The Monitor & Principals
SEADS

By
Edwin Alex Melendez

Secure Enclave Attack Detection System (SEADS)

- The goal is to develop a network monitoring system that can detect attacks on Security protocols by sophisticated intruders.

Intruder  Internet  Principals
One Possible Attack

• Needham/Schoreder Public Key Protocol

A -> B : [na]PKb
B -> A : [na, nb]PKa
A -> B : [nb]PKb

• Attack

A -> M(B) : [na]PKb
M -> B : [na]PKb
B -> M : [na, nb]PKa
M -> A : [na, nb]PKa
A -> M : [nb]PKb
M -> B : [nb]PKb

Right from the beginning this projects became a software engineering challenge
The following are Programming Skills used

• Win32 Programming
  – Win32 is the Windows API
• Thread programming
  – The monitor is multi-threaded
• Network Socket programming
  – The monitor is a network program
• C++ Standard Template Library (STL)
  – The monitor uses complex data structures
The Pros and Cons of Win32

• **Pros**
  – Win32 is easy to use
    • The API is Well-documented
  – Powerful
    • Win32 provides the tools to create advance programs

• **Cons**
  – It is Microsoft proprietary
    • the programmer is ignorant about the inner-working of the OS
  – No support for the POSIX standard
    • POSIX – (The Portable Operating System Interface )
    • It is impossible to write platform-independent code in windows.

More on Win32
Kernel Objects in Windows

• “Kernel objects are operating system resources such as processes, threads, sockets, and events” [1]
  – These objects can be signaled
  – The programmer can create his own kernel objects
  – Synchronization becomes easy when handle to kernel objects are available

![Diagram of primary thread and threads](image-url)
**STL**

“STL is based on a separation of data and operations” [2]

- Vectors, Linked-List, Maps, Queues and others are containers.
- These containers can contained any data type, even other containers.
  - This gives the ability to create complex data structures.

---

**My Job**

- **Create the monitor** which gathers ongoing security protocol activity

- **Simulate the principals** who communicate with each other and the monitor
**SEADS’s Design**

### Principals
{Alex}

### Monitor
{Alex}

### IDE

**Monitor’s Design**

- **Network Thread**
- **Database Thread**
- **IDE Thread**
- **Event Signal**
- **Data Buffer**
  - “queue”
- **Monitor’s Database**
  - “Complex Data Structure”
The Monitor’s Database

Packet received from a principal

[PN] [parties] [nonce] [event]

“DSP” “WP” ...

[102,230,159]

[102,132,221]

...

session 1 session 2 ...

A -> B

B <- A

...

The Monitor’s Database was with STL

Maps

“Associative Arrays”

Linked List

Vector

“DSP” “WP” ...

[102,230,159]

[102,132,221]

...

session 1 session 2 ...

A -> B

B <- A

...
How Do Iterators work?
The same interface works for any type of container

```cpp
list<Session>::iterator sesI;
...
for(sesI = grpI->sessions.begin();
    sesI != grpI->sessions.end(); ++sesI)
{
    ...
    sesI->eventStack.push_back(*e);
    ...
}
```

Remaining Work
Hope to finish by Mid June

- **Principals**
  - Add a GUI interface using the MFC library
  - Enable principals to simulate complicated attacks
- **Monitor**
  - Stress test the Monitor
- **Write Report and Defend Project**
Bookmarks

[1] Win32 Multithreaded Programming, Aaron Cohen and Mike Woodring