
Static Analysis

Static Analysis

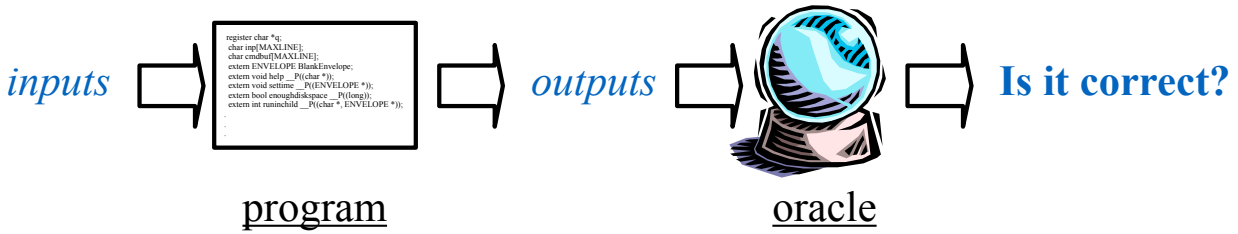
for **Secure Development**

- **Introduction**
  - *Static analysis*: **What**, and **why**?
- **Basic analysis**
  - *Example*: **Flow analysis**
- **Increasing precision**
  - *Context-*, *flow-*, and *path* **sensitivity**
- **Scaling it up**
  - Pointers, arrays, information flow, …

Current Practice

for Software Assurance

- **Testing**
  - Make sure program runs correctly on set of inputs



  - **Benefits**: Concrete failure proves issue, aids in fix
  - **Drawbacks**: Expensive, difficult, *hard to cover all code paths*, no guarantees

Current Practice
(cont'd)

- **Code Auditing**
  - Convince someone else your source code is correct
  - **Benefit**: humans can *generalize beyond single runs*
  - **Drawbacks**: Expensive, hard, no guarantees


```
register char *q;
 char inp[MAXLINE];
 char cmdbuf[MAXLINE];
 extern ENVELOPE BlankEnvelope;
 extern void help __P((char *));
 extern void settime __P((ENVELOPE *));
 extern bool enoughdiskspace __P((long));
 extern int runinchild __P((char *, ENVELOPE *));
 extern void checksmtpattack __P((volatile int *, int, char *, ENVELOPE *));

 if (fileno(OutChannel) != fileno(stdout))
 {
  /* arrange for debugging output to go to remote host */
  (void) dup2(fileno(OutChannel), fileno(stdout));
 }
 settime(e);
 peerhostname = RealHostName;
 if (peerhostname == NULL)
  peerhostname = "localhost";
 CurHostName = peerhostname;
 CurSmtpClient = macvalue('_', e);
 if (CurSmtpClient == NULL)
  CurSmtpClient = CurHostName;

 setproctitle("server %s startup", CurSmtpClient);
#if DAEMON
 if (LogLevel > 11)
 {
  /* log connection information */
  sm_syslog(LOG_INFO, NOQID,
   "SMTP connect from %.100s (%.100s)",
   CurSmtpClient, anynet_ntoa(&RealHostAddr));
 }
#endif

 /* output the first line, inserting "ESMTP" as second word */
 expand(SmtpGreeting, inp, sizeof inp, e);
 p = strchr(inp, '\n');
 if (p != NULL)
  *p++ = '\0';
 id = strchr(inp, ' ');
 if (id == NULL)
  id = &inp[strlen(inp)];
 cmd = p == NULL ? "220 %.*s ESMTP%s" : "220-%.*s ESMTP%s";
 message(cmd, id - inp, inp, id);

 /* output remaining lines */
 while ((id = p) != NULL && (p = strchr(id, '\n')) != NULL)
 {
  *p++ = '\0';
  if (isascii(*id) && isspace(*id))
```

```
cmd < &cmdbuf[sizeof cmdbuf - 2])
   *cmd++ = *p++;
  *cmd = '\0';

  /* throw away leading whitespace */
  while (isascii(*p) && isspace(*p))
   p++;

  /* decode command */
  for (c = CmdTab; c->cmdname != NULL; c++)
  {
   if (!strcasecmp(c->cmdname, cmdbuf))
    break;
  }

  /* reset errors */
  errno = 0;

  /*
  **  Process command.
  **
  **  If we are running as a null server, return 550
  **  to everything.
  */

  if (nullserver)
  {
   switch (c->cmdcode)
   {
    case CMDQUIT:
    case CMDHELO:
    case CMDEHLO:
    case CMDNOOP:
    /* process normally */
    break;

    default:
    if (++badcommands > MAXBADCOMMANDS)
     sleep(1);
    usrerr("550 Access denied");
    continue;
   }
  }

  /* non-null server */
  switch (c->cmdcode)
  {
   case CMDMAIL:
   case CMDEXPN:
   case CMDVRFY:
```

```
while (isascii(*p) && isspace(*p))
    p++;
  if (*p == '\0')
   break;
  kp = p;

  /* skip to the value portion */
  while ((isascii(*p) && isalnum(*p)) || *p == '-')
   p++;
  if (*p == '=')
  {
   *p++ = '\0';
   vp = p;

   /* skip to the end of the value */
   while (*p != '\0' && *p != ' ' &&
          !(isascii(*p) && iscntrl(*p)) &&
          *p != '=')
    p++;
  }

  if (*p != '\0')
   *p++ = '\0';

  if (tTd(19, 1))
   printf("RCPT: got arg %s=\"%s\"\n", kp,
    vp == NULL ? "<null>" : vp);

  rcpt_esmtp_args(a, kp, vp, e);
  if (Errors > 0)
   break;
 }
 if (Errors > 0)
  break;

 /* save in recipient list after ESMTP mods */
 a = recipient(a, &e->e_sendqueue, 0, e);
 if (Errors > 0)
  break;

 /* no errors during parsing, but might be a duplicate */
 e->e_to = a->q_paddr;
 if (!bitset(QBADADDR, a->q_flags))
 {
  message("250 Recipient ok%s",
   bitset(QQUEUEUP, a->q_flags) ?
    " (will queue)" : "");
  nrcpts++;
 }
 else
 {
  /* punt -- should keep message in ADDRESS... */
```

If You're Worried about Security…

A **malicious adversary** is trying to exploit anything you miss!

What more can we do?

Static analysis

- **Analyze program's code without running it**
  - In a sense, we are asking a computer to do what a human might do during a code review
- **Benefit** is (much) **higher coverage**
  - Reason about many possible runs of the program
    - Sometimes *all of them*, providing a **guarantee**
  - Reason about incomplete programs (e.g., libraries)
- **Drawbacks**
  - Can only analyze limited properties
  - May miss some errors, or have false alarms
  - Can be time consuming to run

Impact

- Thoroughly check limited but useful properties
  - **Eliminate categories of errors**
  - **Developers** can **concentrate** on deeper **reasoning**
- **Encourages better development practices**
  - Develop programming models that avoid mistakes in the first place
  - Encourage programmers to think about and make manifest their assumptions
    - Using **annotations** that improve tool precision
- Seeing **increased commercial adoption**