DefCon CTF 2016

Cyber Grand Cyber Cybier Cyber Challenge.

<Emantor>

14. August 2016

CTF WTF?

- **C**apture **T**he **F**lag
- Sicherheitswettbewerbe
- Aufgaben bestehen aus Programmen mit Schwachstellen (Vulnerabilities)
- Diese müssen ausgenutzt werden um Flag zu erlangen (Exploit)
- Flags geben Punkte
- Verschiedene Programmtypen:
  - Web (HTML, JS, Databases, ...)
  - Binäraufgaben (statisch oder dynamisch kompilierte Programme ohne Sourcen)
  - Interpreteraufgaben (z.B. Restriktionen von Operatoren)

Spielart

- Als Team, Verteilung der Aufgaben auf mehrere Leute
- Online oder Lokal
- Lokal meist im Rahmen von Konferenzen
- Teilweise Aufteilung in Qualifikation und Finale
- 2 Typen:
  - Jeopardy
  - Attack/Defense

Jeopardy

- Vorgegebene Aufgaben für alle
- Flag wird in der Aufgabe hinterlegt
- Vorwiegend Online

Attack/Defense

- Jedes Team bekommt eine (virtuelle) Maschine
- Vernetzung der Maschinen über z.B. VPNs
- Direkter Angriff anderer Maschinen
- Verteidigung der eigenen Maschine durch Patchen, Firewall, ...
- Zusätzlich SLA-Checks (Service Level Assurance), prüft ob Service noch läuft
- Flags werden vorher hinterlegt oder während des Spielens übertragen

CGC-Framework

- **C**yber **G**rand **C**hallenge
- Projekt der DARPA (Defense Advanced Research Projects Agency)
- Qualifikation im Vorraus
- Preisgeld von 2 Millionen Dollar
- Zentrales Attack/Defense CTF d.h. ein System managed alles
- dieses verwaltet Binaries, FW regeln, PoVs
- führt jede Runde diese gegeneinander aus

PoV

- **P**roof **o**f **V**ulnerability
- Werden mit dem Server ausgehandelt
- 2 Typen:
  1. Kontrolle des EIP-Registers + eines weiteren Registers
  2. Auslesen eines vorgegebenen Speicherbereichs
- PoV Typ 1 einfach reproduzierbar da crash
- PoV Typ 2 wesentlich schwerer zu erkennen
- PoVs können Executables oder XML Definitionen sein

CGC Syscalls and Binary Format

- 7 Syscalls:
  1. void terminate(int status)
  2. int transmit(int fd, const void *buf, sizet count, sizet *txbytes)
  3. int receive(int fd, void *buf, sizet count, sizet *rxbytes)
  4. int fdwait(int nfds, fdset *readfds, fdset *writefds, const struct timeval *timeout, int *readyfds)
  5. int allocate(sizet length, int isX, void **addr)
  6. int deallocate(void *addr, sizet length)
  7. int random(void *buf, sizet count, sizet *rndbytes)
- Linux: 313
- ELF Like Binaryformat (ELF to CGC)

PoV

- **P**roof **o**f **V**ulnerability
- Werden mit dem Server ausgehandelt
- 2 Typen:
  1. Kontrolle des EIP-Registers + eines weiteren Registers
  2. Auslesen eines vorgegebenen Speicherbereichs
- PoV Typ 1 einfach reproduzierbar da crash
- PoV Typ 2 wesentlich schwerer zu erkennen
- PoVs können Executables oder XML Definitionen sein

Vorbereitung

- Zusammenfassen der VMs (5 zu 1)
- Empfänger und Indexer für Netzwerktraffic
- Function detection script
- Reassembling Framework

Wer ist ESPR

- ESPR: Eat Sleep Pwn Repeat
- Zusammenschluss aus Stratum Auhuur und KitCTF
- Defcon: 13 Leute aus allen Teams (3 Stratum 0, 3 KitCTF, 7 CCCAC)

Defcon CTF

- Final CTF
- ESPR qualifiziert im RuCTFe 2015
- The very first human and AI CTF!
- Komplettes CGC Framework
- Erstelltes Interface von LegitBS

Aufgaben

- Binary Binary Binary
- Aufgaben hatten immer mehrere Vulnerabilities
- Eine Aufgabe mit Kommunikation mehrerer Prozesse
- Ein Dateisystememulator
- Ein Kompiler

Fazit

- Mehr Vorbereitung!
- CGC ist “interessant”
- Kein Web, kaum Crypto, keine Scriptsprachen
- Automatisiertes testen von Networktraffic wäre cool

Referenzen

- ESPR Twitter: `https://twitter.com/EatSleepPwnRpt`
- Gynvael (DS): `https://youtu.be/AKs277vpVSY`
- CGC Docs: `http://cgc-docs.legitbs.net/`

Danke fürs ESPR Template, Comawill!