] = 115;  keydata[12] = 99;
  keydata[1] = 109;  keydata[4] = 115;  keydata[7] = 111;  keydata[10] = 105; keydata[13] = 110;
  keydata[2] = 101;  keydata[5] = 121;  keydata[8] = 99;   keydata[11] = 99;  keydata[14] = 105;
```

Exécution de commandes système

- Concerne le package *com.cisco.ccm.admin.actions*
- Escape shell au sein de *BulkFileUploadAction.grantpermission* :

```
public boolean grantpermission(String theFilePath)
  throws InterruptedException
{
  boolean isSuccess = true;

  Process runMod = null;
  try
  {
    LOG.debug("in the grant permission function");

    String strcmd = "chmod 664 '" + theFilePath + "'";

    LOG.debug("in the grant permission function:the file path is:" + theFilePath);
    LOG.debug("cmd is::" + strcmd);

    runMod = Runtime.getRuntime().exec(new String[] { "/bin/bash", "-c", strcmd });

    runMod.waitFor();

    LOG.debug("Setting permissions: the exit value:" + runMod.exitValue());
```

Exécution de commandes système

- Plusieurs conditions pour *trigger* la vulnérabilité :

```
String sqlsearch = "Select tbf.filelocation from typebatfunction as tbf where tbf.enum=" + functiontype;

rs = con1.executeQuery(sqlsearch);

String strFilePath = "";

if (rs.next())
{
  strFilePath = rs.getString("filelocation");
}
...
strFilePath = strFilePath + filename;
LOG.debug("In upload action1.the file path" + strFilePath);

this.fileexists = new File(strFilePath);
...
LOG.debug("In upload action: Instantiate the Writer to write to the file:location is:" + strFilePath);
out = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(strFilePath), "UTF8"));
...
LOG.debug("Calling the grantpermission function: the path passed" + strFilePath);
isPermSuccess = grantpermission(strFilePath);
```

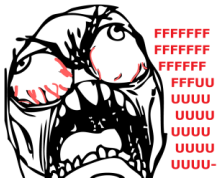
Exploitation

Exécution de commandes système

- Requiert les conditions suivantes pour être *trigger* :
  - Pouvoir insérer une entrée dans la table *typebatfunction*
  - Le *payload* utilisé doit être un chemin d'accès valide

- Problème posé :
  - *Stacked queries* via la première injection sql ?
  - La majorité des requêtes sont exécutées via *dbadminweb*
  - Utilisateur sql possédant des droits limités
  - Cet utilisateur ne peut pas écrire dans *typebatfunction*

```
GRANT SELECT, INSERT, UPDATE, DELETE on typebatfunction to database;
GRANT SELECT, INSERT, UPDATE, DELETE on typebatfunction to poweruser;
GRANT SELECT on typebatfunction to stduser;
...
GRANT stduser TO dbadminweb;
GRANT CONNECT TO dbadminweb;
```



Obtention des droits *poweruser*

- Obtention d'un accès en écriture sur *typebatfunction* ?
  - L'utilisateur sql *dbims* possède le rôle *poweruser*
  - Identification de l'url JDBC associée

```
key="writeurl" value="jdbc:informix-sqli://...;user=dbims;"
```

- Identification des requêtes exécutées dans ce contexte
- Découverte d'un cas remplissant toutes les conditions :

```
protected static boolean updateFullCredential(String credOID, Boolean userCantChange, Boolean
  throws Exception
{
  ...
  try {
    conn = new DBConnector(DBConnector.getWriteUrl());

    String sql = "execute procedure ImsUpdateFullCredential";
    ...
    if (credPolOID != null)
      sql = sql + "'" + credPolOID + "',";
```

Exploitation

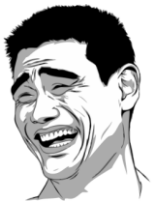
Elévation de privilèges vers *root*

- Exécution de commandes système en tant que *tomcat*
- Audit du système afin d'obtenir les droits *root*
- Analyse du fichier */etc/sudoers* :

```
$ cat /etc/sudoers |grep informix
informix ALL=(root) NOPASSWD: /usr/local/cm/bin/cisco_creve.pl
```

- Quelles sont les propriétés du fichier concerné ?
  - L'utilisateur *informix* est aussi propriétaire du fichier
  - Local root si élévation de privilèges vers *informix*

```
$ ls -lah /usr/local/cm/bin/cisco_creve.pl
-rwxr-xr-x informix informix 3.5K Oct  6 20:38 cisco_creve.pl
```



Exploitation

Élévation de privilèges vers *informix*

- Lors de l'installation, appel de *sec_pwd_change.py*
- Génération du mot de passe de plusieurs utilisateurs système
- Dérivé d'une valeur aléatoire stockée dans un fichier :

```
        f1 = open('/usr/local/cm/db/ifx.txt', 'w')
        f1.write(finalval)
...
        alphabet = "abcdefghijklmnopqrstuvwxyz"
        numalpha = "123456789" + alphabet
        mungedval = ""
        if finalval!= None:
            for c in finalval:
                i = alphabet.find(c)
                if i >= 0:
                    mungedval += numalpha[i]
...
            cmd = "echo '%s' | passwd --stdin -f informix" % (mungedval)
            rc = os.system(cmd)
```

- Le fichier est *world-readable* et non supprimé :

```
$ cat /usr/local/cm/db/ifx.txt
313d8db76d5b
```

Plan

- Introduction
- Méthodologie
- Exploitation
- **Démo**
- Conclusion

Plan

- Introduction
- Méthodologie
- Exploitation
- Démo
- **Conclusion**

Conclusion

Synthèse

- Cisco Unified Communications Manager Remote Root Exploit
- Ne nécessite pas de credentials (*pre-auth*)
- Exploit fiable avec conditions par défaut
- Chaîne d'exploitation de six vulnérabilités
  - Injection sql
  - Clé secrète hardcodée
  - Injection sql *post-auth* avec privilèges élevés
  - Exécution de commandes système
  - Elévation de privilèges vers *informix*
  - Elévation de privilèges vers *root*

Merci de votre attention

Questions ?

www.lexfo.fr @LexfoSecurite contact@lexfo.fr