CTF – Capture the Flag

Hans-Christian Esperer, CDA
hc@hcesperer.org

21. Juni 2008

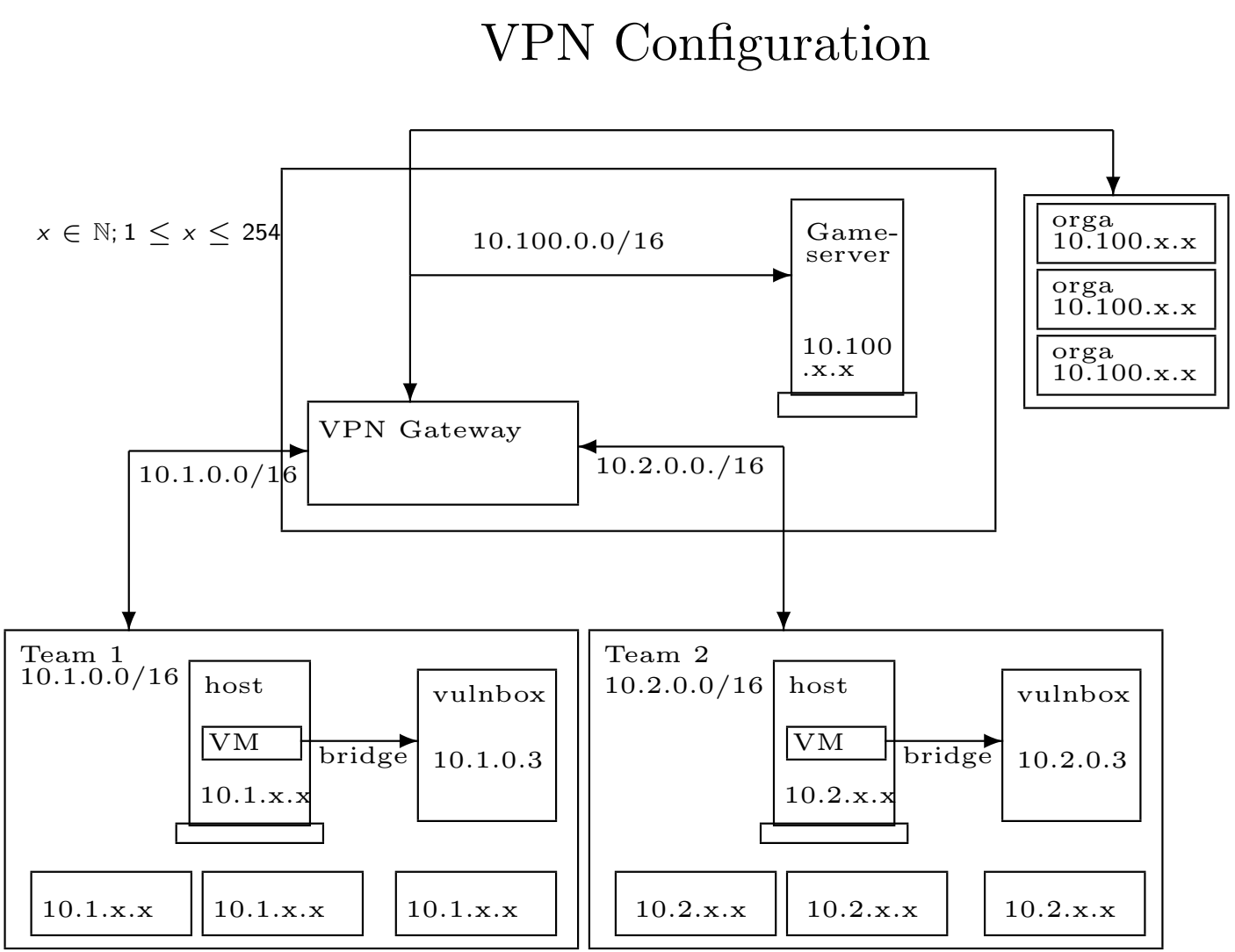
ToC

CTF in der IT-Sicherheit

- Wettbewerb
- 2 oder mehr Teams
- Klassische Sicherheitlücken
- Geregelter Ablauf
- Geschützter Bereich (VPN, lokales Netz)
- „Standardisiert“

Variationen

Netzlayout


VPN Configuration
$x \in \mathbb{N}; 1 \leq x \leq 254$
10.100.0.0/16
Game-server
10.100.x.x
orga 10.100.x.x
orga 10.100.x.x
orga 10.100.x.x
VPN Gateway
10.1.0.0/16
10.2.0.0./16
Team 1
10.1.0.0/16
host
VM
10.1.x.x
bridge
vulnbox
10.1.0.3
10.1.x.x
10.1.x.x
10.1.x.x
Team 2
10.2.0.0/16
host
VM
10.2.x.x
bridge
vulnbox
10.2.0.3
10.2.x.x
10.2.x.x
10.2.x.x

Flagge holen...

```
hc.daedalus ~> telnet localhost 2525
Trying 127.0.0.1...
telnet: connect to address 127.0.0.1: Connection refused
Trying ::1...
Connected to localhost.
Escape character is '^]'.
220 localhost ADLSMTP adl+java smtp
helo foo.de
250 gruess gott
mail from: <>
250 kk
rcpt to: <123';select email_ids from emails;
451-X-Error: Illegal query 'ERROR:  syntax error at or near "';
451-X-Error: 35301B41C628CAE26963967E29E8A9257090A62F399A18584A103D53FFCBCF8D4
D384F1FBDE5E97DE17
quit
451 57B612D334B189ED1461DDEF3AB61EA37953E76C3FBCECD07C4071E77C1213AA D370087612
A2259130D221 localhost CLOSING CONNECTION
Connection closed by foreign host.
hc.daedalus ~>
```

...und abliefern

```
hc.daedalus ~> telnet localhost 8080
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Welcome, 127.0.0.1
Welcome to the CTF scorebot 0.4.37
------------------------------------------------------------------
type "man.reportflag", "man.reportadvisory", "man",
  "copyright" or "license" for more information.
scorebot > reportflag(2, "57B612D334B189ED1461DDEF3AB61EA37953E76C3FBCECD(

You successfully reported a flag for service adlsmtp from team WoD.
You now have 1 offensive points.
scorebot >
```

Ursachen

- Gedankenlosigkeit
- Zeitdruck
- Motivationsmangel
- Ignoranz
- **Wissensmangel**
- **Gesunder Menschenverstand** → Falsche Annahmen → Mathematiker schreiben die sicherste Software ;-)

SQL injection

```
class Email:
    [...]
    def SendEmail(self, e_from, e_to, e_subj, e_hdrs, e_body):
        c = self.db.cursor()
        c.execute("select uid from users where email_to=" \
                + e_to + " limit 2")
        res = c.fetchall()
        if len(res) < 1: raise EmailException("Illegal recipient")
        if len(res) > 1: die("WTF? Serious sh*t happened; dying")
        c.execute("insert into emails (e_from, e_to, e_subj, " \
                "e_hdrs, e_body) " \
                "values (%s,%s,%s,%s,%s)" % \
                (e_from, e_to, e_subj, e_hdrs, e_body))
        self.db.commit()
        return "250 Yay, email sent!!"
```

Buffer overflow

```
#include <stdio.h>
char ADMINUSER[] = "root"; char *ADMINPWD = "reindeerflotilla";
#define LAUFWERK_C 2
int format(int drive){puts("formatting hdd");
        system("format /dev/laufwerkc");}
int main(argc, argv)
    int argc;
    char** argv;
{   char fullname[32]; // no one has a name longer than 32 chars...
    char pass[20]; // who can remember long passwords, anyway?          10
    int is_admin = 0; int i;
    puts("What's your name?\n"); gets(fullname);
    puts("And what would your password be, Sir?");
    gets(pass);
    if (!strcmp(ADMINUSER, fullname) && !strcmp(ADMINPWD, pass))
        is_admin = 1; if (is_admin) format(LAUFWERK_C);
}
```

Dem Compiler gefällt's schon nicht...

```
hc.daedalus ~/doc/ctf> cc -o name name.c
/var/tmp//ccnkXwpH.o(.text+0x4b): In function `main':
: warning: warning: this program uses gets(), which is unsafe.
hc.daedalus ~/doc/ctf> ./name
What's your name?

warning: this program uses gets(), which is unsafe.
abc
And what would your password be, Sir?
def
hc.daedalus ~/doc/ctf> ./name
```

Bis jetzt läuft's noch...

```
hc.daedalus ~/doc/ctf> ./name
What's your name?

warning: this program uses gets(), which is unsafe.
root
And what would your password be, Sir?
reindeerflotilla
formatting hdd
hc.daedalus ~/doc/ctf>
```

Buffer overflow

```
hc.daedalus ~/doc/ctf> ./name
What's your name?

warning: this program uses gets(), which is unsafe.
abcdefghijkllllllllllllllllllllllllllllllllll
And what would your password be, Sir?
ko
formatting hdd
Segmentation fault (core dumped)
hc.daedalus ~/doc/ctf> ./name
```

An alles gedacht? – SUID

- Ziel: Vereinsvorstand soll /var/log/auth.log einsehen können
- Problem: Zugang zu /var/log/auth.log auf Mitglieder von *adm* beschränkt
- Idee: Vorstand in Gruppe *adm* packen?
- $\longrightarrow$ zuviele Rechte
- Lösung: Wir schreiben ein suid-root Binary
- Hint: eigentlich reicht ein sgid-adm Binary, aber suid-root ist lustiger

An alles gedacht? – SUID

```
#include <stdio.h>
int main(argc, argv)
    int argc;
    char** argv;
{   char what[8192]; char *acmd;
    argc--; argv++;
    puts("auth.log viewer. . .");
    setuid(0); setgid(0);
    if (argc) acmd = *argv; else acmd = "-f";
    snprintf(what, 8192, "tail %s /var/log/auth.log", acmd);
    system(what);
}
```

Besser?

```
#include <stdio.h>
int main(argc, argv)
    int argc;
    char** argv;
{   char what[8192]; char *acmd = "";
    argc--; argv++;
    puts("auth.log viewer...");
    setuid(0); setgid(0);
    if (argc) if (*argv[0] == '-') if (argv[0][1] == 'f') acmd = "-f";
    snprintf(what, 8192, "tail %s /var/log/auth.log", acmd);
    system(what);
}
```

. . . oder so vielleicht?

```
#include <stdio.h>
#include <unistd.h>
static const char AUTHFILE[] = "/var/log/auth.log";
int main(argc, argv)
    int argc;
    char** argv;
{   char what[8192]; char *acmd = "";
    argc--; argv++;
    puts("auth.log viewer...");
    setuid(0); setgid(0);
    if (argc) if (*argv[0] == '-') if (argv[0][1] == 'f') acmd = "-f";
    execl("/usr/bin/tail", "/usr/bin/tail",
            *acmd ? acmd : AUTHFILE,
            *acmd ? AUTHFILE : NULL, NULL);
    system(what);
}
```

SUID ist gefährlich

- system() sucht *tail* in $PATH
- Ausführender Benutzer hat *volle Kontrolle* über Environment
- execl() bekommt absoulten Pfad → sicher
- Was macht tail? Gibt es eine Version von tail, die auf das Environment zugreift? → wissen wir nicht
- Lösung: Environment löschen und manuell setzen
- Aber: Gibt es andere Elemente, die zu einem suid-root Programm durchgereicht werden? →**Ja**. *Je nach UNIX-Vendor unterschiedlich*! Beispiele:
  - File descriptors (fds)
  - Arguments (args)
  - Current working directory (cwd)
  - Resource limits (rlimits)
  - POSIX-Signals (signals)

  Quelle: http://cr.yp.to/qmail/guarantee.html

Heute noch relevant?

Eigentlich müsste es doch heute jeder besser wissen...

- „BUGS: ...still crashes randomly; probably has some memory leaks“ – man Xnest
- $\longrightarrow$ Ausbruch aus XEN DOM_u möglich
- „It is possible for a corrupted file system to cause a crash.“ – FreeBSD Manual 12.Juli 2006
- $\longrightarrow$ Niemals fremde USB-Sticks mounten
- „...under certain circumstances, jabberd leaks memory. **This bug is very hard to fix**. In the meantime use a garbage collector.“ April 2008

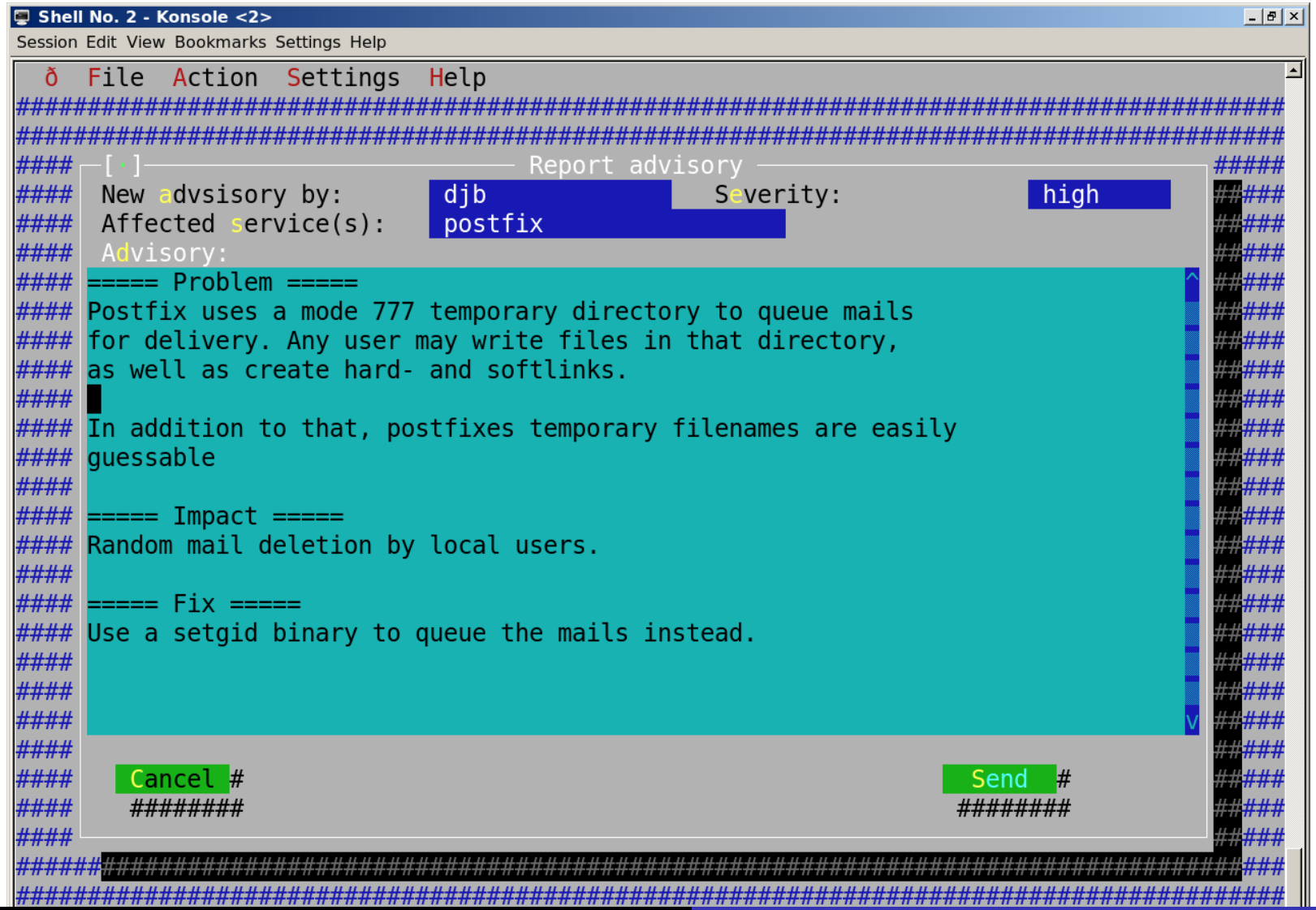
Offensive und defensive Punkte

- Pro Flagge ein Punkt
- Jede Flagge pro Team nur einmal gültig
- Allgemeine Gültigkeitsdauer; Standard: 15min
- **AP:** „Flaggen zählen nur als offensiv, wenn das angreifende Team den Ursprungsservice selbst am Laufen hat“
  - Ist das sinnvoll?
  - Ausbildung von Blackhats?
- Jede innerhalb der Gültigkeitsdauer *von keinem Team* gemeldete Flagge $\longrightarrow$ Ein Defensivpunkt

Defensive & Advisories

- Advisories können unabhängig von Flaggenabgabe und -verteidigung geschrieben werden
- Jedes Advisory je nach Wichtigkeit n Punkte; $n \in \mathbb{N}; n > 0$
- Sollten Advisories sofort veröffentlicht werden?
  - Ja
  - Nein

Beispieladvisory



Warum ein eigener CTF?

- Dienste coden macht Spaß
- $\longrightarrow$ Protokolle und Betriebssystem kennenlernen
- Perspektivenwechsel
- Lustig auf Konferenzen; große CTFs meist über VPN

Dienste schreiben

- Mischung aus komplexen und einfachen Diensten
- Verschiedene Programmiersprachen
  - „C“
  - Scriptsprache (z.B. python, ruby, unrealscript)
  - Binary-only Dienst
- DoS-Attacken möglichst unmöglich machen → werden sonst ausgenutzt – jeglicher Verbote zum Trotz
- „Wohldefinierte“ Sicherheitslücken → Auswirkungen sollten bei Entwurf bekannt sein

Gameserver

- Flaggen an Dienste verteilen
- Flaggen von Diensten einsammeln
- $\longrightarrow$ Flag IDs vergeben
- „Gestohlene“ Flaggen entgegennehmen
- Verwaltung von Advisories

Testscripts

- Schnittstelle zwischen Gameserver und Dienst
- Meistens vom Dienstautor selbst geschrieben
- Funktionen:
  - store IP FLAGID FLAG
  - retrieve IP FLAGID FLAG
  - test IP
- Rückgabewert entscheidend

Beispieltestscript – store

```
#!/usr/local/bin/python
[what, ip, flagid, flag] = sys.argv[1:]
if what == 'store':
  s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
  s.connect(tuple([ip, 2525]))
  s.send("helo score.bot\n")
  s.send("mail from: dood <foo@bar.xxx>\n")
  s.send("rcpt to: " + str(time()) + " <%s@ftwhak.com>\n" % flagid)
  s.send("data\nSubject: hello world\n\n%s\n.\nquit\n" % flag)
  f = s.makefile("r")
  res = f.read()
  for i in res.split("\n"):
    if (len(i) > 3) and (i[0:3] == "221"):
      print "Flag stored!"
      sys.exit(0)
  sys.exit(1)
```

Beispieltestscript – retrieve

```
#!/usr/local/bin/python
[ip, what, flagid, flag] = sys.argv[1:]
elif what == 'retrieve':
  s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
  try:
    s.connect(tuple([ip, 1111]))
    s.sendall("auth user: me pwd: foobar\nget %s\nquit\n" % flagid)
  except: sys.exit(2) # connection failed
  f = s.makefile("r")
  res = f.read()
  for i in res.split("\n"):
    if (len(i) > 3) and (i == flag):
      print "Flag retrieved!"
      sys.exit(0) # all OK
  sys.exit(1) # service broken
```

Beispieltestscript

```
#!/usr/local/bin/python
from sys import exit, argv
def store((ip, flagid, flag)): pass
def retrieve((ip, flagid, flag)): pass
def test((ip)): pass
try:
    {   'store': store,
        'retrieve': retrieve,
        'test': test
    }[argv[1]](tuple(argv[2:]))
except Exception, e:
    print "Error: " + str(e)
    print "Usage: %s store|retrieve IP FLAGID FLAG" % argv[0]
    print "       %s test IP" % argv[0]
```

Probleme

- Teams schlecht vorbereitet
- Keine Kenntnisse über CTF-Ablauf
- Netzwerk/Hardware läuft nicht wie geplant → für Redundanz sorgen
- Lokalitäten nicht verfügbar
- Platzregen
- Strom- und/oder Netzwerkausfälle
- Versehentliches Steckerziehen

Upcoming CTFs

CIPHER 4 http://www.cipher-ctf.org/

DA-OPEN http://ctf.sec-tud.de/

→ MRMCDs111b 6. September 2008 in Darmstadt