Session 1165

Building Secure Applications on OpenVMS

Tuesday, September 11, 2001 – 1:00 PM, Room 201D
Wednesday, September, 12, 2001 – 2:45 PM, Room Hilton-Oceanside

Robert Gezelter Software Consultant
35 – 20 167th Street, Suite 215
Flushing, New York 11358 – 1731
United States of America

+1 (718) 463 1079
gezelter@rlgsc.com
http://www.rlgsc.com

What Makes a Secure OpenVMS Application?

Good fences make good neighbors

- “Mending Wall”
North of Boston, 1914
Robert Frost

Why?

Primary Reason – Control Business Risk
Risks:

- Personnel Disclosure
(SSN, Medical, Personnel)
- Business Disclosure
(Publicity, Loss of Advantage, SEC)
- Accountability
- Corruption/Contamination

Technical Goals

Secondary Reasons – Maintain

- System Integrity
- Accountability
- Auditibility

How?

“For your protection and ours, this envelope will be opened in the presence of two bank staff members”

– Citibank Deposit/Payment Envelope (1980)

Is performance an issue?

- Not generally an issue
- Carefully identify bottlenecks
- Eliminate Bottlenecks
- Security is almost NEVER the reason for a PERFORMANCE problem

What Makes a secure OpenVMS Application?

OpenVMS itself is rated C2.

Running a C2-rated operating system is not sufficient. Applications must be designed to not compromise the integrity and containment of the C2-criteria.

Security Critical Areas

- Access Control
- Privileges
- Re-invention
- Contamination

Access Control

Five sample areas:

- Password Management
- DECnet TASK Object
- File Protection and Applications
- Account/Access Management (SYSUAF, RIGHTSLIST, SYLOGIN)
- Access Method Restrictions

Password Management

Changing Passwords:

- Change Frequency – Too Often is not good
- Pronounceability – Important
- Machine Generated
  – Good, if pronounceable

DECnet TASK Object

- facility used for worm attacks
- worm attacks have used GUEST and default accts
- No alternative if network applications
- are to be developed
- (alternatives require >= SYSPRV)

DECnet TASK Object (cont'd)

Safe if used properly

- NO DEFAULT ACCOUNTS
- NO GUEST ACCOUNT
- /NONETWORK qualifier
- NONETMBX qualifier

File Protection and Applications

Access Control Lists and Identifiers

- Do NOT grant access to individuals
- Files may be accessed by identified classes of users
- Individual accounts are given access to classes of data (Rights Identifiers)
- Procedures at access removal/de-briefing

File Protection and Applications (cont'd)

- Do NOT block attempts beyond authorization – let the OpenVMS Security Alarms be triggered
- Break single files into multiple files to permit different security levels

File Protection and Applications (cont’d)

- Data Files (Read/Write/No Access)
- Executable Files (Execute/No Access)
- Protected Subsystems
  Good:

```
(IDENTIFIER=PAYROLL_CLERK,ACCESS=READ)
(IDENTIFIER=PAYROLL_SUPERVISOR,ACCESS=READ+WRITE)
(IDENTIFIER=PAYROLL_CLERK,ACCESS=EXECUTE)
```

  Bad:

```
(IDENTIFIER=SMITH_J,ACCESS=READ)
(IDENTIFIER=DOE_JA,ACCESS=READ+WRITE)
(IDENTIFIER=SMITH_J,ACCESS=EXECUTE)
```

Account/Access Management

SYSUAF

- Automatic Account Expiration
- NO Generic Accounts
- Automatic Logon Facility (ALF)
- Captive Flag

Account/Access Management (cont'd)

- RIGHTSLIST –
  - By Application Function
  - Separate from UIC (SOGW)
  - Paperwork policies

  Examples:

      PAYROLL_CLERK - Read Access
      PAYROLL_ENTRY - Write Access
                      Working Hours-only
      PAYROLL_SUPERVISOR - Modify Access

Account/Access Management (cont'd)

- System Login
  - Check access based upon source
  - More complicated than SYSUAF
  - Use Rights Identifiers as Input
- Group/Application Logins
  - Enforce Group/Role Requirements
  - Remember, User cannot override
  - Check for safe environment

Access Method Restrictions

- Protected Subsystems
- Type of Access
- Take the alarm

Privileges

In a word: Just Say NO.

Permissible:	TMPMBX
Possible:	NETMBX
Never:	Any Devour Class
	NO SYSPRV, CMKRNL, etc.

Reasons:

- Too Broad
- No granularity
- Subverts accountability
- Compromises system integrity

Contamination

Single Thread Application:
Generally safe and within the OpenVMS security model.

Multi-theaded Applications:
Integrity and security outside of the OpenVMS model; You are on your own!

Contamination (Cont'd)

Suggestion:

Use Shareable Libraries to get the memory advantages of common executables without the Contamination hazard.

Re-Invention

When you re-write something, it is a reliable bet that you will forget about some seemingly small feature. Unfortunately, system security depends upon the integration of many small, seemingly baroque details.

Re-Invention (cont'd)

Example:

If your application needs a LOGIN authentication mechanism, use LOGINOUT and AUTHORIZE in concert with SYSUAF and RIGHTSLIST to validate and login your users. Attempting to replicate the functionality is more likely to lead to a security breach.

Re-Invention (cont'd)

If you require some capability not in standard LOGINOUT, consider using the exit or use or use an image executed through SYLOGIN.COM.

Summary:

It is possible to build extremely robust and secure applications under OpenVMS; provided that you do not compromise the integrity of the system; instead use OpenVMS and its underlying capabilities to maximal advantage and leverage your own efforts.

Questions?

Robert Gezelter Software Consultant
35 – 20 167th Street, Suite 215
Flushing, New York 11358 – 1731
United States of America

+1 (718) 463 1079
gezelter@rlgsc.com
http://www.rlgsc.com

Session Notes & Materials:
http://www.rlgsc.com/cets/2001/index.html