25 OCTOBER 2023 3:20PM-3:40PM PT Forum 120

Common Aerospace Threats from Coding Practice to Full Exploit: Tracing Threat Angles

Bryce L. Meyer

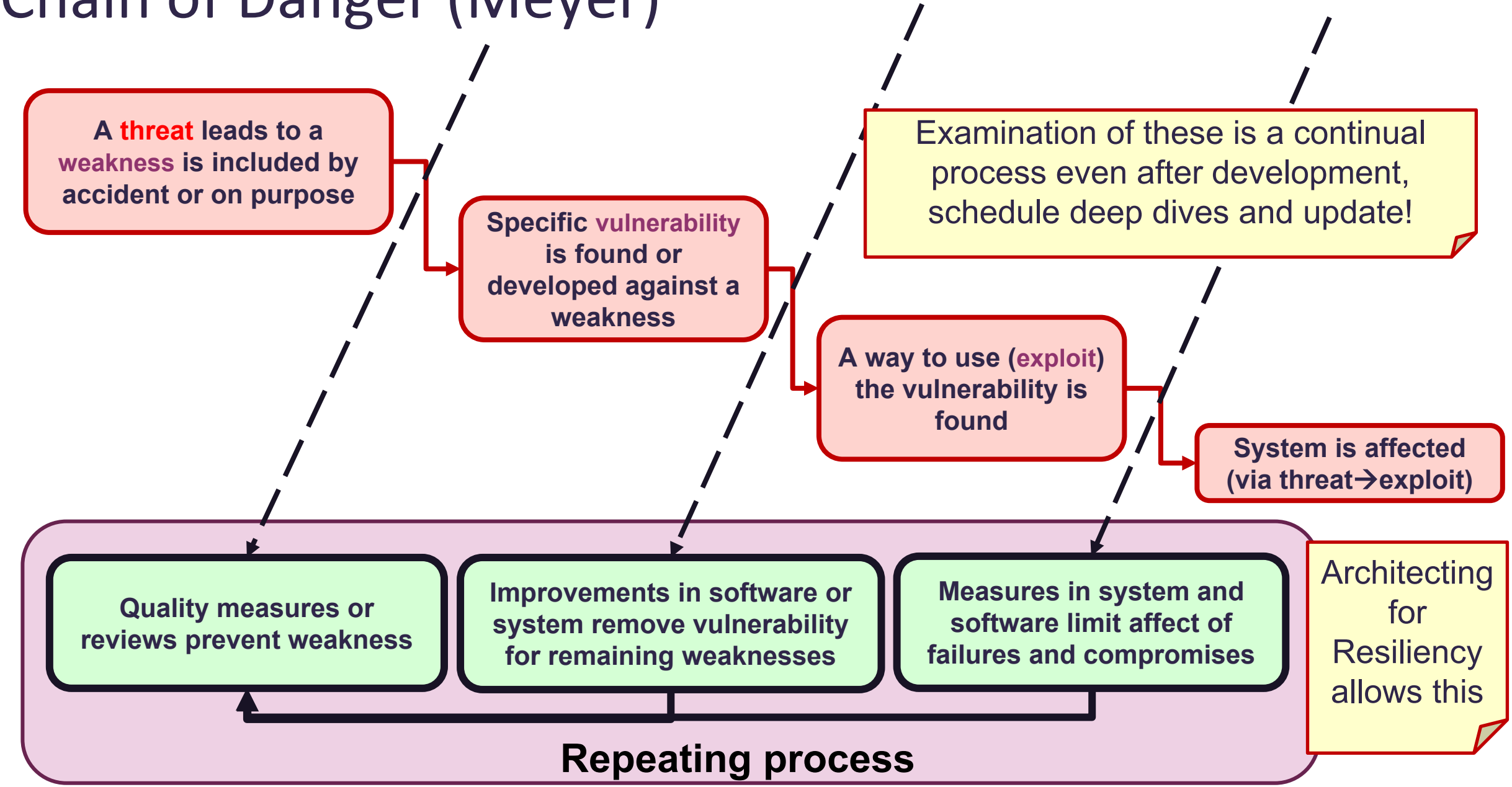
AIAA Associate Fellow
AIAA Aerospace Cybersecurity Working Group

ASCEND

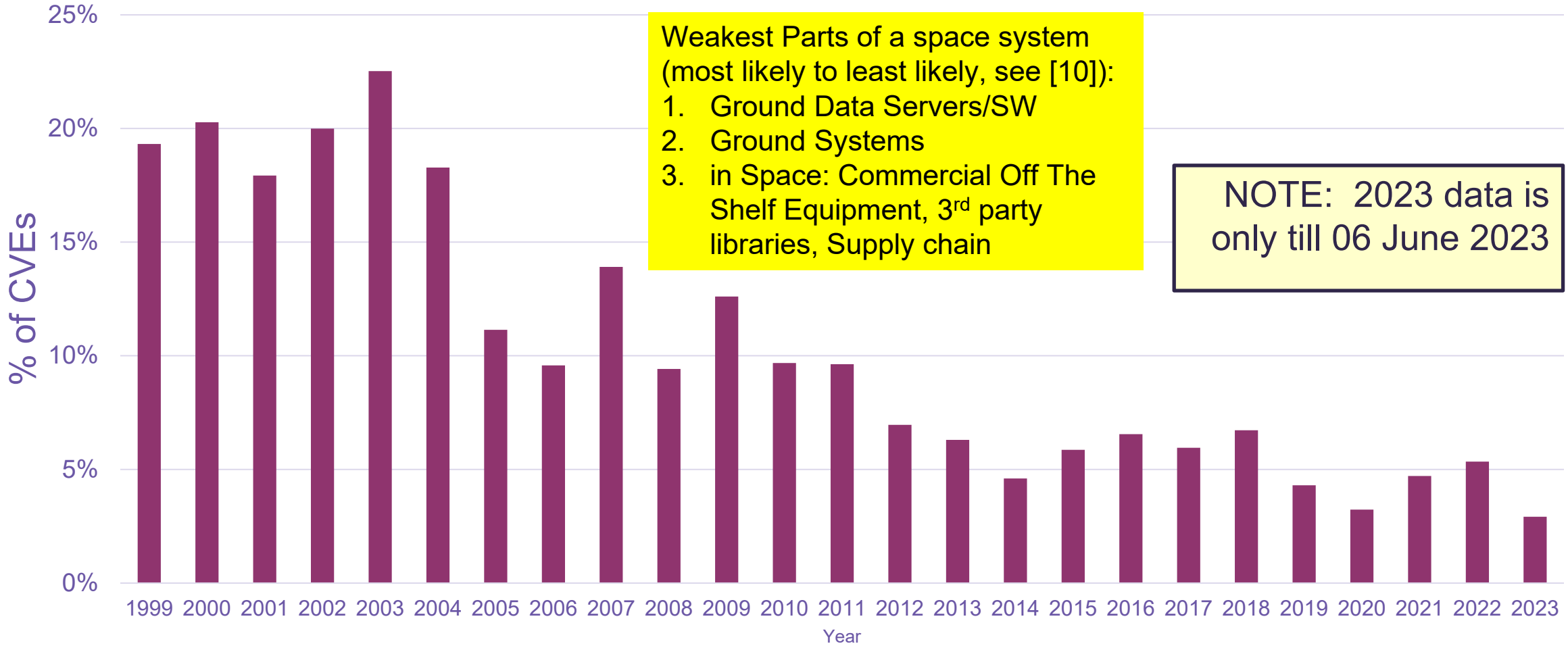
Agenda

- Chain of Danger and Vulnerabilities
- Sample Attack: Memory Overflow with Tools
- Threat Angles:
    - **Rush to Include Code Angle: Included code has a threat**
    - **Rush to Compile and Start Tests Angle: Compiler settings or ignored compiler warnings cause threat**
    - **Ignorance is Bliss Angle: Tests are performed, but warnings are ignored**
- Prevention

Chain of Danger (Meyer)



% of Vulnerabilities (CVEs) Tied to Overflows



Weakest Parts of a space system (most likely to least likely, see [10]):

1. Ground Data Servers/SW
2. Ground Systems
3. in Space: Commercial Off The Shelf Equipment, 3rd party libraries, Supply chain

NOTE: 2023 data is only till 06 June 2023

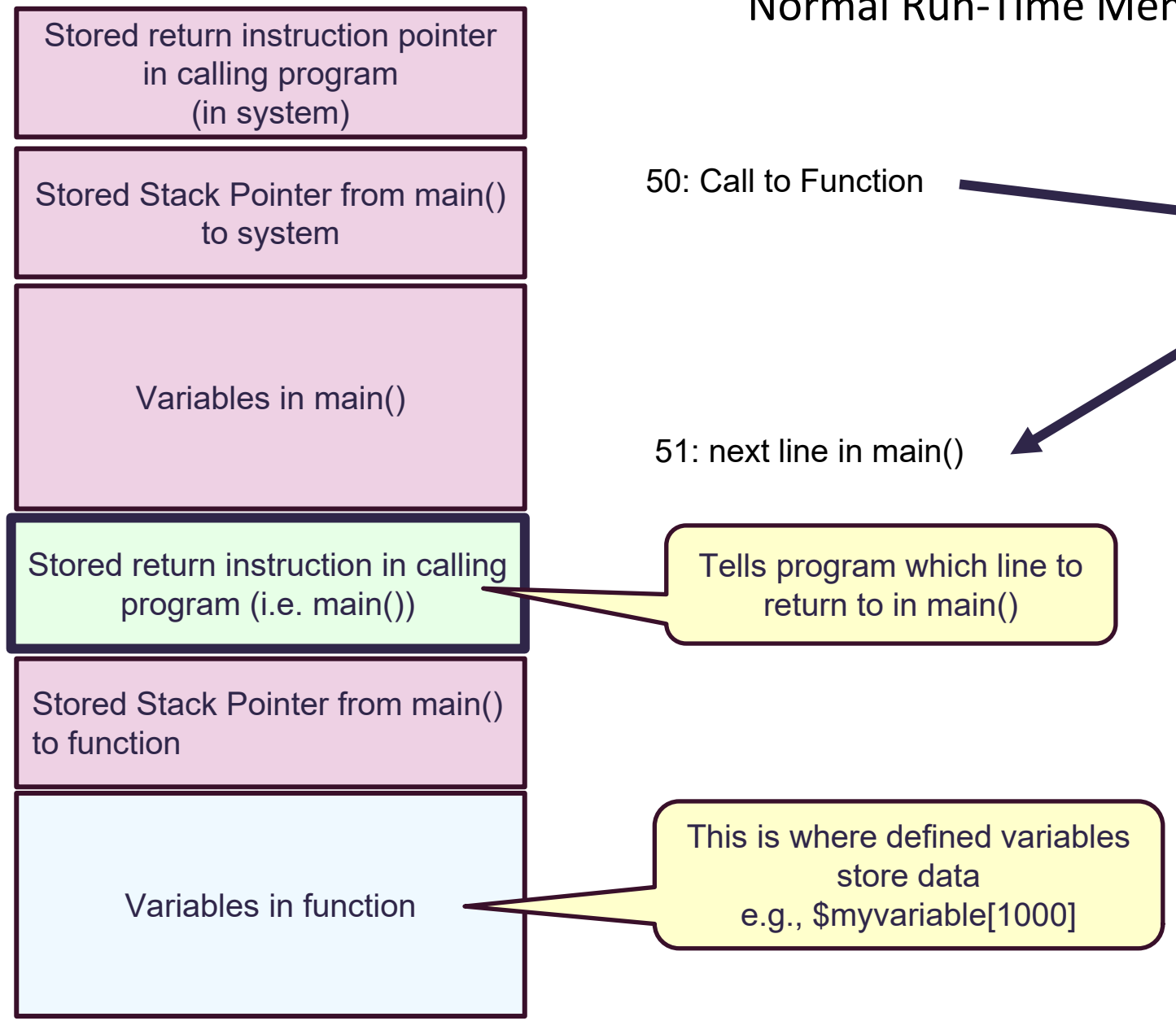
Only a small percentage of threats are overflows, yet they persist.

The older the code, the more likely a memory overflow opportunity exists.

Program Memory Basics

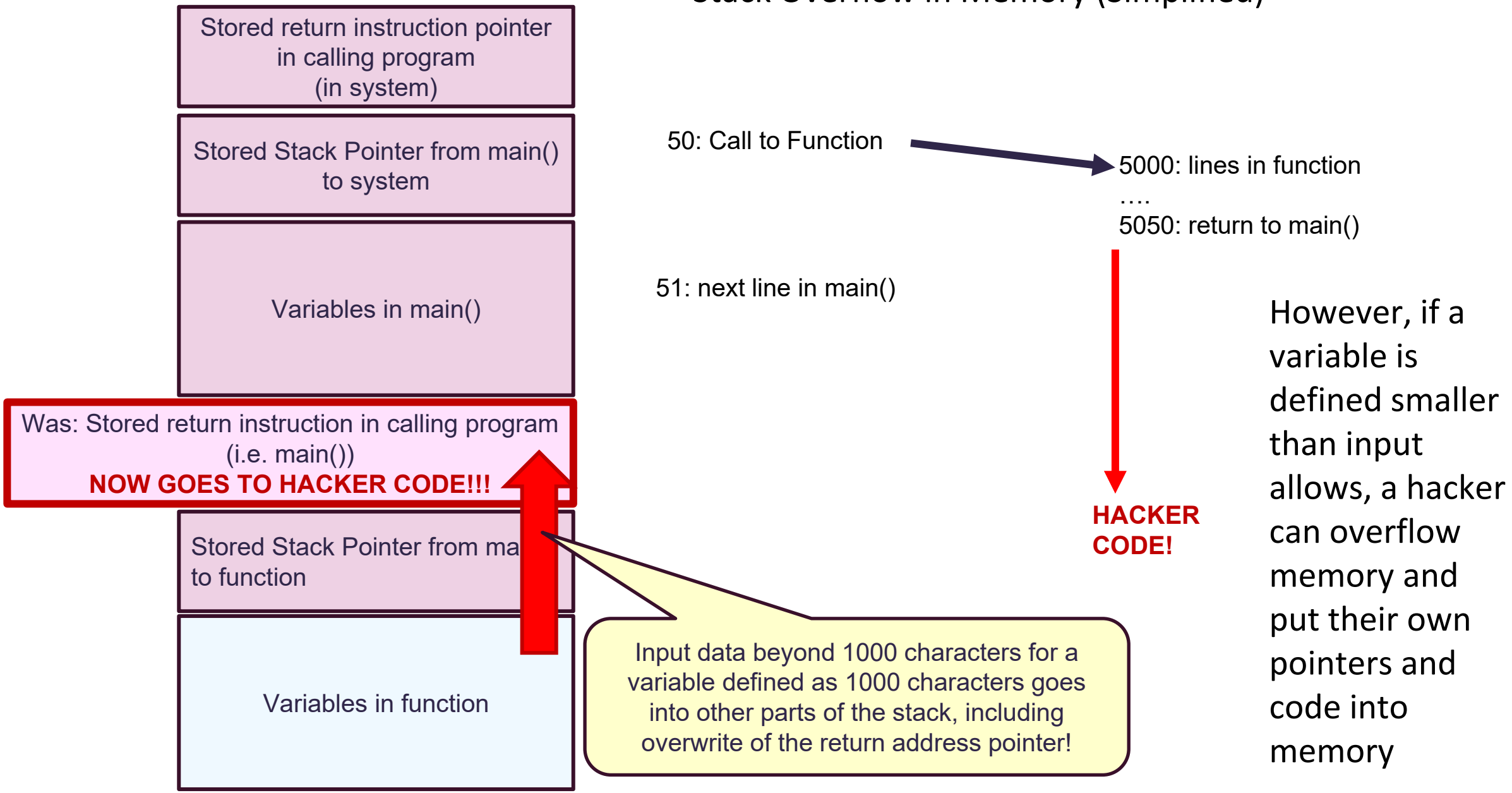
- At Run-time, the operating system creates a virtual memory space for each function in a program
- AMD vs. Intel may have slightly different architectures
- Memory is allocated for various purposes:
  - To hold the binary instructions of function (.text)
  - To hold defined global variables
  - To hold global uninitialized variables
  - To hold dynamically allocated variables (the ‘Heap’)
  - To hold local defined variables and function calls (the ‘Stack’)
  - To hold environment and argument data
- Sometimes the Heap and Stack are together called a buffer.

Normal Run-Time Memory Use



When a program runs, OS allocates run-time memory to the binary code and variables for each function…

Stack Overflow in Memory (Simplified)

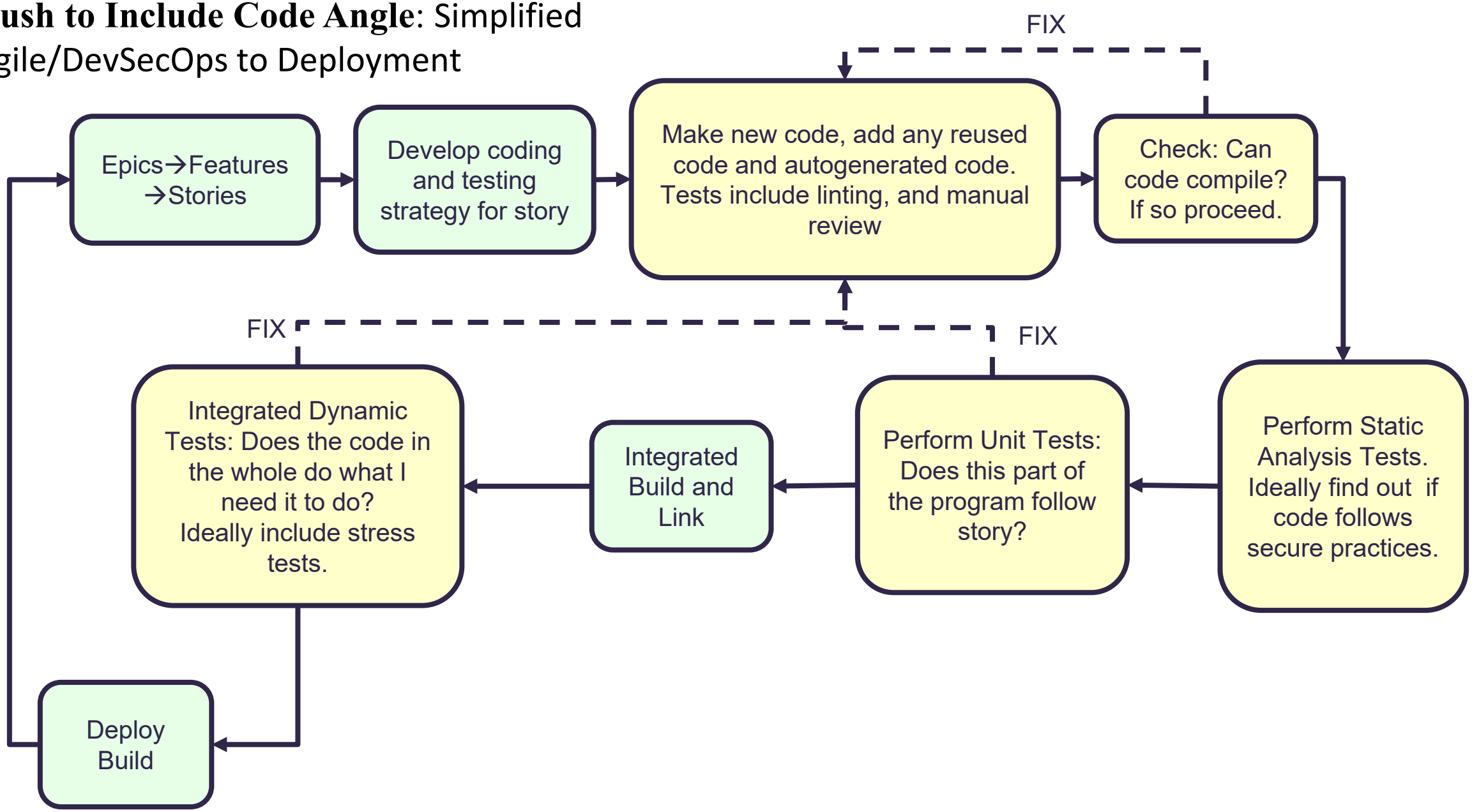


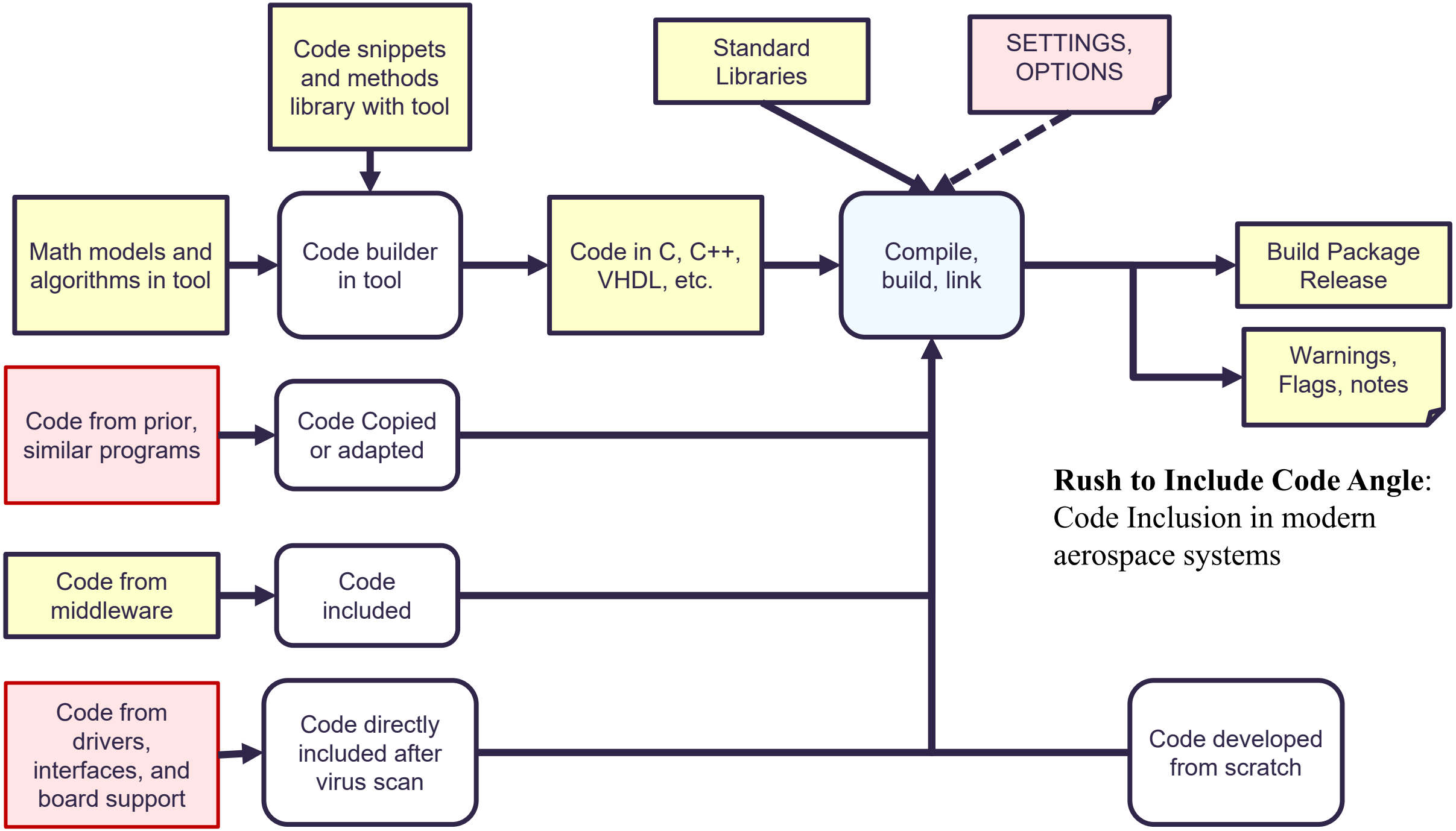
However, if a variable is defined smaller than input allows, a hacker can overflow memory and put their own pointers and code into memory

Tools used by both Red Teams and Attackers

Example Tool	Purpose
Kali Linux	Red Team/Penetration Testing aggregation of tools to run exercises and make code more secure
PwnTools and Ghidra	Capture The Flag framework of tools, Linux.
Ropper	Tool Implement Return Oriented Programming (ROP) tests, by finding gadgets, largely Windows.
Metasploit	Penetration Testing Framework, includes methods to create a command-and-control channel into an exploited system, typically for ground systems
HackRF/Universal Radio Hacker (URH)	Toolset for Software Defined Radio (SDRs) to enable wireless exploitation

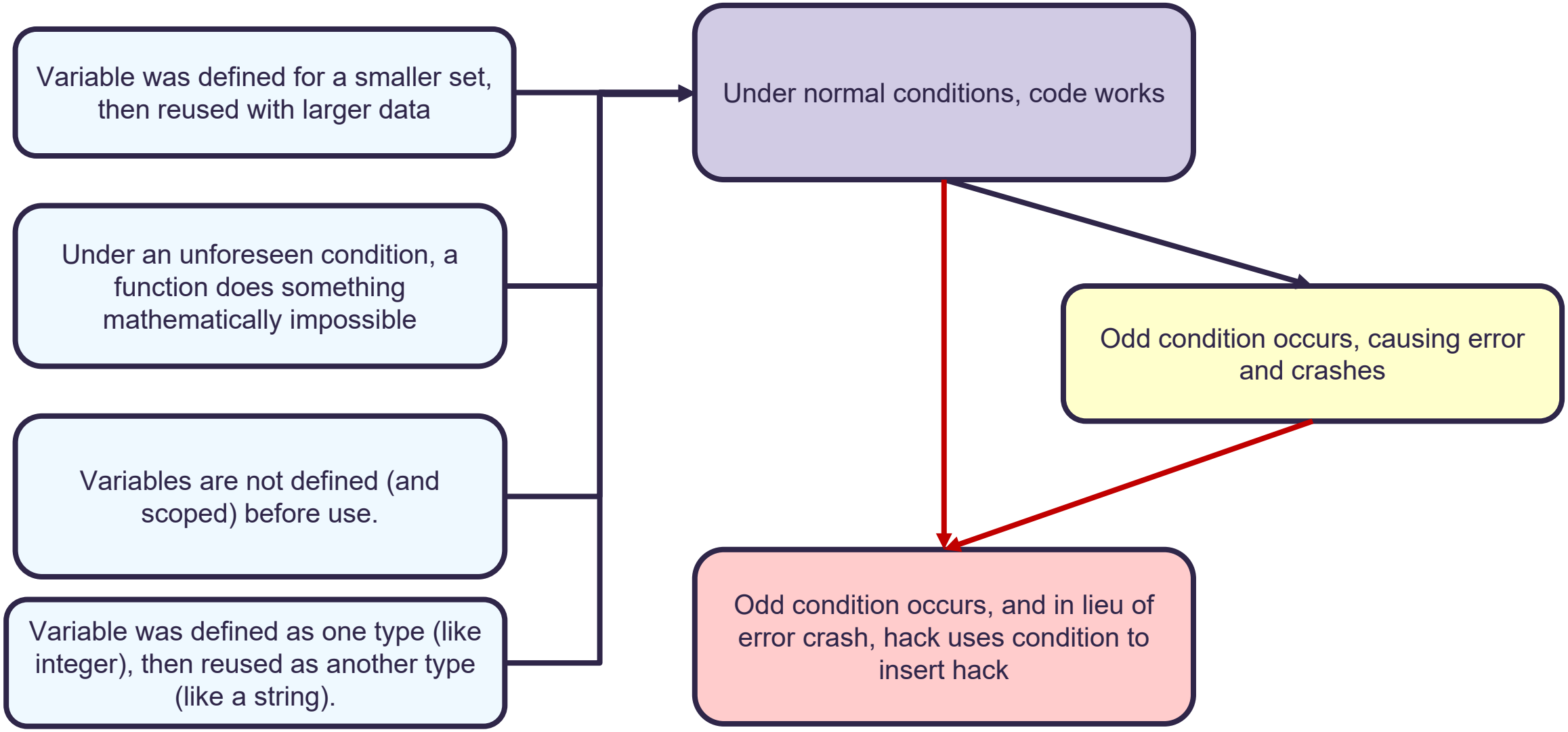
Rush to Include Code Angle: Simplified Agile/DevSecOps to Deployment



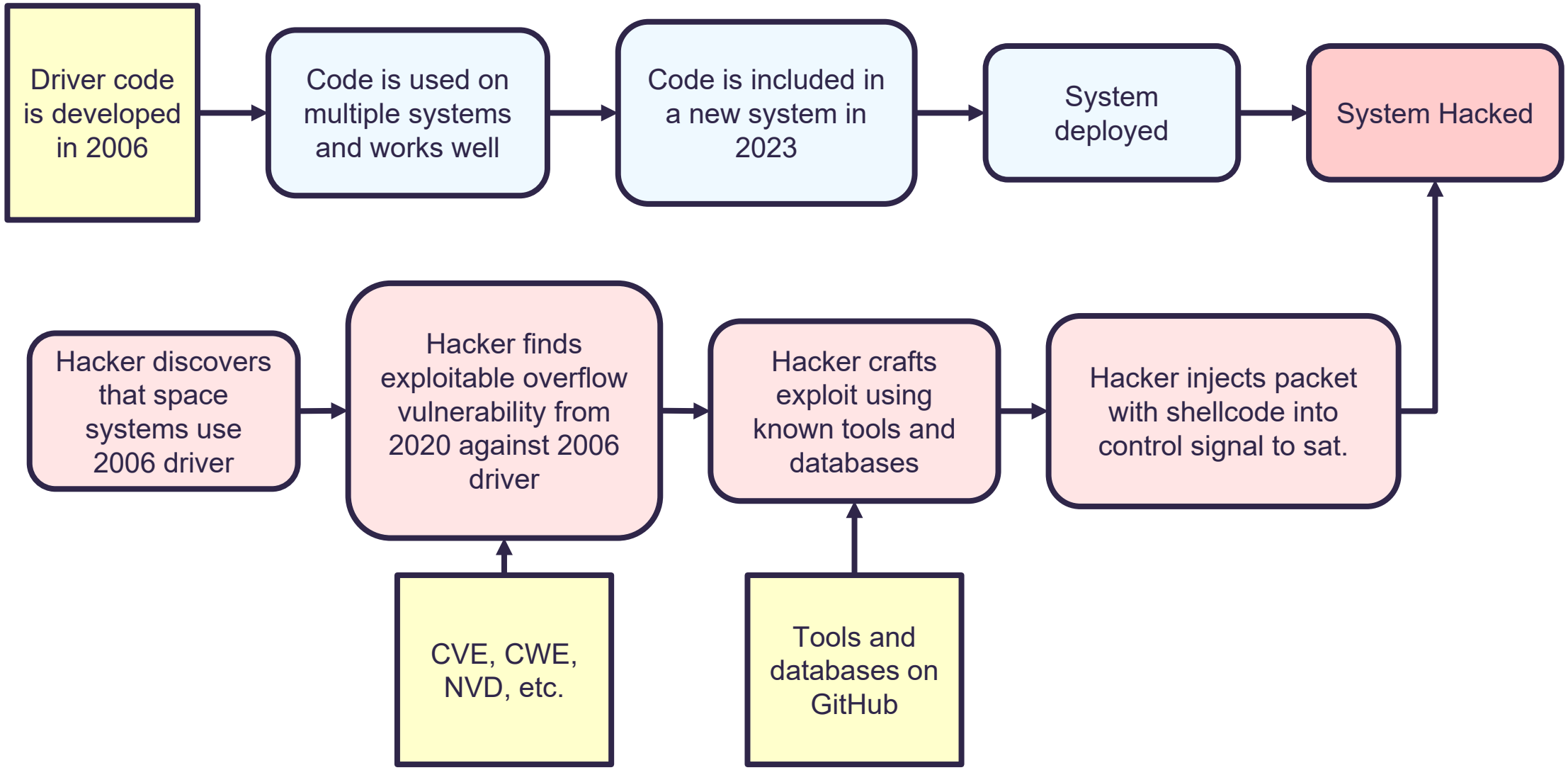


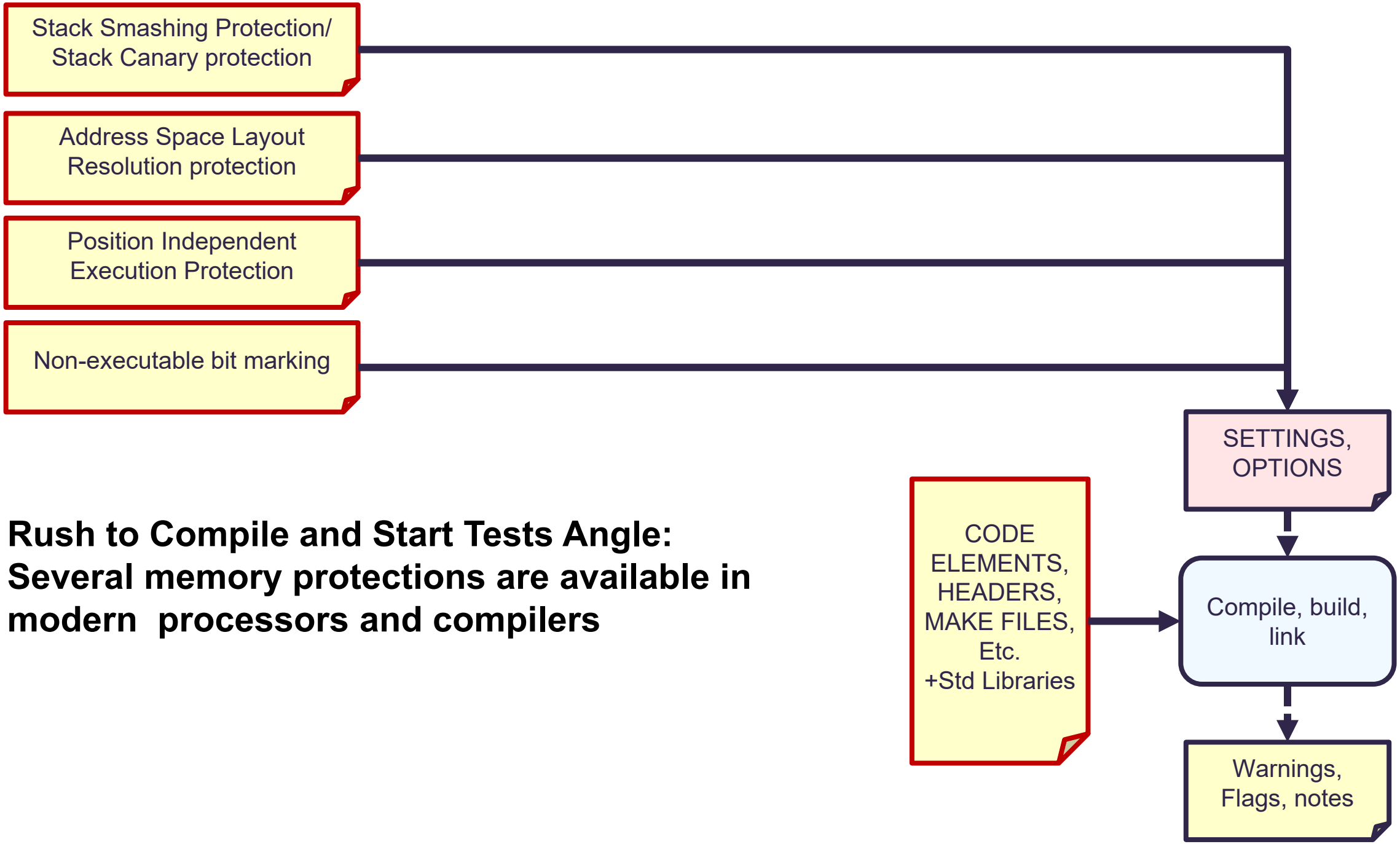
Rush to Include Code Angle: Code Inclusion in modern aerospace systems

Rush to Include Code Angle: Off-nominal conditions are opportunities, especially in included code.



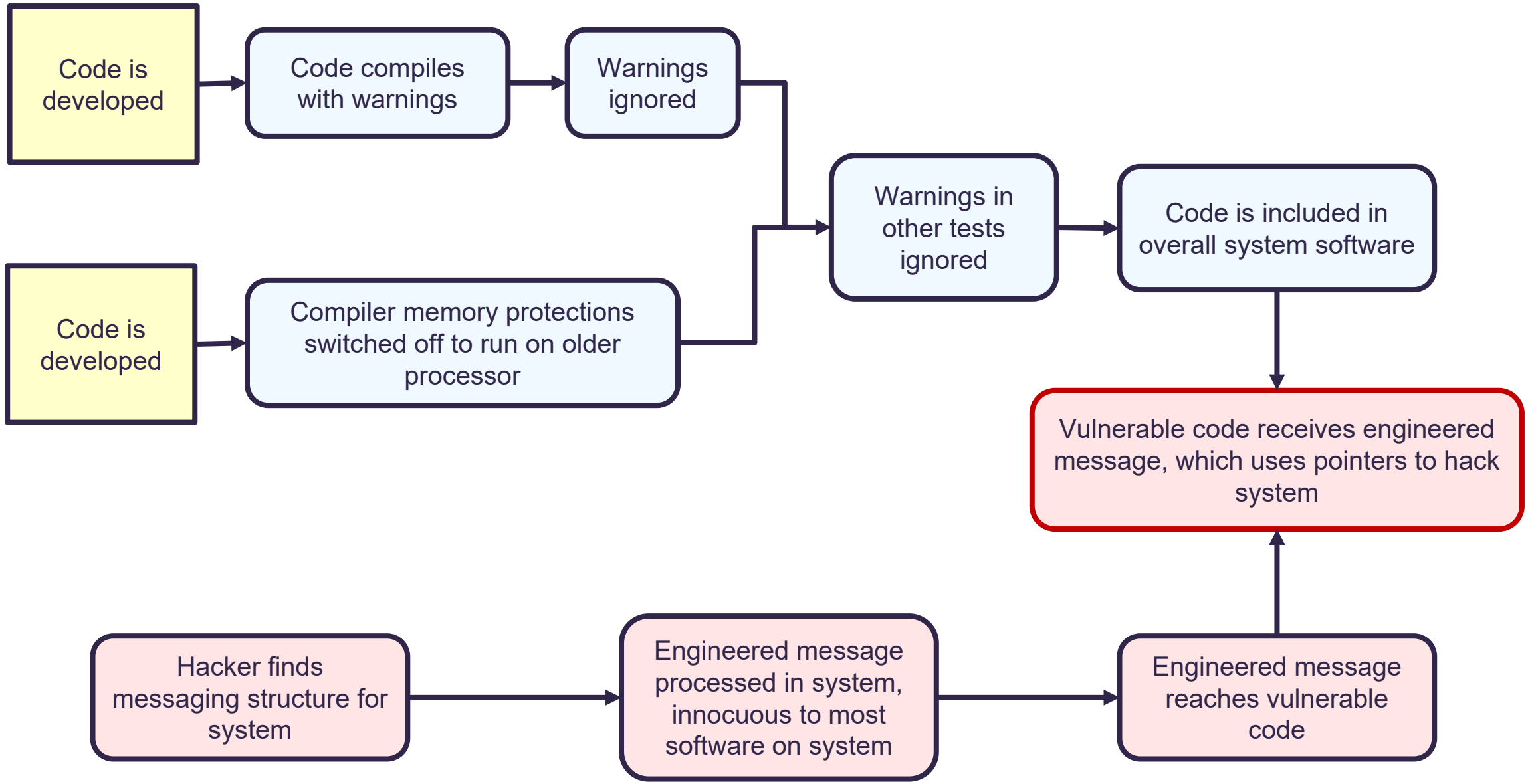
Rush to Include Code Angle: Hacker uses opportunity in included code…



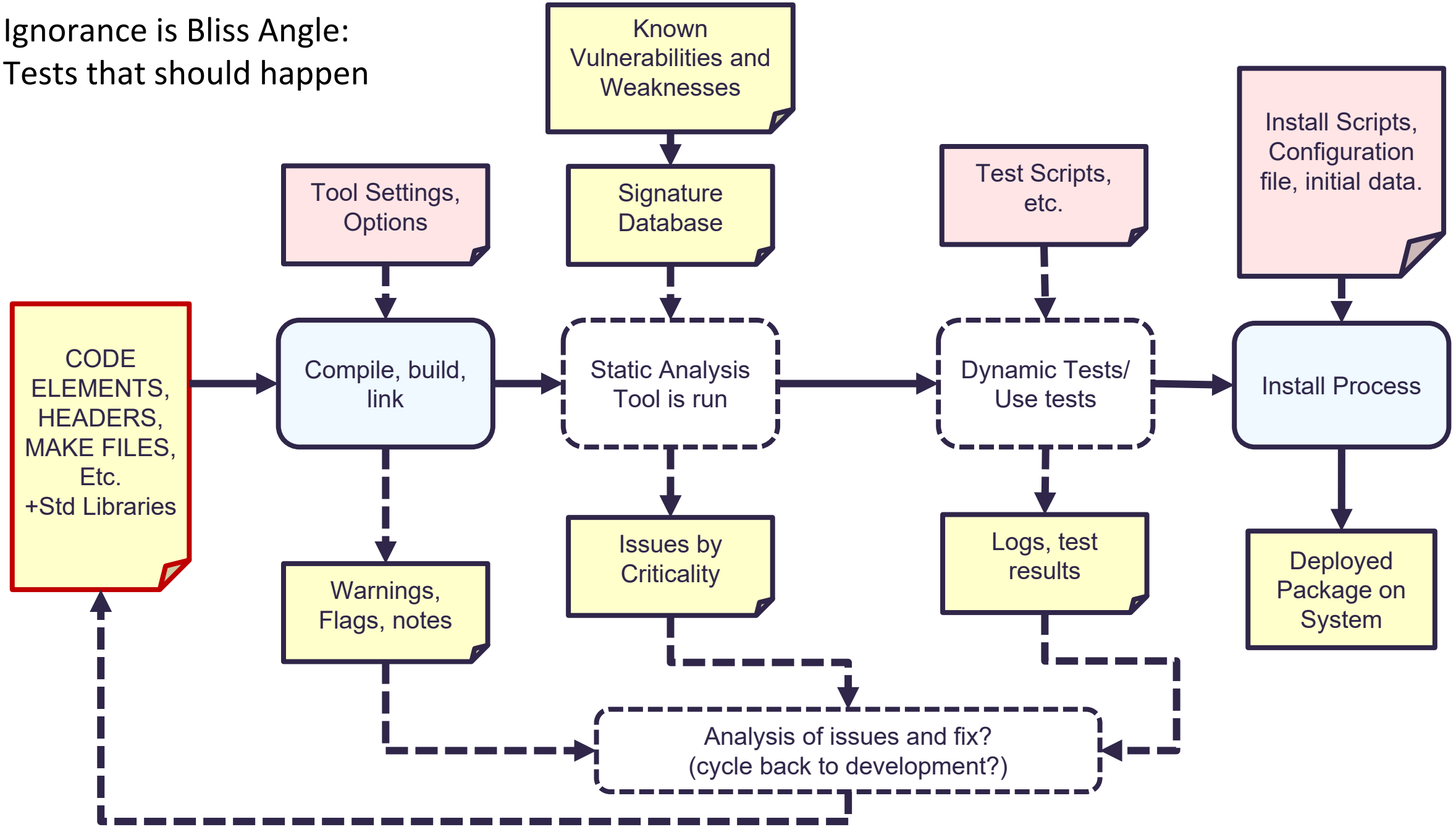


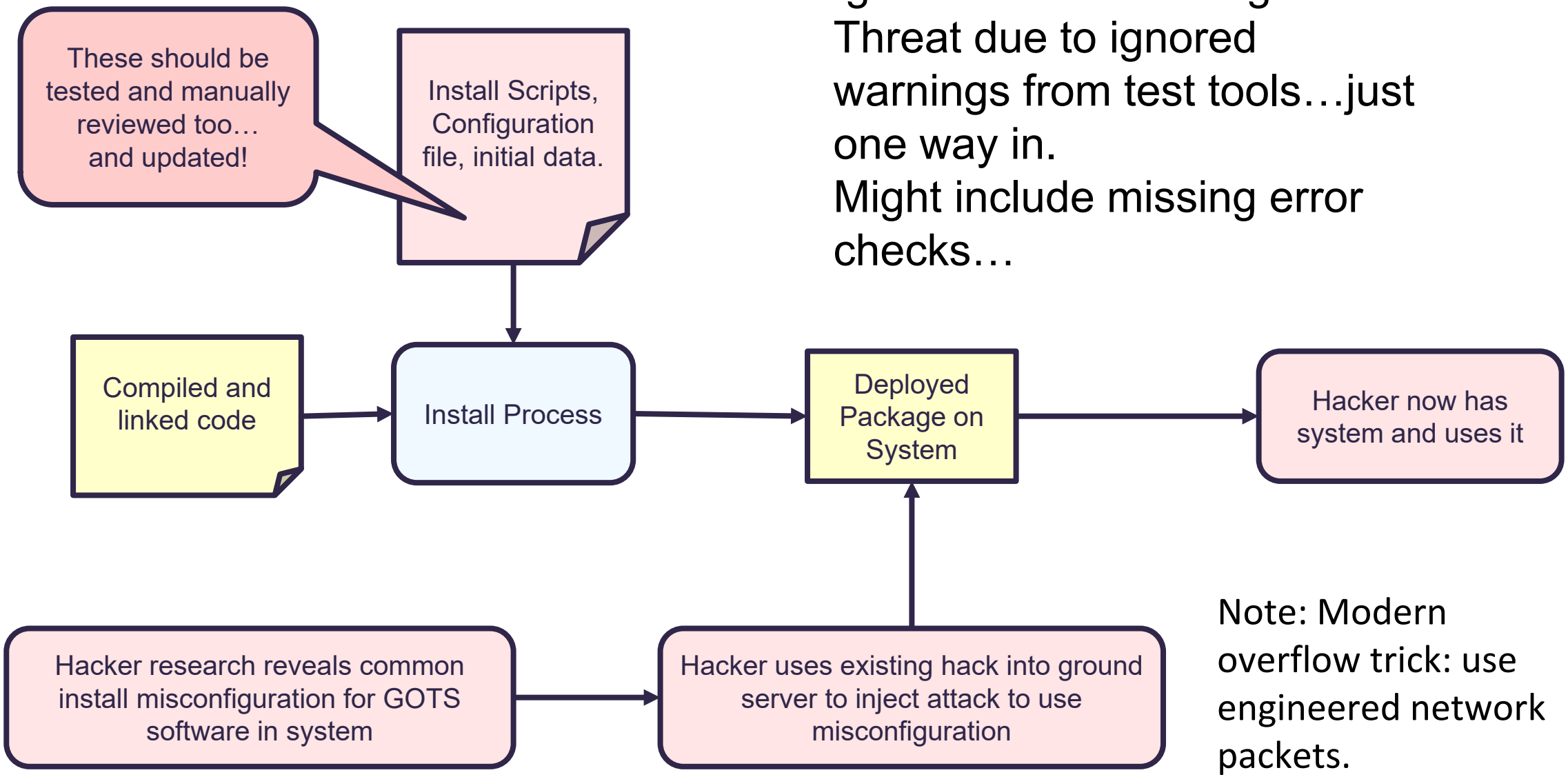
Rush to Compile and Start Tests Angle: Several memory protections are available in modern processors and compilers

Rush to Compile and Start Tests Angle: Compilers can be a line of defense.



Ignorance is Bliss Angle:
Tests that should happen





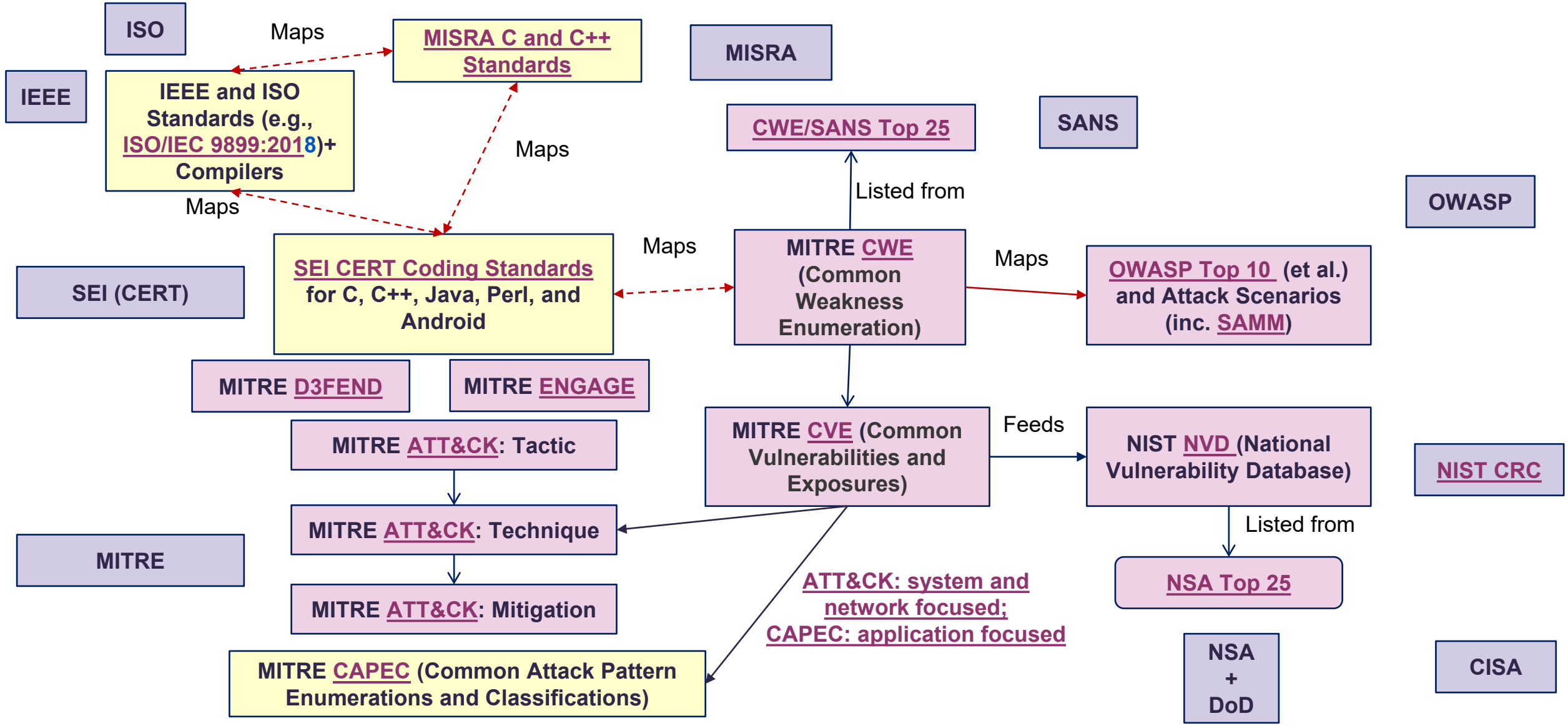
Ignorance is Bliss Angle:
Threat due to ignored warnings from test tools…just one way in.
Might include missing error checks…

Note: Modern overflow trick: use engineered network packets.

Prevention

- PLAN TO ALLOW FOR TIME and FUNDING TO EXAMINE THE THREAT ANGLES! Rushing means including issues for later!
- Do not push past critical examinations!
- Following the DevSecOps process, and pay attention to included code, compile settings, and test warnings
- Examine outputs from compile, linting, static analysis in detail, and act upon those that lead to threats
- Use secure coding standards (see next slide)
- Plan for updates!

Lists of Threats and Security Practices



Conclusion

Memory overflows are but one vulnerability of many, and older software and systems are more susceptible that newer systems

Memory overflows capitalize on mismatch between design and code

Many aerospace systems may not be updatable…satellites, probes, etc.

Threat angles are the result of a ***rush to finish***, so managers must plan and budget for time to avoid the angles:

1. Rush to Include Code Angle: Included code has a threat
2. Rush to Compile and Start Tests Angle: Compiler settings or ignored compiler warnings cause threat
3. Ignorance is Bliss Angle: Tests are performed, but warnings are ignored

Independent Verification and Validation including Red Teams can catch them!

References

[1] “Smashing The Stack For Fun And Profit” by Aleph One, 1996, Underground.Org, https://inst.eecs.berkeley.edu/~cs161/fa08/papers/stack_smashing.pdf
[2] Common Vulnerabilities and Exposures (CVE): https://www.cve.org/
[3] Top 10 Secure Coding Practices, created by R. Seacord, found at: https://wiki.sei.cmu.edu/confluence/display/seccode/Top+10+Secure+Coding+Practices
[4] “DevSecOps Fundamentals Guidebook: DevSecOps Tools & Activities”, March 2021, v2.0, U.S. Department of Defense,
[5] ISO/IEC Standard 9899:2018, International Organization for Standardization, Chemin de Blandonnet 8 CP 401, 1214 Vernier, Geneva, Switzerland, © 2018, found at: https://www.iso.org/standard/74528.html
[6] Open Worldwide Application Security Project (OWASP) Security Knowledge Framework, found at: https://owasp.org/www-project-security-knowledge-framework/
[7] SEI CERT C Coding Standard, available online at: https://wiki.sei.cmu.edu/confluence/display/c/SEI+CERT+C+Coding+Standard
[8] Seacord, R. C., *SEI CERT C Coding Standard: Rules for Developing Safe, Reliable, and Secure Systems (2016 Edition)*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2016, URL: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=454220.
[9] Seacord, R. C., *Secure Coding in C and C++ (SEI Series in Software Engineering)*, 2nd ed., Addison-Wesley Professional, Upper Saddle River, NJ, 2013.
[10] Bryce L. Meyer, “Perilous Paths: The Cybersecurity Issues of a Space-Based Cloud Imagery Processing System” AIAA 2022-4327, 15 Oct 2022, https://doi.org/10.2514/6.2022-4327
[11] K. Lukin and M. Haselberger, "Hacking Satellites With Software Defined Radio," 2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC), San Antonio, TX, USA, 2020, pp. 1-6, doi: 10.1109/DASC50938.2020.9256695.
[12] Alberts, C. J., Dorofee, A. J., Stevens, J. F. Stevens, and Woody, C., *Introduction to the OCTAVE Approach*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2016, URL: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=51546.
[13] National Institute of Standards and Technology, “Security and Privacy Controls for Information Systems and Organizations,” Special Publication 800-53, Revision 5, NIST, Gaithersburg, MD, September 2020. URL: https://doi.org/10.6028/NIST.SP.800-53r5.
[14] Common Weakness and Enumeration (CWE): https://cwe.mitre.org/
[15] MITRE ATT&CK ™: https://attack.mitre.org/
[16] MITRE Engage ™: https://engage.mitre.org/
[17] MITRE D3FEND ™: https://d3fend.mitre.org/
[18] Allen Harper, Ryan Linn, Stephen Sims, Michael Baucom, Huascar Tejeda, Daniel Fernandez, Moses Frost, Gray Hat Hacking: The Ethical Hacker's Handbook, Sixth Edition 6th Edition, McGraw Hill, ISBN-10, 1264268947, ISBN-13, 978-1264268948
[19] Sanders and Kordella, “Cyber Best Practices for Small Satellites”, https://www.mitre.org/sites/default/files/2021-11/prs-19-03753-01-cyber-best-practices-for-small-satellites.pdf