INTRODUCTION À L'ANALYSE AUTOMATIQUE DE MALWARE AVEC LA SANDBOX *CUCKOO*

SECURIMAG - 14 MARS 2019

0hexit

DÉFINITION D'UNE SANDBOX

Wikipédia :

> *A "sandbox" is a security mechanism for separating running programs, usually in an effort to mitigate system failures or software vulnerabilities from spreading.*

⚠ Sandbox ⇏ VM et VM ⇏ sandbox

CADRE COURANT D'UTILISATION D'UNE SANDBOX

- Prévention (levée de doute)
- Réponse aux incidents de sécurité
- Analyse forensique (post-mortem)
- Threat Intelligence (profilage des attaquants et des vecteurs d'attaque) : IoC

RÉPONSE AUX INCIDENTS DE SÉCURITÉ

- Accès, création, suppression de fichiers ?
- Communications réseau ?
- Ciblé ? Opportuniste ?
- Local ? Externe ?
- Temporaire ? Persistent ?
- Données insignifiantes ? Sensibles ?
- Périmètre de compromission ?
- Type et but de l'attaque ?
- Identité de l'attaquant ?

THREAT INTELLIGENCE : INDICATEURS DE COMPROMISSION (IOC)

- Hash de fichiers
- Signatures de virus
- Noms de fichiers créés
- Mutex (accès exclusif à des ressources partagées)
- IP utilisées, noms de domaine
- Clefs du registre créées
- Tendent à être standardisés

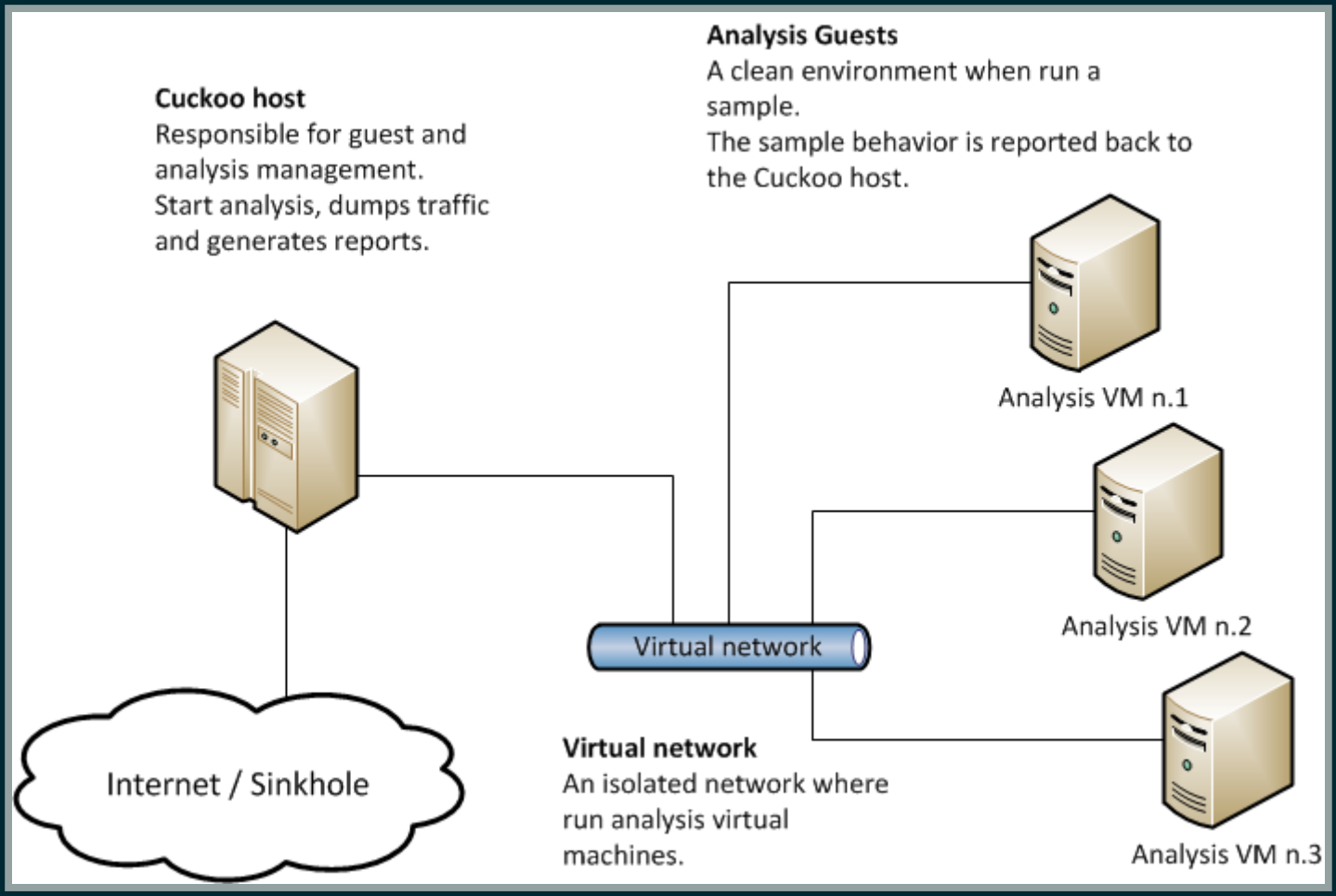
4 NIVEAUX D'ANALYSE DE MALWARE

- Analyse totalement automatique
- Analyse des propriétés statiques
- Analyse interactive du comportement
- Reverse engineering manuel

SANDBOX CUCKOO

- Système d'analyse **automatique** de malware
- Génération de rapports détaillés
- Environnement protégé (isolé)
- Types d'analyse :
  - statique
  - dynamique
  - mémoire

ARCHITECTURE PRINCIPALE



RÉSULTATS DU RAPPORT

- Trace des appels effectués (API calls) par les processus créés par le malware
- Fichiers créés, supprimés et téléchargés par le malware durant son exécution
- Dumps mémoire des processus liés au malware
- Trace du trafic réseau (même chiffré) au format PCAP
- Screenshots pris durant l'exécution du malware
- Dumps mémoire complets des machines

QUELQUES OUTILS UTILISÉS PAR CUCKOO

- YARA : identification et classification des malware à partir de patterns textuels ou binaires → signatures
- Volatility : analyse avancée de dumps mémoire
- ssdeep : computing context triggered piecewise hashes (CTPH) = fuzzy hashes
- tcpdump

AVANTAGES DE CUCKOO (1)

- Analyse de malware dans des env. virtuels Windows, OS X, Linux et Android
- Open source
- Fonctionne bien ! (utilisé par VirusTotal par ex.)
- Supporte les hyperviseurs classiques (KVM, VirtualBox, VMware Workstation, XenServer…)
- Pas d'upload en ligne → créateur du malware pas forcément informé que son programme a été détecté

AVANTAGES DE CUCKOO (2)

- Automatique : gain de temps, moins de compétences spécifiques requises
- Possibilité d'analyser des URL
- Interface Web
- Possibilité de simuler des connexions à Internet (INetSim), d'utiliser Tor ou un VPN, de rejeter tous les paquets ou de tous les accepter.

MISE EN PLACE ET FONCTIONNEMENT D'UNE SANDBOX CUCKOO

- Configurer l'hôte : machines cibles, options d'analyse et de rapport, modules, connexion à Internet...
- Configurer la machine cible (ici une VM) : OS, logiciels tiers, désactivation des màj, du pare-feu et de l'UAC, ajout de l'agent
- Déploiement peut être facilité avec VMCloak

CUSTOMISATION (1)

- Modules auxilaires :
  - Procédures à exécuter en parallèle de chaque processus d'analyse (par ex. sniffer avec tcpdump)
- Modules machines :
  - Interactions entre Cuckoo et le logiciel de virtualisation
- Packages d'analyse :
  - Customiser la procédure d'analyse d'un fichier (par ex. exécution/pause, injection de DLL, dump mémoire à des moments précis)

CUSTOMISATION (2)

- Modules de traitement :
  - Customiser l'analyse des résultats bruts générés
- Signatures personnalisées :
  - Permettent d'identifier des patterns prédéfinis
  - Donner un contexte aux analyses pour simplifier l'interprétation des résultats et identifier des comportements particuliers des malware (par ex. installation de drivers, mutex) et les catégoriser
- Modules de rapport :
  - Customiser les formats de sortie des données

DEMO TIME!

Source du malware : malware-traffic-analysis.net

RAPPORTS VIRUSTOTAL DU MALWARE :

- Analyse .doc
- Analyse .exe

RISQUES / DÉFAULTS (1)

- Techniques "Anti-VM / sandbox" (détection d'environnements simulés)
  - Malware agit différemment (cache ses traces) ou se stoppe
  - Mais de - en - courant car de + en + de VM utilisées sur des serveurs de prod.

RISQUES / DÉFAULTS (2)

- Utilisation de hooks par Cuckoo :
  - Malware peut inspecter son propre espace mémoire et réécrire les hooks de Cuckoo
  - Malware peut aussi utiliser des hooks qui réécrivent ceux de Cuckoo et crasher
  - Mais en pratique, Cuckoo vérifie que ses hooks sont toujours en place et fonctionnels.

RISQUES / DÉFAULTS (3)

- Pas d'analyse des apps iOS
- Analyse n'est jamais sûre à 100% !
- Attention aux infections sur le LAN et aux métadonnées récupérées par les malware lors de requêtes Internet → isolation, simulation du réseau

TECHNIQUES ANTI-VM / SANDBOX (1)

OUTILS DE DÉTECTION D'ARTÉFACTS DUS AUX SANDBOXES

- Pafish (Paranoid Fish)
- VMDE
- anticuckoo

TECHNIQUES ANTI-VM / SANDBOX (2)

EXEMPLE : DÉTECTION DE DEBUGGER

```
int debug_isdebuggerpresent() {
    if (IsDebuggerPresent())
        return TRUE;
    else
        return FALSE;
}
```

TECHNIQUES ANTI-VM / SANDBOX (3)

EXEMPLE : DÉTECTION AVEC L'INSTRUCTION RDTSC

```
static inline unsigned long long rdtsc_diff() {
    unsigned long long ret, ret2;
    unsigned eax, edx;
    __asm__ volatile("rdtsc" : "=a" (eax), "=d" (edx));
    ret  = ((unsigned long long)eax) | (((unsigned long long)edx)
    __asm__ volatile("rdtsc" : "=a" (eax), "=d" (edx));
    ret2  = ((unsigned long long)eax) | (((unsigned long long)edx
    return ret2 - ret;
}
```

TECHNIQUES ANTI-VM / SANDBOX (4)

EXEMPLE : DÉTECTION AVEC L'INSTRUCTION CPUID

```
static inline void cpuid_vendor_00(char * vendor) {
    int ebx = 0, ecx = 0, edx = 0;

    __asm__ volatile("cpuid" \
            : "=b"(ebx), \
            "=c"(ecx), \
            "=d"(edx) \
            : "a"(0x00));
    sprintf(vendor  , "%c%c%c%c", ebx, (ebx >> 8), (ebx >> 16), (
    sprintf(vendor+4, "%c%c%c%c", edx, (edx >> 8), (edx >> 16), (
    sprintf(vendor+8, "%c%c%c%c", ecx, (ecx >> 8), (ecx >> 16), (
    vendor[12] = 0x00;
}
```

TECHNIQUES ANTI-VM / SANDBOX (5)

EXEMPLE : DÉTECTION DE HOOKING

```
static int check_hook_m1(DWORD * dwAddress) {
    BYTE *b = (BYTE *)dwAddress;
    return (*b == 0x8b) && (*(b+1) == 0xff) ? FALSE : TRUE;
}

/* Causes FP in Win 8 */
int check_hook_DeleteFileW_m1() {
    return check_hook_m1((DWORD *)DeleteFileW);
}

int check_hook_ShellExecuteExW_m1() {
    return check_hook_m1((DWORD *)ShellExecuteExW);
}

int check_hook_CreateProcessA_m1() {
    return check_hook_m1((DWORD *)CreateProcessA);
```

TECHNIQUES ANTI-VM / SANDBOX (6)

EXEMPLE : RED PILL (INSTRUCTION SIDT)

- Présentée la 1ère fois par Joanna Rutkowska
- Instruction SIDT exécutable en mode non privilégié.
- VMM relocalise l'IDTR de l'invité pour éviter les conflits avec l'IDTR de l'hôte.
- Mais VMM pas notifiée (aucune exception générée) lorsque SIDT est executé par l'invité.
- Donc retourne IDTR relocalisé, dont l'adresse commence par un octet caractéristique.

TECHNIQUES ANTI-VM / SANDBOX (7)

EXEMPLE : NO PILL (INSTRUCTIONS SGDT / SLDT)

- Structure LDT assignée à un processeur, pas à un OS.
- Localisation = 0 sur l'hôte, ≠ 0 sur la VM

TECHNIQUES ANTI-ANTI-VM / SANDBOX

- Bien configurer hyperviseur + VM
- Rendre la VM moins "propre" (artéfacts d'usure "wear-and-tear") : système, disque, réseau, registre, navigateur...
- Outils :
  - VMCloak, Antivmdetection, VBoxHardenedLoader
- Émulation > virtualisation ??

LISTE DE TECHNIQUES D'ÉVASION

http://unprotect.tdgt.org/index.php/Sandbox_Evasion

ALTERNATIVES À CUCKOO (1)

- Gratuites :
  - Limon : sandbox open source pour malware Linux
  - DRAKVUF : outil d'analyse de binaires basé sur la virtualisation (pas d'agent donc très discret, trace uniquement les appels système)
  - Sandboxie (version gratuite)
  - Analyseurs en ligne

ALTERNATIVES À CUCKOO (2)

- Payantes :
  - Joe Sandbox (virtualisation niveau hyperviseur, depuis un agent en mode noyau)
  - Crowdstrike Falcon Sandbox (idem)
  - VMRay Analyzer (pas d'agent, monitoring implémenté directement au niveau hyperviseur → quasi invisible)
  - Lastline Analyst (émulateur système → moins détectable, voit toutes les instructions)
  - FireEye AX (appareil matériel basé sur l'exécution virtuelle)

UN BON CONCENTRÉ D'INFOS AUTOUR DES MALWARE

- https://github.com/rshipp/awesome-malware-analysis

QUESTIONS ?