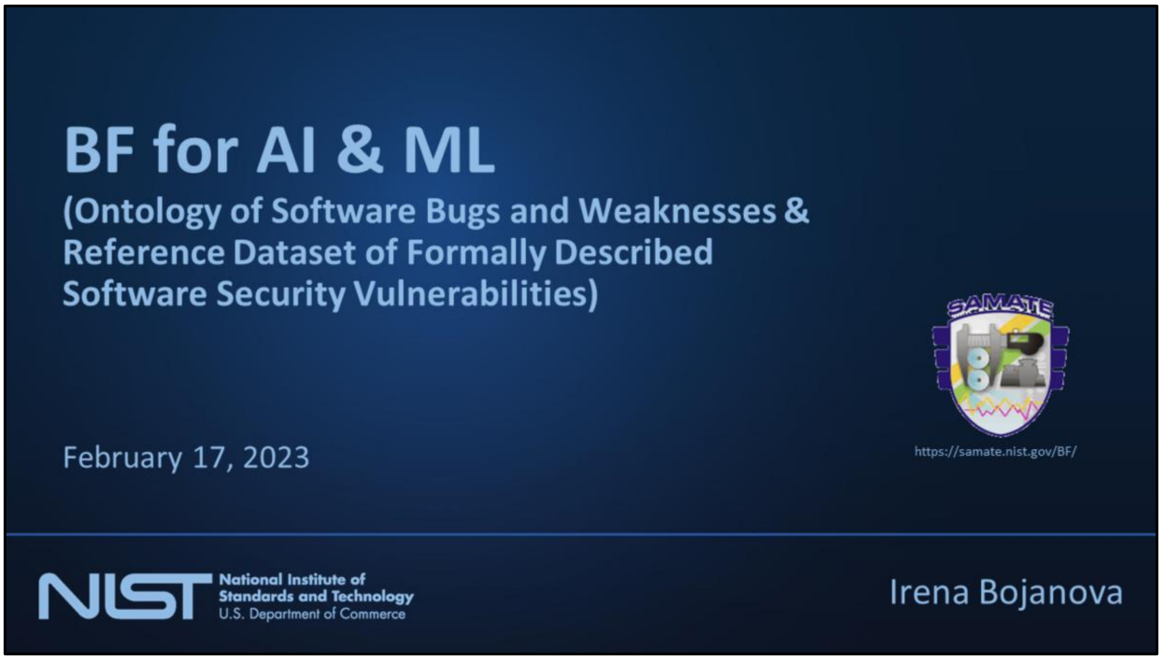


BF for AI & ML

(Ontology of Software Bugs and Weaknesses & Reference Dataset of Formally Described Software Security Vulnerabilities)

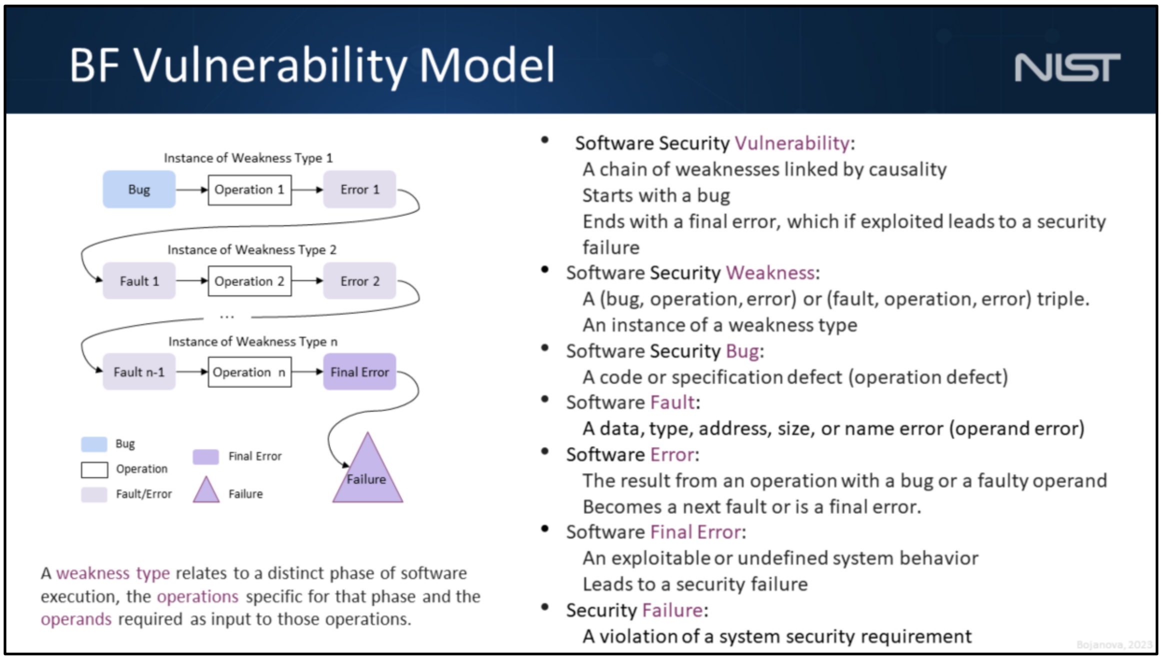
February 17, 2023

https://samate.nist.gov/BF/

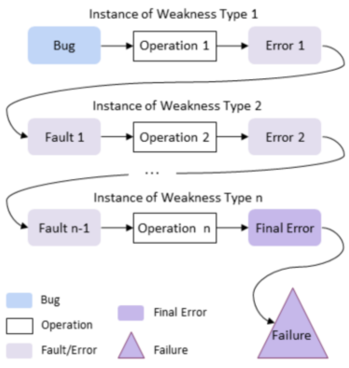
NIST National Institute of Standards and Technology U.S. Department of Commerce

Irena Bojanova

I am Irena Bojanova – a Computer Scientist at NIST and
the Primary Investigator and Lead of the Bugs Framework project.



BF Vulnerability Model



A weakness type relates to a distinct phase of software execution, the operations specific for that phase and the operands required as input to those operations.

- Software Security Vulnerability:
  A chain of weaknesses linked by causality
  Starts with a bug
  Ends with a final error, which if exploited leads to a security failure
- Software Security Weakness:
  A (bug, operation, error) or (fault, operation, error) triple.
  An instance of a weakness type
- Software Security Bug:
  A code or specification defect (operation defect)
- Software Fault:
  A data, type, address, size, or name error (operand error)
- Software Error:
  The result from an operation with a bug or a faulty operand
  Becomes a next fault or is a final error.
- Software Final Error:
  An exploitable or undefined system behavior
  Leads to a security failure
- Security Failure:
  A violation of a system security requirement

Bojanova, 2023

We are classifying software bugs and weaknesses to allow precise descriptions of vulnerabilities that exploit them.

This is our high-level vulnerability description model, which would allow backtracking to the bug from an observed security failure.

A software security vulnerability is a chain of weaknesses linked by causality.
It starts with a bug and ends with a final error, which if exploited leads to a security failure.

A software security weakness is a (bug, operation, error) or (fault, operation, error) triple.
It is an instance of a weakness type that relates to a distinct phase of software execution,
the operations specific for that phase and the operands required as input to those operations.

A software security bug is a code or specification defect -- an operation defect.

A software fault is a name, data, type, address, or size error -- an operand error.
Name is in reference to a resolved or bound object, function, data type, or namespace. The others are in reference to an object.

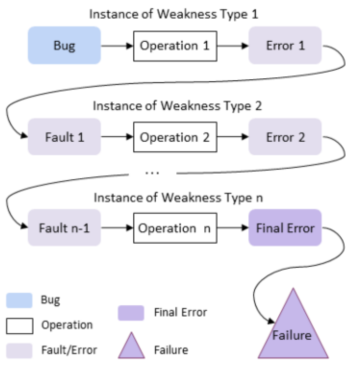
A software error is the result from an operation with a bug or a faulty operand. It becomes a next fault or is a final error.

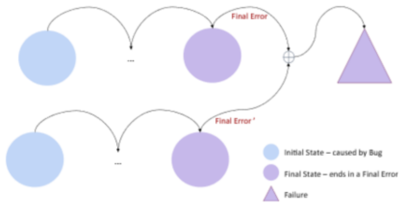
A software security final error is an exploitable or undefined system behavior that leads to a security failure.

A security failure is a violation of a system security requirement.

A chain of weaknesses starts with a bug, propagates through errors that become faults, and ends with a final error.
The final error is the one exploited by attackers towards a security failure. For example, missing input validation
may propagate to integer overflow, followed by buffer overflow, which if exploited may lead to a remote code execution failure.

BF Vulnerability Model





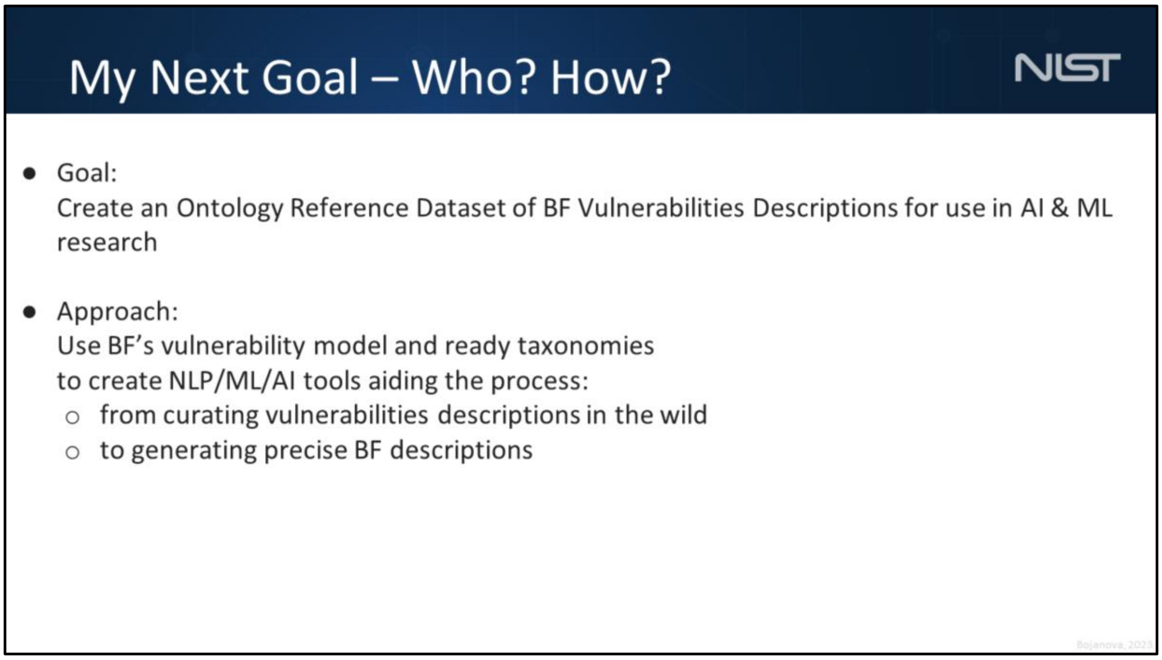
Final Error
...
Final Error'
...
Initial State – caused by Bug
Final State – ends in a Final Error
Failure

```
START := Vulnerability Converge

Vulnerability:= Bug Operation Error

Error := Fault Operation
       | FinalError

Converge:= Vulnerability Converge
         | Failure END
```

A security failure may be caused by the converging final errors of several vulnerabilities.
The bug in at least one of the chains must be fixed to avoid the failure.

BF's vulnerability model and BF's class taxonomies
allow BF to be expressed as a formal language.
This is the high-level LL1 grammar of BF.

Show BFCVE tool.
Show BF LL1 grammar.


My Next Goal – Who? How?

- Goal:
  Create an Ontology Reference Dataset of BF Vulnerabilities Descriptions for use in AI & ML research
- Approach:
  Use BF's vulnerability model and ready taxonomies
  to create NLP/ML/AI tools aiding the process:
  - from curating vulnerabilities descriptions in the wild
  - to generating precise BF descriptions

Bojanova, 2023

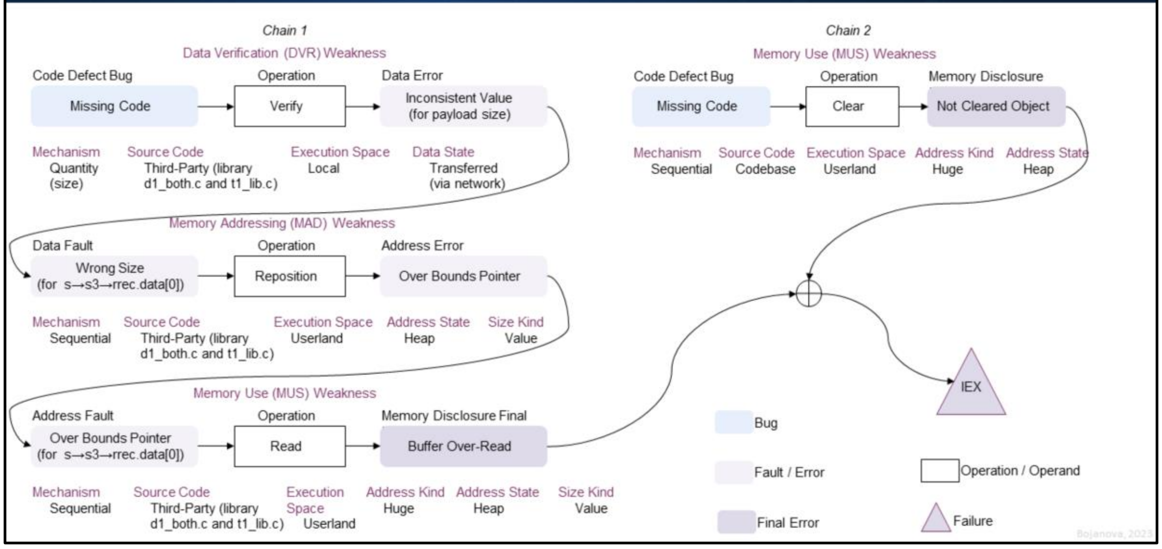
Ready BF class types (clusters):

_DAT: Data Type Bugs – Bugs/Faults supplying type compute exploit vectors:
- Declaration (DCL) class – An object, a function, a type, or a namespace is declared or defined improperly.
- Name Resolution (NRS) class – The name of an object, a function, or a type is resolved improperly or bound to an improper type or implementation.
- Type Conversion (TCV) class – Data are converted or coerced into other type improperly.
- Type Computation (TCM) class – An arithmetic expression (over numbers, strings, or pointers) is calculated improperly, or a boolean condition is evaluated improperly.

_INP: Input/Output Check Bugs – Bugs/Faults supplying injection exploit vectors.
- Data Validation (DVL) class – Data are validated (syntax check) or sanitized (escape, filter, repair) improperly.
- Data Verification (DVR) class – Data are verified (semantics check) or corrected (assign, remove) improperly.

_MEM: Memory Bugs – Bugs/Faults supplying memory corruption/disclosure exploit vectors.
- Memory Addressing (MAD) class – The pointer to an object is initialized, repositioned, or reassigned to an improper memory address.
- Memory Management (MMN) class – An object is allocated, deallocated, or resized improperly.
- Memory Use (MUS) class – An object is initialized, read, written, or cleared improperly.

BF Description of Heartbleed

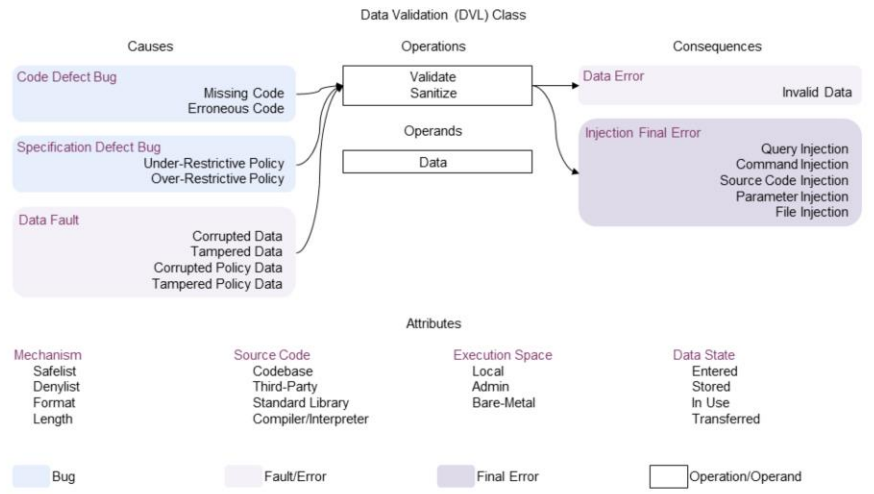


For example, this is the BF description of Heartbleed that we created
by manually examining sources with textual descriptions and code.

Now how to automatize this?

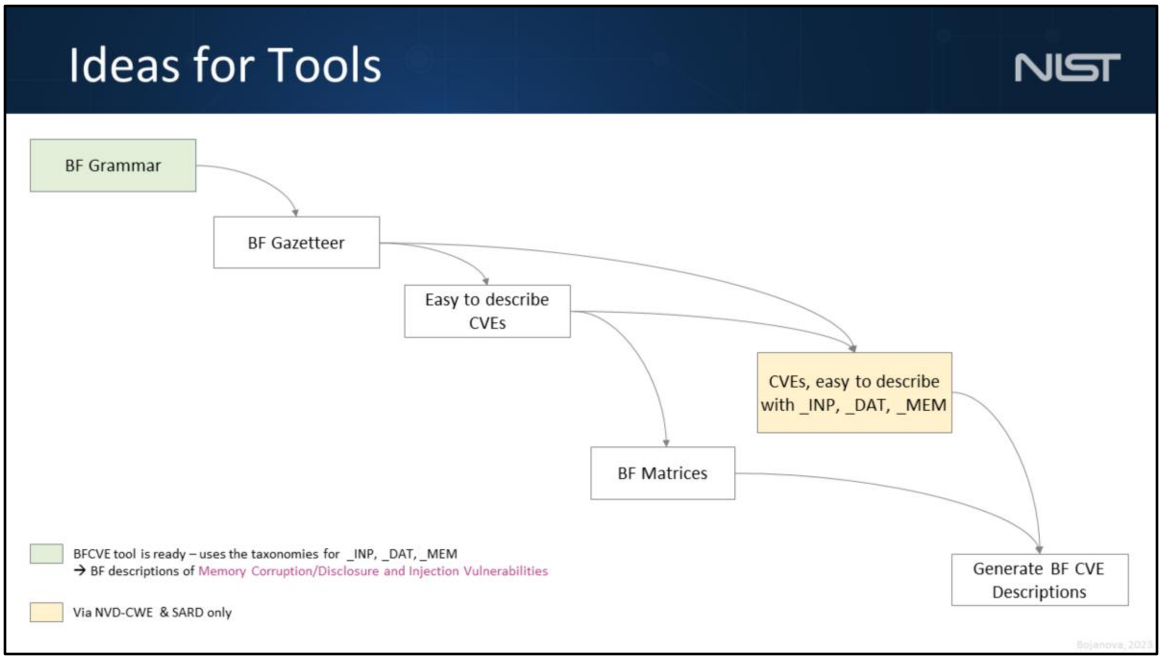
We could reason that BF is to CVEs as the Semantic Web is to Wild World Web (WWW).
A BF description would be like metadata over a set of documents that describe a vulnerability.
If we look at BF as an ontology, we need to parse any available documents to find the entities of that ontology.
The BF entities are operation, bug, fault, error, and attributes.

BF DVL Class Taxonomy



We also have relationships based on the class definitions:
For each class, for example the DVL class here,
there is a set of operations + a group of bug/fault causes
that lead to a group of errors
(note that we still need to develop matrices of meaningful transitions).

This should correspond to NPL Relationship Extraction (RE), so we could use Named-Entity Recognition (NER).



🟩 BFCVE tool is ready – uses the taxonomies for _INP, _DAT, _MEM
→ BF descriptions of Memory Corruption/Disclosure and Injection Vulnerabilities

🟨 Via NVD-CWE & SARD only

My BFCVE Tool is practically ready – we can describe memory corruption and injection vulnerabilities caused by input and data type related bugs – and these are anyway the most dangerous and frequent.

What next? Who and How?:
- Create a BF Gazetteer – NLP?
- Select easy to describe CVEs – ML?
- Select from these such that are easy to describe with _INP, _DAT, _MEM
- Create the BF matrices
- Generate BF descriptions

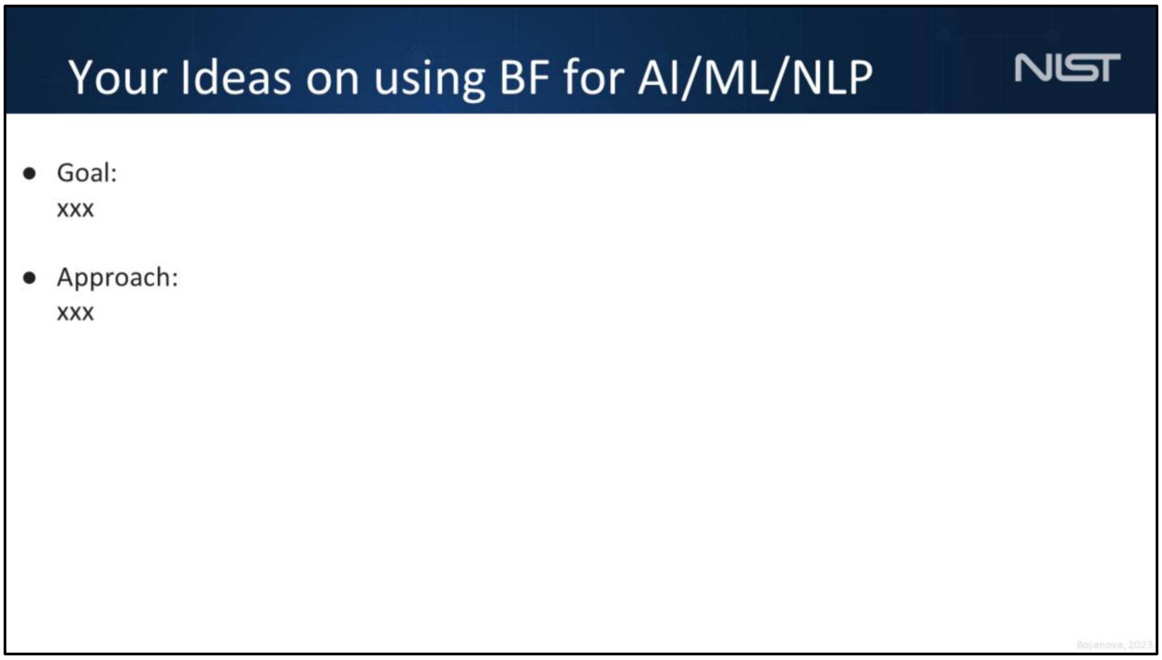
Some of these five projects could go in parallel. Then we chain them.

The BF Gazetteer step, should get us a list of words that people use to refer to BF terms.
Using NLP to find out for each value of each BF entity what other words people use it in plain English texts.
This could be done manually or using rule-based RE, unsupervised ML, or supervised ML.
____

A first guess would be to try rule base approach to select easy-to-describe CVEs and other texts.
Also, we should use something to easily detect CVEs we can describe with the classes we have.
This is connected to the gazetteer previously mentioned.

From there, we can go manually describing CVEs or attempt some ML until we reach a good amount of data (4k annotated words in 50k-100k words).

After this, it should be easy to create a deep learning network to BF-classify vulnerability text.



Your Ideas on using BF for AI/ML/NLP

NIST

- Goal:
  xxx
- Approach:
  xxx

What would be your ideas about AI/ML research with or without such a reference dataset?

The CHIPS initiative! Extending the BF taxonomies to cover firmware bugs and weaknesses.
Developing the tools described above.

Questions & Discussion

irena.bojanova@nist.gov

Please do not hesitate to contact me with questions and ideas for collaboration.