

NVD-BF (or ^{Alt.}NVD^{BF})

Formal Vulnerability Classifications Platform to Accelerate AI and FM Cybersecurity R&D

Concept Pitch
IMS FY 2027
January 22, 2026

NVD – National Vulnerability Database
BF – NIST Bugs Framework
AI – Artificial Intelligence
FM – Formal Methods
R&D – Research and Development



NIST National Institute of
Standards and Technology
U.S. Department of Commerce

Irena Bojanova
(ITL/775)

NVD-BF is envisioned as a formal vulnerability classifications platform that would advance hardware and software vulnerability analytics and accelerate AI and FM-based cybersecurity R&D.

Reality

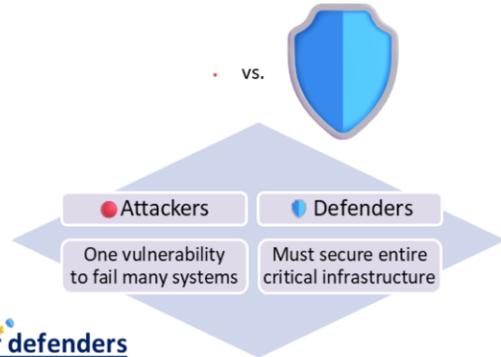
Vulnerability R&D → Cybersecurity

NIST

- Vulnerability exploits – primary infection vector



- Defender's Paradox



- Finding/fixing **code** defects: **\$600+ B**
CISQ, 2022

➔ Accelerate vulnerability R&D to empower defenders

- ✓ AI-based agents and systems
- ✓ Formal verification methods

NIST Priority: Secure US critical infrastructure.

1. Bolognini, 2023

The cybersecurity reality is:

- Hardware and software vulnerability exploits account for 33% of the initial vectors in cyberattacks.
- Finding and fixing code bugs/faults costs more than 600 billion dollars per year.
- Attackers need single exploitable vulnerability to compromise many systems, but defenders must secure the entire critical infrastructure.

→ To empower defenders, we must accelerate cybersecurity vulnerability R&D.

Goal

Advance Vulnerability Analytics

NIST

- Vulnerability repositories
 - [NVD](#), [KEV](#)
 - [CVE](#) (& [CWE](#))
- Problem – limited analytics
 - Narrative descriptions
 - Static, subjective data
 - Lack of labeled datasets



➔ Multi-dimensional vulnerability formalism?

- ✓ Formal specification
- ✓ Secure coding principles
- ✓ Automatic generation
- ✓ Dynamic vulnerability classifications
- ✓ Contextual connections
- ✓ Analytics and scoring

NIST Priority: AI-based agents to secure US critical infrastructure.

1. Background, 2024

- Current vulnerability repositories are based on NL descriptions and static subjective data.
- Cybersecurity experts and automated systems struggle due to the lack of comprehensively labeled vulnerability datasets for AI and FM R&D.

→ To advance vulnerability analytics, we need dynamic, formal, multi-dimensional classifications.

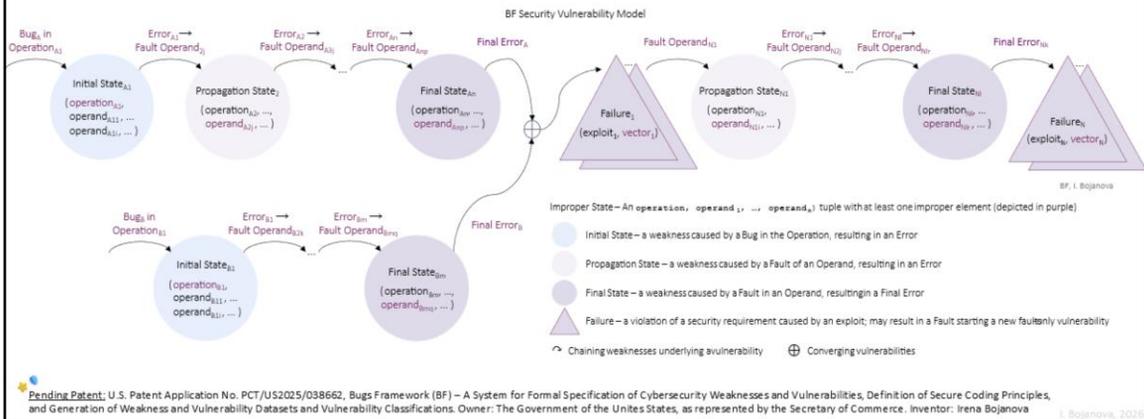
→ Descriptions and classifications should be automatically generated utilizing a powerful vulnerability formalism.

Utilize

Bugs Framework (BF)

NIST

- Multi-dimension classification of security bugs and related to them faults
- Understand vulnerabilities – weakness causation, propagation, convergence



The Bugs Framework (BF) is a structured multi-dimensional classification of security bugs and related to them faults, supporting deeper understanding of vulnerabilities as chains of weaknesses, adhering to strict causation, propagation, and convergence rules.

BF's formal language is based on rigorous BF bugs models, causal taxonomies, and a vulnerability specification model.

Example

NVD-BF: Heartbleed



NATIONAL VULNERABILITY DATABASE **NIST**

CVE-2014-0160 Detail

Description

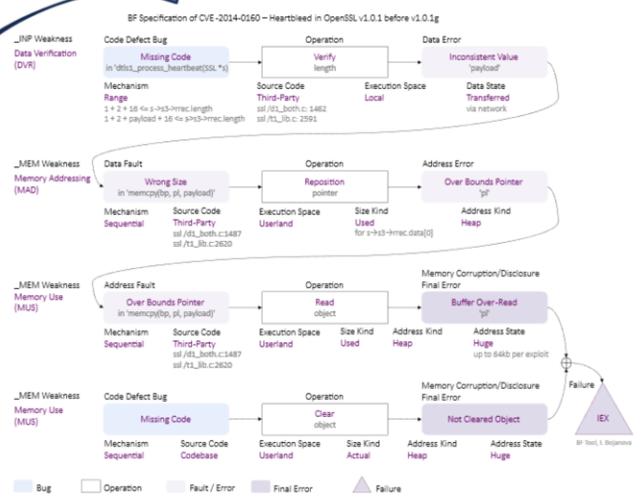
The (1) TLS and (2) DTLS implementations in OpenSSL 1.0.1 before 1.0.1g do not properly handle heartbeat Extension packets, which allows remote attackers to obtain sensitive information from process memory via crafted packets that trigger a buffer over-read, as demonstrated by reading private keys, related to ssl_both.c and tl_1st.c, aka the Heartbleed bug.

Metrics

CVSS Version 4.0	CVSS Version 3.0	CVSS Version 2.0
NIST: NVD	Base Score: 8.8	Vector: CVSS:3.1/AV:N/AC:L/PRN/UI:N/SU:C/H:N/A/R
ADP: CISA-ADP	Base Score: 8.8	Vector: CVSS:3.1/AV:N/AC:L/PRN/UI:N/SU:C/H:N/A/R

Weakness Enumeration

CWE-ID	CWE Name	Source
CWE-122	Out-of-Bounds Read	NIST - CISA-ADP



- Augment**
- ✓ NVD description → BF specification
 - ✓ CVSS scores → BF attributes analytics
 - ✓ CWE assignments → BF security rules

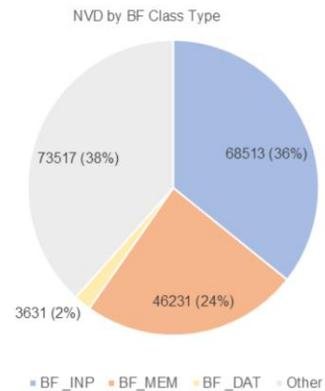
For example, the Heartbleed description in NVD will be augmented with its BF specification of two converging chains of weaknesses, as well as, provided in machine-readable formats.

The CVSS severity scores – via BF attributes analytics and the CWE weakness assignments – via specific BF security rules.

NVD labels 190,000+ CVEs with CWEs

- 60% map to two BF Class Types
 - ✓ 68,000+ → BF_INP
 - ✓ 46,000+ → BF_MEM
- Most relate to
 - ✓ Injection ← SQL Injection
 - ✓ Memory Corruption/Disclosure ← Buffer Overflow

➔ **Now it's the time!**



J. Rodriguez, 2015

- NVD labels close to 200,000 CVEs with CWEs, which allows mapping them to BF classes.
- 60% of these CVEs map to BF Input Check or Memory Bugs classes.
- Many lead to the most dangerous exploitable errors: Injection (e.g., SQL injection) and Memory Corruption (e.g., buffer overflow).

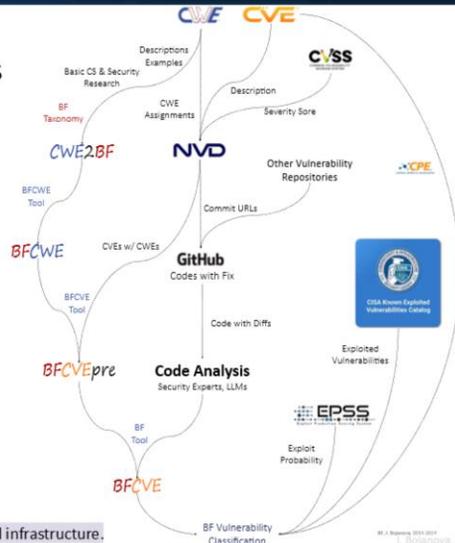
→The thousands of known vulnerabilities form the basis for a first comprehensively labeled vulnerability dataset and multi-dimensional classifications.

Approach

AI-Based Agents

NIST

- Analyze ← BF definitions, taxonomy, rules
 - ✓ Narrative descriptions ← NVD, CVE, CWE, KEV, CPE, EPSS
 - ✓ Software ← source code
 - ✓ Hardware code ← firmware, microcode
 - Generate ← BF formal language
 - ✓ Vulnerability specifications
 - ✓ Secure coding principles
 - Validate/Verify
 - ✓ Toward BF Formal Language
- ➔ **AI-centered vulnerability platform**
- ✓ Comprehensively labeled vulnerability datasets
 - ✓ AI agents, APIs, UIs, and Apps



We will create BF-based AI autonomous agents:

- agents to autonomously process NL descriptions from NVD and other vulnerability repositories
- agents to analyze firmware and software code with bugs and fixes
- agents to generate formal vulnerability specifications and secure coding principles
- agents to validate and verify generated specifications toward the BF formal language.

Impact

NVD–BF (or NVD^{BF}) Platform

- **Why NIST?**
 - ✓ Team expertise in cybersecurity, FM, AI-based agent
 - ✓ NIST BF inventor, patent in NIST custody
 - ✓ Industry, government, academia partnerships

➔ Potential impact

- ✓ Unprecedented cybersecurity analytics capabilities
- ✓ Highly informed cybersecurity R&D innovations
- ✓ Unambiguous communication about cybersecurity
- ✓ Solidifying NIST NVD as authoritative reference

Outcome

- ✓ Breakthrough vulnerability classifications
- ✓ AI-based agents, APIs, UIs, and Apps

→ Establish NIST BF as the standard for specifying cybersecurity vulnerabilities

- **Risks if not done**
 - ✓ Cybersecurity experts and automated systems will continue to struggle due to lack of precise descriptions of the publicly disclosed vulnerabilities
 - ✓ Parallel efforts—[EUVD](#), [OSV](#), etc.—will eventually do it and gain advantage

NIST Priority: Secure US critical infrastructure.
1. Berman, 2018

Why NIST?

- We will form a team with the exact expertise in cybersecurity, FM, AI models, and AI-based agents.
- I am the inventor of the BF, as the theoretical basis for this project. The patent is owned by the Government of the United States, as represented by the Secretary of Commerce; the patent is in the custody of NIST.
- Strong industry, government, academia partnerships

The resulting outcome will be a breakthrough platform with formal multi-dimensional vulnerability classifications and AI-based agents, APIs, UIs, and Apps for cybersecurity R&D.

The potential impact would be

- Unprecedented multi-dimensional cybersecurity analytics capabilities
- Highly informed innovations in bug/fault detection, vulnerability resolution or mitigation, and development of effective countermeasures against potential security threats and specific exploits
- Unambiguous communication about cybersecurity across industry, government, and academia

- Solidifying NIST NVD as authoritative reference

→ Finally, the NVD-BF would establish NIST BF as the standard for specifying cybersecurity vulnerabilities

It this is not done, cybersecurity experts and automated systems will continue to struggle due to lack of precise descriptions of the publicly disclosed vulnerabilities.

Parallel efforts—European Union Vulnerability Database (EUVD), Open Source Vulnerability (OSV) Database, etc.—will eventually go in this direction and gain advantage.



Potential Collaborators



- **NIST Researchers**

- Irena Bojanova —cybersecurity, formal methods— ITL/775.01
- Tim Blattner —AI models, AI-based agents— ITL/775.02
- Jason Collard—AI, NLP, formal methods— ITL/775.04
- Jeff Voas —cybersecurity, digital twin— ITL/773.03
- Rick Kuhn —combinatorial testing— ITL/773.02

- **Non-NIST PhD Candidates**

- Purdue University
- Colorado State University
- Idaho National Laboratory
- JHU, Applied Physics Laboratory
- University of West Attica
- University of Quebec

I. Bojanova, 2024

References

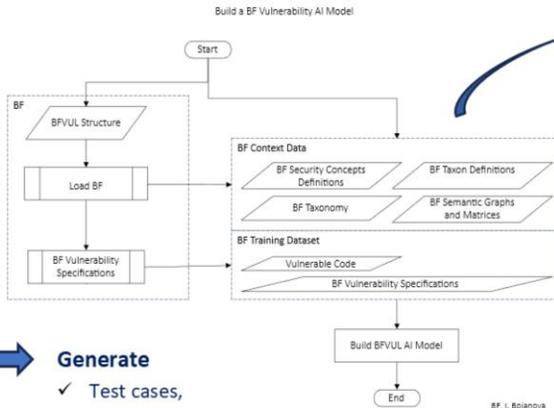


- [1] Bojanova I (2024) Bugs Framework (BF): Formalizing Cybersecurity Weaknesses and Vulnerabilities. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP), NIST SP 800-231. <https://doi.org/10.6028/NIST.SP.800-231>

PATENT PENDING

- [2] U.S. Patent Application No. PCT/US2025/038662, Bugs Framework (BF) – A System for Formal Specification of Cybersecurity Weaknesses and Vulnerabilities, Definition of Secure Coding Principles, and Generation of Weakness and Vulnerability Datasets and Vulnerability Classifications. Inventor: Irena Bojanova, NIST.

Extras



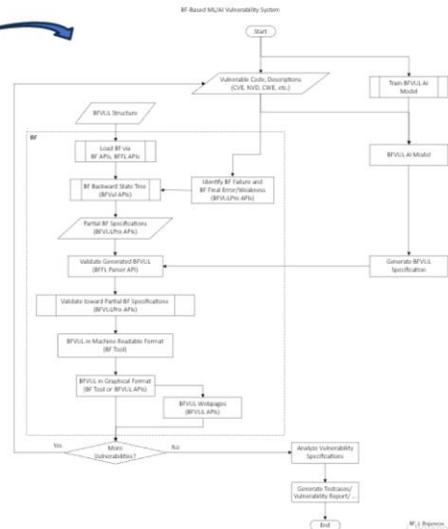
Generate

- ✓ Test cases,
- ✓ Vulnerability reports

Automate

- ✓ Bug/fault detection and prioritization
- ✓ Vulnerability resolution or mitigation

BF, I. Bojanova



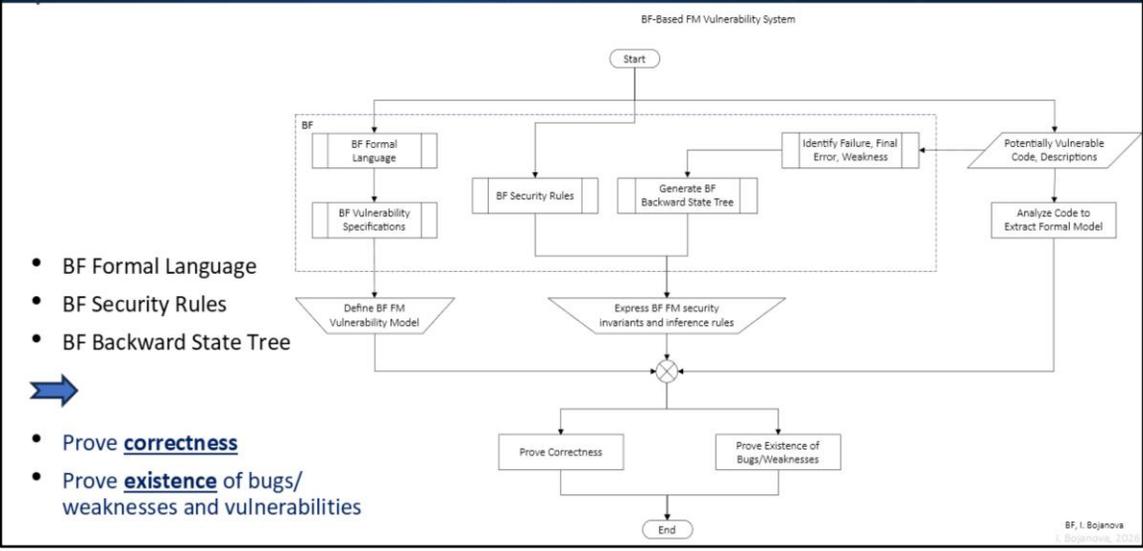
M.S. Rahman, 2021

Broader applications support would be for:

- Automation of bug/fault detection and vulnerability resolution or mitigation
- Generation of testcases and vulnerability reports

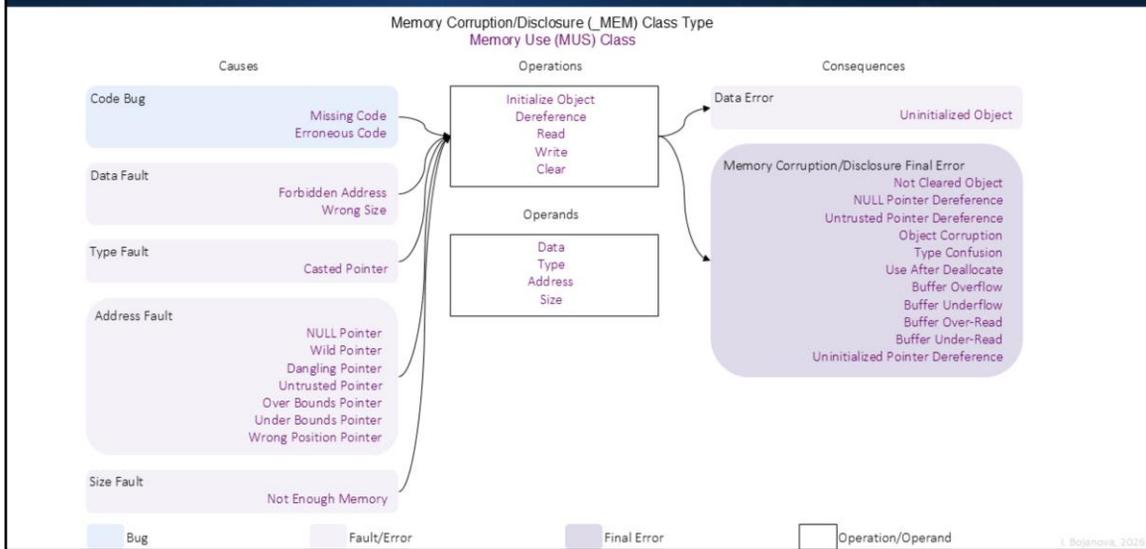


BF-Based FM-Powered Systems



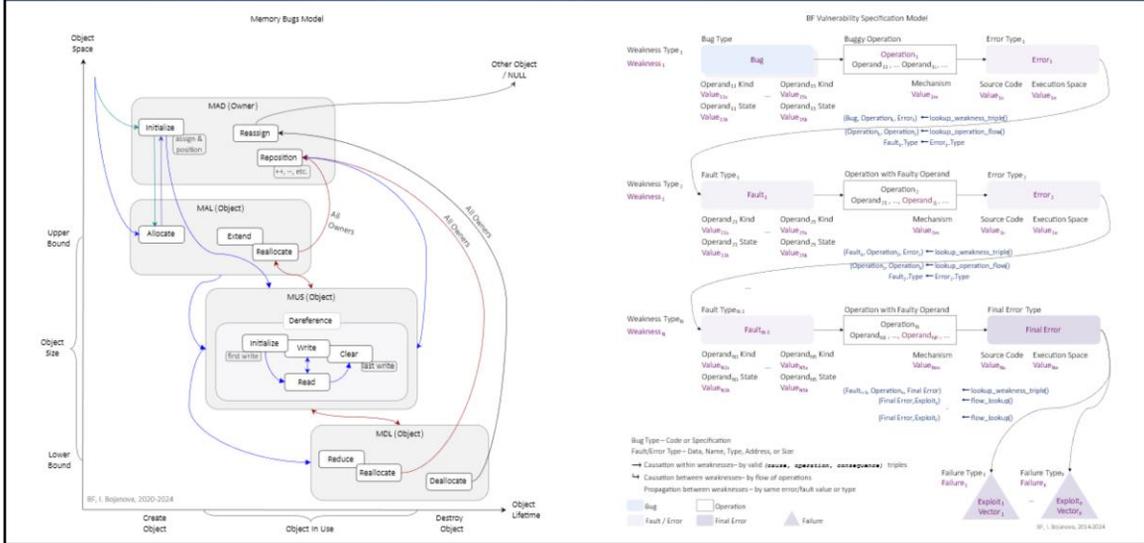
Given the formal specification of code and the BF security concepts definitions, FM could be applied to prove correctness or prove the existence of bugs/faults and related vulnerabilities.

BF Causal Taxonomies



BF organizes bugs by the operations of distinct execution phases, faults by their input operands, and errors by their output results. Bugs and faults are the possible causes for security weaknesses; errors and final errors -- their possible consequences.

BF Bugs and Specification Models



BF's weakness taxonomies and bugs and specification models form the basis for defining secure coding principles, such as memory safety, input/output safety, and data type safety.

- Specification of cybersecurity weaknesses and vulnerabilities

```

SyntaxRules :
    S ::= Vulnerability Converge_Failure
    Vulnerability ::= Bug_Fault Operation OperAttrs_Error_FinalError
    Bug_Fault ::= Bug
                | Fault
    OperAttrs_Error_FinalError ::= OperationAttribute OperAttrs_Error_FinalError
                | Error Fault1 OprndAttrs_Operation
                | FinalError
    OprndAttrs_Operation ::= OperandAttribute OprndAttrs_Operation
                | Operationk OperAttrs_Error_FinalError
    Converge_Failure ::= ⊕ Vulnerability Converge_Failure
                | Vector Exploit NextVulner_Failure
    NextVulner_Failure ::= Fault2 OprndAttrs_Operation
                | Failure ε

SemanticRules :
    (Bug, Operation1, Error) ← lookup_weakness.triple()
    (Bug, Operation1, FinalError) ← lookup_weakness.triple()
    (Fault1, Operationk, Error), k > 1 ← lookup_weakness.triple()
    (Fault1, Operationk, FinalError), k > 1 ← lookup_weakness.triple()
    (Operation1, ..., Operationk), k > 1 ← lookup_operation_flow()
    Fault1 ← if (Fault1.ClassType == Error.ClassType) then Error

Predicates :
    Fault1.Type == Error.Type
    Vector.Type == FinalError.Type
    Fault2.Type == ExploitResult.Type

```

BF's formal language is based on rigorous BF bugs models, causal taxonomies, and vulnerability specification models.