Microsoft Research

Microsoft

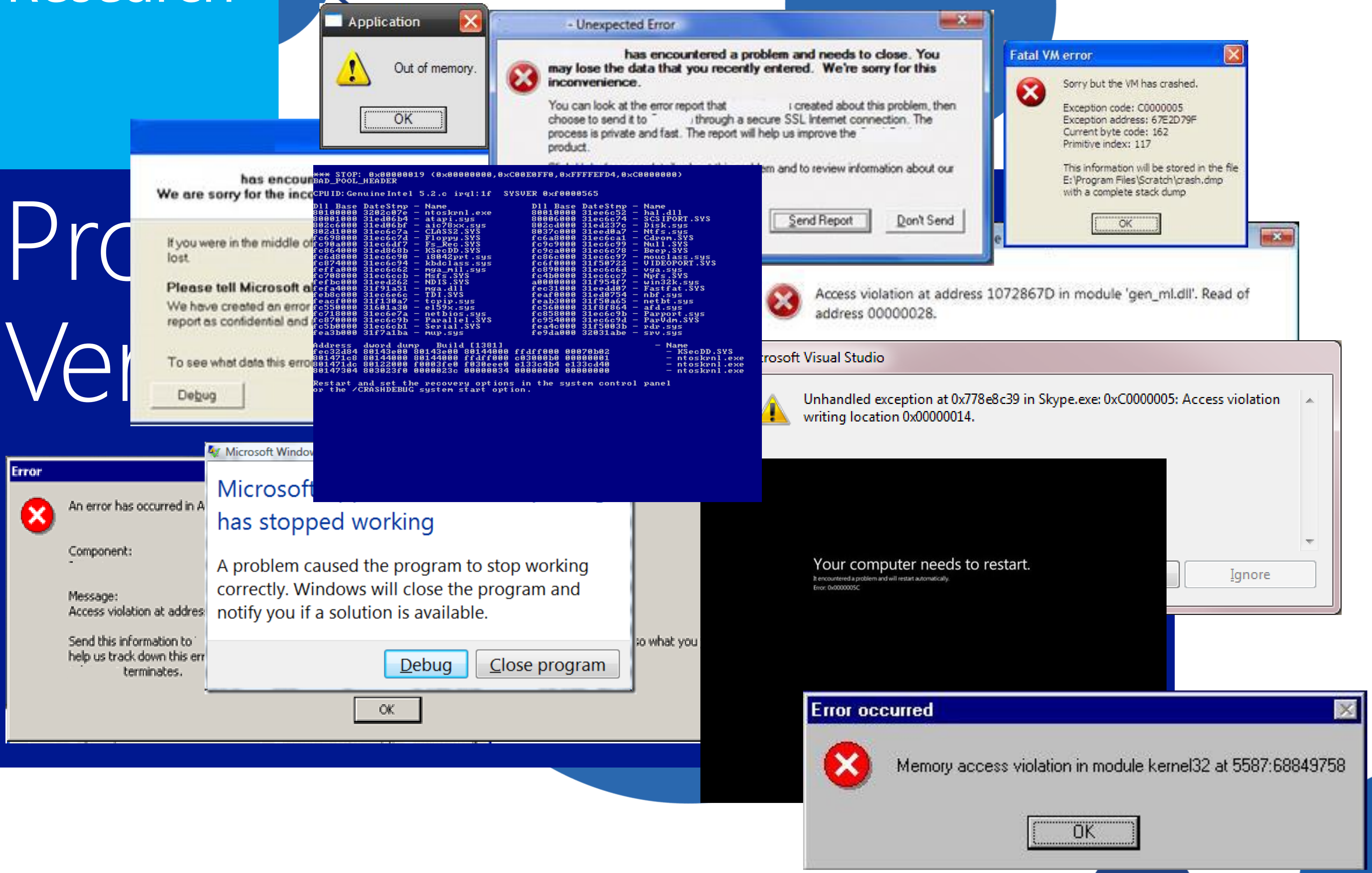
Program Analysis and Verification at Microsoft

Nick Benton

Microsoft Research

Microsoft

Pro
Ver



Application

Out of memory.

- Unexpected Error

has encountered a problem and needs to close. You may lose the data that you recently entered. We're sorry for this inconvenience.

You can look at the error report that created about this problem, then choose to send it to through a secure SSL Internet connection. The process is private and fast. The report will help us improve the product.

em and to review information about our

Fatal VM error

Sorry but the VM has crashed.

Exception code: C0000005
Exception address: 67E2D79F
Current byte code: 162
Primitive index: 117

This information will be stored in the file E:\Program Files\Scratch\crash.dmp with a complete stack dump

has encoun
We are sorry for the inco

If you were in the middle of
lost

Please tell Microsoft a
We have created an error
report as confidential and

To see what data this erro

```
*** STOP: 0x00000019 (0x00000000,0xC00E0FF0,0xFFFFEFD4,0xC0000000)
BAD_POOL_HEADER

CPUID:GenuineIntel 5.2.c irql:1f  SYSVER 0xf0000565

Dll Base DateStmp - Name                    Dll Base DateStmp - Name
80100000 3202c07e - ntoskrnl.exe            80010000 31ee6c52 - hal.dll
80001000 31ed06b4 - atapi.sys               80006000 31ec6c74 - SCSIPORT.SYS
802c6000 31ed06bf - aic78xx.sys             802cd000 31ed237c - Disk.sys
802d1000 31ec6c7a - CLASS2.SYS              8037c000 31eed0a7 - Ntfs.sys
fc698000 31ec6c7d - Floppy.SYS              fc6a8000 31ec6ca1 - Cdrom.SYS
fc90a000 31ec6df7 - Fs_Rec.SYS              fc9c9000 31ec6c99 - Null.SYS
fc864000 31ed868b - KSecDD.SYS              fc9ca000 31ec6c78 - Beep.SYS
fc6d8000 31ec6c90 - i8042prt.sys            fc86c000 31ec6c97 - mouclass.sys
fc874000 31ec6c94 - kbdclass.sys            fc6f0000 31f50722 - VIDEOPORT.SYS
feffa000 31ec6c62 - mga_mil.sys             fc890000 31ec6c6d - vga.sys
fc708000 31ec6ccb - Msfs.SYS                fc4b0000 31ec6cc7 - Npfs.SYS
fefbc000 31eed262 - NDIS.SYS                a0000000 31f954f7 - win32k.sys
fefa4000 31f91a51 - mga.dll                 fec31000 31eedd07 - Fastfat.SYS
feb8c000 31ec6e6c - TDI.SYS                 feaf0000 31ed0754 - nbf.sys
feacf000 31f130a7 - tcpip.sys               feab3000 31f50a65 - netbt.sys
fc550000 31601a30 - el59x.sys               fc560000 31f8f864 - afd.sys
fc718000 31ec6e7a - netbios.sys             fc858000 31ec6c9b - Parport.sys
fc870000 31ec6c9b - Parallel.SYS            fc954000 31ec6c9d - ParVdm.SYS
fc5b0000 31ec6cb1 - Serial.SYS              fea4c000 31f5003b - rdr.sys
fea3b000 31f7a1ba - mup.sys                 fe9da000 32031abe - srv.sys

Address  dword dump   Build [1381]                           - Name
fec32d84 80143e00 80143e00 80144000 ffdff000 00070b02        - KSecDD.SYS
801471c8 80144000 80144000 ffdff000 c03000b0 00000001        - ntoskrnl.exe
801471dc 80122000 f0003fe0 f030eee0 e133c4b4 e133cd40        - ntoskrnl.exe
80147304 803023f0 0000023c 00000034 00000000 00000000        - ntoskrnl.exe

Restart and set the recovery options in the system control panel
or the /CRASHDEBUG system start option.
```

Access violation at address 1072867D in module 'gen_ml.dll'. Read of address 00000028.

rosoft Visual Studio

Unhandled exception at 0x778e8c39 in Skype.exe: 0xC0000005: Access violation writing location 0x00000014.

Error

An error has occurred in A

Component:
-

Message:
Access violation at addres

Send this information to
help us track down this err
terminates.

o what you

Microsoft Windo

Microsoft
has stopped working

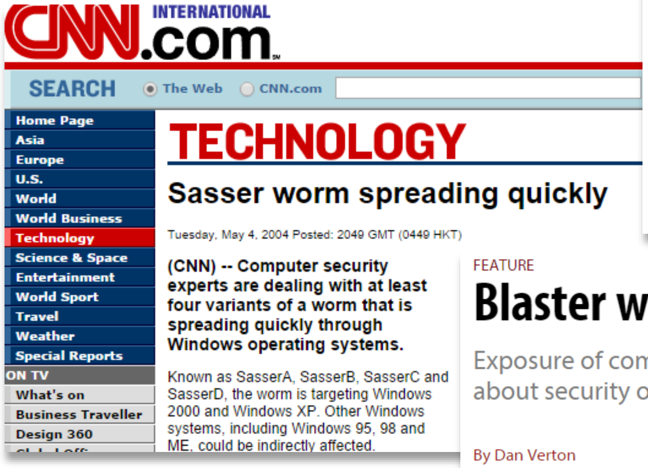
A problem caused the program to stop working correctly. Windows will close the program and notify you if a solution is available.

Your computer needs to restart.
It encountered a problem and will restart automatically.
Error: 0x0000005C

Error occurred

Memory access violation in module kernel32 at 5587:68849758

And worse…



CNN INTERNATIONAL .com

SEARCH ◉ The Web ○ CNN.com

Home Page
Asia
Europe
U.S.
World
World Business
Technology
Science & Space
Entertainment
World Sport
Travel
Weather
Special Reports
ON TV
What's on
Business Traveller
Design 360

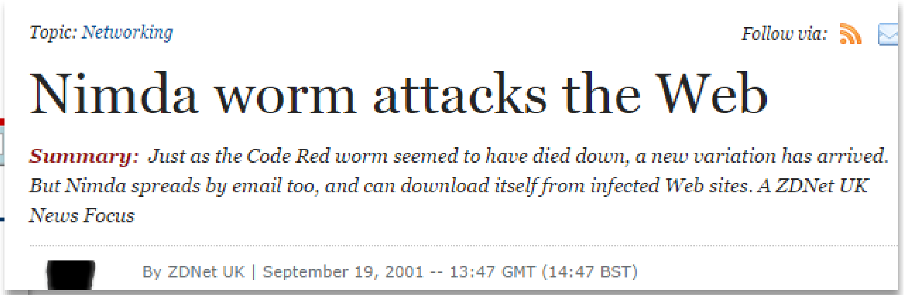
TECHNOLOGY

Sasser worm spreading quickly

Tuesday, May 4, 2004 Posted: 2049 GMT (0449 HKT)

(CNN) -- Computer security experts are dealing with at least four variants of a worm that is spreading quickly through Windows operating systems.

Known as SasserA, SasserB, SasserC and SasserD, the worm is targeting Windows 2000 and Windows XP. Other Windows systems, including Windows 95, 98 and ME, could be indirectly affected.

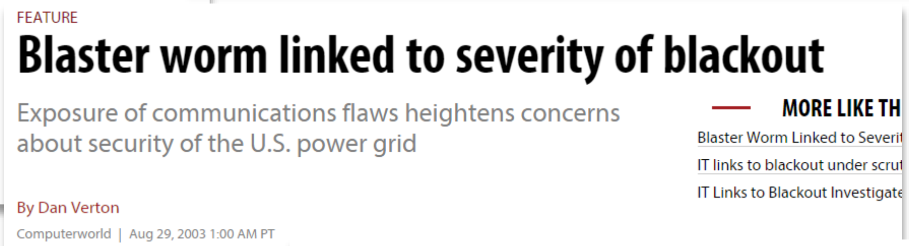


Topic: Networking — *Follow via:*

Nimda worm attacks the Web

Summary: *Just as the Code Red worm seemed to have died down, a new variation has arrived. But Nimda spreads by email too, and can download itself from infected Web sites. A ZDNet UK News Focus*

By ZDNet UK | September 19, 2001 -- 13:47 GMT (14:47 BST)



FEATURE

Blaster worm linked to severity of blackout

Exposure of communications flaws heightens concerns about security of the U.S. power grid

By Dan Verton

Computerworld | Aug 29, 2003 1:00 AM PT

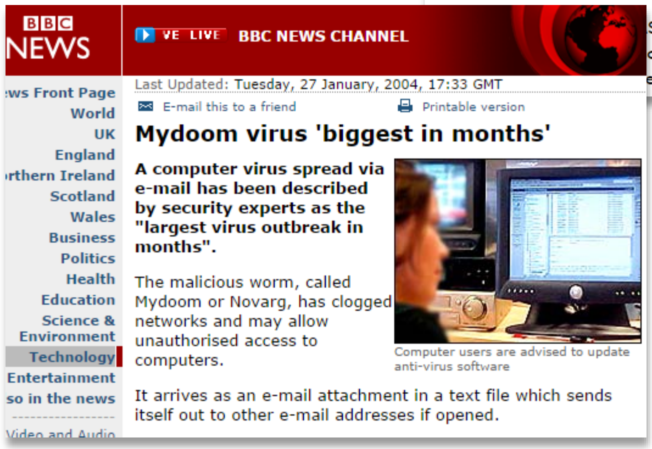
MORE LIKE TH

Blaster Worm Linked to Severit

IT links to blackout under scrut

IT Links to Blackout Investigate

SHINGT
cading e
ealed th



BBC NEWS — VE LIVE BBC NEWS CHANNEL

ws Front Page
World
UK
England
rthern Ireland
Scotland
Wales
Business
Politics
Health
Education
Science & Environment
Technology
Entertainment
so in the news
Video and Audio

Last Updated: Tuesday, 27 January, 2004, 17:33 GMT

E-mail this to a friend — Printable version

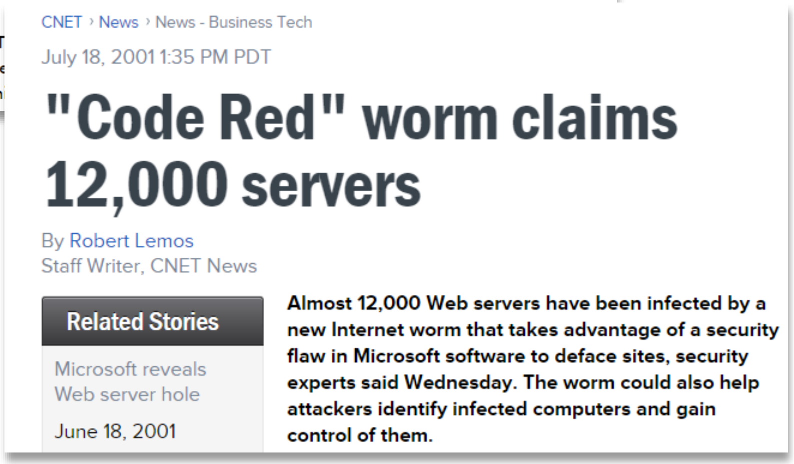
Mydoom virus 'biggest in months'

A computer virus spread via e-mail has been described by security experts as the "largest virus outbreak in months".

The malicious worm, called Mydoom or Novarg, has clogged networks and may allow unauthorised access to computers.

Computer users are advised to update anti-virus software

It arrives as an e-mail attachment in a text file which sends itself out to other e-mail addresses if opened.



CNET › News › News - Business Tech

July 18, 2001 1:35 PM PDT

"Code Red" worm claims 12,000 servers

By Robert Lemos
Staff Writer, CNET News

Related Stories

Microsoft reveals Web server hole
June 18, 2001

Almost 12,000 Web servers have been infected by a new Internet worm that takes advantage of a security flaw in Microsoft software to deface sites, security experts said Wednesday. The worm could also help attackers identify infected computers and gain control of them.

PreFast, PreFix, Esp, …

Rich ecosystem of pluggable analysis tools for finding defects in C/C++ code

Established pillar of Microsoft engineering practice

Quality gate for checkins/integrations

Ship in Visual Studio

357 rules, 13 defect categories

6011	Dereferencing NULL pointer	PREfast
6387	Invalid parameter value	PREfast
6001	Using uninitialized memory	PREfast
6101	Returning uninitialized memory	PREfast
6031	Return value ignored	PREfast
6385	Read overrun	PREfast
6262	Excessive stack usage	PREfast
6386	Write overrun	PREfast
26115	Failing to release lock	EspC
28196	Returning bad result	Drivers
28719	Banned API usage	WindowsPRefast
26035	Precondition null termination violation	EspX
26018	Potential buffer overflow nullterminated	EspX

PreFast, PreFix, Esp, …

Rich ecosystem of pluggable analysis tools for finding defects in C/C++ code

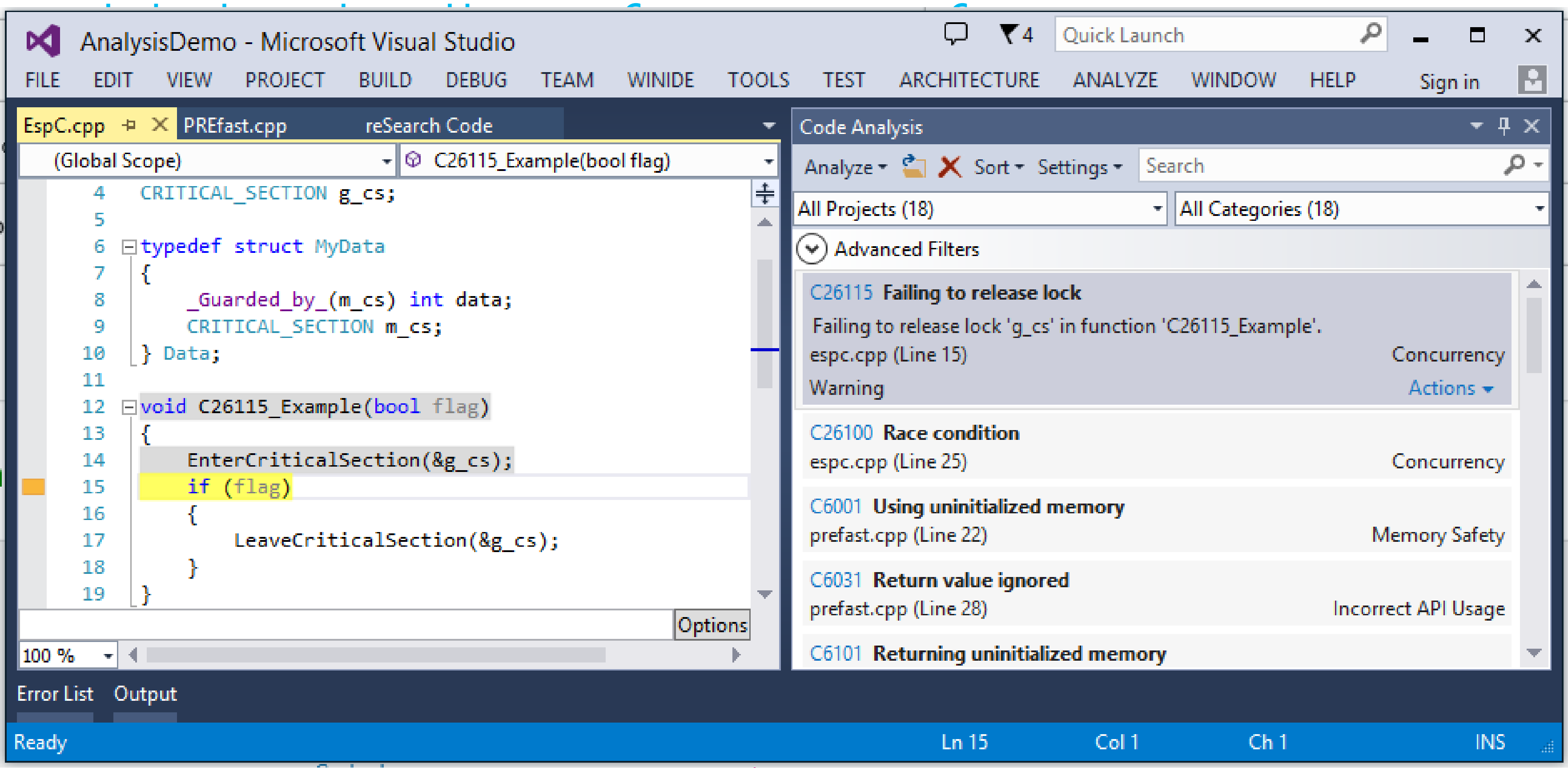
E

p

Q

S

35



```
 4  CRITICAL_SECTION g_cs;
 5
 6  typedef struct MyData
 7  {
 8      _Guarded_by_(m_cs) int data;
 9      CRITICAL_SECTION m_cs;
10  } Data;
11
12  void C26115_Example(bool flag)
13  {
14      EnterCriticalSection(&g_cs);
15      if (flag)
16      {
17          LeaveCriticalSection(&g_cs);
18      }
19  }
```

C26115 **Failing to release lock**
Failing to release lock 'g_cs' in function 'C26115_Example'.
espc.cpp (Line 15) Concurrency
Warning

C26100 **Race condition**
espc.cpp (Line 25) Concurrency

C6001 **Using uninitialized memory**
prefast.cpp (Line 22) Memory Safety

C6031 **Return value ignored**
prefast.cpp (Line 28) Incorrect API Usage

C6101 **Returning uninitialized memory**

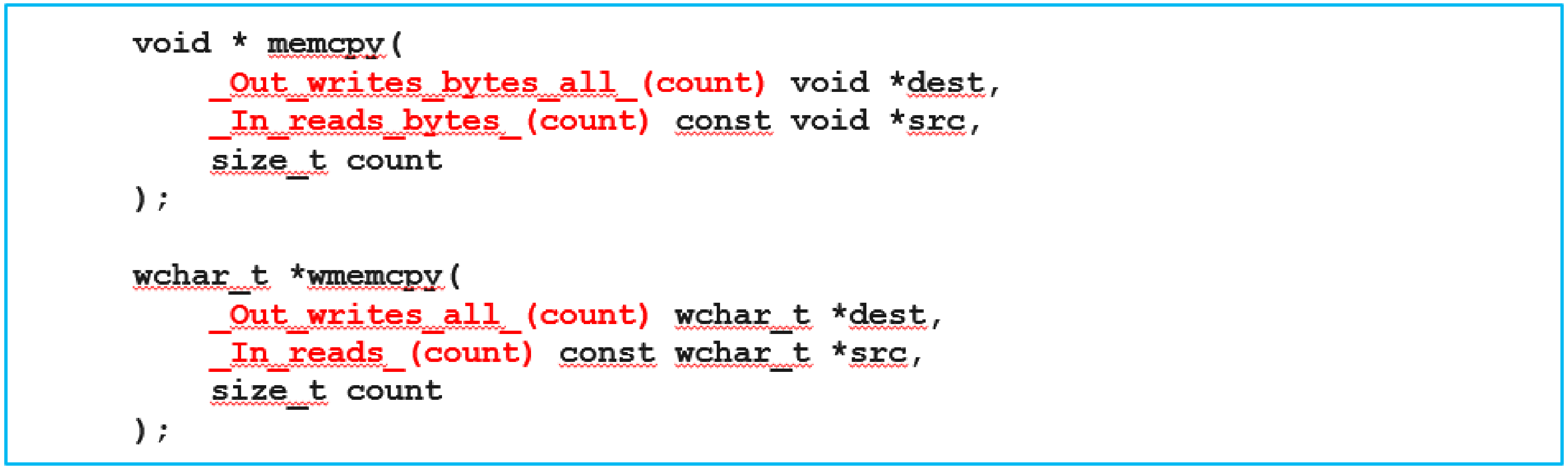
ef

26018	Potential buffer overflow nullterminated	EspX

SAL

Source code Annotation Language

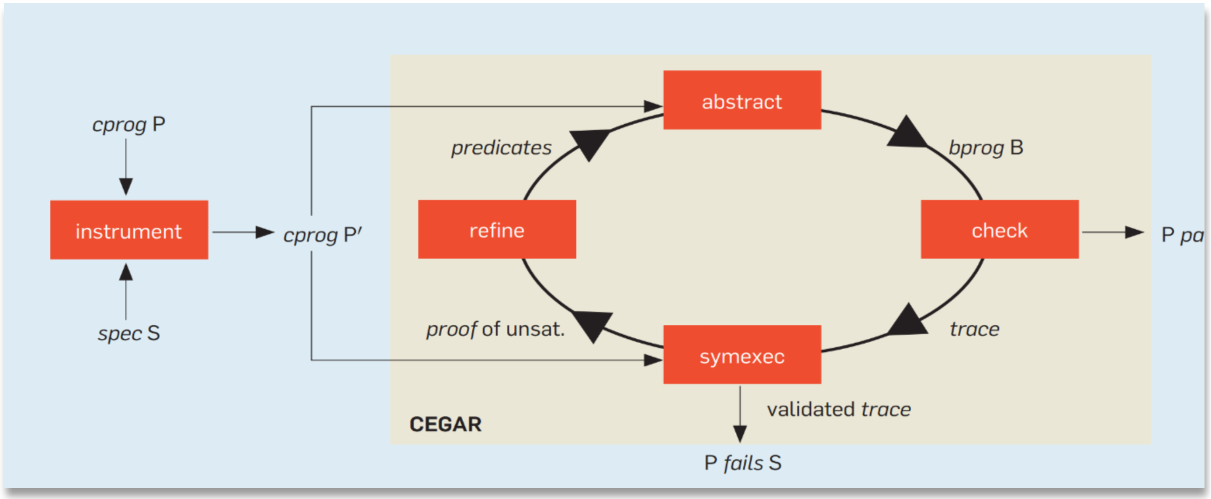
Making programmer intent explicit
Consumed by analysis tools such as PreFast
Over 3 million annotations in Windows alone!
Tens of thousands of bugs found and fixed with help of SAL
Preconditions, postconditions, invariants, concurrency



```
void * memcpy(
    _Out_writes_bytes_all_(count) void *dest,
    _In_reads_bytes_(count) const void *src,
    size_t count
);

wchar_t *wmemcpy(
    _Out_writes_all_(count) wchar_t *dest,
    _In_reads_(count) const wchar_t *src,
    size_t count
);
```

SLAM and Static Driver Verifier



"Counter-example driven abstraction refinement"

SLAM: Implements CEGAR, uses Z3

Boolean program abstraction, CFL reachability

SLIC: Defines safety automaton

e.g. lock is alternately acquired and released

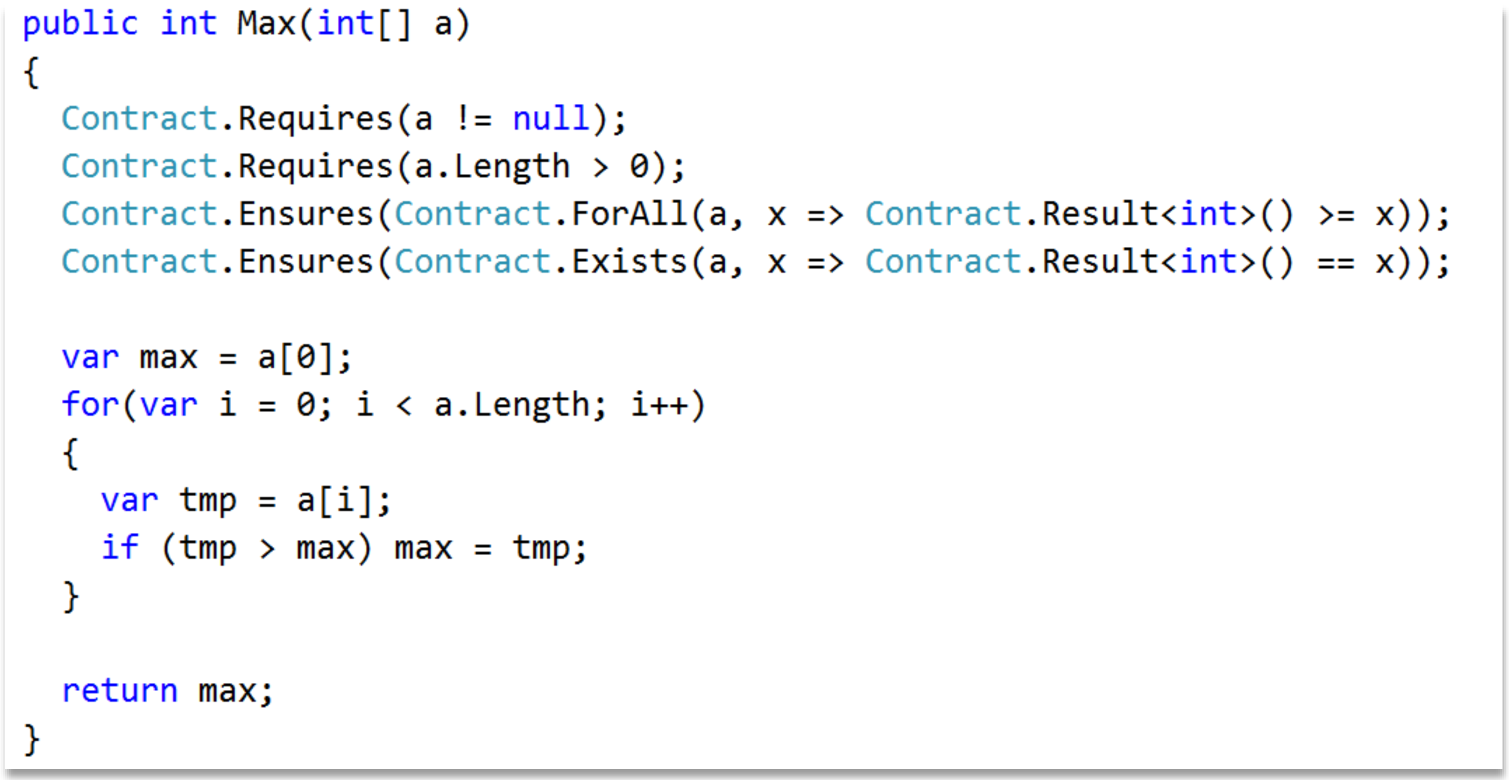
SDV: OS model stubs + ~200 driver rules

Ships externally with WDK

Quality gate for Windows 7 drivers at Microsoft, 270 real bugs found

90-98% of bugs reported are real, nonresults on < 3.5% of runs

Code Contracts



```
public int Max(int[] a)
{
  Contract.Requires(a != null);
  Contract.Requires(a.Length > 0);
  Contract.Ensures(Contract.ForAll(a, x => Contract.Result<int>() >= x));
  Contract.Ensures(Contract.Exists(a, x => Contract.Result<int>() == x));

  var max = a[0];
  for(var i = 0; i < a.Length; i++)
  {
    var tmp = a[i];
    if (tmp > max) max = tmp;
  }

  return max;
}
```

Contracts written in source language (C#, VB, …)

Dynamic or static checking by abstract interpretation (Clousot)

VS plugin

Used inside and outside MS (>120k downloads)

Basis for further work on automated testing, suggested repairs, contract inference, verification modulo versions

Dafny

Imperative language designed for verification of functional correctness

Built on Boogie/Z3, largely automatic

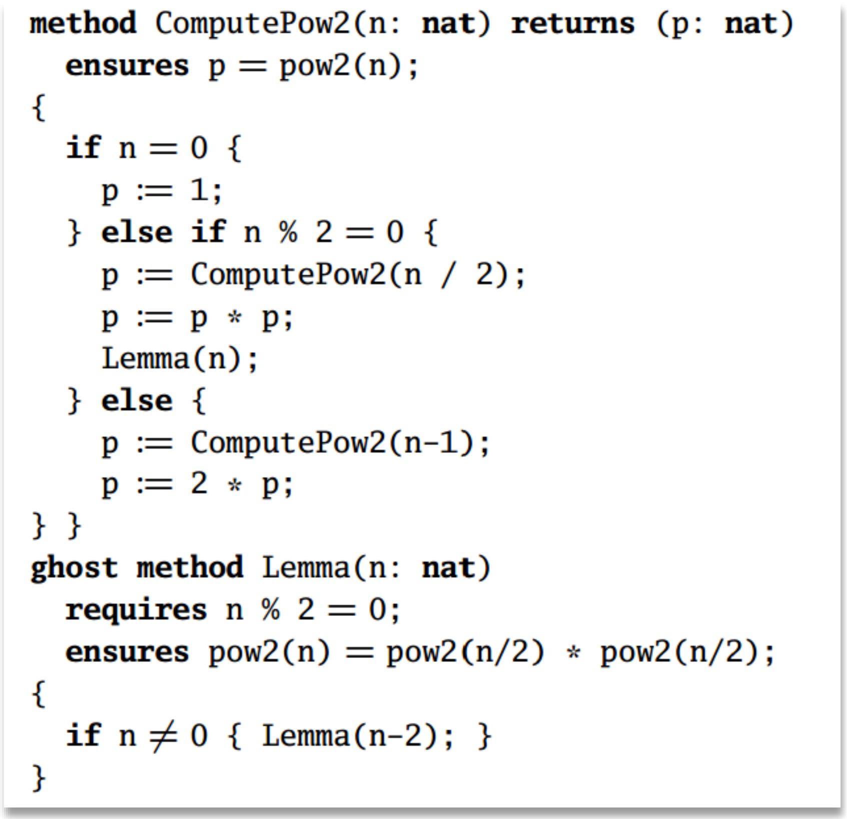
Pre/post, invariants, termination

Ghost variables, sets, sequences, alg types

Dynamic frames

Extensively used in teaching

And other verification projects in MSR



```
method ComputePow2(n: nat) returns (p: nat)
  ensures p = pow2(n);
{
  if n = 0 {
    p := 1;
  } else if n % 2 = 0 {
    p := ComputePow2(n / 2);
    p := p * p;
    Lemma(n);
  } else {
    p := ComputePow2(n-1);
    p := 2 * p;
} }
ghost method Lemma(n: nat)
  requires n % 2 = 0;
  ensures pow2(n) = pow2(n/2) * pow2(n/2);
{
  if n ≠ 0 { Lemma(n-2); }
}
```

x86 Proved

Language

```
Definition allocImp (heapinfo:DWORD)
           (bytes:nat) (fail:DWORD) :=
  mov ESI, heapinfo;;
  mov EDI, [ESI];;
  add EDI, bytes;;
  jc fail;;   (* wrap-around *)
  cmp [ESI+4], EDI;;
  jc fail;;   (* no memory *)
  mov [ESI], EDI.
```

```
Definit
 (proc(
 const
 const
 block
   (var
 if (a=
 then goto k (a,a)
 else goto k (b,b))%twiddle.
```

Specification

```
Definition allocSpec n fail inv code :=
  Forall i, Forall j, (
     safe @ (EIP ~= fail ** EDI?) //\\
     safe @ (EIP ~= j ** Exists p,
                 EDI ~= p +# n **
                 memAny p (p +# n))
   -->>
     safe @ (EIP ~= i ** EDI?))
   @ (ESI? ** OSZCP_Any ** inv)
   <@ (i -- j :-> code).
```

Logic

```
Lemma spec_at_or_and S R1 R2
     {HNeg: AtContra S}:
  S @ (R1 \\// R2) |-- S @ R1 //\\ S @ R2.
Proof.
  rewrite ->land_is_forall, lor_is_exists.
  transitivity (Forall b,
  S @ (if b then R1 else R2)); last first.
  - apply: lforallR => [[|]].
    - by apply lforallL with true.
    - by apply lforallL with false.
  apply: at_ex'.
Qed.
```

Compiler

```
 FTOP dword op dst ShiftCountCL =>
  encodeOpcode dword #x"D2" $$
  writeNext (inj op, dst)

| IMUL dst src =>
  writeNext #x"0F" $$
  writeNext #x"AF" $$
  writeNext (inj dst, src)
```

```
| CON
let:n
let:y
let (
    (
mov E
cmp E
JCC c
);
jmp (nth #0 cmap (size noblocks).-1)
```

Coq

Proof

```
 (* mov [ESI], EDI *)
specintro. move/eqP => Hcarry0.
subst carry0.
specapply MOV_M0R_rule.
- by ssimpl.
rewrite <-spec_reads_frame.
apply limplValid.
autorewrite with push_at.
 apply: landL2. cancel1.
 rewrite /OSZCP_Any /flagAny /regAny
    /allocInv. ssplits.
```

Binary

```
"BB 00 80 0B 00 E9 0B 00 00
 43 01 4F FF C3 FF C3 81 FB
 82 E9 FF FF FF BE 00 80 0B
 00 E9 10 00 00 00 8B 06 89
 00 00 81 C7 04 00 00 00 81
 0F 82 E4 FF FF FF B9 14 00
 00 00 BF D1 06 30 00 C1 E2
  2 03 FA C1 EA 07 C6 04 4F
   00 0F 84 07 00 00 00 FF
   39 00 00 00 00 81 FA 31
   00 00 00 FF C2 E9 05 00
```

x86 Architecture

```
Inductive NonSPReg := | EAX | EBX | ECX |
EDX | ESI | EDI | EBP.
(* General purpose registers,
   including ESP *)
Inductive Reg :=
| nonSPReg :> NonSPReg -> Reg
| ESP.
(* All registers, including EIP
   but excluding EFL *)
Inductive AnyReg :=
| regToAnyReg :> Reg -> AnyReg
| EIP.
```

x86 Semantics

```
| MUL src =>
  let! v1 = getRegFromProcState EAX;
  let! v2 = evalRegMem src;
  let res := fullmulB v1 v2 in
  let cfof := high 32 res == #0 in
  do! setRegInProcState EAX (low 32 res);
  do! setRegInProcState EDX (high 32 res);
  do! updateFlagInProcState CF cfof;
  do! updateFlagInProcState OF cfof;
  do! forgetFlagInProcState SF;
  do! forgetFlagInProcState PF;
  forgetFlagInProcState ZF
```

- Visual Studio Code Analysis
  - http://msdn.microsoft.com/en-us/library/ms182025.aspx
- SAL
  - http://msdn.microsoft.com/en-us/library/ms182032.aspx
- SLAM & Static Driver Verifier
  - http://research.microsoft.com/en-us/projects/slam/
- Code Contracts
  - http://research.microsoft.com/en-us/projects/contracts/
- Dafny
  - http://research.microsoft.com/en-us/projects/dafny/
- x86proved
  - http://x86proved.codeplex.com/