PATHWAY System Security

Most applications on Tandem systems make use of PATHWAY as their run time environment:
Apart from its general purpose to provide Nonstop applications with the fundamental basics to achieve scalability and availability, it provides an automatic load balancing, keeps track of servers and their status and restarts them, when needed, and is easy to use.

For system administration purpose it provides a very basic security system to prevent:

- PATHMON from being re-configured by unauthorized users
- PATHWAY programs from being executed by unauthorized users and
- PATHWAY servers from being called by unauthorized PATHSEND communication events

These pretty basic security features generate the impression of a good level of security.

However, apart from providing the basic security settings described above, an important aspect of security is missing:
PATHWAY servers - in terms of the operating system - are processes, and a processes can be opened by anybody, independent of the user ID.
When a process can be opened, it can be used to exchange information with requestors outside the PATHWAY environment. All that is needed is some knowledge of the expected message structure!

In other words: Everybody, knowing a PATHWAY server process and its message structure can abuse the server to deliver any information to the unauthorized requestor, e.g. to

- to retrieve and abuse unauthorized data, thus compromising confidentiality
- to manipulate data, thus compromising data integrity
- to "overheat" the server process through a large number of 'opens' and messages, thus compromising application availability

Not knowing the message structure can lead to the fact, that

- invalid messages may cause the server to end up in trouble, thus bringing the whole application down.
- a Denial of Service attack could be fired against the server process, in the best case causing response time delays, more likely causing response timeouts, or aborted transactions leading to loss of business.

There are only two methods currently available to prevent this:

1. By SAFEGUARD ACLs on the process name.
2. By adding logic in the server process, controlling open events and rejecting the unexpected ones.

This sounds easy to establish, but:

- To maintain SAFEGUARD ACLs, the processes in question have to have known/defined names. To maintain hundreds of process names is a night mare, and adding process names to production environments is not an easy task at all.
  In addition this approach is not viable, as processes are mostly started dynamically with their process names varying.
- From security assessments already delivered to customers it is our observation, that NONE of the Pathway servers examined include control of OPEN messages, in order to reject unexpected opens. To implement this logic, the servers have to be modified.

GreenHouse became aware of this lack of security by delivering a security review, and has developed a solution to this critical vulnerability, so customers can save time and budget for modifying thousands of servers on their system.

Product requirements were:

1. No SAFEGUARD ACLs on processes
2. No change in PATHWAY servers = no additional code to detect OPEN requests
3. Easy to install and maintain = plug-and-play
4. Three run-time modes:
   - ALL (collects and logs all system wide OPEN events to processes and sub-processes)
   - TEST (collects and logs all PATHWAY server relevant OPEN events)
   - PRODUCTION (collects and logs all PATHWAY server relevant OPEN events, and rejects the unexpected ones)
5. Easy configuration of exceptions
6. No re-start of product when configuration is changed
7. Action log
8. Real time reporting

PATHWAY Security Shell (PS-Shell) combines all these requirements. In PRODUCTION mode, it prevents all unexpected OPEN events by generating an error 48 (security violation) to the OPEN request and reports those to a log file.

To demonstrate how easy it is to open a process, we developed a tool named TESTOPEN.

Only execution access is required to run it, no other security attribute is necessary.
All it performs is an OPEN to a given process, nothing else: It is NOT a Trojan Horse, nor an NSA corrupted piece of code!

Command syntax is:

```
TESTOPEN <proc-name>|<CPU,PIN>|ALL
```

where

proc-name	name of process to open
CPU,PIN	PIN of process to open
ALL	checks the entire system for PATHWAY servers and tries to open them

Run it against any process to find out, if it:
- accepts OPEN messages
- does not react on an OPEN message
- rejects an OPEN message

OPEN accepted:

```
$GHS1 PSSHELL 86> testopen $sman
Process $sman successful opened.
$GHS1 PSSHELL 87>
```

$SMAN can be opened, and a communication is possible!
This server is in danger!

Process does not read $RECEIVE, thus does not reply to the OPEN message:

```
$GHS1 PSSHELL 88> testopen $ich
OPEN error 40 on process $ich
$GHS1 PSSHELL 89>
```

This process does not accept an OPEN, thus does not allow any IO.
This server is secure.

When PS-Shell is installed, an unexpected OPEN is rejected, e.g.:

```
$GHS1 PSSHELL 89> testopen $sman
OPEN error 48 on process $sman
$GHS1 PSSHELL 90>
```

$SMAN now is protected by the PS-Shell and the OPEN is rejected with error 48.
This server is secure!

To check the vulnerability of all PATHWAY servers on the system, execute TESTOPEN with the ALL parameter.

This example shows the situation WITHOUT PS-Shell:

```
$GHS1 PSSHELL 279> testopen all
OPEN error   48 on process $GUI1      ($GHS1.SEGUISRV.SEGUI800)
OPEN error   48 on process $SEIW      ($GHS1.SEGUISRV.SEGUI800)
Process $Z03M successful opened ***   ($GHS1.PWM.PWMLOG)
OPEN error   48 on process $Z06T      ($GHS2.CDS.GWPWSRV1)
Process $Z096 successful opened ***   ($GHS2.CDS.OTTACL)
OPEN error   48 on process $GUI0      ($GHS1.SEGUISRV.SEGUI800)
OPEN error   48 on process $GUI2      ($GHS1.SEGUISRV.SEGUI800)
OPEN error   48 on process $GUIT1     ($GHS1.SEGUISRV.SEGUI800)
OPEN error   48 on process $GUIT2     ($GHS1.SEGUISRV.SEGUITST)
OPEN error   48 on process $Z03L      ($GHS1.SECWIN.SEGUI800)
Process $SMAN successful opened ***   ($GHS1.SECOM700.SECMAN)
Process $Z03Z successful opened ***   ($GHS1.WEBADMIN.IWAMSLOG)
Process $SMANS successful opened ***  ($GHS2.CHRISTI.SECMAN)
$GHS1 PSSHELL 280>
```

The green marked servers check open messages and reject unexpected ones, while the red marked servers accept the open, thus are in danger.

When PS-Shell is active, all unexpected opens are rejected by error 48:

```
$GHS1 PSSHELL 280> testopen all
OPEN error   48 on process $GUI1      ($GHS1.SEGUISRV.SEGUI800)
OPEN error   48 on process $SEIW      ($GHS1.SEGUISRV.SEGUI800)
OPEN error   48 on process $Z03M      ($GHS1.PWM.PWMLOG)
OPEN error   48 on process $Z06T      ($GHS2.CDS.GWPWSRV1)
OPEN error   48 on process $Z096      ($GHS2.CDS.OTTACL)
OPEN error   48 on process $GUI0      ($GHS1.SEGUISRV.SEGUI800)
OPEN error   48 on process $GUI2      ($GHS1.SEGUISRV.SEGUI800)
OPEN error   48 on process $GUIT1     ($GHS1.SEGUISRV.SEGUI800)
OPEN error   48 on process $GUIT2     ($GHS1.SEGUISRV.SEGUITST)
OPEN error   48 on process $Z03L      ($GHS1.SECWIN.SEGUI800)
OPEN error   48 on process $SMAN      ($GHS1.SECOM700.SECMAN)
OPEN error   48 on process $Z03Z      ($GHS1.WEBADMIN.IWAMSLOG)
OPEN error   48 on process $SMANS     ($GHS2.CHRISTI.SECMAN)
$GHS1 PSSHELL 281>
```

TESTOPEN can be executed by anybody - no special access rights are needed!

To add security to your PATHWAY systems, consider using PS-Shell.

A test version can be made available for a one month test period for free.
For a commercial proposal please contact Info@GreenHouse.de