The Static Driver Verifier Research Platform

Thomas Ball[1], Ella Bounimova[1], Vladimir Levin[2], Rahul Kumar[2], and Jakob Lichtenberg[2]

[1]Microsoft Research

[2]Microsoft Windows

http://research.microsoft.com/slam/

Plan

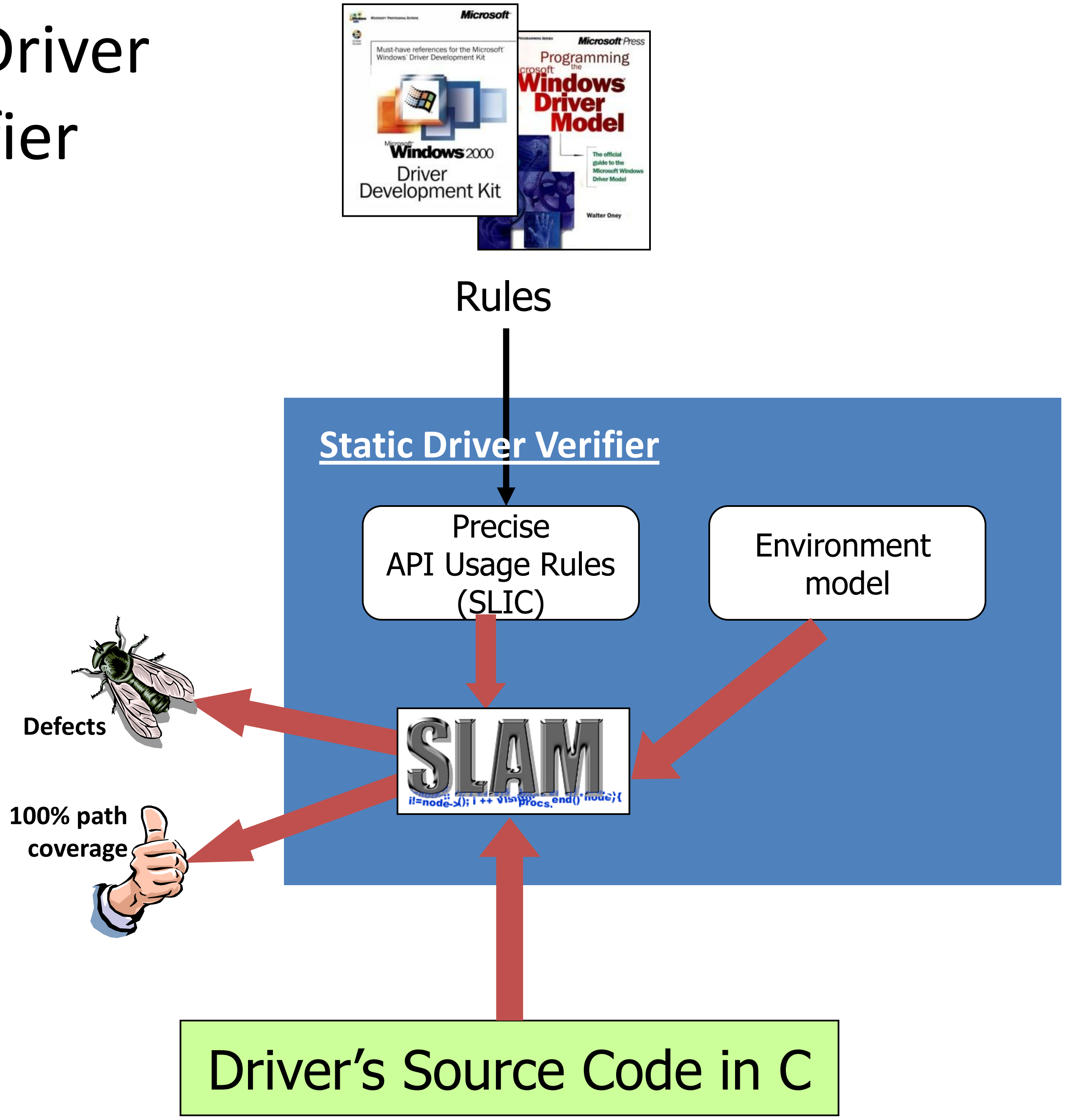
- Static Driver Verifier (SDV)
- SDV Research Platform
  - Creating SDVRP Plugins
  - Boolean Program repository
  - SLAM2 verification engine
- Conclusion

The Static Driver Verifier

Static Driver Verifier (SDV):

- Compile-time verification tool
- Ships with Windows 7 Driver Kit (WDK)
- Less than 4% false alarms on real drivers
- Supports many driver APIs (WDM, KMDF, NDIS, …)
- Uses SLAM as the verification engine
  - ✓ Based on CEGAR loop
  - ✓ Boolean abstraction of input C programs
- API-specific components:
  - ✓ environment model
  - ✓ API rules in SLIC language

Static Driver Verifier


Microsoft Windows 2000 Driver Development Kit
Programming the Microsoft Windows Driver Model
Rules
Static Driver Verifier
Precise API Usage Rules (SLIC)
Environment model
SLAM
Defects
100% path coverage
Driver's Source Code in C

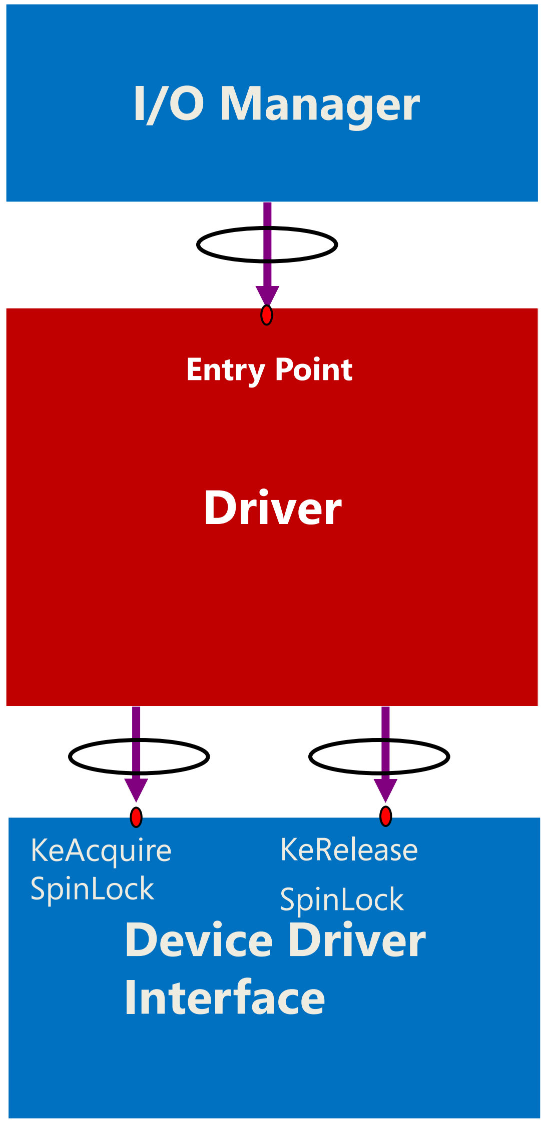
SDV Research Platform

- Academic release of SDV, based on the code that ships with Windows 7 WDK
- Write custom plugins for APIs other than device drivers and custom API rules
- Apply SDV to verify modules (clients) written in C that use the APIs
- Based on the new, robust SLAM2 engine [see upcoming FMCAD2010 paper]

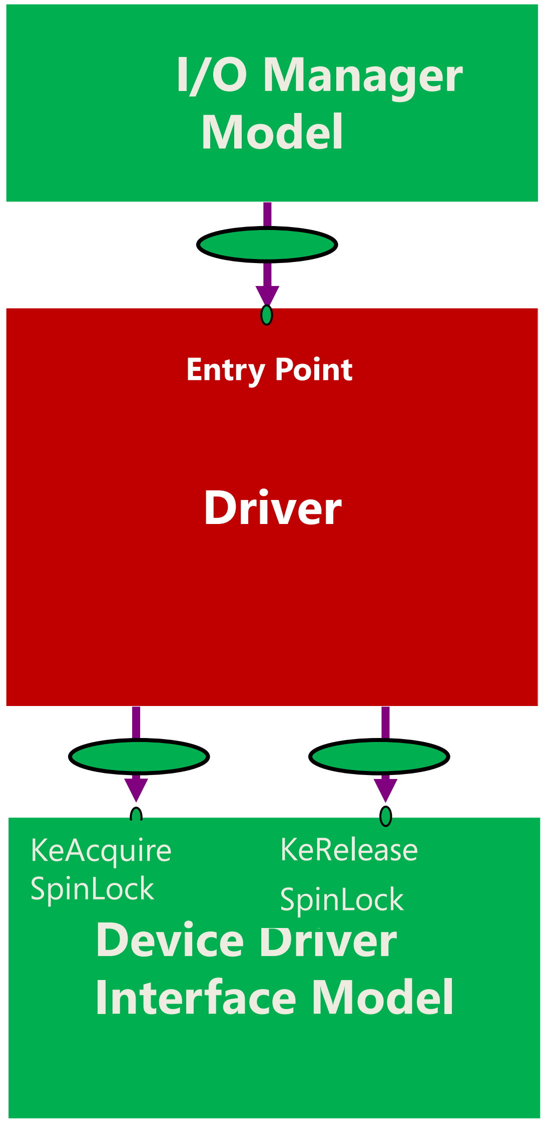
New in This Release: Boolean Program Test Suite

- About 2,800 Boolean programs (BPs) from SDV runs on Windows 7 Device Drivers
  - BP size: 1 - 31 Mb
- Results from running SDV Boolean program model checker Bebop on these programs
- Test scripts used to run Bebop – substitute your BP model checker in place of Bebop!

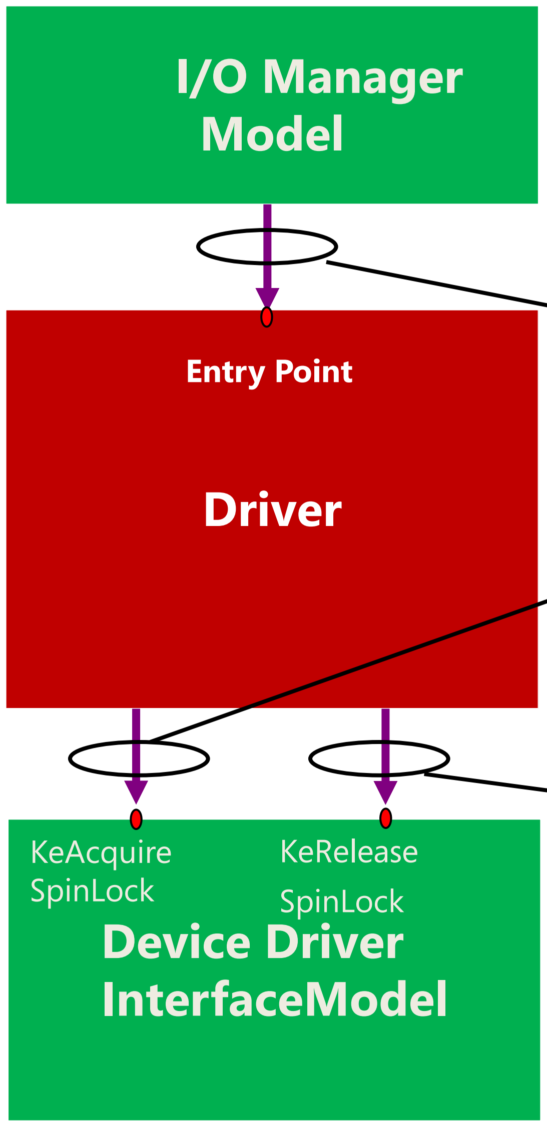
Driver and Operating System: A Module and an Environment



Replace Environment by Environment Model



API SLIC Rule



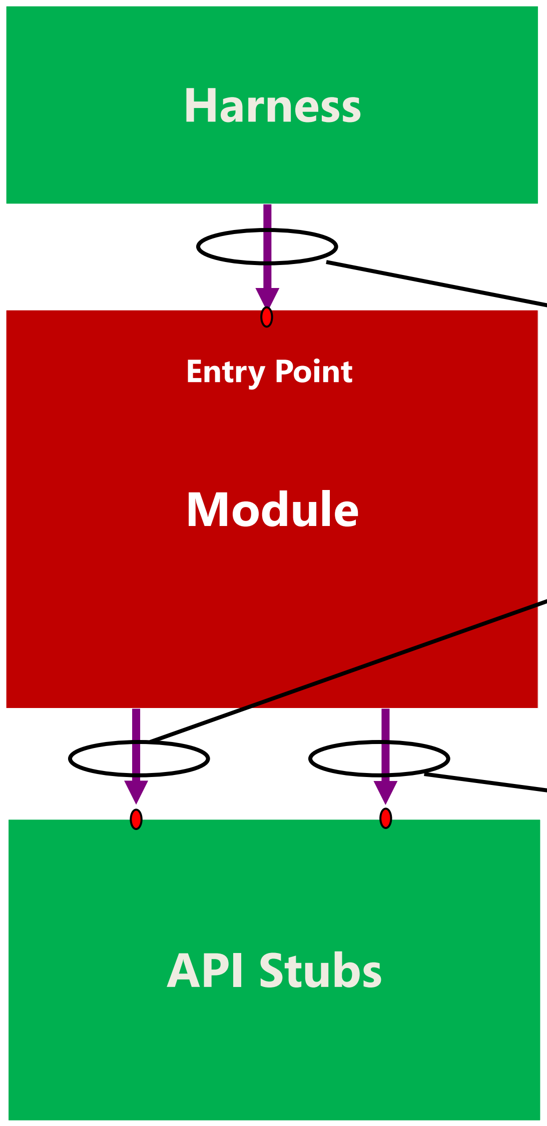
```
state {
  enum {unlocked, locked} s = unlocked;
}
```

```
RunDispatchFunction.exit
{
  if (s != unlocked) abort;
}
```

```
KeAcquireSpinLock.entry
{
  if (s != unlocked) abort;
  else s = locked;
}
```

```
KeReleaseSpinLock.entry
{
  if (s != locked) abort;
  else s = unlocked;
}
```

Generalized to Arbitrary Module



```
state {
   ...
}
```

Your

Rule

Here

SDVRP Plugin: Two Parts

- Platform model (in C)
  - Harness
  - API stubs
- API usage rules (in SLIC)

Plugin Examples in SDVRP

- Plugins for driver platforms: WDM, KMDF, NDIS – can be extended

- Custom platform and plugin – a simple, but complete example

Example: Custom Platform and Plugin

- Custom Platform (what to verify)
  - Platform implementation
  - Sample driver (with a bug)
- Custom Plugin (what to write)
  - sample platform rule
  - platform model

Platform APIs and Data Types

Data:
CUSTOM_IRP: request packet
CUSTOM_LOCK: int
CUSTOM_STATUS: return status for APIs

APIs:
CUSTOM_READ
CUSTOM_WRITE
CustomAcquireLock
CustomReleaseLock
CustomMemMove

The Rule: CustomLock.slic

```
...
state{ enum {unlocked, locked} s = unlocked;}

watch CustomAcquireLock.exit.$1;

CustomAcquireLock.exit[guard $1]
{  if(s==locked)
        { abort "The driver is calling $fname after already acquiring the lock.“;
  } else { s=locked;}}

CustomReleaseLock.exit[guard $1]
{  if(s==unlocked)
        { abort "The driver is calling $fname without first acquiring the lock.“;
   } else {  s=unlocked;}}

sdv_stub_custom_main_end.entry
{  if(s==locked) { abort "The driver has returned from an entry point without releasing the lock.";}}
```

Sample Driver

Entry points:

```
CUSTOM_STATUS DriverWrite(PCUSTOM_IRP irp) {…}

CUSTOM_STATUS DriverRead(PCUSTOM_IRP irp)
{
   CUSTOM_STATUS status;
   CustomAcquireLock(&(DriverData.Lock));

    /* Left out: read DriverData.buffer  from disk. */
   status=CustomMemMove(irp->buffer, DriverData.buffer, 512);
   if (status==CUSTOM_STATUS_UNSUCCESSFUL)
   {
      return CUSTOM_STATUS_UNSUCCESSFUL;
   }
   CustomReleaseLock(&(DriverData.Lock));
   return CUSTOM_STATUS_SUCCESS;
}
```

Platform API model

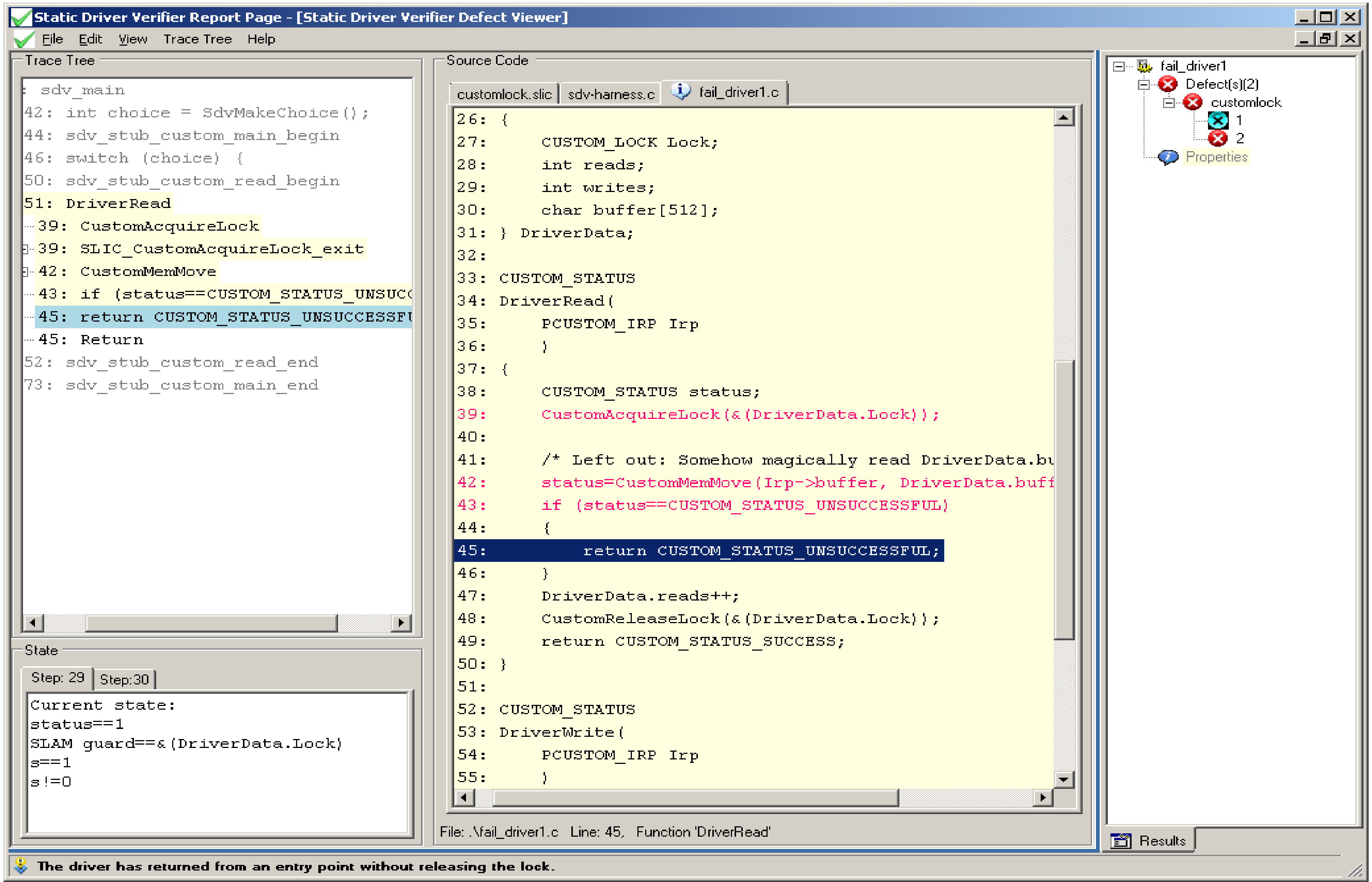
CustomMemMove stub:

```
CUSTOM_STATUS CustomMemMove(char *dst, char *src, int
    bytes)
{
   int choice = SdvMakeChoice();
   switch (choice) {
      case 0: return CUSTOM_STATUS_SUCCESS;
      default: return CUSTOM_STATUS_UNSUCCESSFUL;
   }
}
```

Platform model: test harness

```
int sdv_main()  {
   CUSTOM_STATUS status;
   int choice = SdvMakeChoice();

   switch (choice) {

      case 0:
          status=fun_CUSTOM_READ(sdv_pcustom_irp);
          break;

      case 1:
           status=fun_CUSTOM_WRITE(sdv_pcustom_irp);
           break;

      default:
           status=sdv_DoNothing();
           break;
   }}
```

Defect in sample driver



```
: sdv_main
42: int choice = SdvMakeChoice();
44: sdv_stub_custom_main_begin
46: switch (choice) {
50: sdv_stub_custom_read_begin
51: DriverRead
  39: CustomAcquireLock
  39: SLIC_CustomAcquireLock_exit
  42: CustomMemMove
  43: if (status==CUSTOM_STATUS_UNSUCC
  45: return CUSTOM_STATUS_UNSUCCESSFU
  45: Return
52: sdv_stub_custom_read_end
73: sdv_stub_custom_main_end
```

```
26: {
27:     CUSTOM_LOCK Lock;
28:     int reads;
29:     int writes;
30:     char buffer[512];
31: } DriverData;
32:
33: CUSTOM_STATUS
34: DriverRead(
35:     PCUSTOM_IRP Irp
36:     )
37: {
38:     CUSTOM_STATUS status;
39:     CustomAcquireLock(&(DriverData.Lock));
40:
41:     /* Left out: Somehow magically read DriverData.bu
42:     status=CustomMemMove(Irp->buffer, DriverData.buff
43:     if (status==CUSTOM_STATUS_UNSUCCESSFUL)
44:     {
45:         return CUSTOM_STATUS_UNSUCCESSFUL;
46:     }
47:     DriverData.reads++;
48:     CustomReleaseLock(&(DriverData.Lock));
49:     return CUSTOM_STATUS_SUCCESS;
50: }
51:
52: CUSTOM_STATUS
53: DriverWrite(
54:     PCUSTOM_IRP Irp
55:     )
```

```
Current state:
status==1
SLAM guard==&(DriverData.Lock)
s==1
s!=0
```

File: .\fail_driver1.c Line: 45, Function 'DriverRead'

The driver has returned from an entry point without releasing the lock.

SLAM2 Verification Engine

Improvements include

- Boolean abstraction on basic blocks
- Error Trace validation: combination of forward and backwards symbolic execution
- Optimized predicate discovery
- Uses Z3, new axiomatization of pointers

SLAM2 Verification Engine

SLAM 2.0 released with SDV 2.0, part of Windows 7 WDK

Parameter for WDM drivers	SDV 2.0 (SLAM2)	SDV 1.6 (SLAM1)
False defects	0.4% (2/512)	19.7% (31/157)
Give-up results	3.2% (187/5727)	6% (285/4692)

Download/Installation

- Download and installation instructions on http://research.microsoft.com/slam/
- SDVRP requires that the (freely available) Windows Driver Kit Version 7.1.0 (WDK) be installed ***first***
- Install the SDVRP on top of WDK

Conclusion

- SDVRP toolkit for customizable verification of client code against API rules
- SDV for Windows 7 based on SLAM2
- Boolean program repository
- Licensed for research purposes

SDVRP discussion alias:

sdvrpdis@microsoft.com