Securing web-based applications with hole-in-the-chroot

Anne Bennett

<anne@alcor.concordia.ca>

Instructional and Information Technology Services

Concordia University, Montreal, Canada

Our mission

- Several sensitive (and fragile) report-generating applications should be available on the web.
- It should be possible for our web-inexperienced, mostly-COBOL programmers to do it.
- The security of the underlying databases and appplications should be protected.
- The security of the system itself should remain protected.

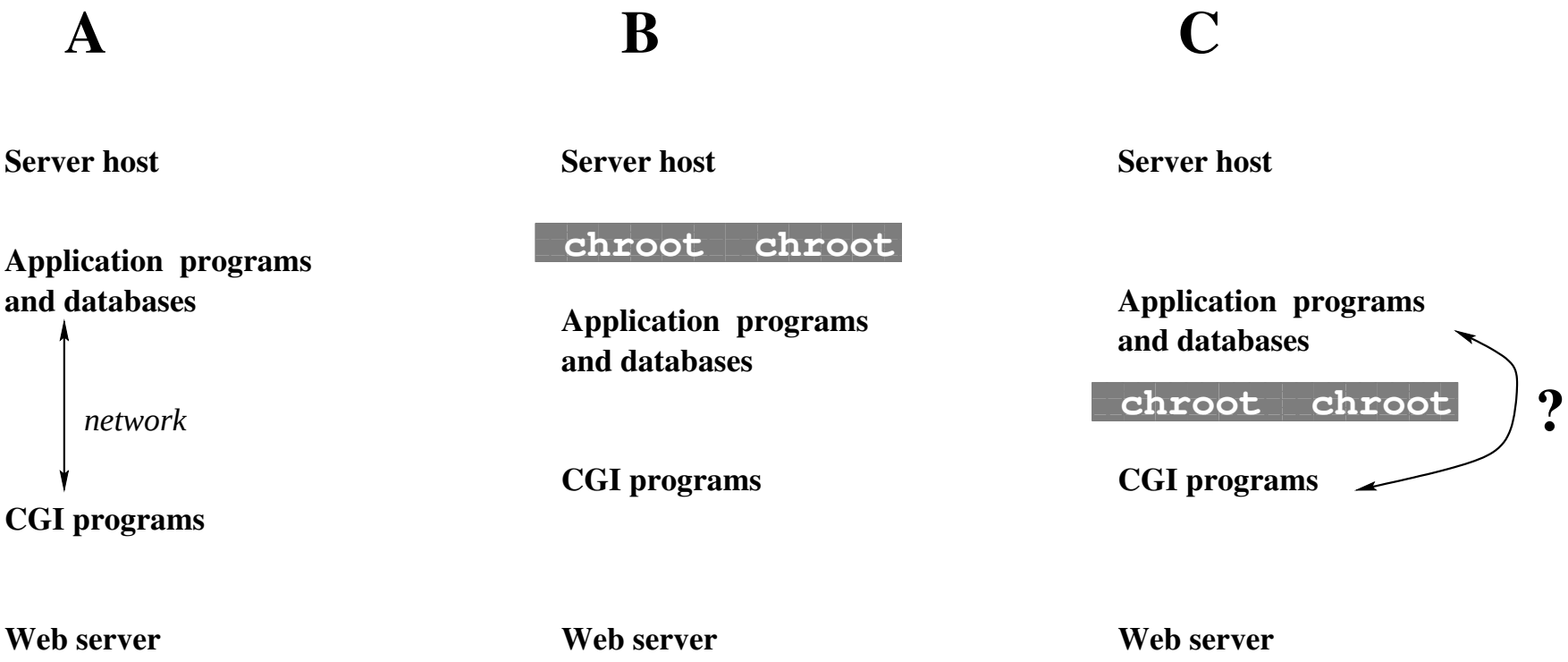
Components of a web-based service, vulnerabilities and defenses

Server host	*Various attacks!*
Application programs	*Buffer overflows and other programming errors*
CGI programs and Web server modules	*Bugs, misconfiguration*
Web server software	*Bugs, misconfiguration*
Network	*Sniffing, hijacking*
Browser	*Bugs, misconfiguration*
Client host	*Malware, misconfiguration*

Components of a web-based service, vulnerabilities and defenses

Server host	*Various attacks!*	Keep system well patched Monitor file integrity and logs
chroot chroot chroot chroot chroot chroot chroot chroot ch		
Application programs	*Buffer overflows and other programming errors*	Use cgiwrap to isolate damage Careful programming practices
CGI programs and Web server modules	*Bugs, misconfiguration*	
Web server software	*Bugs, misconfiguration*	Apache runs mostly as "web user" and is now fairly stable; section that runs as root is very small
Network	*Sniffing, hijacking*	Use encryption (SSL)
Browser	*Bugs, misconfiguration*	Outside scope of my task
Client host	*Malware, misconfiguration*	

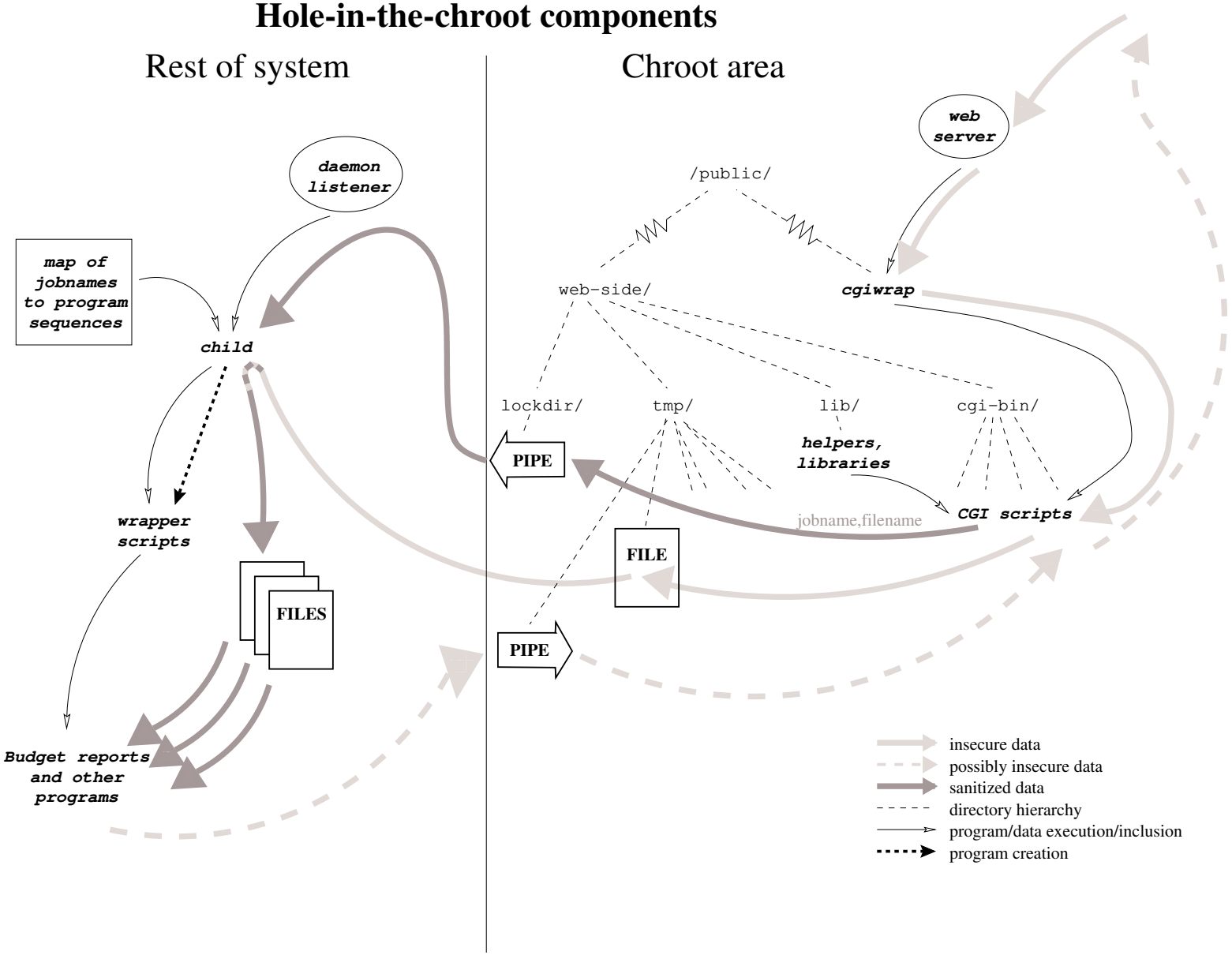
Three approaches to designing our system



Components of the hole-in-the-chroot package

- Programs:
  - *On the chroot side:* ***CGI scripts and code libraries,*** *which send requests to:*
  - *On the system side: a* ***daemon listener,*** *which receives and processes requests. It will spawn a* ***child*** *to process each request.*
- Communications channel:
  - *Meta-communication: a* ***pipe to daemon,*** *used for submitting requests.*
  - *Per-request communication: a* ***file to daemon child*** *and a* ***pipe from daemon child,*** *used to send input to and collect results from the application program.*
- Additional components not supplied by package:
  - *Web server.*
  - *cgiwrap.*
  - *Application programs.*

Hole-in-the-chroot components



Programmers' view of the hole-in-the-chroot package

Lines of generously-commented Perl:

- Hole-in-the-chroot supplies:
  - ***The daemon listener*** 1257
  - ***Code libraries for the CGI scripts*** 895
  - ***A CGI script "stub"*** 147
- The programmers supply:
  - *Application program* n/a
  - *The "meat" of the CGI script:* 113
    - ***issue form***
    - ***prepare input for application program***

```
# ================ configuration section ====================

# The program name to send to the other side.
$progname        = 'abcde';

# How many seconds to wait for a response after we
# tell the other side we have a job for it.
$timeout         = 300;

# Whether access to this form should be restricted to our domain.
$concordia_only = 0;

# The web page title for this form.
$title           = "Report ABCDE";
$more_title      = "(Grant Expenditures Report)";

# The last update information for use in the footer.
$last_update     = "December 13, 1999";
```

```
# ========== MIS-supplied code to print form ==============

print "\n\n<P>\n", "<TABLE>",

  $new_row,         "Username: ",
  $next_column,     CGI::textfield('-name'       => 'username',
                                   '-default'    => '',
                                   '-size'        => 8,
                                   '-maxlength'  => 16 ),

  $new_row,         "Password: ",
  $next_column,     CGI::password_field('-name'       => 'password',
                                        '-default'    => '',
                                        '-size'        => 8,
                                        '-maxlength'  => 16 ),

  $new_row,         "As Of Date (DD-MON-YYYY): ",
  $next_column,     CGI::textfield('-name'       => 'asof_date',
                                   '-default'    => '',
                                   '-size'        => 11,
                                   '-maxlength'  => 11 ),

  $new_row,         "Grant Code: ",
  $next_column,     CGI::textfield('-name'       => 'grant',
                                   '-default'    => '',
                                   '-size'        => 6,
                                   '-maxlength'  => 6 ),

  "</TABLE>", "<P>",
  CGI::submit('-name' => 'Submit'),
  "\n\n";
```

```
# ========== MIS-supplied code to prepare data ==============

# Check that we have all the data we're supposed to have.
#
unless ( $query->param("username") )
{
  die_html "You did not supply a username; try again.";
}
$query->param("username" => uc($query->param("username")));

unless ( $query->param("password") )
{
  die_html "You did not supply a password; try again.";
}

if ( $query->param("asof_date") )
{
  $query->param("asof_date" => uc($query->param("asof_date")));
  unless ( $query->param("asof_date") =~ /^$dd_mon_yyyy$/ )
  {
    die_html sprintf("Your 'as of' date, '%s', was not in the format "
                     ."'DD-MON-YYYY'; try again.", $query->param("asof_date"));
  }
}
else
{
  $query->param("asof_date" => "");
}

unless ( $query->param("grant") )
{
  die_html "You did not supply a grant code; try again.";
}
$query->param("grant" => uc($query->param("grant")));
                                                    # continued...
```

```
# Write out the data (with the appropriate intervening blank
# lines) to the input file for the program.

#-----------------------------------------------------------------
# Data for database ready check
#
printresult "DBNAME\n";
printresult $input_delimiter;

#-----------------------------------------------------------------
# Data for user access to data check
#
printresult sprintf "%s\n", $query->param("username");
printresult "i\n";
printresult sprintf "%s\n", $query->param("grant");
printresult $input_delimiter;

#-----------------------------------------------------------------
# Data for application program, including user authentication
#
printresult sprintf "%s\n", $query->param("username");
printresult sprintf "%s\n", $query->param("password");
printresult "\n";
printresult "C\n";
printresult sprintf "%s\n", $query->param("asof_date");
printresult "S\n";
printresult sprintf "%s\n", $query->param("grant");
printresult "\n";
printresult "999\n";
printresult "N\n";
```

Operational experience

- Programmers are happy: little extra work for them.
- System is stable; no problems introduced, performance good.
- Users very pleased to have their reports on the web, but...
- Now they want more features!!!

Hole-in-the-chroot version 2

New features

- ***Persistent sessions (encrypted cookie/hidden field contains authentication data)***
- ***More complex (back and forth) interactions using a "token language"***
- ***Run as different users (not just "database user") on system side***
- ***Multi-language support (rudimentary internationalization)***

Size comparison

	Lines of Perl: v.1	v.2
The daemon listener	1257	2231
Code libraries for the CGI scripts	895	2563
A CGI script "stub"	147	291
Shared code (CGI and daemon)	0	845
Support programs	59	328
	2358	6258

Questions?

Some questions I have been asked...

- How is username/password authentication done?
- Any incidents due to CGI errors since this system in place?
- How safe is chroot, really?