This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

SAND2019-3993C

Safety Threats from Electronic Products:

Software, Firmware, Programmable Logic Devices

Philip Huffman, PE LWP
System Surety Engineering
Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525

SAND2019-XXXX

Sponsored by: NNSA National Nuclear Security Administration

Safety Threats from Electronic Products, Part 1:

Part 1: Escapes, Defects, Design Errors

Sponsored by: NNSA National Nuclear Security Administration

Case Study

- Therac 25: 1985 through 1987

- Atomic Energy of Canada Limited (AECL)
- Entered service in 1983
- Based on Therac-6 and Therac-20
- Used PDP-11 computer system
- Lacked hardware interlocks from previous
  - Hardware interlocks removed
  - Manual controls removed
  - Relied on computer-controlled servos
  - Less setup time: major benefit for hospitals

Leveson, N.G., “An Investigation of the Therac-25 Accidents,” Computer, (26), 7 July 1993.

Case Study

- Therac 25: 1985 through 1987
  - Patient 1: Marietta
    - Tremendous rush of heat, red hot sensation,
    - swelling, redness, spasms
    - Loss of breast, loss of use of shoulder and arm
    - Dosage determined to be 15,000-20,000 rad
  - Patient 2: Ontario, Canada
    - Error message, machine indicated “no dose”
    - Operator re-attempted treatment 5 times, each showing “no dose”
    - Described as a “normal issue” with the machine
    - Patient experienced “electric tingling shock”,
    - hospitalized for radiation exposure, died 3 months later

Case Study

- Therac 25: 1985 through 1987
  - Patient 3: Yakima Valley
    - Patient experienced burns in a “striped pattern”
    - Determined to be radiation exposure over a year later
    - Required surgery, experienced minor disability and scarring
  - Patient 4: Tyler, Texas
    - Error message, machine indicated “underdose”
    - Operator re-attempted treatment
    - Patient experienced “electric shock, burning” after first attempt
    - While trying to exit the bed, received the second exposure on arm
    - Lost use of arm, legs, speech, died 5 months later

Case Study

- Therac 25: 1985 through 1987
  - Patient 5: Tyler, Texas
    - Error message, machine indicated “underdose”
    - Operator heard loud moan from patient on intercom
    - Patient experienced dose to side of head
    - fell into coma, severe neurological damage, died 3 weeks later
  - Patient 6: Yakima Valley
    - Error message, operator repeated procedure, second error message
    - Patient complained of “burning sensation”
    - Died 3 months later, complications from overexposure

Case Study

- Therac 25: Manufacturer's response
  - Patient 1
    - Improper dose was "impossible"
  - Patient 2
    - Sent service engineer, could not reproduce malfunction
    - Problem identified with turntable switch, voluntary recall initiated
  - Patient 3
    - Notified hospital in writing, overdose was "impossible", there had been no other incidents
  - Patient 4
    - Suggested electrical malfunction, machine checked and no issues found
    - "could not possibly overdose a patient", said no other incidents reported

Case Study

- Therac 25: Manufacturer's response
  - Patient 5
    - A hospital physicist had managed to reproduced the error message response, problem related to rapid entry/changing of system parameters
    - Simultaneously, FDA had determined the Therac-25 was defective and required corrective action plan
    - Responded with thorough review due to FDA investigation of the system, but sent an "understated" letter to users.
      - FDA concluded the letter was inadequate and not commensurate with the urgency of the problem
    - Added new circuit requiring reset if high pulse detected. No hardware safety interlocks addressed
      - Declared machine now "many times safer" and "virtually eliminated" possibility of overdose
      - FDA unsatisfied, concerned with software engineering practices, absence of documentation, testing
      - FDA required extensive testing on any software change with formal test plan
  - Patient 6
    - FDA declared the system defective, sent letter to all users
    - Corrective action included software/hardware changes, hardware interlock

Case Study

- Therac 25: Investigation
  - Multiple software defects associated with the control software
    - Same defects identified on the previous model, Therac 20, but hardware safety devices prevented overdosing.
    - Software Error: Entering treatment data
      - Any changes within 1-8 seconds of initial entry caused radiation level and aiming error
    - Software Error: Real time controller
      - Turntable set to test mode to check beam alignment, increments alignment variable
      - Every 256th increment resulted in zero assigned, presented full strength beam with no attenuation
  - Poor software engineering, quality assurance
    - Systems typically produced up to 40 error messages each day
    - No requirements specification, no formal software or system testing
    - Correcting errors introduced new errors
    - Manufacturer's denials of possibility of malfunction or overdose
  - Slow FDA Response
    - Trusted inflated reliability/safety statistics from manufacturer

Definitions

- Software
  - ANSI N42.42-2006

    A set of instructions that tell a computer what to do
  - ISO/IEC 26514:2008

    Program or set of programs used to run a computer
  - IEEE Std 7-4.3.2-2010

    Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system

Definitions

- ## Software Product

A software deliverable typically consisting of an executable software program and associated operation data from a product realization process.

Software product may include:

- Software requirements specifications,
- Design documentation,
- Source code,
- Executable code,
- Operational data,
- Maintenance procedures,
- Test plans,
- Quality plans,
- Configuration management plans, and
- Other variations of the pieces.

Safety Threats from Electronic Products

CASE STUDIES

Case Study

- 1982 Trans-Siberian Gas Pipeline Explosion
  - 2900 miles, 41 compressor stations
  - December 1981: small fire at a compressor station destroyed monitoring devices, control panels
  - October 1982: massive explosion detected by satellite
  - Software reported as faulty, causing valves and pump malfunctions
  - Continual build-up in pipeline pressure prior to explosion
  - 100 ft section of pipeline exploded
  - No physical casualties, but major financial contribution to fall of Soviet Union

"The Soviet Gas Pipeline in Perspective," Special National Intelligence Estimate, SNIE 3-11/2-82, Director of Central Intelligence, 21 September 1982.
"Soviets Burned by CIA Hackers?", Wired, 26 March 2004.
Hill, G., "Review of the 1982 Soviet Explosion," https://blogs.ncl.ac.uk/ghill4/sample-page/review-2/.

Case Study

- 1983 World War III – Almost…
  - September: Soviet early warning system identified 5 ICBM launches from the US.
  - Software detected rare alignment of the sun with high level clouds
  - The sun's reflection was interpreted as launch
  - Soviet colonel Stanislav Petrov prevented full scale retaliatory launch because "it didn't feel right"
    - Launch detection system was new
    - Nothing detected by ground radar
    - "why only launch 5?"
  - Review of the system found additional software defects, embarrassed his superiors and the scientists that developed it

Thomson, I., "30 Years On: The Day a Computer Glitch Nearly Caused World War III," The Register, 27 September 2013.

Case Study

- 1990 AT&T Telephone System
  - January afternoon, 72 screen video array representing network displayed rapidly spreading tangle of red malfunctions
  - Lost 50% of long distance service for nine hours
  - New York switch performing routine self test
    - Indicated nearing load limits
    - Performed reset and sent error message to accept no more calls
    - Second message sent within 10 mS indicating back online
    - At downstream switches, software defect couldn't handle messages this close, second message overwrote crucial data
    - Software detected overwrite and performed a reset, also sending a message from downstream switches
    - Domino effect through 114 switches on the network
  - Software design was highly tested, possessed many fault tolerant features
    - Fault was in a fault monitor/error handling routine
    - Rare event for AT&T system

at&t

Neumann, P.G., "The Cause of AT&T Network Failure," The Risks Digest, Vol. 9, Issue 62, 26 February 1990.

Case Study

- 1991 Patriot Missile System
  - February 1991 during Gulf War, off-track Patriot allowed Scud missile to strike army barracks in Dhahran
    - 28 American soldiers killed
    - 97 people injured
  - Range gate failed to predict future location of Scud
    - Software stored time in tenths of a second as integers
    - 24-bit registers allowed only 14 hours operational time
    - Actual use was over 100 hours
  - Software precision degraded with length of time it had been running

Schmitt, E., "U.S. Details Flaw in Patriot Missile," The New York Times, 6 June 1991.

Case Study

- 1992 A320 “Fly by Wire” system
  - First fully “fly-by-wire” airplane.
  - Four crashes between 1988 and 1992
  - “Safety Feature” in the autopilot increased the descent rate on landing – 4x higher
  - Investigation found lack of checks on output of automated actions
  - In another of the crashes, the “fly-by-wire” system did not recognize the plane had landed, blocking attempts at reverse thrust to brake the plane

Hamelink, C.J., The Ethics of Cyberspace, Sage Publications, 1999.

Case Study

- 1996 Ariane 5 Flight 501
  - Ariane 5 generates 64 bit float values,
    converts to an integer, which overflowed
    the 16 bit variable representing horizontal bias
    - Overflow caused an error trap to execute that cleared the memory
    - Memory dump interpreted by the rocket as instructions to the rocket nozzles
  - Controller trapped the overflow error and crashed during launch,
  - System switched to backup computer
  - Backup had already crashed – running the same software (redundancy)
  - Failure 37 seconds into flight, $370 million loss

Lions, J.L., "Ariane 5 Flight 501 Failure, Report by the Inquiry Board," European Space Agency, July 1996.

Case Study

- 1997 USS Yorktown
  - IT Modernization program, “smart ship”
    - Automate navigation
    - Monitor sensor systems
    - Control machinery and fuel
  - Fuel valve was closed, but system said open
  - The RDM manager program attempted a division operation, throwing divide-by-zero error, then crashed
  - Failure dominoed through other smart ship systems, including motor and propulsion control
  - Ship dead in the water for 3 hours before returning to port under emergency power
  - Input values not validated, error handling nonexistent

Slabodkin, G., “Software Glitches Leave Navy Smart Ship Dead in the Water,” Government Computing News, 13 July 1998.

Case Study

- 2003 Northeastern Blackout
  - 3500 megawatt surge
  - Over 30 minutes impacted Ohio, New York, Michigan, New Jersey, Ontario
  - Race condition within the Unix-based energy management system
    - First Energy, Ohio
    - Software defect stalls alarm event processor
    - Operators deprived of alerts
    - Unprocessed events enter queue, leading to server failure
    - Control room itself loses power
  - Ohio plant drops, 3 138 kV lines overvoltage and drop out, domino effect
  - 256 plants drop offline, grid separation occurs
  - 100 deaths (transportation, fire, electrical, medical)

Technical Analysis of the August 14, 2003, Blackout, Report to the NERC Board of Trustees by the NERC Steering Group, July 13, 2004.

Case Study

- 2015 Airbus A400M Crash
  - Crash occurred in Spain killing four during test flight
  - Investigation showed software defect caused power off in three engines
    - Electronic Control Unit (ECU), one per engine
    - Fuel tank trim control shut off fuel to three engines
    - Error in the configuration setup, parameter data had been erased
    - “quality escape” at the Airbus assembly facility attributed to weakness in the software testing

Hepher, T., “A400M Probe Focuses on Impact of Accidental Data Wipe,” Reuters, 9 June 2015.

Case Study

Software Failures of 2014-2017

- O2 lost access to data services
- TSB Bank customers locked out of accounts for months
- Welsh NHS IT failure unable to access patient files
- Meltdown & Spectre : HW vulnerabilities
- WannaCry ransomware
- Cloudflare software error releasing customer passwords, data,
- Bitcoin memory leak affecting 70% of service
- British Airways global IT failure, over 1000 flights
- Nest Thermostat left users in the cold, drained devices batteries,
- HSBC outage, 2 day unable to access accounts
- Glitch releases 3200 prisoners early , been going for 13 years
- Bloomberg cancelled debts
- RBS 600,000 payments disappear
- Nissan Airbag glitch, would not inflate
- F35 detects targets incorrectly
- Amazon prices drop to 1cent before Christmas
- UK Airspace closed
- Toyota prius recall
- Apple IOS 8 upgrade: loss of signal, phone freeze
- 7 states lose emergency services for 6 hours

Safety Threats from Electronic Products

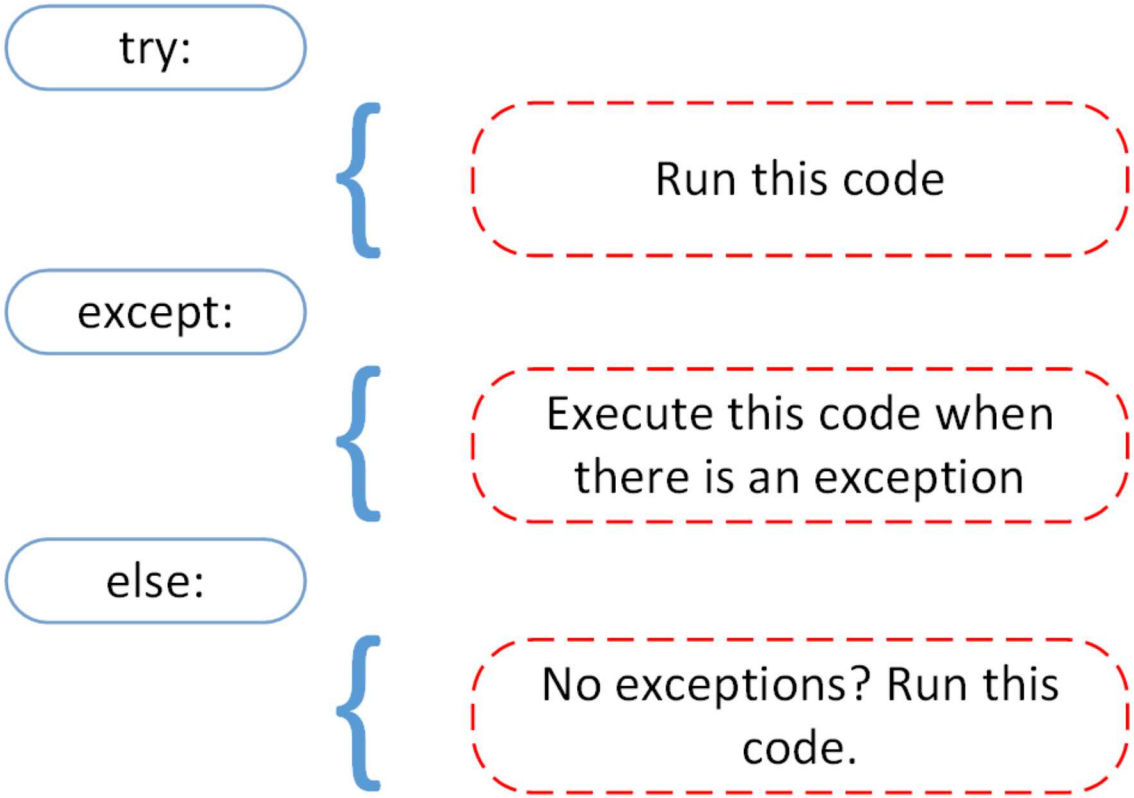
FAILURE MECHANISMS

- Types of Software Failures
  - Functional
    - Calculation
  - Error handling
  - Boundary conditions
    - Variable conversion
    - Buffer overflow
  - Race conditions
  - Hardware compatibility
  - User interface

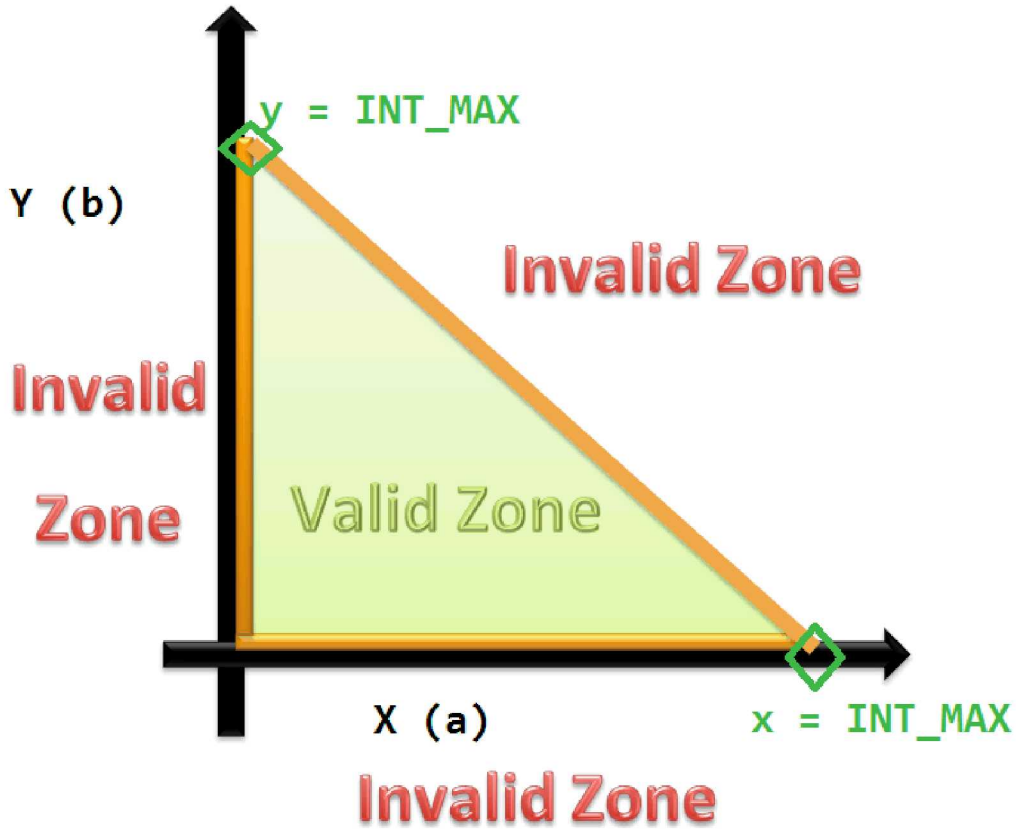
Functional Errors

I felt a great disturbance in the force. As if someone had just attempted something and a million integers cried out at once and then were silenced.

Error Handling



Boundary Conditions



Boundary Conditions

32 - bit

0000 0000 0000 0000 0000 0000 0000 0001

return 1000 0000 0000 0000 0000 0000 0000 0000

---

64 - bit

0000 0000 0000 0000 0000 0000 0000 0001

return 0000 0000 … 0000 0000 0000 0000 0000 0000 0000 0001

Boundary Conditions

X strcpy writes entire string, overwriting whatever is after the destination string

✓ strlcpy truncates and adds terminating null character

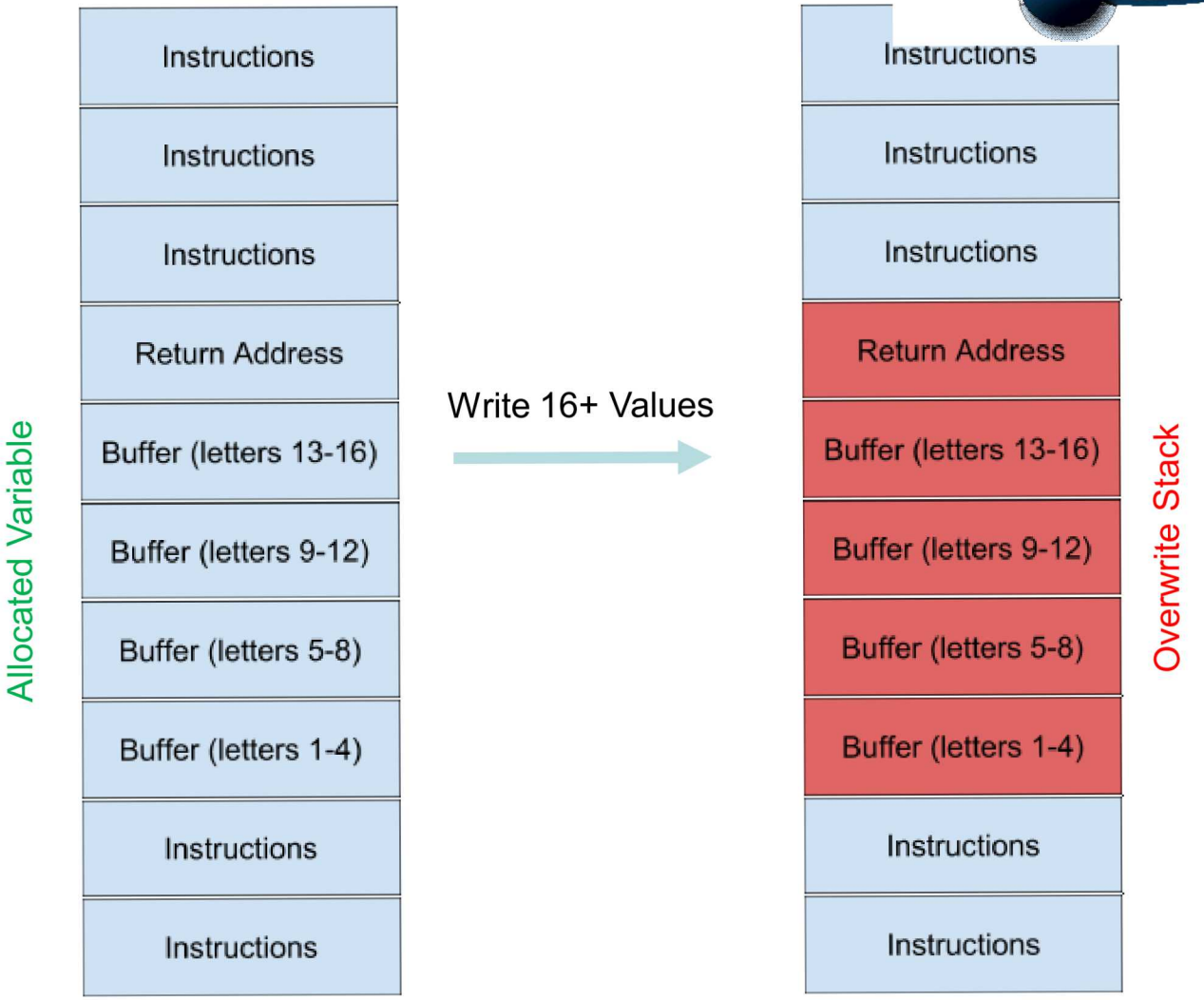
```
Char destination[5]; char *source = "LARGER";

strcpy(destination, source);
```

L	A	R	G	E	R	\0	

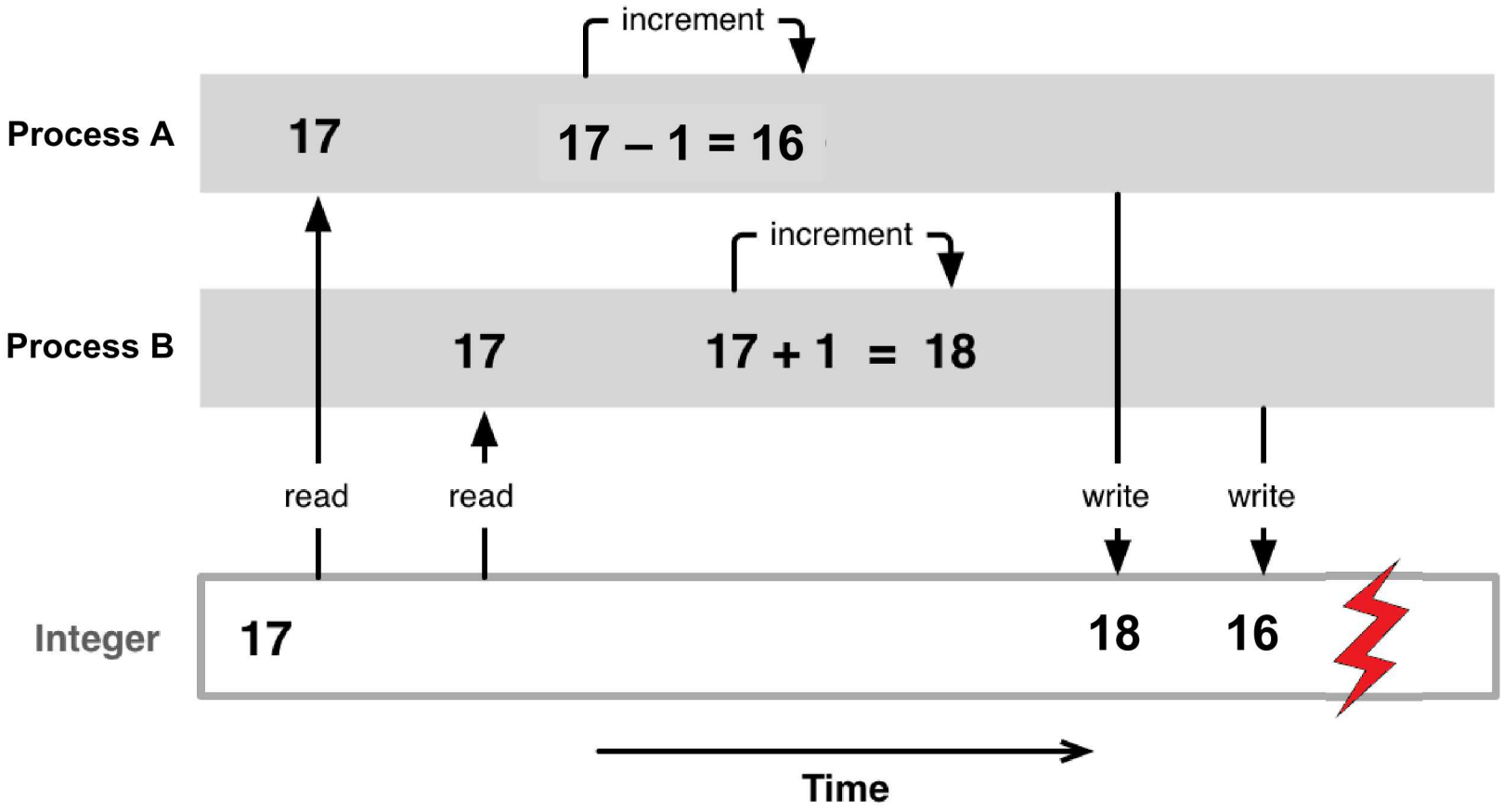
```
strlcpy(destination, source, sizeof(destination));
```

L	A	R	G	\0			

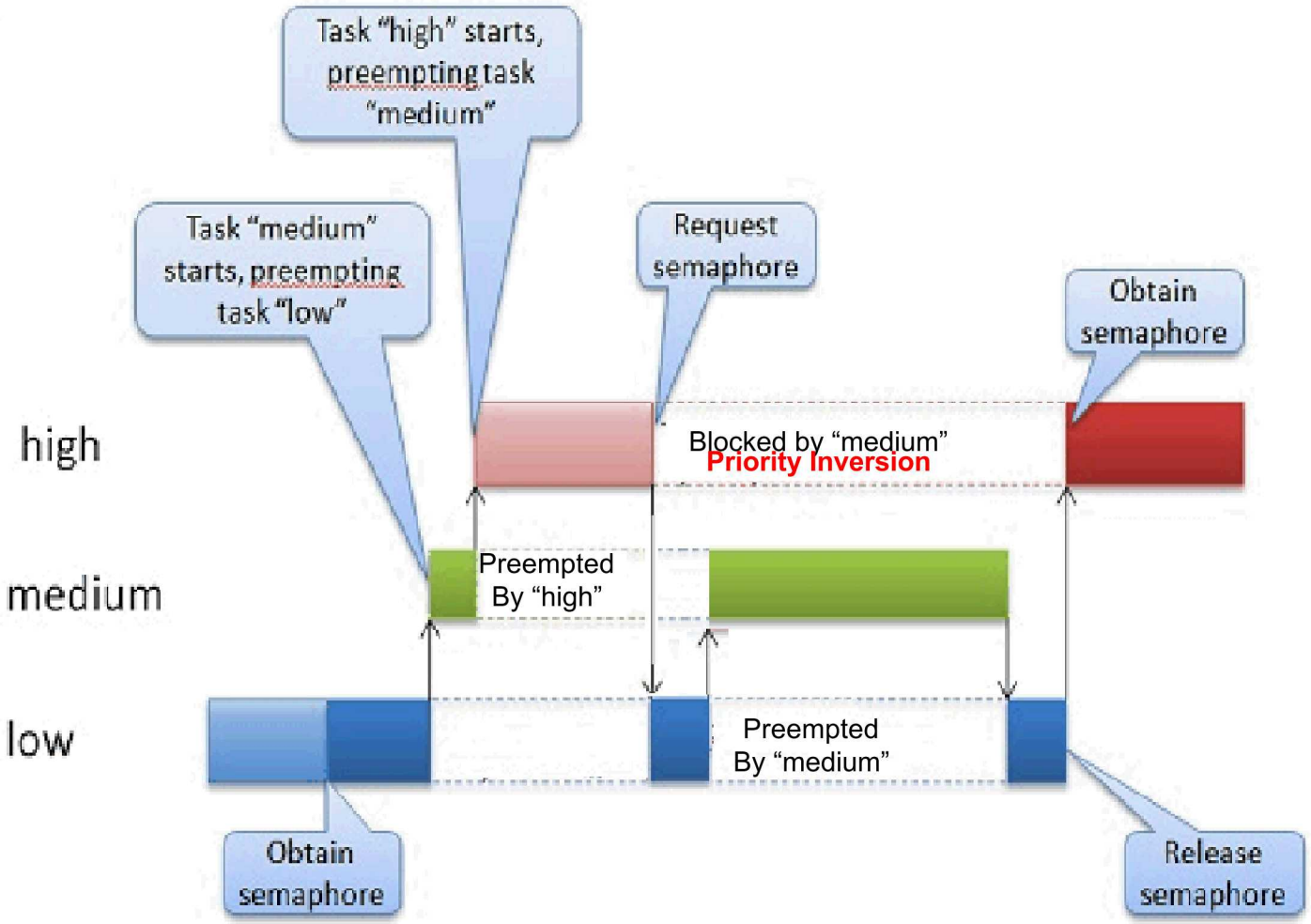
Boundary Conditions



Race Conditions

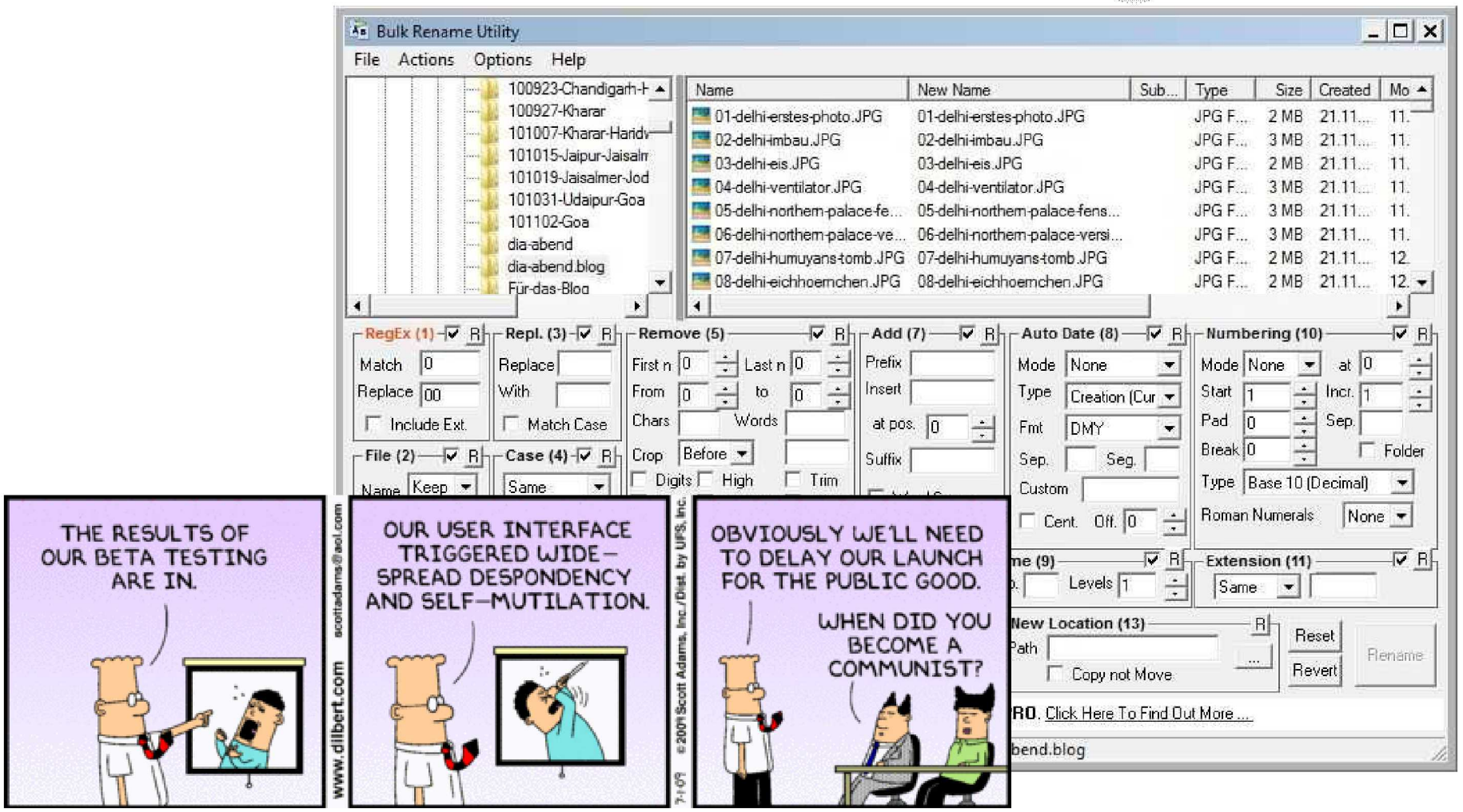


Race Conditions



Hardware Compatibility

User Interface



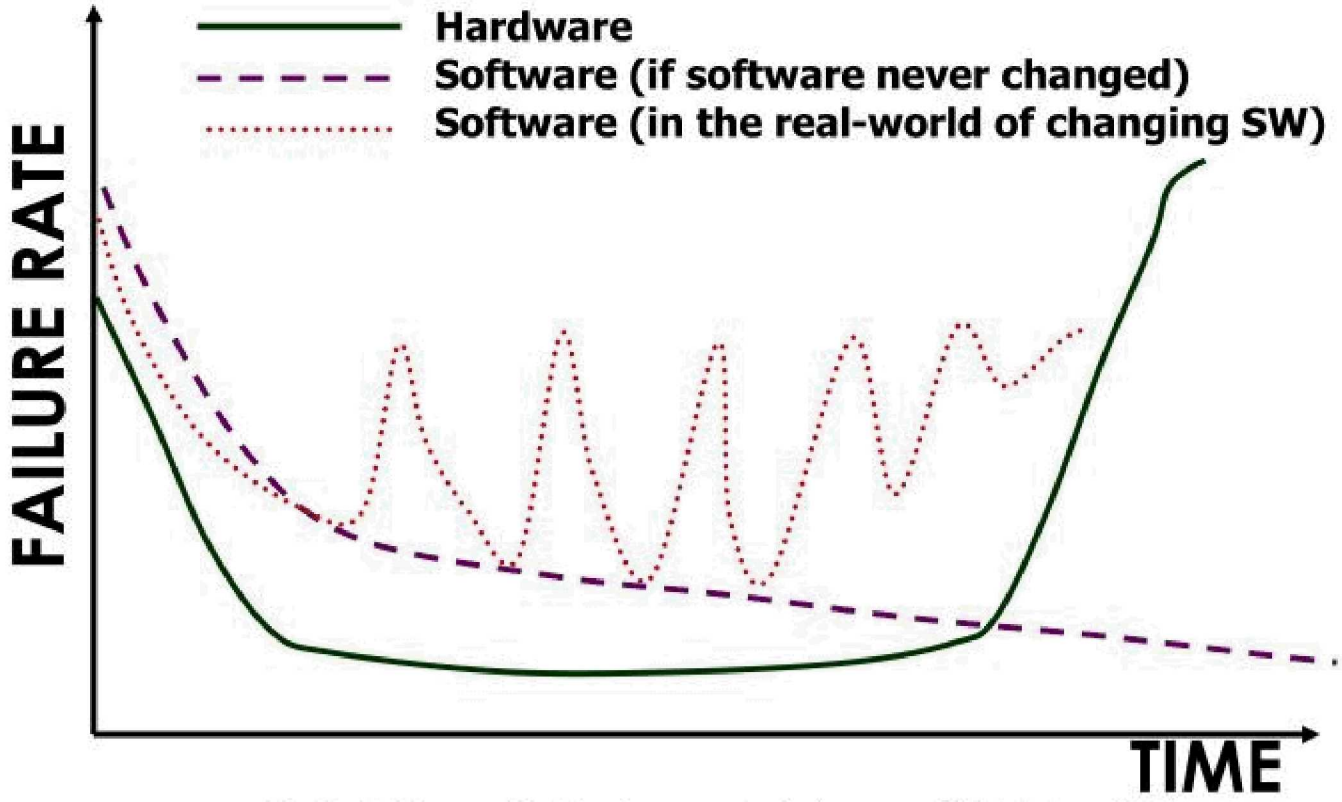
Safety Threats from Electronic Products

MITIGATION

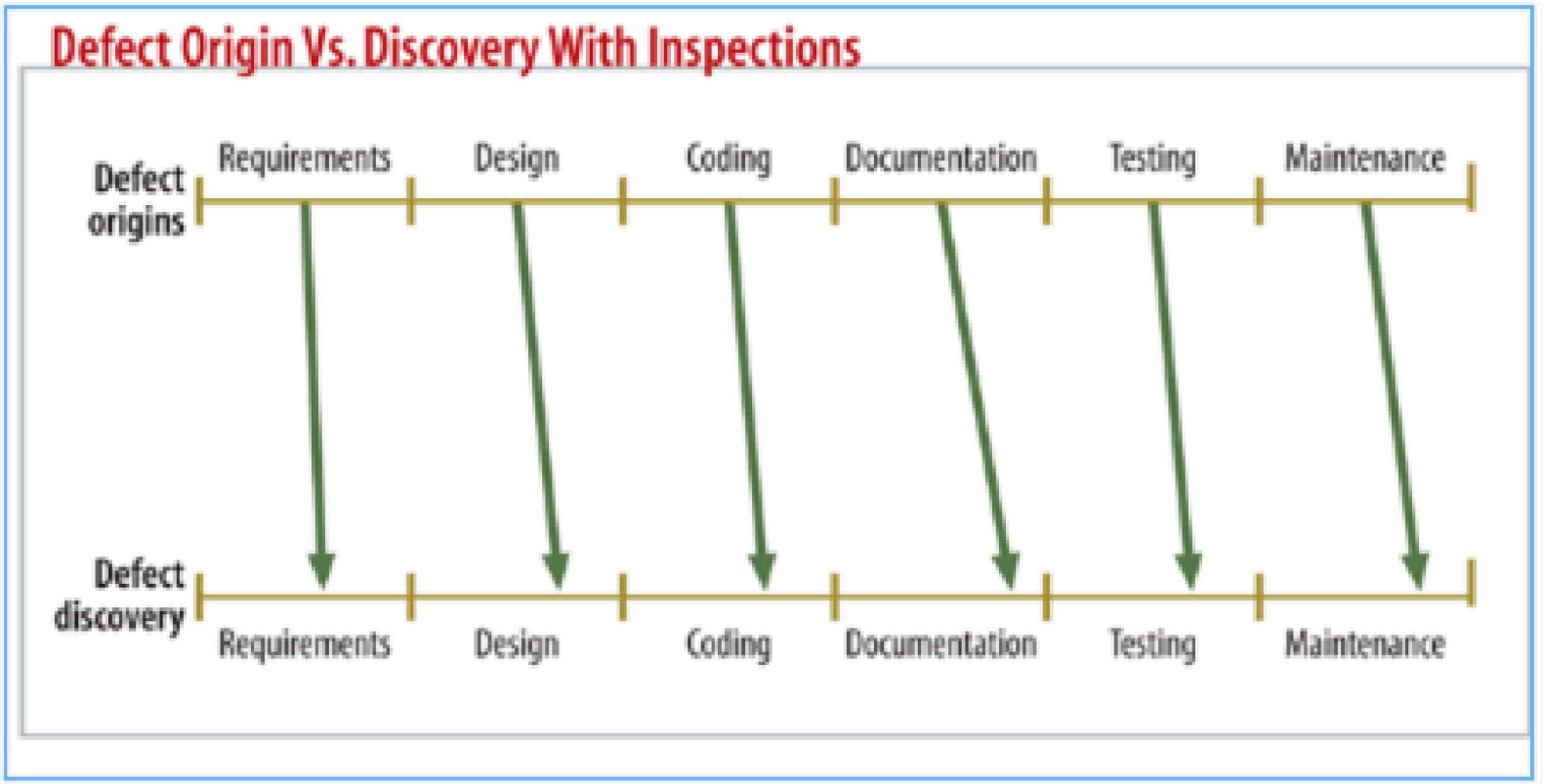
Verification & Validation

- The Verification Dilemma

Verification & Validation



Verification & Validation



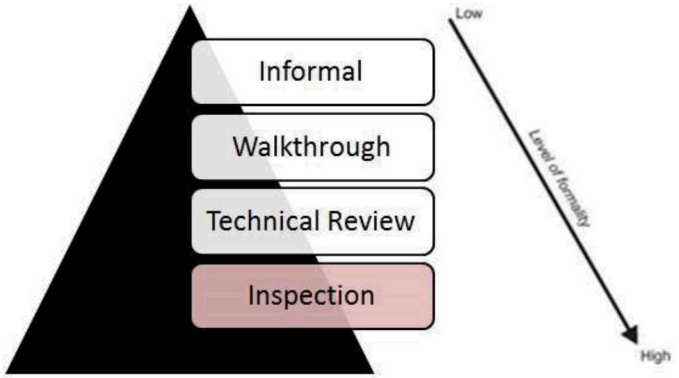
Verification & Validation

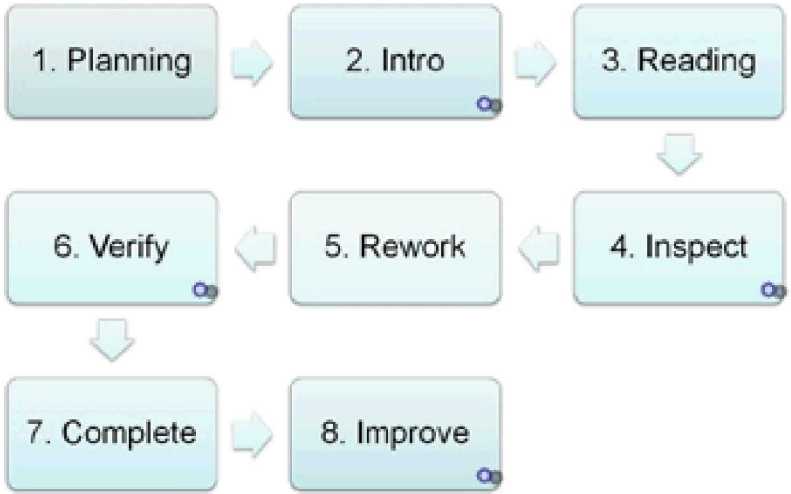
- Static Verification
  - Formal Inspection
    - Critical reading and analysis of code
      - Multiple inspectors
      - Designated reader/recorder
      - Designated Arbiter
    - Formal follow-up
  - Technical Reviews
  - Walkthroughs
    - Error detection through group code reading
  - Desk Checking
    - One-person version of walkthrough

Static Verification

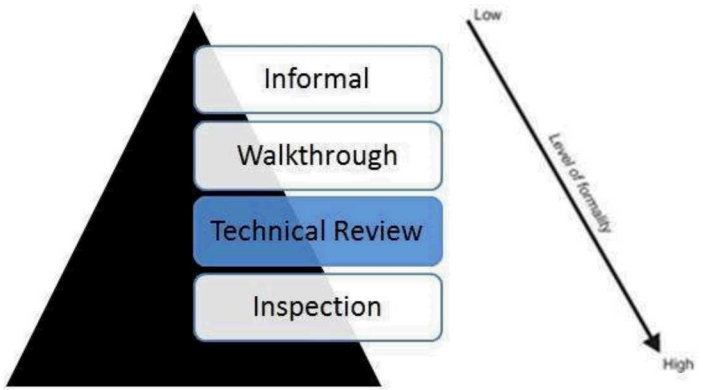
Formal Inspection

- Specific Roles
- Moderator Led
- Defined Process
- Formal inputs and follow-up
- Corrective actions





Static Verification



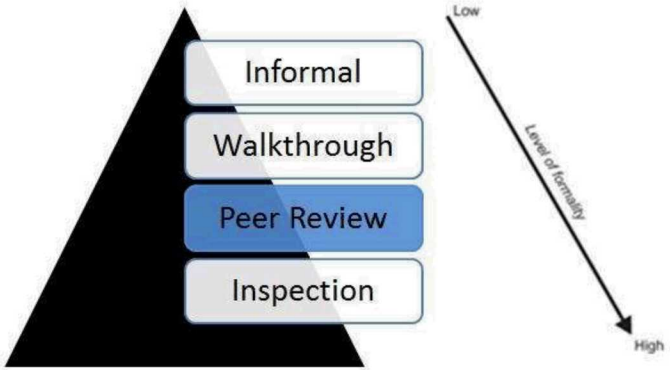
Technical Reviews

- Advanced preparation
- 3-5 people involved
- Developer Led
- Formal process
- Focus on specific part of software

Static Verification

Code Walkthroughs

- Peer Review
- Developer Led
- Informal process
- “read through the code” and ask questions



Static Verification

Desk Checking

- One person, usually developer
- Manually step through code
- Informal Review
- Usually limited to a specific function or algorithm

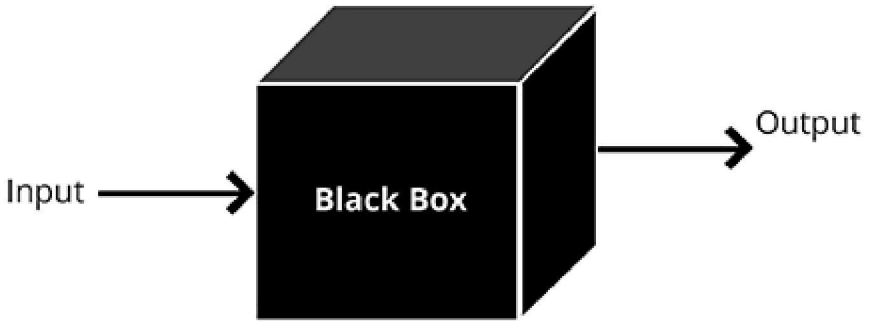
Dynamic Verification

- Dynamic Testing
  - Black-box/functional testing
    - Correct handling of external functions
    - Reduce risk of functional problem at the end user
  - White-box/structural testing
    - Correct implementation of internal units, structures, interfaces between them
    - Detect/remove failures internal to the implementation

Dynamic Verification

Black Box Testing

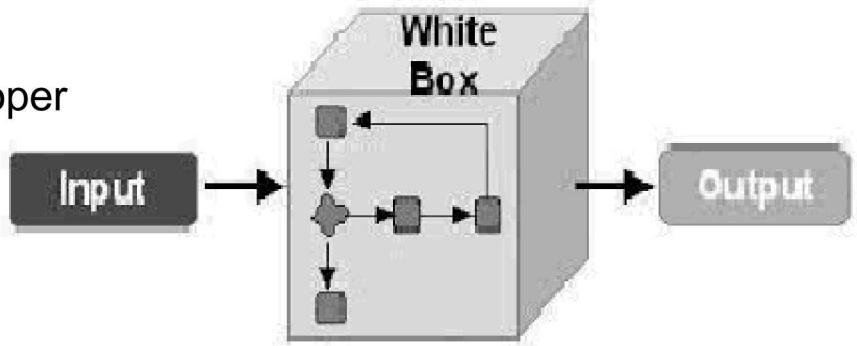
- No knowledge of inner workings
- Only “expected” inputs injected
- Specification driven
- Identifies:
  - Functional errors
  - Performance errors
  - External boundary errors
- Does not identify:
  - Latent errors
  - Intrinsic defects
  - Program path problems



Dynamic Verification

White Box Testing

- Multilevel Testing
  - Units, Functions, Algorithms
- Much better coverage
  - Statement coverage
  - Branch coverage
  - Path coverage
- Fault injection/Error Handling/Dead Code/Poor Coding
- Disadvantages
  - Time Consuming
  - Requires Knowledgeable Developer
  - Not Comprehensive



Failure Containment

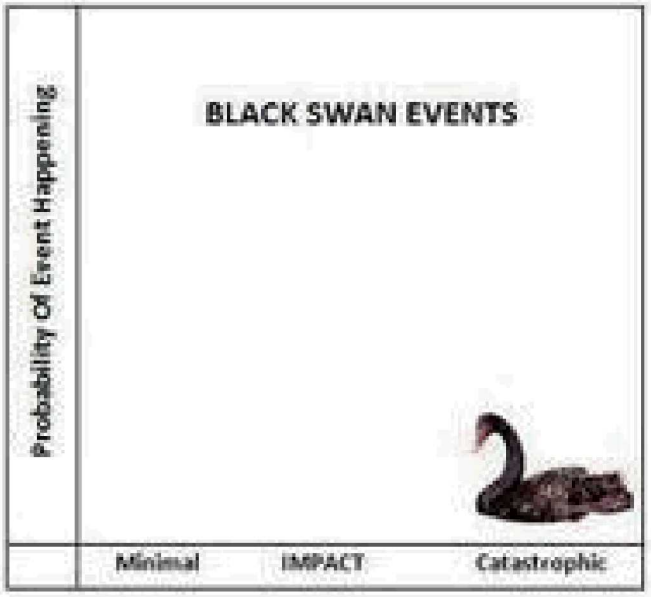
Software will fail

- Software defects cannot be eliminated do to size and complexity
- Must consider rare events, unusual dynamic scenarios
- Techniques to break the causal relation between faults and failures
  - Formal analysis
  - Error detection
  - Temporal monitoring – watchdogs
  - N-version Replication
  - Fault Injection
- Measures to avoid catastrophic consequences

Failure Containment

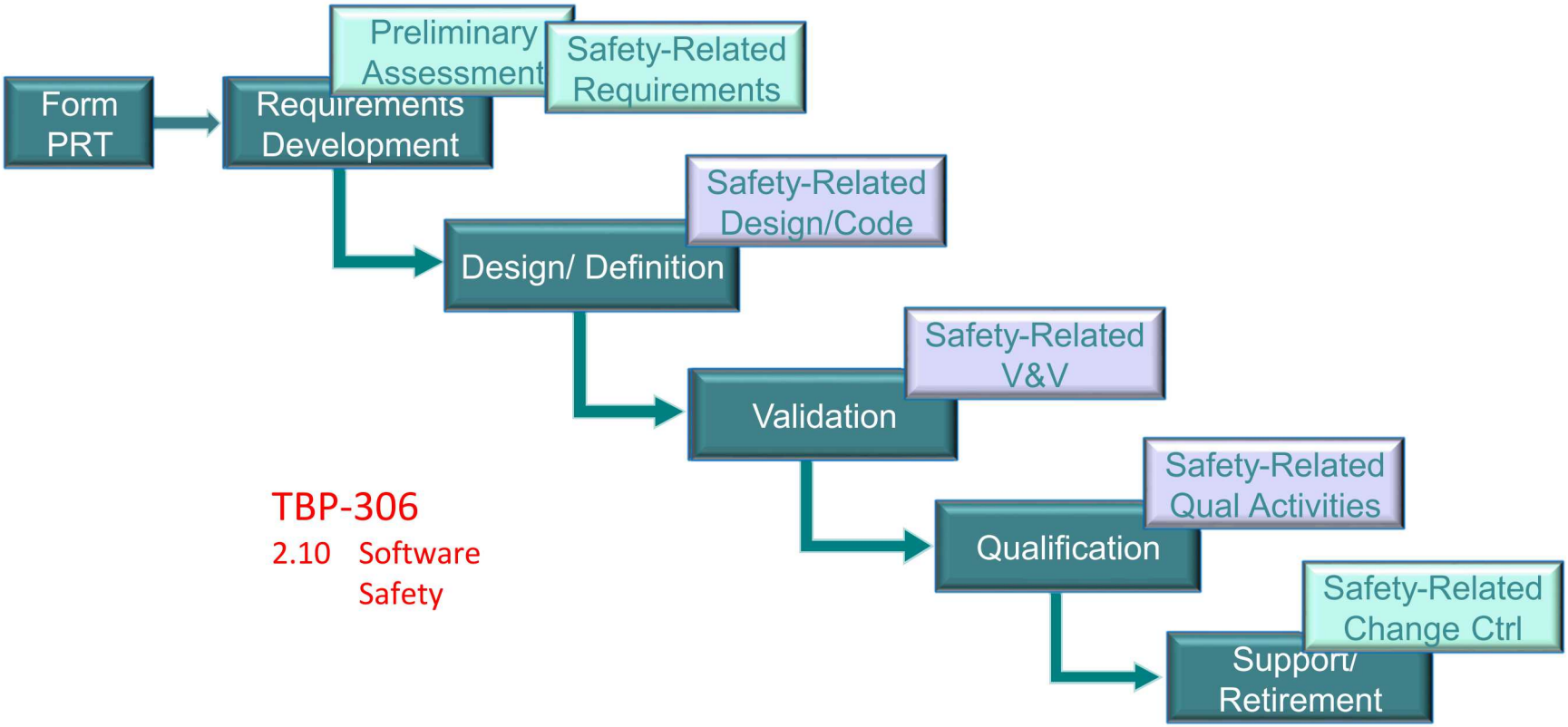
Rare Events

- Cannot be detected through testing because of the volume of cases required for the events to show
- Single and Dual failure modes, surveillance activities a must
- Near misses are indicators, require formal review and response



Failure Containment

Formal Analysis



TBP-306

2.10 Software Safety

Failure Containment

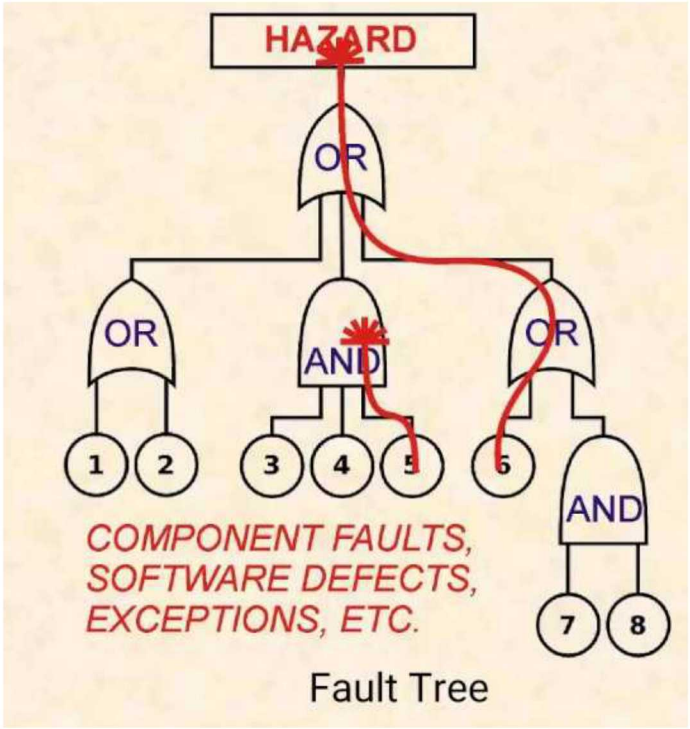
Formal Analysis

SOFTWARE SAFETY CRITICALITY MATRIX				
SEVERITY CATEGORY				
SOFTWARE CONTROL CATEGORY	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
1	SwCI 1	SwCI 1	SwCI 3	SwCI 4
2	SwCI 1	SwCI 2	SwCI 3	SwCI 4
3	SwCI 2	SwCI 3	SwCI 4	SwCI 4
4	SwCI 3	SwCI4	SwCI 4	SwCI 4
5	SwCI 5	SwCI 5	SwCI 5	SwCI 5
SwCI	LEVEL OF RIGOR TASKS			
SwCI 1	Program shall perform **analysis of requirements, architecture, design and code**; and conduct **in-depth safety-specific testing.**			
SwCI 2	Program shall perform **analysis of requirements, architecture, and design**; and conduct **in-depth safety-specific testing.**			
SwCI 3	Program shall perform **analysis of requirements and architecture**; and conduct **in-depth safety-specific testing.**			
SwCI4	Program shall conduct **safety-specific testing.**			
SwCI 5	Once assessed by safety engineering as Not Safety, then no **safety specific analysis or verification** is required.			

Failure Containment

Fault Mitigation

- Work forward from fault to mishap (SWFMEA)
- Work backward from hazard to causes (FTA)
- Identify/Mitigate single point failures
- Implement Mitigation
  - Independent
  - Redundant



Fault Tree

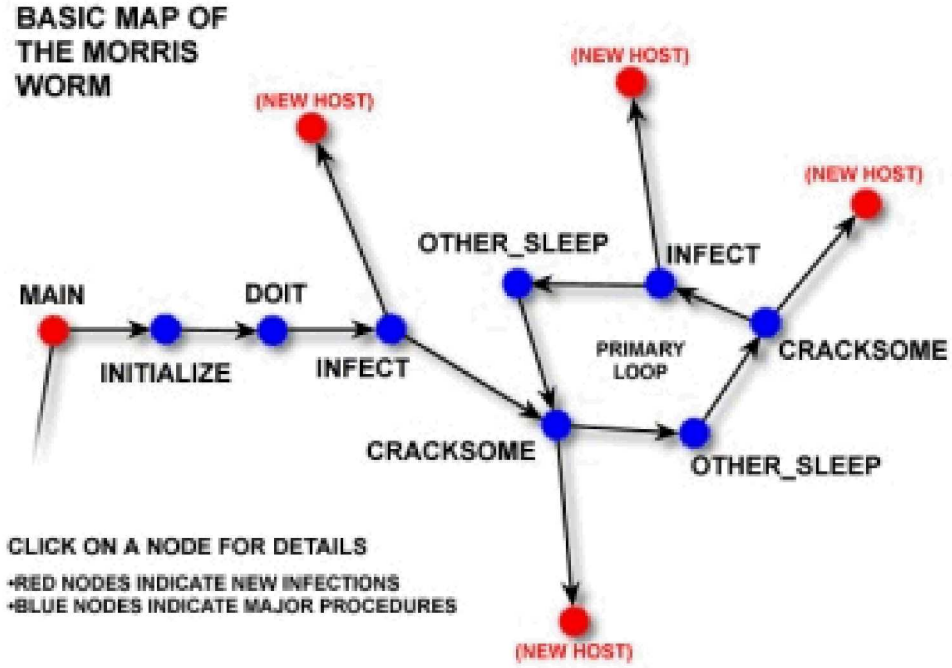
Safety Threats from Electronic Products, Part 2:

DELIBERATE ACTS

Intent

The Morris Worm

- 1988, Robert Morris
- A science experiment gone wrong: attempt to map connected systems
- 6000 University and Military systems: 10% of connected systems



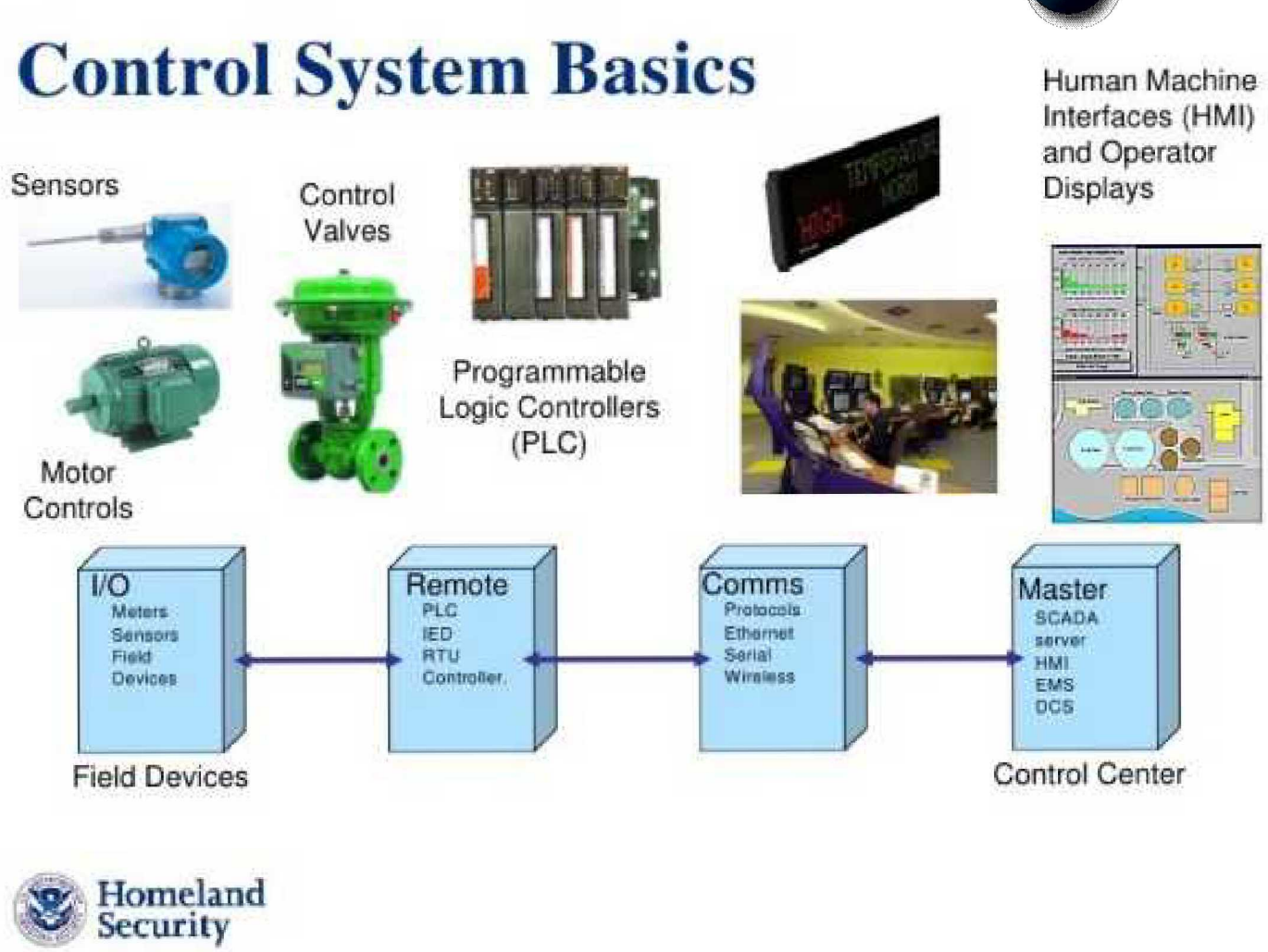
The Morris Worm: 30 Years Since First Major Attack on the Internet, 2 November 2018, https://www.fbi.gov/news/stories/morris-worm-30-years-since-first-major-attack-on-internet-110218

Intent - Stuxnet

- 100% Coded Weapon
  - Penetration
    - Penetrated across 3 platforms: Windows, Step 7, Siemens PLC
  - Observation
  - Attack
    - 32 Payload options
    - Identified/Neutralized multiple types of anti-malware software
- Timeline:
  - June 2009: Earliest detected copy, Attack wave 1
  - March 2010: Attack wave 2
  - April 2010: Attack wave 3
  - July 2010: Reports of infections in Siemens SCADA systems
- 130,000 systems infected

W32.Stuxnet Dossier, v.1.4, Symantec Security Response, 11 February 2011.

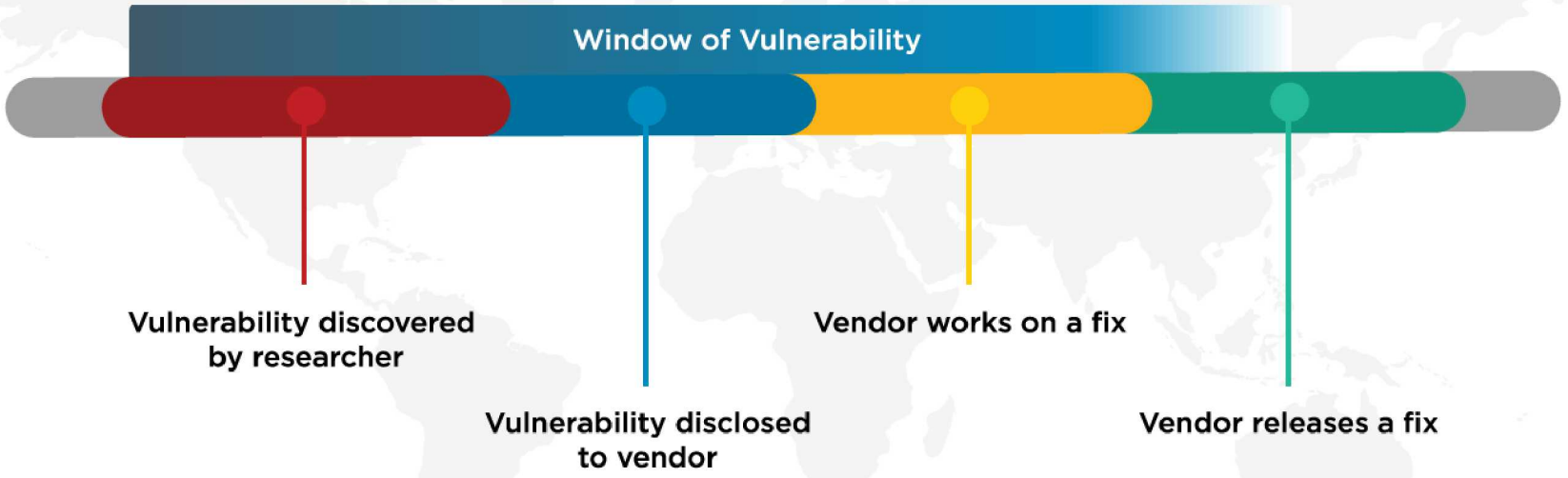
Intent - Stuxnet



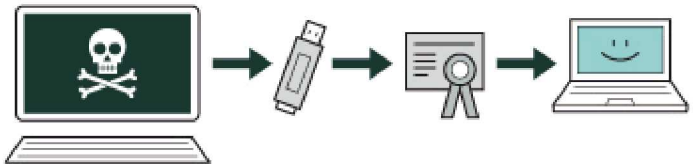
Homeland Security

Intent - Stuxnet

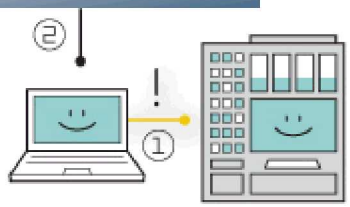
What is a Zero Day Exploit?



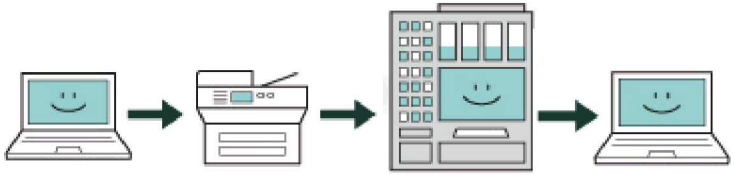
Intent - Stuxnet



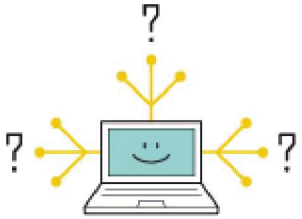
Infection



Communicate
Update

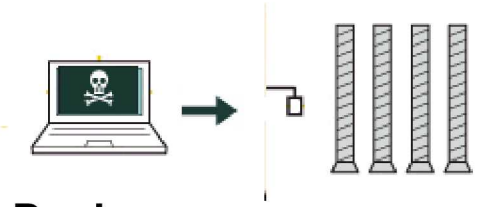


Spread

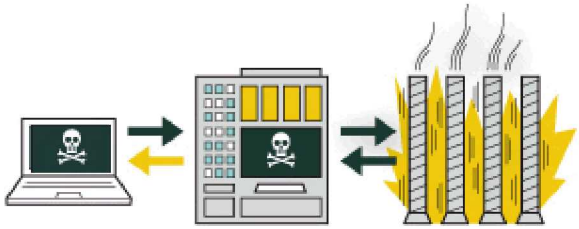


Search

Penetrate



Deploy



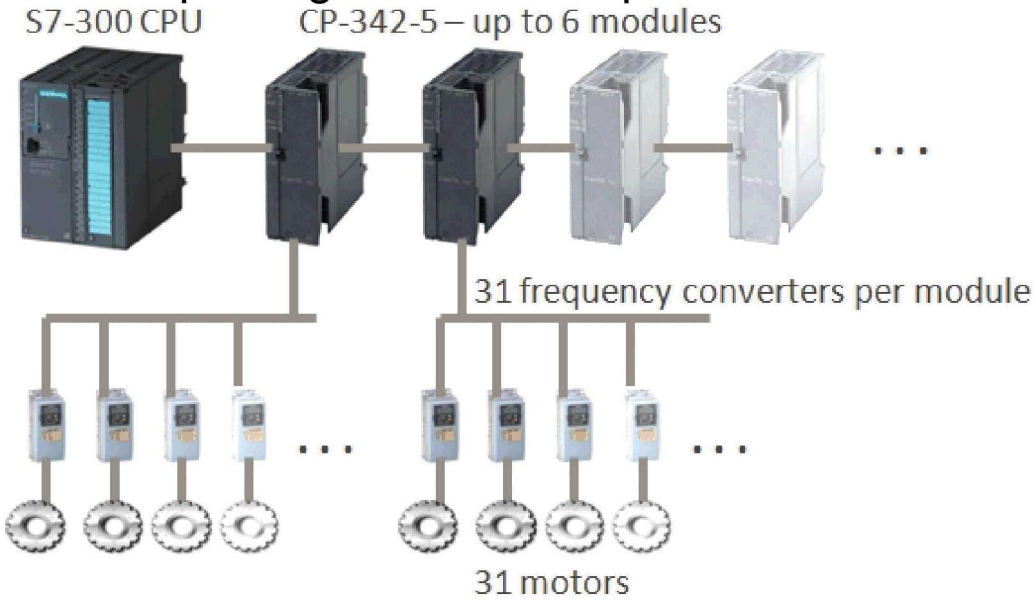
Control
Attack

Intent - Stuxnet

Intent - Stuxnet

Selective Targeting

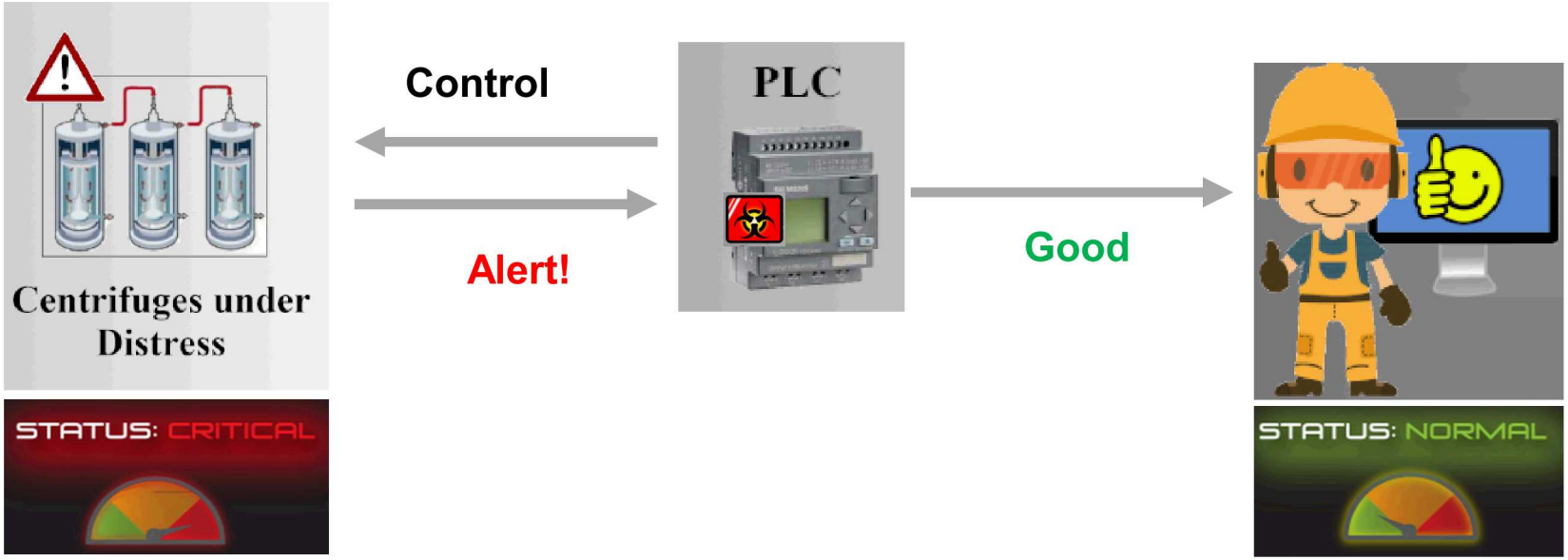
- Specific Step 7 environment: S7-300
- Specific network topology: Profibus
- Specific type of frequency controlled motors: Vacon or Fararo Paya
- Specific number of those motors on the network
- Attacks only those spinning at a certain speed



Intent - Stuxnet

Malicious Payload

- Vary motor speed low, then high, then low
- Report expected speed to monitors
- Attack continues over 17 months
- 5000 of 8000 centrifuges taken offline

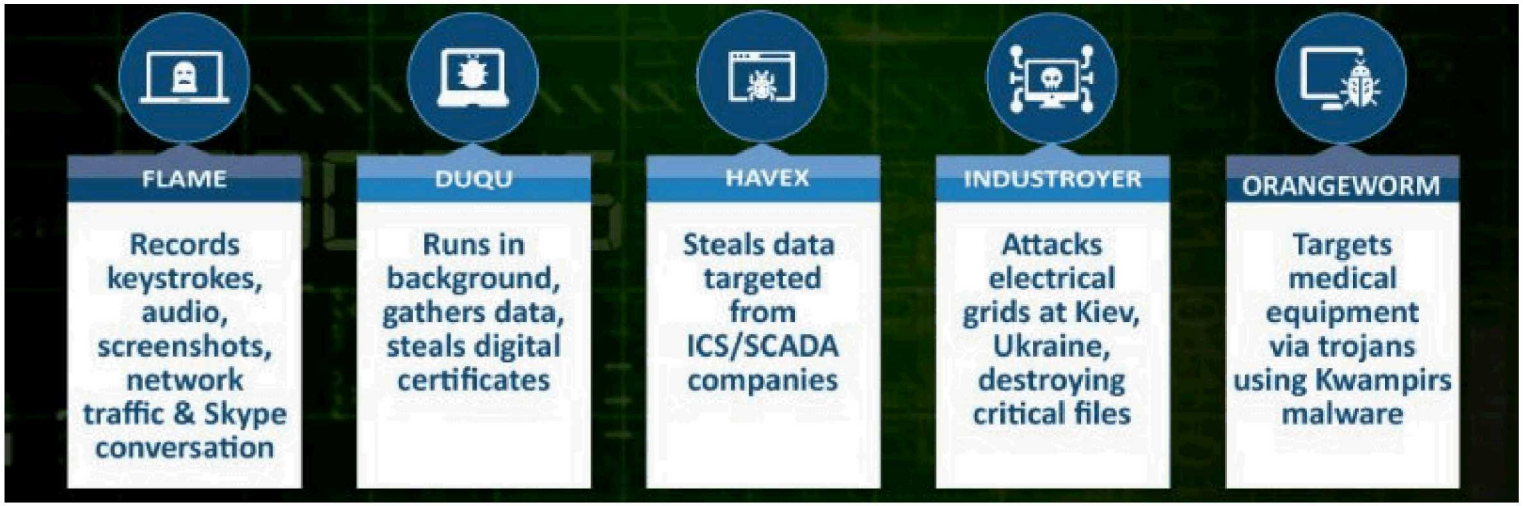


Intent - Stuxnet

Intent

Intentional Defects

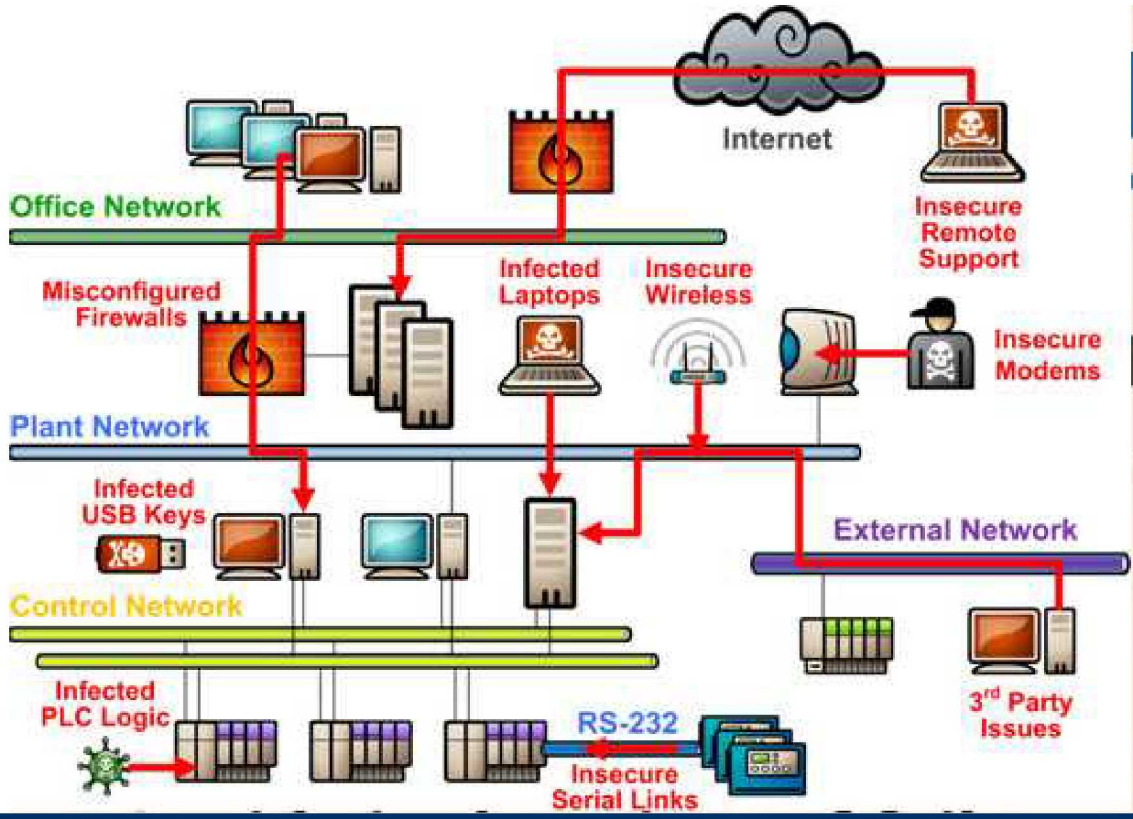
- Morris Worm
- Stuxnet



Intent

Perimeter Defense – Air Gaps?

- "The completely isolated control system is history" Director of Security, Siemens
- USB peripherals
- Software updates
- Wireless interfaces
- 3rd Party tools
- Internal networks



Safety Threats from Electronic Products

PROGRAMMABLE LOGIC

Programmable Logic

- September 2007[1,2]
  - Israel attacks suspected nuclear installation in Syria
  - State-of-the-art Syrian radar failed to warn of incoming attack
  - Speculation is that COTS microprocessors had a kill switch programmed into them by the French contractors

[1] Adee, S., "The Hunt for the Kill Switch," IEEE Spectrum, 1 May 2008.

2 Fulghum, D.A., et al, "Cyber-Combat's First Shot," Aviation Week & Space Technology, 26 November 2007.

Programmable Logic

- FPGA
  - Field Programmable Gate Array
- SoC
  - System on a Chip
- CPLD
  - Complex Programmable Logic Device
- Hardware Trojan
  - Malicious payloads or circuit modifications within an integrated circuit

Programmable Logic

Complex Programmable Logic Device (CPLD)

- Medium complexity device
- Macrocell architecture
- 10,000+ logic gates
- Non-volatile memory (EEPROM)

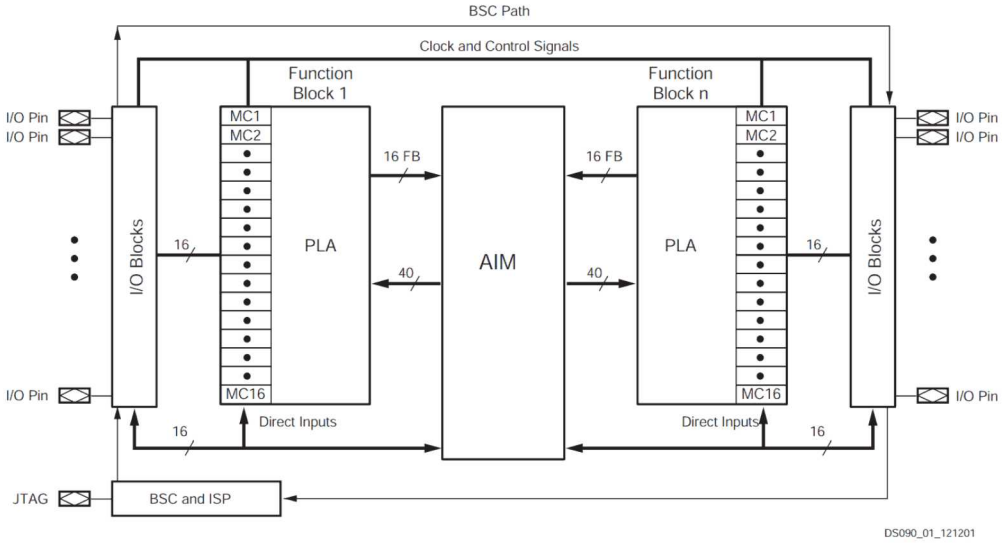


Figure 1: **CoolRunner-II CPLD Architecture**

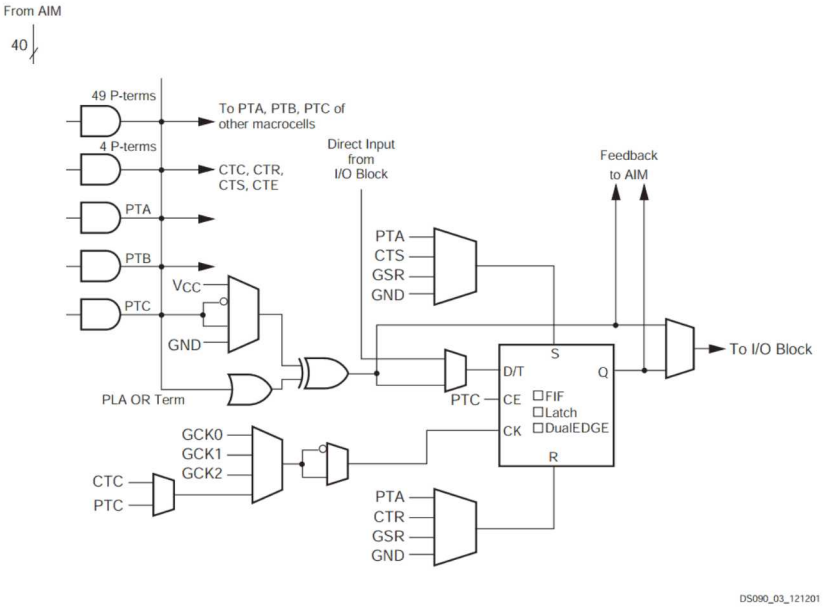
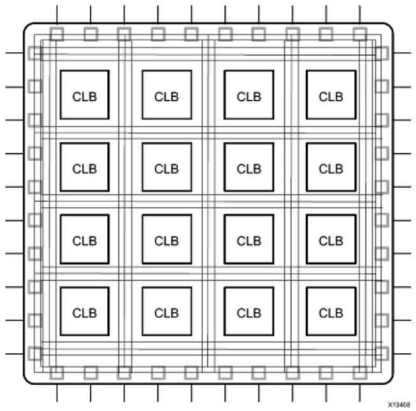


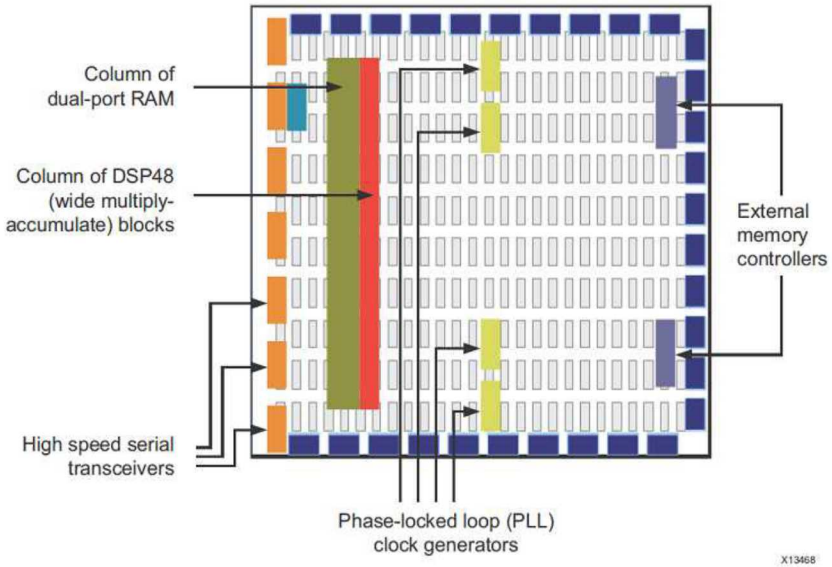
Figure 3: **CoolRunner-II CPLD Macrocell**

Programmable Logic

Field Programmable Gate Array (FPGA)

- High complexity device
- Array of programmable logic blocks
- Fine grain structure
- 50 million+ gates
- Embedded functions: memory, adders, multipliers, even microprocessors (SoC)
- Reprogammable *after* deployment

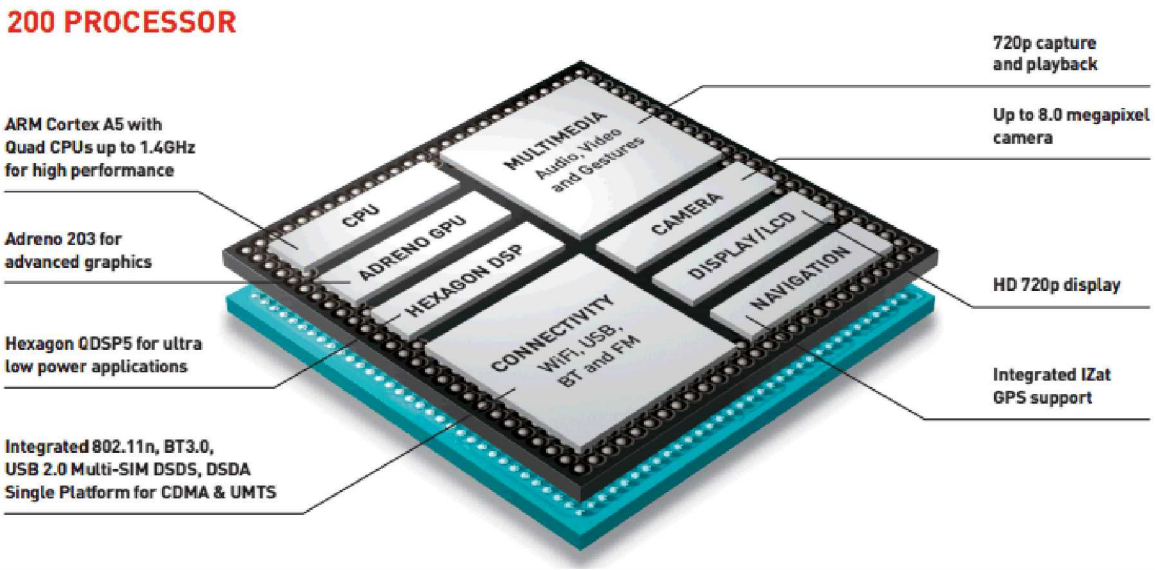


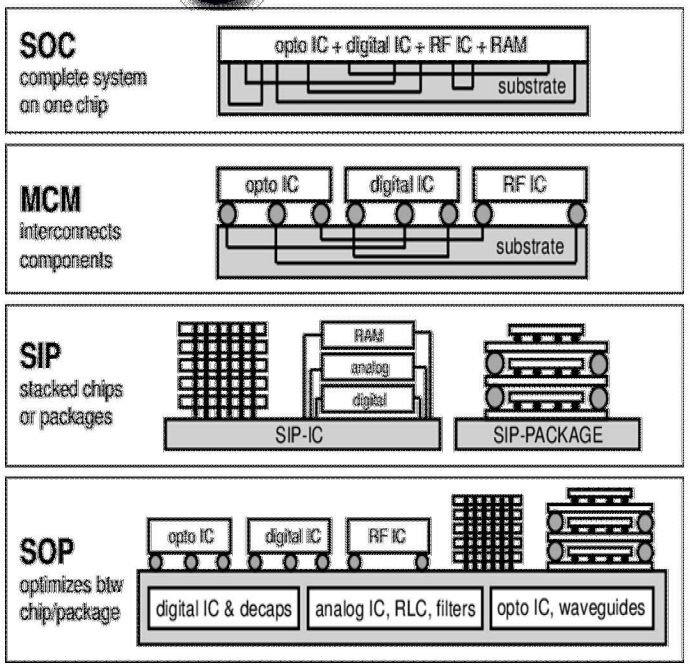


Programmable Logic

System on a Chip (SoC)

- CPU, Memory, I/O, Programmable Logic
- Wired/Wireless
- Analog and Digital
- ARM, Pi, Atom, Snapdraggon





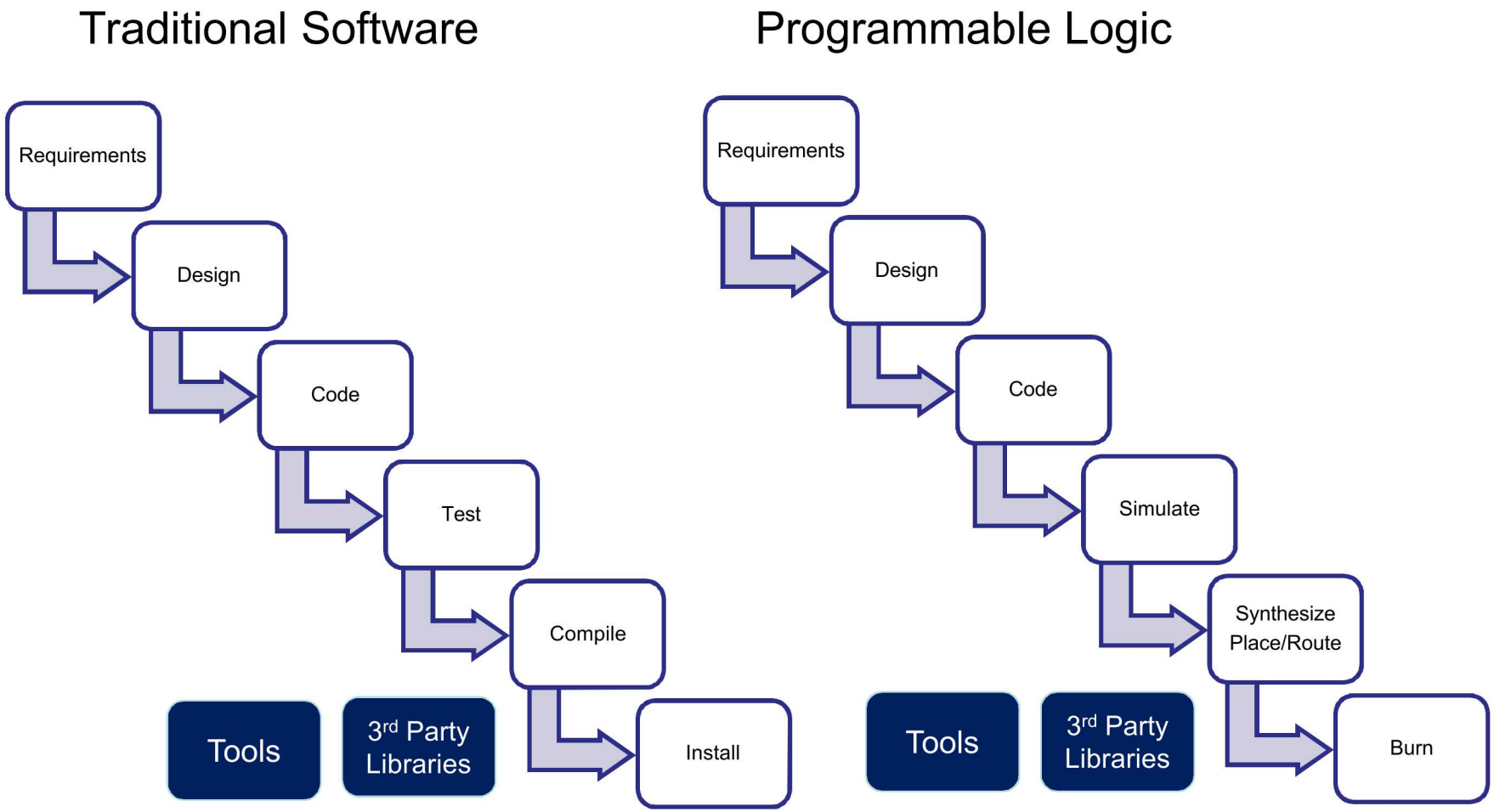
Programmable Logic

- Firmware
  - IEEE 12207-2008: Combination of a hardware device and computer instructions or computer data that reside ***as read-only software*** on the hardware device
  - IEEE STD 7-4.3.2-2010: For compliance with this standard, firmware shall be treated as software in a programmable device

- HDL – Hardware Design/Description Language

  A specialized computer language used to program the structure, design, and operation of electronic circuits. The output is a gate-level net list, which can be used to lay out a PCB, program a PLD, or fabricate a custom IC

Programmable Logic



Programmable Logic

C vs Verilog

C/C++	Verilog	
case sensitive	case sensitive	
{ ... }	begin ... end	*(like in Pascal)*
if (...) else	if (...) else	
type var;	type var;	
a = b;	a = b;	
==, <, >	==, <, >	
&, |, ^	&, |, ^	
+, -, *, /	+, -, *, /	
= (..) ? (..) : (..)	= (..) ? (..) : (..)	*(ternary operator)*
for (.., ..., ...)	for (.., ..., ...)	
while (...)	while (...)	
"bla-bla"	"bla-bla"	
// comment	// comment	

Programmable Logic

Examples of HDL Code

VHDL:

```
process (S0,S1),A,B,C,D)
begin
   case (S0,S1), is
      when "00" => Y <= A;
      when "01" => Y <= B;
      when "10" => Y <= C;

   end
end pr
```

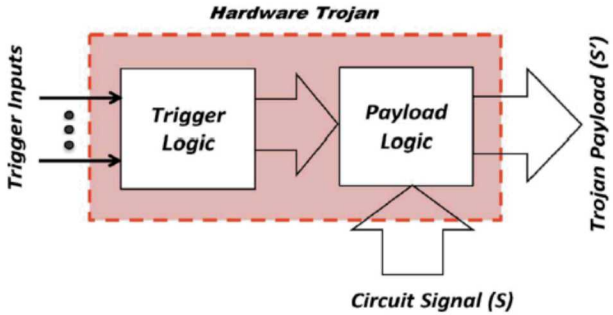
```
always @((S0,S
      case ((S
         2'b00
         2'b01
```

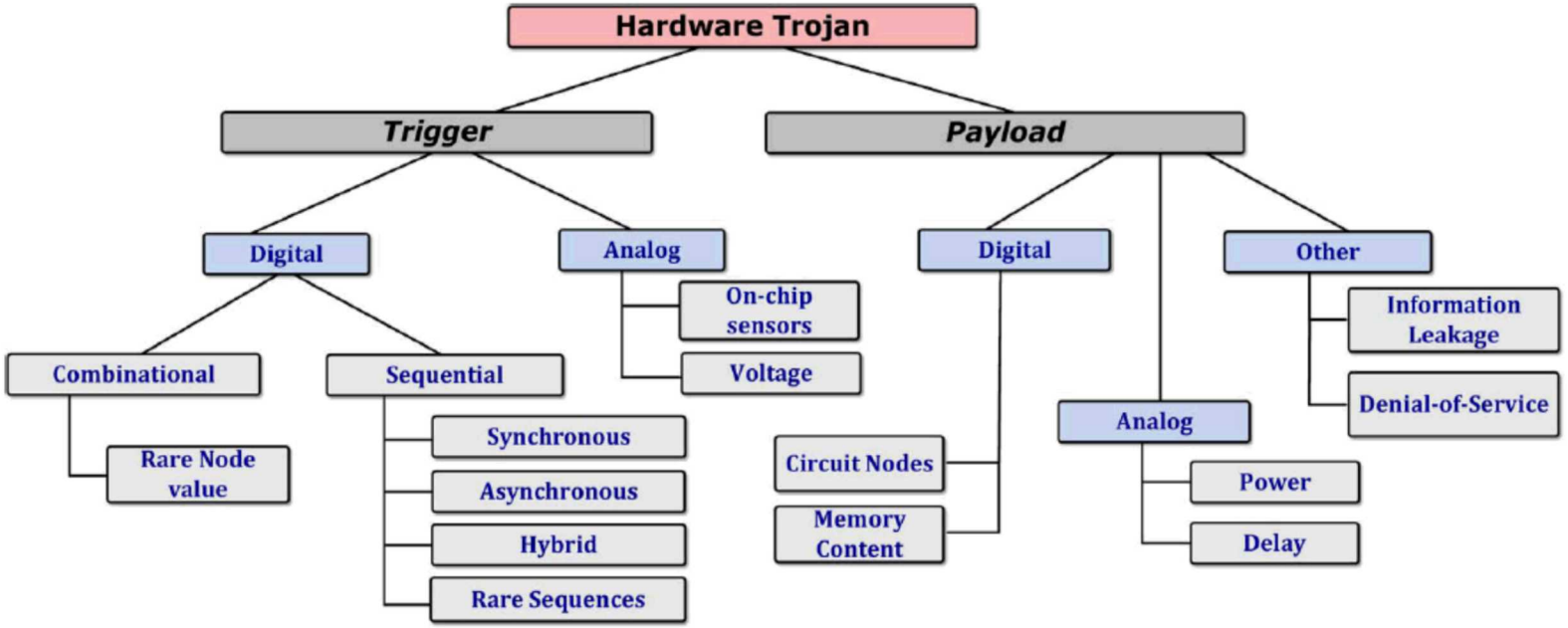
```
process_coordinate : process (RESET_
begin
            if(RESET_N ='0') then

             counter_int.h <=(othe
             counter_int.v <=(othe

             else

             X <= 320;--Starting v
             Y <= 240;--Starting v

             if SW_INTERNAL(0) = '
             if SW_INTERNAL(1) = '
             if SW_INTERNAL(3) = '
             if SW_INTERNAL(4) = '
             when others => '0';
             end if;
end process;
```

```
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ClkDiv is
    Port (  InByte : in STD_LOGIC_VECTOR(3 downto 0);       --<-- Seq_CPLD
            RegSel : in STD_LOGIC_VECTOR(1 downto 0);       --<-- Seq_CPLD
            RegStrb : in STD_LOGIC;                         --<-- Seq_CPLD
            MClk : in  STD_LOGIC;                           --<-- OSC
            SeqReset : in STD_LOGIC;                        --<-- Power Monitor
            ADC_Clk : out STD_LOGIC);                       -->-- ADC
end ClkDiv;

architecture Behavioral of ClkDiv is
   signal   ADC_div :       STD_LOGIC_VECTOR(5 downto 0) := "001111";
   signal   ADCClk :        STD_LOGIC := '0';
   signal   ClkSel :        STD_LOGIC_VECTOR(2 downto 0) := "100";

begin

   ClkDivP : process(Mclk,SeqReset)
      begin
      if SeqReset = '0' then
         ADCClk <= '0';
         ADC_div <= "001001";
      elsif Mclk = '0' and Mclk'event then
         if ADC_div = "000000" then
            ADCClk <= not(ADCClk);
            case ClkSel is
               when "000" =>                    -- 20MHz - divide by 2
               when "001" =>                    -- 10MHz
                  ADC_div <= "000001";             -- divide by 4
               when "010" =>                    -- 4MHz
                  ADC_div <= "000100";             -- divide by 10
               when "011" =>                    -- 2MHz
                  ADC_div <= "001001";             -- divide by 20
               when "100" =>                    -- 1MHz
                  ADC_div <= "001001";             -- divide by 40
               when others =>                   -- 400KHz
```

Programmable Logic

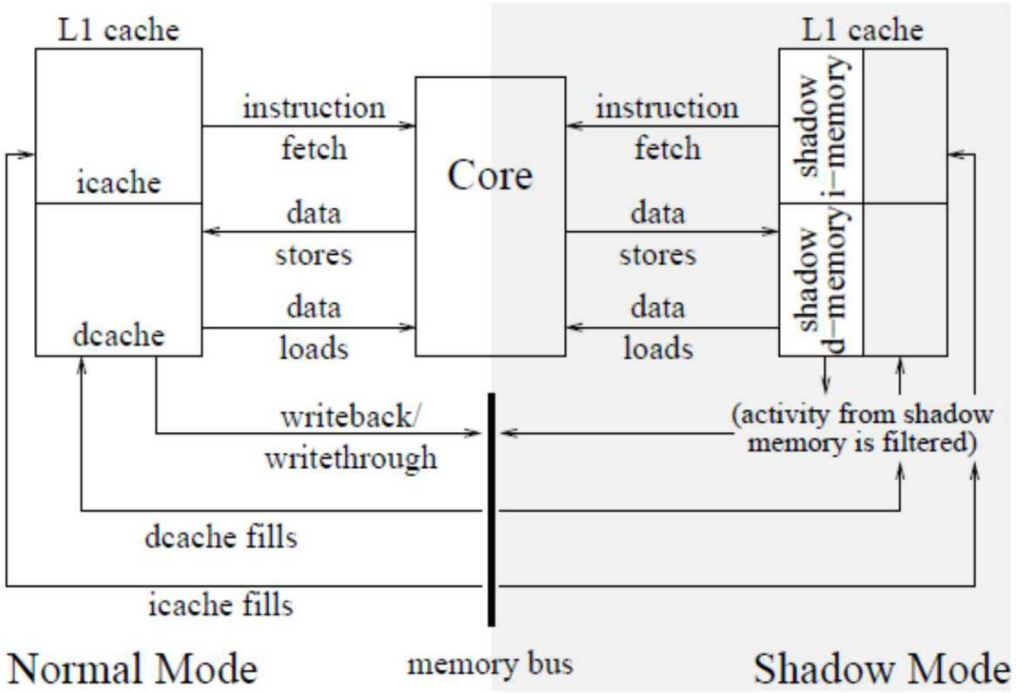
- Undetected circuitry
- Trigger mode
- Payload and delivery





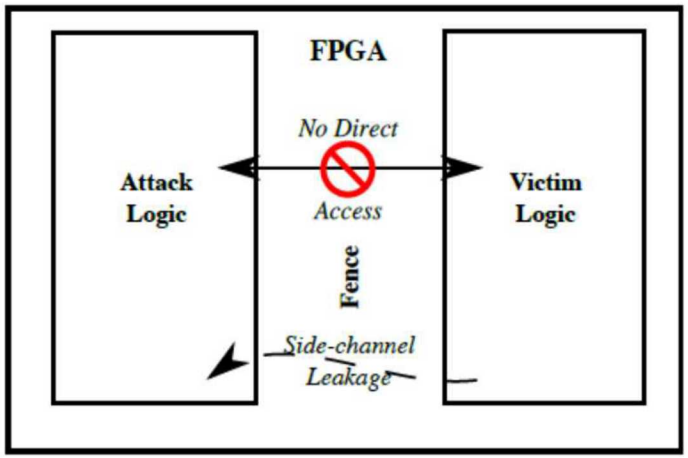
Programmable Logic

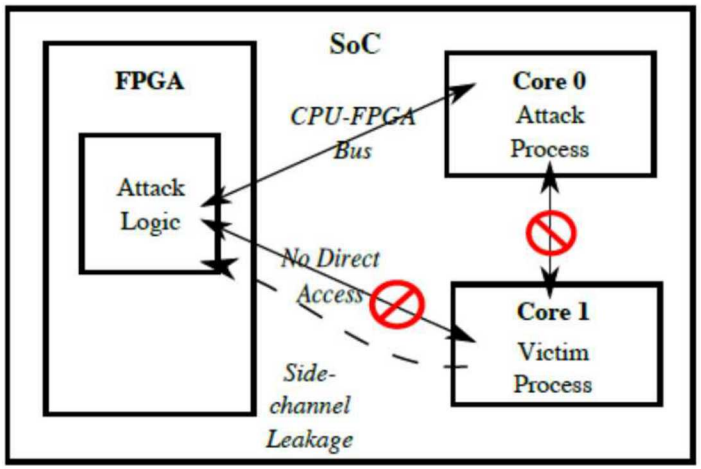
- Shadow Mode Attack
  - Programmable logic with microcontrollers - SoC
  - Memory access mechanism provides attacker bypass of memory protection
  - Shadow mode mechanism to execute invisible malicious firmware
- Login Backdoor
  - Use of Shadow Mode attack to provide permanent back door for attacker
- Kill Switch



Programmable Logic

- Side-Channel Attack
  - Pre-programmed power monitor circuit
  - Limitation is resolution
- Covert Channel Attack
  - Pre-programmed circuit to control switching activities
  - Uses power/timing channels for communication
  - Bidirectional

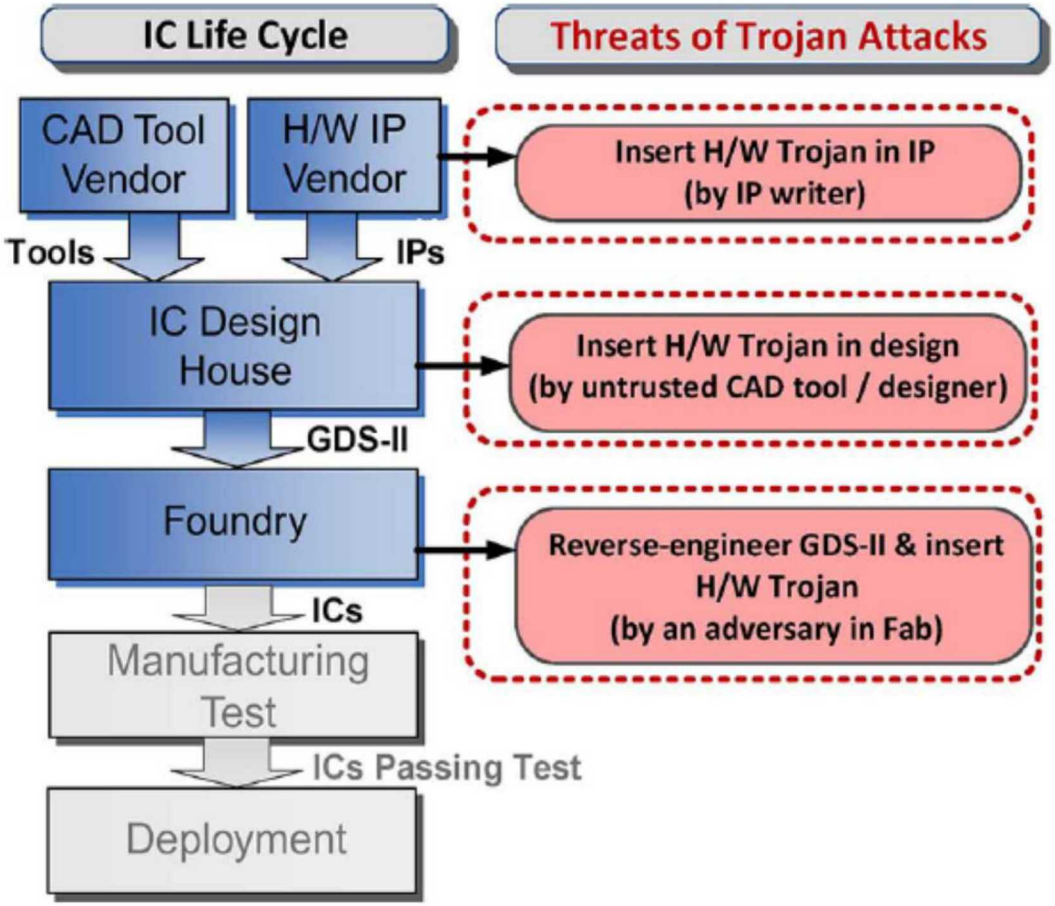




Programmable Logic

Sources

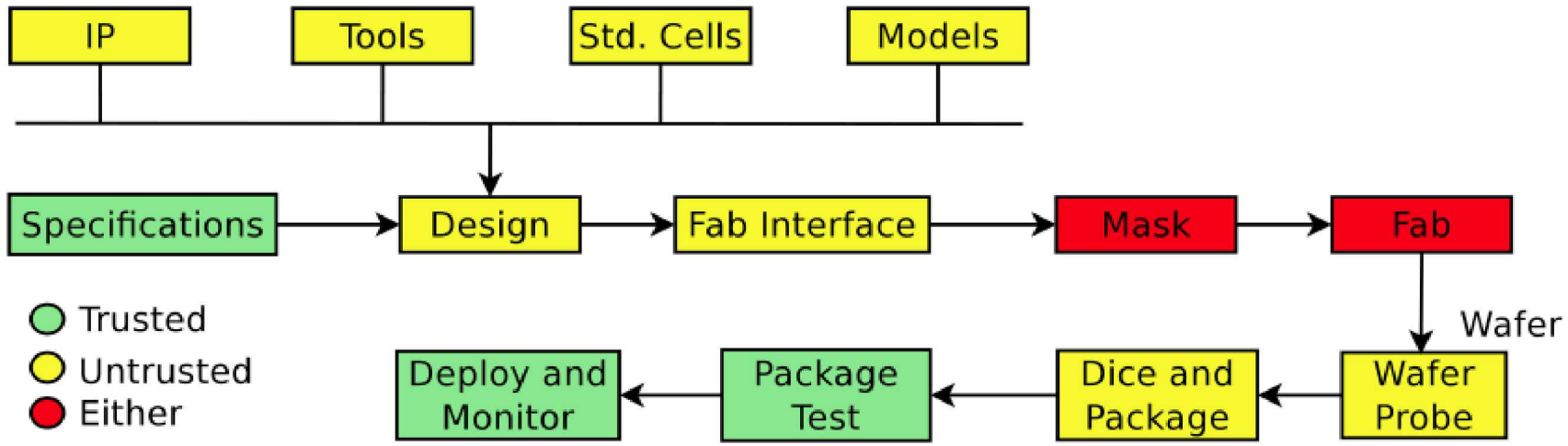
- Procurement
- HDL, CAE, Design Entry
- Standard Cells
- 3rd Party Libraries/Models
- Fabrication facilities



Programmable Logic

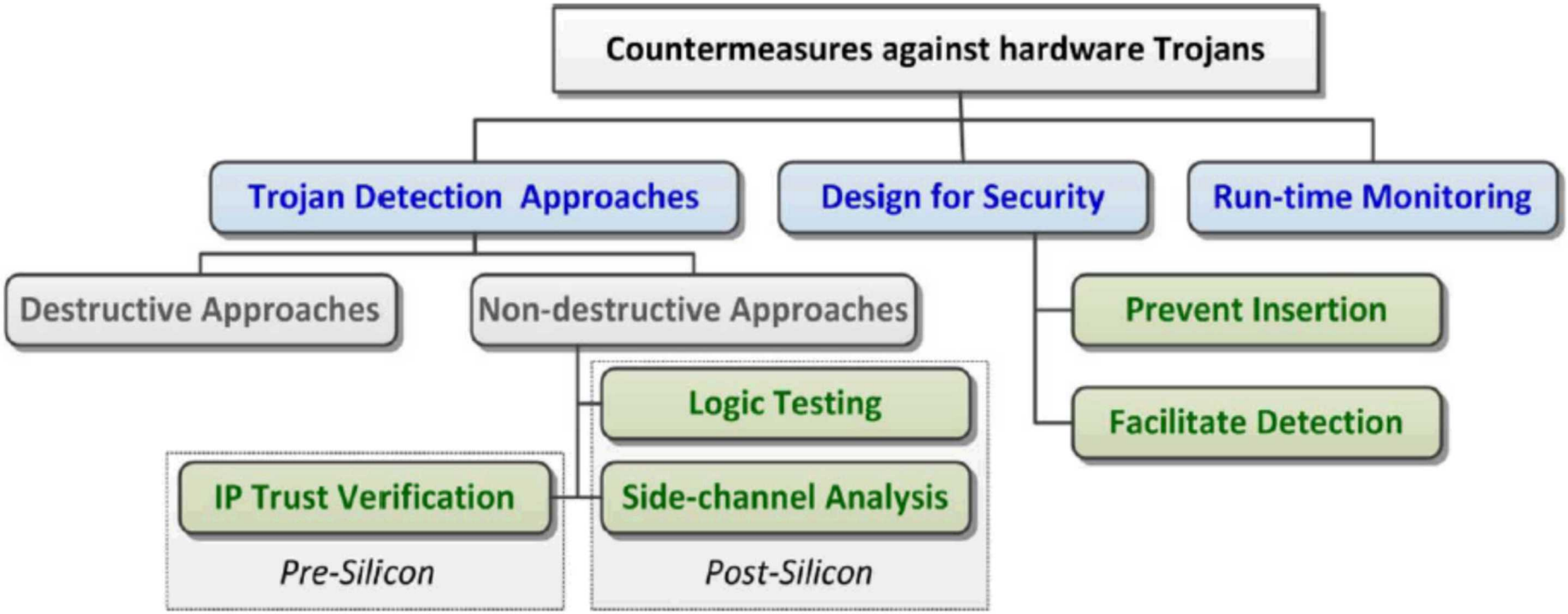
Prevention

- Developing a trusted process



Programmable Logic

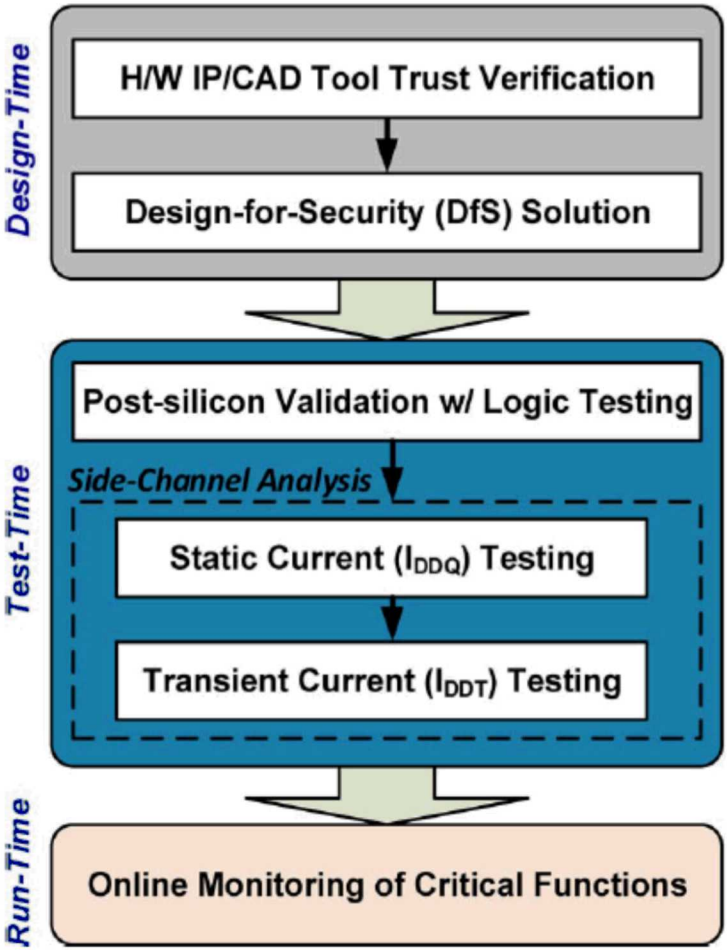
Detection



Programmable Logic

Detection

- Design Verification
- Post-manufacture Verification
- Real time monitoring



Conclusion

- Software is a unique, integral component of a system
  - Design, Verify, Qualify as with any other custom component
- Inspection and Testing help, but break the fault cycle, contain the failure
  - Post-manufacture Verification
- It's not always oversight or accidental
  - Weaponized software is a reality
- Hardware is becoming more dependent on code
  - Programmable logic only differs in the media
- Hardware is not inherently trustworthy
  - Take measures to assure safety, mitigate, develop trust

Post Script

And another….?

- Boeing 737 Max 8
  - Certified by the FAA in March 2017
  - Direct competitor with the Airbus A320 family
  - 4th Generation of the 737 airframe
    - New engines provide 10-12% efficiency increase, reduced emissions
    - Longer nose gear
    - Re-engineered tail
  - 2 crashes:
    - Indonesia, October 2018
    - Ethiopia, March 2019
  - Black box and satellite data show “clear similarities” between them[3]

Seattle Times
Fortune
The Atlantic
Leeham News
Wired
Fox News
FAA

Official Report Not Released

[3] Schemm, P., “Ethiopian Official: Black Box Data Shows ‘Clear Similarities’ between Ethiopian Airlines, Lion Air Crashes,” The Washington Post, 17 March 2019.

Post Script

And then it gets interesting….

- New LEAP engines physically larger, didn't fit
  - Would require raising the plane, new landing gear system
- Mounting position changed,
  - Forward of and above previous configurations
- New position introduced instability to the plane[4]
  - Destabilizes in pitch
  - When new plane approaches stall angle, engine nacelles generate lift, generating nose up motion
  - Departure from naturally stable base requires "augmented flight control" to address the instability
    - First time for the 737 platform

[4] Fehrm, B., "Boeing's Automatic Trim for the 7373 MAX was not disclosed to the Pilots," Leeham News and Analysis, 14 November 2018.

Post Script

And then it gets interesting….

- Certification by the FAA by "accelerated process"[5]
  - Delegated most assessment roles to Boeing, though FAA denies
  - Former FAA safety engineer on the Max 8:
    - "we had retained too much" and should delegate more to Boeing
    - Incomplete reviews, rushed dates, managers override technical staff
- Safety Analysis of the MCAS has crucial errors[6]
  - Understates level of trim control of tail horizontal stabilizers
    - 0.6 degrees vs 2.5 degrees of control
  - Fails to consider automatic reset of the MCAS
    - Resets 5 seconds after disconnect or override
  - Comments on the use of only one Angle of Attack (AoA) sensor, but no action taken
    - The aircraft has two
  - Classifies the system as "hazardous" with potential for loss of life
    - Requires input sensors to be rated 1 in 10 million failure probability.
    - Single sensor only rated to 1 in 100,000
  - Boeing: "there are some significant mischaracterizations"

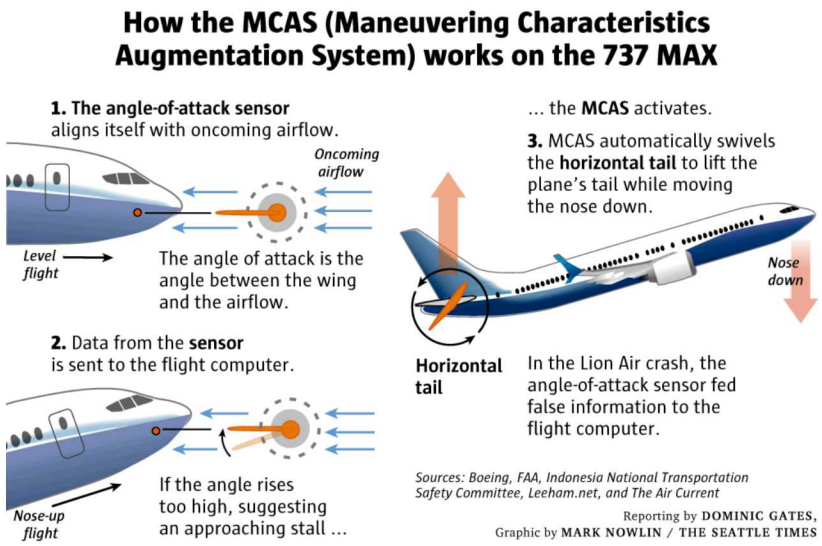
[5] Mann, T. and S. Hughes, "Fast-Tracked Aircraft Certification, Pushed by Boeing, Comes Under the Spotlight," The Wall Street Journal, 24 March 2019.

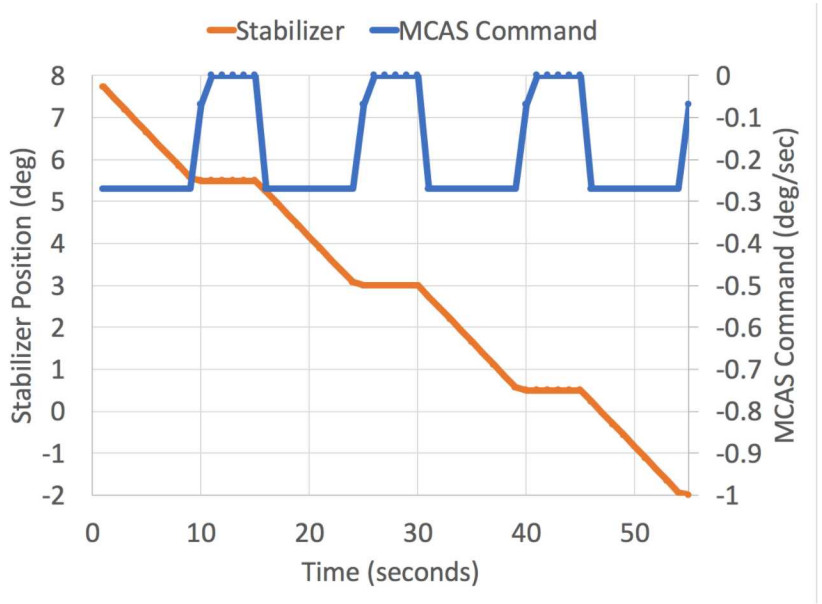
[6] Gates, D, "Flawed Analysis, Failed Oversight: How Boeing, FAA Certified the Suspect 737 MAX Flight Control System," Seattle Times, 21 March 2019.

Post Script

And then…it gets interesting

- Indonesia crash, October 2018[7]
  - MCAS detects a high angle of attack (AoA) and swivels the tail horizontal stabilizers to nose down the aircraft
  - Pilot overrides MCAS and tries to correct
  - MCAS resets and, 5 seconds later, swivels stabilizers down again
  - Pilot again overrides and corrects
  - Repeats 23 times before plane hits the ocean at 500 mph
  - Horizontal stabilizer at maximum position





[7] Karmin, N. and D. Koenig, "Lion Air Pilots Struggled to Save Boeing 737 MAX before Indonesia Crash, Black Box Data Show," Associated Press, 28 November 2018.

Post Script

And then, it gets interesting!

- No training on MCAS system…not even any knowledge of it[8]
  - MCAS not in the pilot training for the plane
  - MCAS not in the flight manuals for the aircraft
  - Boeing deemed it "unnecessary"
- Pilot reports the previous day on the same Indonesian plane[9] had similar problems
  - Experienced repeated nose down swings
  - Activated stabilizer cutoff switches and regained control
- First pilot reports of MCAS issues after the crash[10]
  - November 2018, pilot experiences nose down and "Don't Sink!" annunciation while on autopilot
    - MCAS is not supposed to operate while under autopilot
    - Pilot disengaged system and resumed flight
  - November 2018, pilot concerns of MCAS, "flight manual is inadequate and almost criminally insufficient"
  - November 2018, pilot experiences 1200-1500 fpm descent with "Don't Sink" annunciation while on autopilot
    - Pilot disconnects and entered a climb
- Boeing issues advisory bulletin about the AoA sensor and MCAS in November 2018[11]

BOEING

Flight Crew Operations Manual Bulletin for The Boeing Company

The Boeing Company
Seattle, Washington 98124-2207

737

Number: TBC-19
IssueDate: November 6, 2018

Airplane Effectivity: 737-8 / -9

Subject: Uncommanded Nose Down Stabilizer Trim Due to Erroneous Angle of Attack (AOA) During Manual Flight Only

Reason: To Emphasize the Procedures Provided in the Runaway Stabilizer Non-Normal Checklist (NNC).

Information in this bulletin is recommended by The Boeing Company, but may not be FAA approved at the time of writing. In the event of conflict with the FAA approved Airplane Flight Manual (AFM), the AFM shall supersede. The Boeing Company regards the information or procedures described herein as having a direct or indirect bearing on the safe operation of this model airplane.

THE FOLLOWING PROCEDURE AND/OR INFORMATION IS EFFECTIVE UPON RECEIPT

[8] Liebermann, O., "737 Pilots Trained for Max 8 with Short Online Course," CNN, 22 March 2019.
[9] Levin, A. and H. Suhartono, "Pilot who Hitched a Ride Saved Lion Air 737 Day Before Deadly Crash," Bloomberg, 19 March 2019.
[10] Wolfe, K.A., "Pilots Complained at least 5 Times about Boeing 737 MAX Problems, Records Show," Politico, 12 March 2019.
[11] Levin, A., J. Johnsson, and H. Suhartono, "Boeing Issues Bulletin for 737 MAX after Indonesia Jet Crash," Bloomberg, 7 November 2018.

Post Script

And then it gets interesting…

- Discussions are in process on corrective action for the issue
- The proposed fix?

Mandatory Patch to the software and training

[12] "Continued Airworthiness Notification to the International Community," FAA, 20 March 2019.

Conclusion

Questions?

Understand, Mitigate!

Don't Trust Software for Safety