The Bugs Framework (BF) – New Classes

BF – software Developers' and Testers' "Best Friend."

Irena Bojanova

National Institute of Standards and Technology (NIST)

NITRD, SPSQ IWG Monthly Meeting – July 11, 2019

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

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

Outline

1. Information Exposure (IEX)
2. Randomness Cluster (RND)
   - Pseudo-Random Number Bugs (PRN)
   - True-Random Number Bugs (TRN)
3. Cryptography Cluster (CRY)
   - Key Management Bugs (KMN)
   - Encryption Bugs (ENC)
   - Verification Bugs (VRF)
4. Access Control Cluster (ACC)
   - Authentication Bugs (ATN)
   - Authorization Bugs (AUT)
5. Memory Cluster (MEM)
   - Memory Allocation Bugs (MAL)
   - Memory Initialization Bugs (INI)
   - Memory Use Bugs (MUS)
     Buffer Overflow (BOF) → refined
6. Injection (INJ) → refined
7. Control of Interaction Frequency (CIF) → refined

Information Exposure (IEX)

Information Exposure

- Information and data can be stored, transferred, and used by digital systems.
- Information exposure (information leaks) occurs when
  the system inadvertently reveals sensitive information inappropriately.

→ Exposure of sensitive information can be harmful on its own.
In addition, it could enable further attacks.

- Through information exposure the software may reveal:
  - login credentials
  - private keys
  - state and system data
  - personal data
  - financial data
  - health data
  - business data.

Information Exposure Faults

Formalizing information exposure faults would help researchers and practitioners identify them and avoid related failures.

To describe them, we developed:

- A general descriptive Model of Information Exposure
- One new BF class:
  - ✓ Information Exposure (IEX)

Information Exposure – Related Terms

- The terms “data” and “information” are often used interchangeably
  - *Data* is a set of values of qualitative or quantitative variables.
  - *Information* is any entity or form that provides the answer to a question of some kind or resolves uncertainty.

- To what extent data is informative to someone depends on how unexpected it is to that person.
  - Data has no meaning, while
  - Information has meaning.

- Information and data are on a continuum:
  - Bits in memory are data. Without external context (meaning), the bits might represent an integer, a memory address, a set of flags or other low-level information.
  - At a higher level, the integer could be someone’s age, the number of characters in a document, or a temperature.

Information Exposure – Related Terms

- In software, information is generated by processing data.
- We distinguish between information that is sensitive and information that is not
  - Certain kinds of information can be indirectly sensitive:
    when revealed can lead to harmful consequences.

→ If sensitive information at rest is properly encrypted (see BF ENC),
then information cannot be exposed, assuming a secure decryption key.

→ If information is communicated via a secure channel, it cannot be exposed either.

- Sensitive information includes:
  - ✓ Credentials
  - ✓ System data
  - ✓ State data
  - ✓ Cryptographic data
  - ✓ Digital documents
  - ✓ Personally identifiable data
  - ✓ Business data.

Information Exposure – Related Terms

- Credentials include: passwords, tokens, smart cards, digital certificates, and biometrics, such as fingerprints, hand configuration, retina, iris, and voice characteristics.
- System data could be: pathnames, configurations, logs, and Web usage.
- State data includes: operational data, such as SQL (Structured Query Language) table and column names, and server names.
- Cryptographic data is: hashes, keys, and keying material, such as cryptographic keys, initialization vectors, shared secrets, domain parameters, random seeds, salts, and nonces.

Information Exposure – Related Terms

- Personally identifiable data – could be used to distinguish people:
  - personally identifiable information (PII):
    e.g. SSN, driver's license number, and ID card number
  - personally identifiable financial information (PIFI) –
    e.g. financial account numbers with security codes/ access codes/ passwords
  - payment card information:
    e.g. cardholder name, expiration date, card verification value (CVV2 for Visa), card validation code, for MasterCard (CVC2), personal identification number (PIN) or PIN block, content of magnetic stripe.
  - protected health information:
    e.g. patient medical record or payment history.
- Business data:
  - e.g. intellectual property and trade secrets, operational and inventory data, and industry-specific data, in addition to customer and employee data.

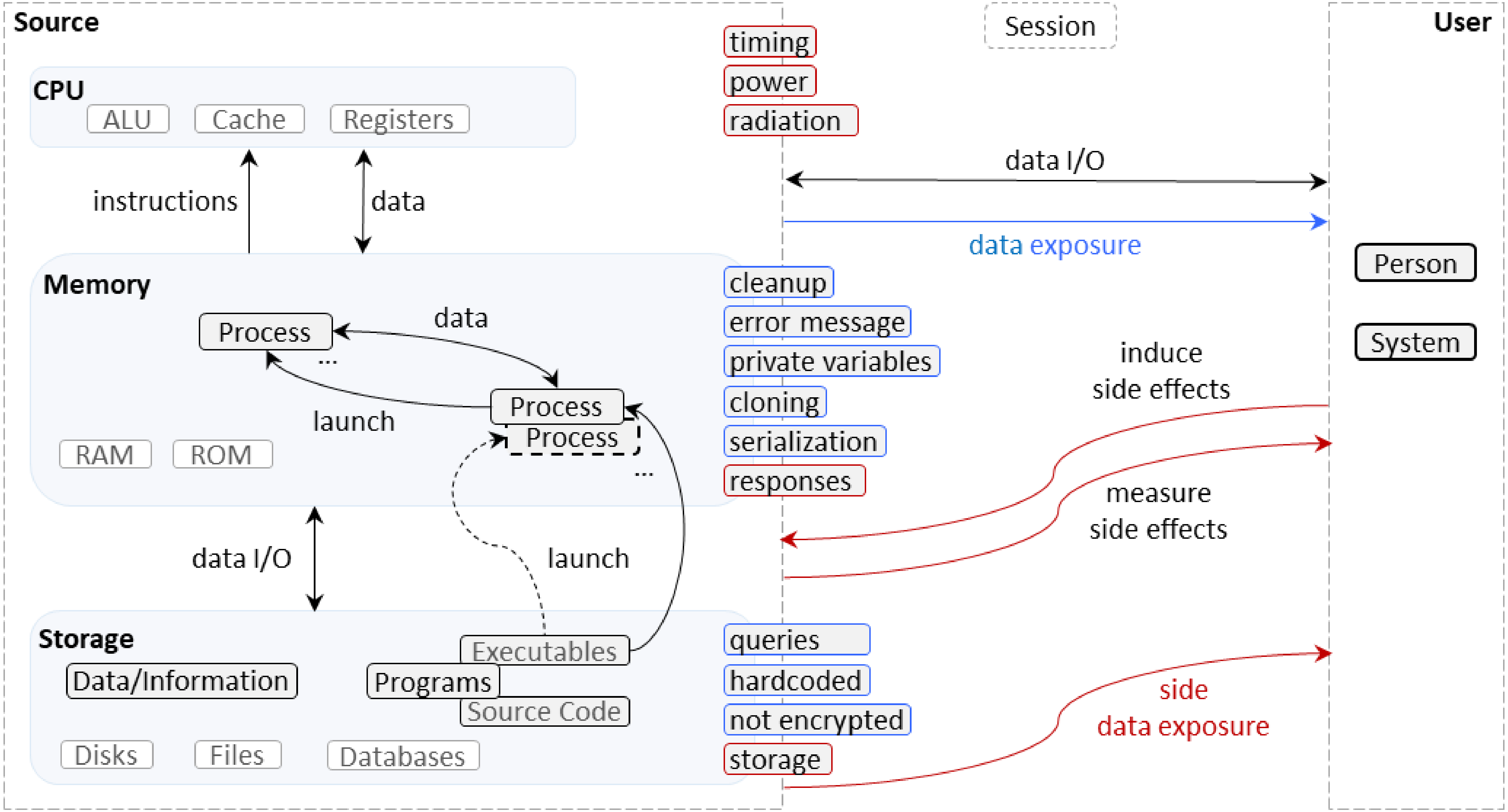
BF: Information Exposure Model

To understand information exposure,

- We developed a general descriptive model that
- Shows through what channels software could expose information.

→ *Exposure* is to any entity that should not have that information,
not just information that is a security concern.

BF: Information Exposure Model



BF: Information Exposure Model

- Information is stored on disks in files or in databases.
- Programs (source code and executables) are also stored on a disk in files and do not require any other resources.
  - A program is comprised of functions invocations.
  - Most functions process input data into output information or data.
  - A process is a program in execution and holds resources such as CPU, memory, storage, and I/O.
  - A program can involve more than one process.
  - A session is a temporary, interactive information interchange (I/O) between two or more devices, or between a computer and a user (e.g. login session).

BF: Information Exposure Model

- Information exposure may happen when:
  - either unintended information is carried, or
  - an unintended recipient gets the information.
- The exposure may be:
  - accidental or
  - because of intentional attacker actions.
- Exposure is through:
  - Legitimate Channels
    - incl. Diagnostic Channels
  - Side Channels
    - incl. Covert Channels

BF: IEX Model – Legitimate Channels

Information could leak through *legitimate channels* during normal use of software via:

- Information display
- Queries → e.g. query strings in SQL queries or GET requests
- Hardcoded information → e.g. passwords or cryptographic keys
- Class cloning
- Serializable classes
- Removing previously used information
- Buffer claeanup dead store removal
- Use of `realloc()`
- Session-ID length
- Sessions state boundaries
- Caching
- Session cleanup.

BF: IEX Model – Diagnostic Channels

- A *diagnostic channel* (error channel) is a legitimate channel that helps users and developers diagnose, find, and correct input or code errors.
- Information may be leaked via:
  - error messages
  - exception handling messages
  - other responses to erroneous inputs or erroneous data processing.

- If an attacker forces an internal fault,
  it may divulge sensitive information,
  including details on the software implementation logic.

BF: IEX Model – Side Channels

- A *side channel* is not intended to transmit information; however, it does transmit information.
- Information may be revealed or deduced due to discrepancies or behavioral inconsistencies:
  - conveying different responses (e.g. depending on if an operation is successful or not)
  - taking different time (e.g. CPU timing)
  - consuming different power
  - using different storage,
  - emitting different electromagnetic radiation.

- Behavioral inconsistency could be:
  - internal
  - external.

BF: IEX Model – Covert Channels

- A *covert channel* is a side channel that is created deliberately as a hidden communication channel.

→ A covert channel, for example, can be created by abusing TCP/IP.

- Unfortunately, a covert channel may be created by optimization techniques, such as:
  - compiler optimizations
  - speculative executions.

Examples of side/covert information exposure attacks are:

✓ Meltdown
✓ Spectre
✓ and the inference attacks.

BF IEX Model – More on Side Channels

- In a side channel, it is common for an attacker to control both:
  - the part that induces the side effect and
  - the part that measures it.
- In other cases, there could be two collaborating attackers:
  - an unauthorized user controlling the part that induces the side effect and
  - a third party controlling the part that measures it.
- There could be also:
  - only a passive attacker, who observes an existing (not induced) behavioral inconsistency.

→ Usually, statistical analysis of the measurements is involved.

BF: Information Exposure (IEX)

- Our Definition:
  Information is leaked through legitimate or side channels.

  Note that leakage to an entity that should not have information is included, not just leakage that is a security concern.

  IEX is related to: BOF, INJ, CIF, ENC, VRF, KMN, TRN, PRN.

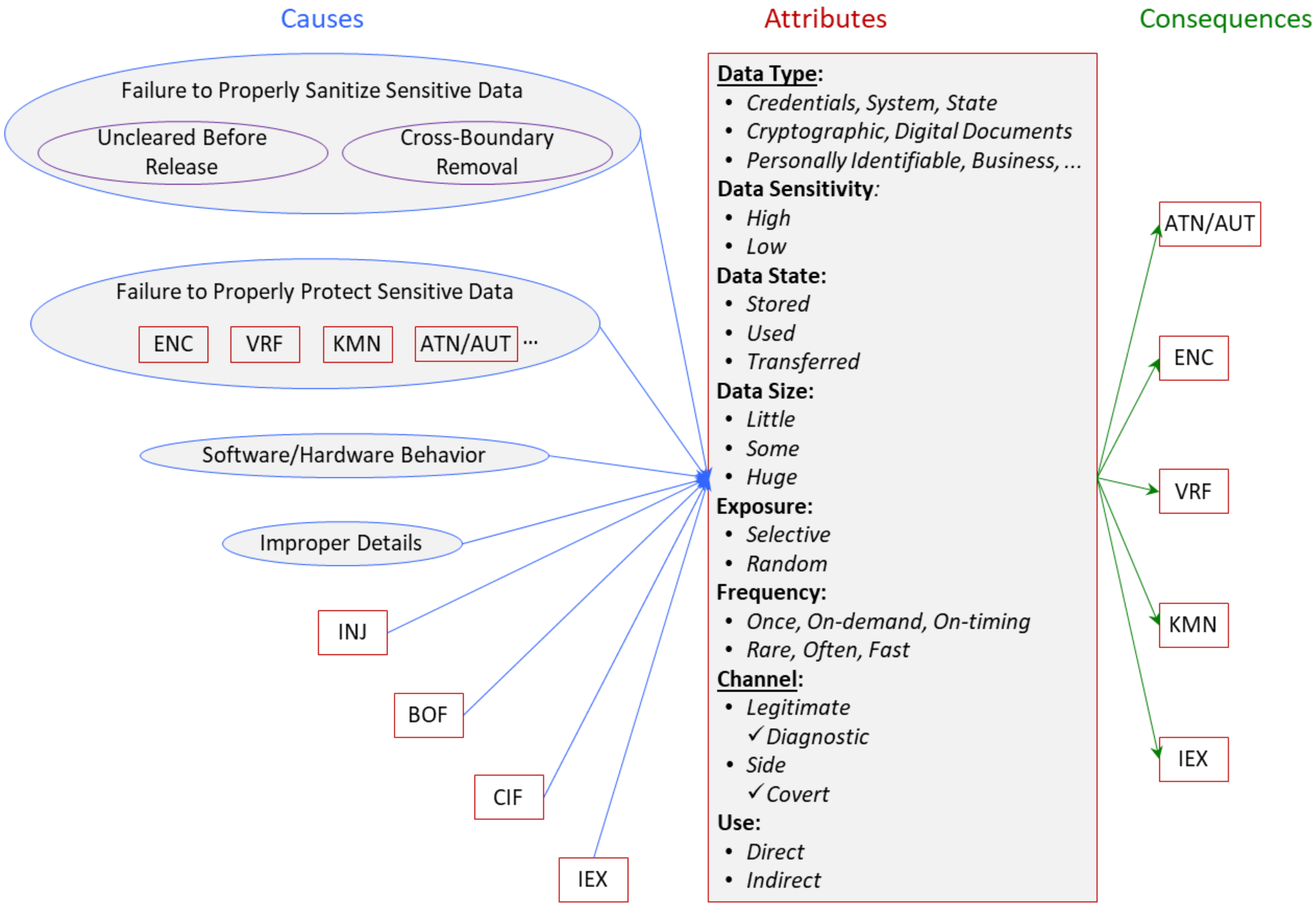
Related CWEs and SFPs:

- CWEs related to IEX are: 8, 11, 13, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 226, 244, 260, 359, 377, 385, 402, 403, 433, 488, 492, 495, 497, 498, 499, 524, 514, 515, 525, 527, 528, 529, 530, 532, 535, 536, 537, 538, 539, 540, 541,546, 548, 550, 552, 555, 598, 612, 615, 642, 651, 668.

  There are many related CWEs, because information exposure can be the consequence of many weaknesses.

- The only related SFP cluster is SFP Primary Cluster: Information Leak.

IEX: Causes, Attributes, and Consequences



Causes

Failure to Properly Sanitize Sensitive Data

- Uncleared Before Release
- Cross-Boundary Removal

Failure to Properly Protect Sensitive Data

- ENC
- VRF
- KMN
- ATN/AUT …

Software/Hardware Behavior

Improper Details

INJ

BOF

CIF

IEX

Attributes

Data Type:
- *Credentials, System, State*
- *Cryptographic, Digital Documents*
- *Personally Identifiable, Business, …*

Data Sensitivity:
- *High*
- *Low*

Data State:
- *Stored*
- *Used*
- *Transferred*

Data Size:
- *Little*
- *Some*
- *Huge*

Exposure:
- *Selective*
- *Random*

Frequency:
- *Once, On-demand, On-timing*
- *Rare, Often, Fast*

Channel:
- *Legitimate*
  - ✓ *Diagnostic*
- *Side*
  - ✓ *Covert*

Use:
- *Direct*
- *Indirect*

Consequences

ATN/AUT

ENC

VRF

KMN

IEX

IEX: Attributes

- **Data Type** – Credentials, System Data, State Data, Cryptographic Data,
  Digital Documents, Personally Identifiable Data, Business Data, etc.

- **Data Sensitivity** – High, Low. This indicates the sensitivity level of leaked data/information.
  - Highly sensitive information that is properly encrypted,
    or information that is non-sensitive
    would not result in harm if exposed.
  - Non-sensitive information could be:
    - public records
    - phone books, or
    - online directories.

IEX: Attributes

- **Data State** – Stored, Used, Transferred. This reflects if the data is at rest, in use, or in transit.
  - Data can be at rest in:
    - files (e.g. initialization, include, temporary, configuration, log server, debug, cleanup, email attachment, login buffer, executable, backup, core dump, access control list, private data index)
    - directories (e.g. Web root, FTP root, CVS repository), or
    - on discs.
  - Data can be in use by functions/programs via:
    - source code (incl. comments)
    - threads
    - registries,
    - cookies
    - GUI
    - environmental variables.
  - Data can be also in transit:
    - between processes or
    - over a network.

IEX: Attributes

- **Data Size** – Little, Some, Huge.
  This indicates how much data/information is leaked.
  - These distinctions are important in some cases:
    → For example, Heartbleed might not have been a severe problem if it just exfiltrated a little data. The fact that it may exfiltrate a huge amount of data greatly increases the chance that very important information will be leaked.
- **Exposure** – Selective, Random.
  This reflects if an attacker can choose what information to expose or where.
  - Selective means the attacker can choose where and what to read.
  - Random is like going through the trash (e.g. Heartbleed).
- **Frequency** – Once, On-demand, On-timing, Rare, Often, Fast.
  This indicates how often the exposure can/does occur.
  - On-timing means depending on timing (e.g. in a race condition).

Note: `Frequency * Size = Rate.`

IEX: Attributes

- **Channel** – Legitimate, Diagnostic, Side, Covert.
  This indicates the medium by which information was leaked.

- **Use** – Direct, Indirect**.**
  - Direct means leaked data/information is valuable on its own.
  - Indirect means it is only useful for launching other attacks.

→ IEX is a high level class, so sites do not apply.

IEX: Causes and Consequences

In the graph of causes:

- *Uncleared Before Release* means information going from one control sphere back to the general pool.
- *Cross-Boundary Removal* means information going from one control sphere to another control sphere.
  (A control sphere is a set of resources and behaviors that are accessible to a single actor or a group of actors that all share the same security restrictions.)
- *Protect Sensitive Data* also covers preparing sensitive data.
- *Software/Hardware Behavior* covers algorithms and execution.
  Observable behavior (time, power, cache lines) depends on the data.
- *Improper Details* include: passwords, paths, SQL query structure/logic, etc. in error/exception, etc. messages.

IEX: Causes and Consequences

- ENC includes:
  - failure to encrypt (cleartext storage, recoverable format storage, cleartext transmission)
  - failure to properly encrypt (inadequate encryption strength, use of risky/broken cryptographic algorithm, missing required cryptographic step, use of hard-coded cryptographic key).
- ATN/AUT includes:
  - improper authentication
  - credentials compromise
  - account access.
- INJ includes:
  - adding commands and
  - masking legitimate commands or information.
- CIF includes limiting the number of failed log in attempts
  (if there is no limit, account names or passwords may be discovered by brute force attacks).

Note: One IEX fault may lead to another IEX.

IEX: Causes and Consequences

Note: One IEX fault may lead to another IEX.

→ e.g.,

IEX of all client credit cards may be caused by
earlier IEX of the password for a privileged account.

IEX: Example 1 – CVE-2007-5172

CVE-2007-5172:

“Quicksilver Forums before 1.4.1 allows remote attackers to obtain sensitive information by causing unspecified connection errors, which reveals the database password in the resulting error message.“ [1]

[1] The MITRE Corporation, CVE-2001-1141, http://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2007-5172.

IEX: Example 1 – CVE-2007-5172

CVE-2007-5172 description using IEX taxonomy:

IEX 1 of password leads to ATN leads to IEX 2.

IEX 1

Cause: Improper Details (error message displays password)

Attributes:

- Data Type: Credentials (password)
- Data Sensitivity: High
- Data State: Stored
- Data Size: Little
- Exposure: Selective
- Frequency: On-Demand
- Channel: Diagnostic (connection error message)
- Use: Indirect

Consequences: ATN.

ATN (to be described once the ATN class is developed).

IEX 2

Cause: Failure to Properly Protect Sensitive Data (password)

Attributes:

- Data Type: Any (user data)
- Data Sensitivity: Low/High
- Data State: Stored
- Data Size: Huge
- Exposure: Selective
- Frequency: On-Demand
- Channel: Legitimate
- Use: Direct (valuable on its own)

Consequences: Any IEX consequence.

IEX: Example 2 – CVE-2004-0243

CVE-2004-0243:

“AIX 4.3.3 through AIX 5.1, when direct remote login is disabled, displays a different message if the password is correct, which allows remote attackers to guess the password via brute force methods.“ [1]

[1] The MITRE Corporation, CVE-2001-1141, http://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-CVE-2004-0243.

IEX: Example 2 – CVE-2004-0243

CVE-2004-0243 description using IEX taxonomy:

IEX 1 of password leads to ATN leads to IEX 2.

IEX 1

Cause: Program Behavior
(different responses for correct vs incorrect password)

Attributes:

- Data Type: Credentials (password)
- Data Sensitivity: High
- Data State: Used
- Data Size: Little
- Exposure: Selective
- Frequency: On-Demand
- Channel: Side (response inconsistency – message replies allow brute force password guessing)
- Use: Indirect

Consequences: ATN.

ATN (to be described once the ATN class is developed).

IEX 2

Cause: Failure to Properly Protect Sensitive Data (password)

Attributes:

- Data Type: Any (user data)
- Data Sensitivity: Low/High
- Data State: Stored
- Data Size: Huge
- Exposure: Selective
- Frequency: On-Demand
- Channel: Legitimate
- Use: Direct (valuable on its own)

Consequences: Any IEX consequence.

IEX: Example 3 – CVE-2017-5753 and CVE-2017-5715 (Spectre)

CVE-2017-5753:

“Systems with microprocessors utilizing speculative execution and branch prediction may allow unauthorized disclosure of information to an attacker with local user access via a side-channel analysis.“ [1]

CVE-2017-5715:

“Systems with microprocessors utilizing speculative execution and indirect branch prediction may allow unauthorized disclosure of information to an attacker with local user access via a side-channel analysis.“ [2]

[1] The MITRE Corporation, CVE-2001-1141, http://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-CVE-2017-5753.

[2] The MITRE Corporation, CVE-2001-1141, http://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-CVE-2017-5715.

IEX: Example 3 – CVE-2017-5753 and CVE-2017-5715 (Spectre)

CVE-2017-5753 and CVE-2017-5715 description using IEX taxonomy:

Cause: Hardware Behavior (CPU speculative execution)

Attributes:

Data Type: Any (user’s data)
Data Sensitivity: High
Data State: Stored
Data Size: Huge
Exposure: Selective
Frequency: On-Demand
Channel: Side (cache-based timing)
Use: Any

Consequences: Any IEX consequence.

IEX: Example 4 – CVE-2017-5754 (Meltdown)

CVE-2017-5754:

“Systems with microprocessors utilizing speculative execution and indirect branch prediction may allow unauthorized disclosure of information to an attacker with local user access via a side-channel analysis of the data cache.“ [1]

[1] The MITRE Corporation, CVE-2017-5754, http://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-CVE-2017-5754.

IEX: Example 4 – CVE-2017-5754 (Meltdown)

CVE-2017-5754 description using IEX taxonomy:

Cause: Hardware Behavior (CPU out-of-order execution)
Attributes:

- Data Type: Any (passwords in password manager or browser, photos, emails, even business-critical documents)
- Data Sensitivity: High
- Data State: Stored (in kernel-memory registries of other processes or virtual machines in the cloud)
- Data Size: Huge
- Exposure: Selective
- Frequency: On-Demand
- Channel: Covert (cache-based timing)
- Use: Any

Consequences: Any IEX consequence.

IEX – Conclusion

- We presented:
  - a general model of Information Exposure and
  - defined a new BF class, (IEX), including:
    - a rigorous definition
    - static attributes of the class, and
    - their related dynamic properties, such as proximate causes, consequences and sites.
- IEX is a very pervasive class, as:
  - many vulnerabilities lead to IEX, and
  - IEX may further lead to other faults.
- We analyzed particular vulnerabilities related to IEX and provide clear IEX descriptions.
- We showed that the BF-structured descriptions are quite concise,
  while still far clearer than unstructured explanations that we have found.

- We had to refine many of the previously defined BF classes.

IEX – Conclusion

- IEX joins other rigorously-defined BF classes, such as:

1. Randomness Cluster (RND)
   - Pseudo-Random Number Bugs (PRN)
   - True-Random Number Bugs (TRN)
2. Cryptography Cluster (CRY)
   - Key Management Bugs (KMN)
   - Encryption Bugs (ENC)
   - Verification Bugs (VRF)
3. Access Control Cluster (ACC)
   - Authentication Bugs (ATN)
   - Authorization Bugs (AUT)

IEX – Conclusion

- Lessons Learned and Future Work
  - At first, we had difficulties distinguishing strictly security-related leaks from other information exposure. However, we realized that “information exposure” is to any entity that should not have that information, not just leaks that are a security concern.
  - An IEX fault could actually be a cause of another IEX or the consequence of a preceding IEX. We learned that as we add more classes, such as IEX, we start chaining classes, consequences and causes in BF.
  - Work on explaining more information exposure faults using IEX and chains of BF classes will help us determine where our BF taxonomy needs refinement.
  - We will have to expand and refine many of the previously created BF descriptions of faults.

→ Our goal is for BF to become software developers’ and testers’ “*Best Friend*.”

Randomness Cluster (RND):

- Pseudo-Random Number Bugs (PRN)
- True-Random Number Bugs (TRN)

RND: Randomness Classes in BF

Random number generators may have weaknesses (bugs) and applications using such may become vulnerable to attacks.

Formalization of randomness bugs would help researchers and practitioners identify them and avoid security failures.

We have developed a general descriptive model of randomness and have defined two Randomness BF classes:

- True-Random Number Bugs (TRN)
- Pseudo-Random Number Bugs (PRN)

Randomness

- Randomness has application in many fields, including:
  - ✓ Cryptography
  - ✓ Simulation
  - ✓ Statistics
  - ✓ Politics
  - ✓ Science
  - ✓ Gaming.
- Any specific use has its own requirements for randomness
  - e.g., random bit generation for cryptography or security purposes has stronger requirements than generation for other purposes.

    For cryptography or security purposes, NIST recommends use of cryptographically secure PRBGs: subject to the requirements in NIST SP 800-90A, NIST SP 800-90B and NIST SP 800-90C.
- Satisfying the requirements for a particular use can be surprisingly difficult.
- Weaknesses (bugs) in RNGs may lead to:
  - ✓ wrong results from the algorithms that use the generated numbers or
  - ✓ allow attackers to recover secret values, such as passwords and cryptographic keys.

Randomness Generation

We separate randomness generation in two distinct processes:

- True-random number generation:
  - ➢ nondeterministic (full entropy)
  - ➢ uses entropy sources.
- Pseudo-random number generation:
  - ➢ deterministic
  - ➢ uses true-random numbers as seeds.

– PRBGs are used to extend the true-random seeds, produced by a TRBG – if the seed has length `n`, the output of the PRBG can have length `m`, where `m > n`.

→ However, a PRBG cannot increase the entropy of its seed.

– It is possible though for a PRBG to use non-random seeds (e.g., for generating random numbers for simulation or game algorithms).

Randomness Attacks

Examples of attacks related to randomness generation are:

- Direct RSA common factor attack
- Cryptanalytic attack
- Input based attack
- State compromise attack.

BF: RND Model

Our BF randomness bugs (RND) model:

- Shows in which software components of TRNG and PRNG bugs can occur

- Is descriptive and not prescriptive

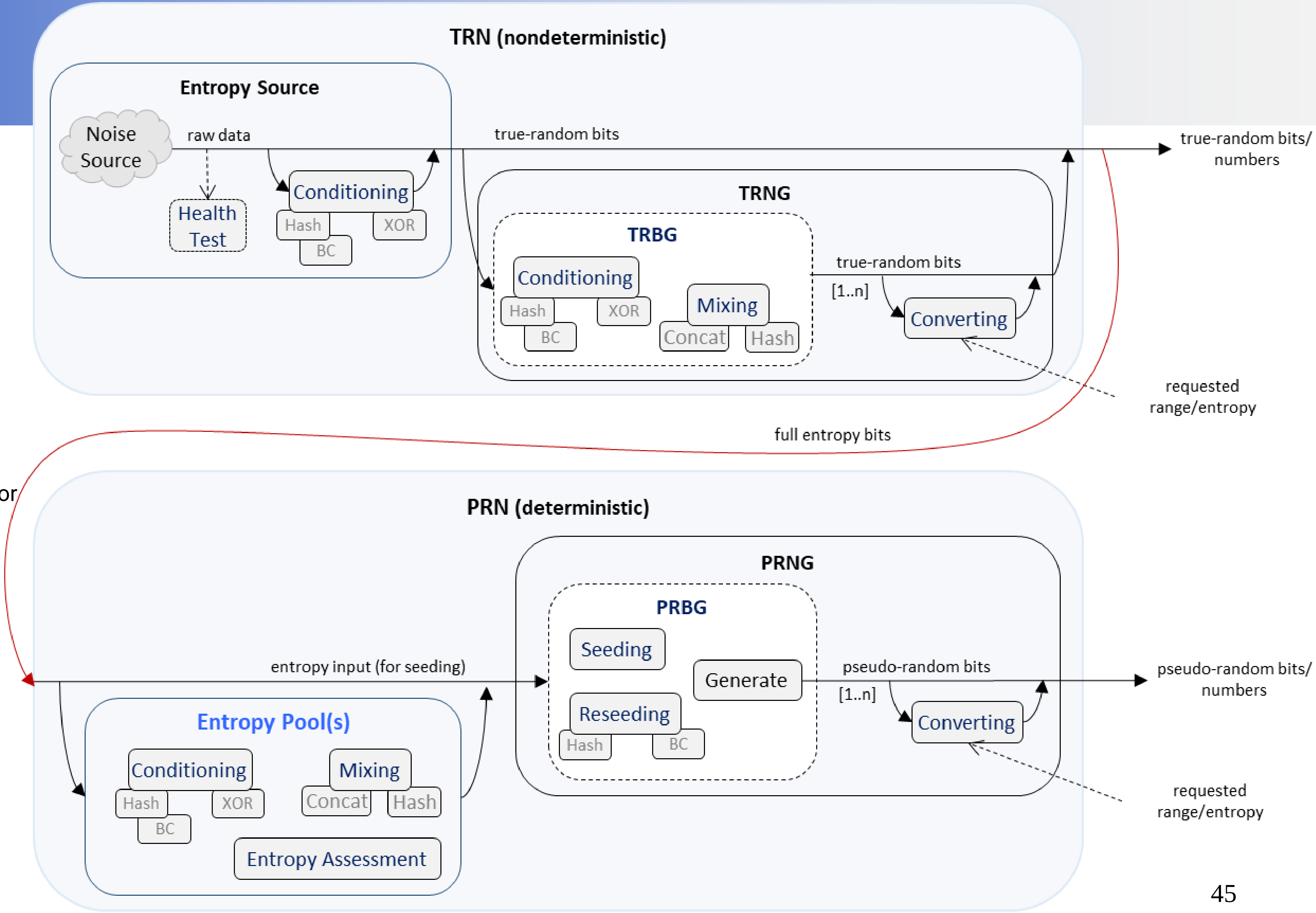
  → Should not be used as a model for construction of RBGs
  (🔒 NIST SP 800-90C specifies construction of RBGs using the mechanisms and entropy sources described in SP 800-90A and SP 800-90B, respectively.)

BF: RND Model

TRN – True-Random Number Bugs
PRN – Pseudo-Random Number Bugs

TRBG: True-Random Bit Generator
TRNG: – True-Random Number Generator

PRBG – Pseudo-Random Bit Generator
PRNG – Pseudo-Random Number Generator
BC – Block Cipher



BF: RND Model

Two distinct processes:

- TRN covers bugs related to:
  - entropy sources
  - TRBGs
  - TRNGs.
- PRN covers bugs related to:
  - entropy pools
  - PRBGs
  - PRNGs.

Although, output from the former process may be used as input to the latter (see the red arrow), they are distinct as bugs related to each have different causes, attributes, and consequences.

- The random bits are optionally converted in a pseudo-random number based on the range that applications provide as an argument.
- If live entropy source is used, the PRBG is said to support prediction resistance.
- PRNGs are algorithmic and can have bugs. Most PRNGs are not cryptographically secure.

BF: True-Random Number Bugs (TRN)

- We define True-Random Number Bugs (TRN) as:

  The software generated output does not satisfy all use-specific true-randomness requirements.

  Note that the output sequence is of random bits, where values are obtained from one or more sources of entropy.

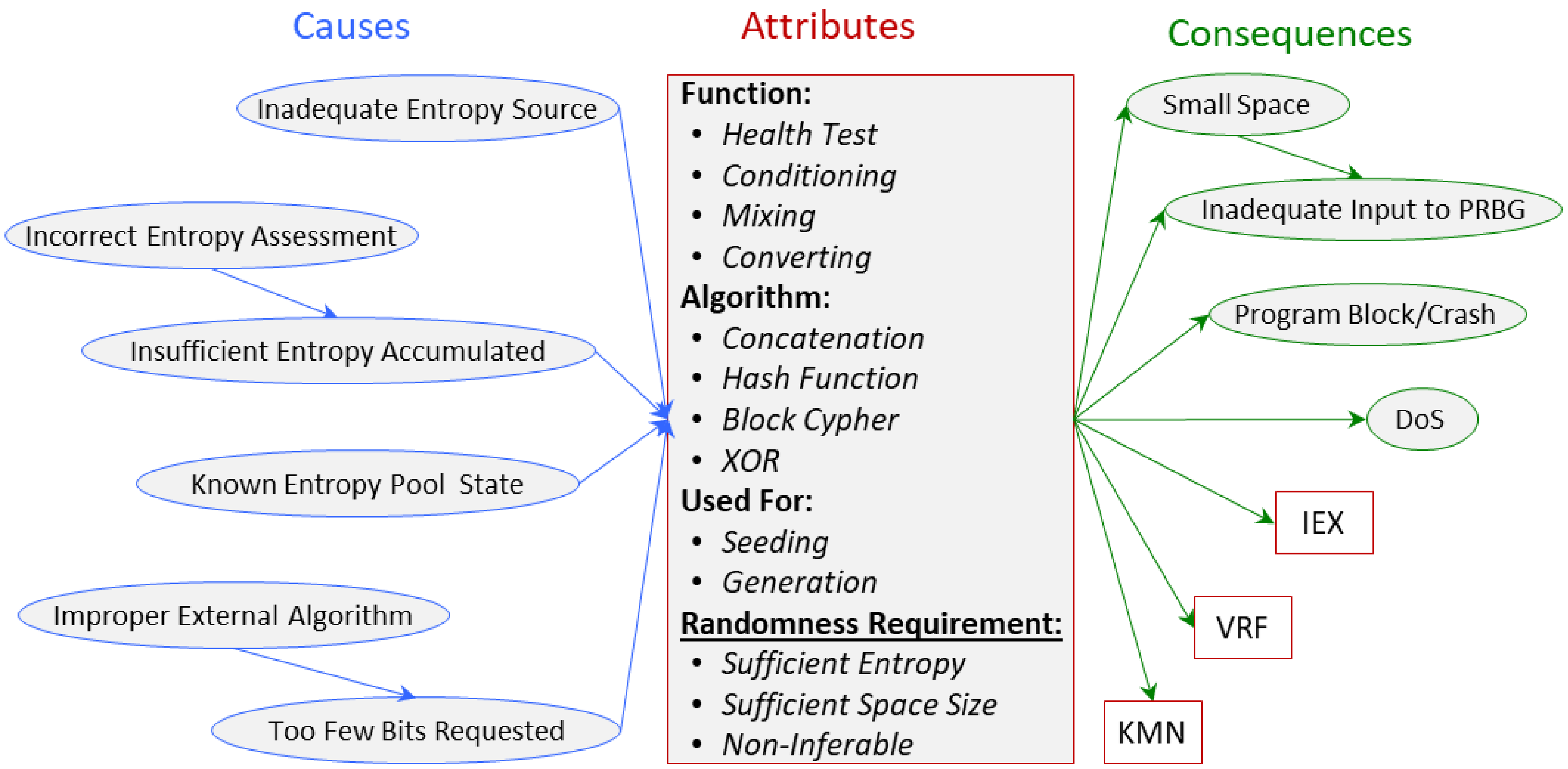
  TRN is related to: PRN, ENC, VRF, KMN, IEX.

Related CWEs, SFPs:

✓ SFP Primary Cluster: Predictability, which is CWE-905 with members CWE 330 to 344.

✓ Among them, TRN CWEs: 330, 331, 332, 333, 334, 337, 339, 340, 341, 342, 343.

TRN: Causes, Attributes, and Consequences



Function:

- *Health Test*
- *Conditioning*
- *Mixing*
- *Converting*

Algorithm:

- *Concatenation*
- *Hash Function*
- *Block Cypher*
- *XOR*

Used For:

- *Seeding*
- *Generation*

Randomness Requirement:

- *Sufficient Entropy*
- *Sufficient Space Size*
- *Non-Inferable*

TRN: Attributes

- **Function:**
  - ✓ Health Test
  - ✓ Conditioning
  - ✓ Mixing
  - ✓ Output
  - ✓ Converting.
- **Algorithm:**
  - ✓ Hash Function
  - ✓ Block Cipher
  - ✓ XOR, etc.
- **Used For** (what the output sequence is used for)**:**
  - ✓ Seeding

    – as a seed for a PRNG
  - ✓ Generation

    – of passwords or cryptographic keying material (keys, nonces).

TRN: Attributes

- **Randomness Requirement** (this is the failed requirement):
  - ✓ Sufficient Entropy
  - ✓ Sufficient Space Size
  - ✓ Non-Inferable.

- ➢ The notion of entropy used here is min-entropy, which is a measure of how difficult it is to guess the most likely entropy source output. (Let `X` be a random variable such that the set of its possible values is finite. Let `P` be the set of probabilities of `X` having those values. Then the min-entropy of `X` is $-\log_2 \max(P)$.)
- ➢ Space size – the number of elements of the space of possible outputs. (If the number of different outputs is not sufficiently large, there is a vulnerability to a brute force attack.)
- ➢ Non-inferable – one cannot recover from known (guessed) information anything about the TRBG output. (TRBGs used for cryptography/security must satisfy the Non-Inferable randomness requirement.)

→ TRN is a high level class, so sites do not apply.

TRN: Causes & Consequences

- In the graph of causes:
  - Incorrect Entropy Assessment may result in TRN output having insufficient entropy.

    For example, a certain number of bits with full entropy may be needed, and because
    of Incorrect Entropy Assessment, the output may have insufficient entropy.

- In the graph of consequences:
  - Inadequate Input to PRNG could be:
    repeating, weak, insufficiently random, predictable, small space seed or other input.
  - A program may crash or block if it runs out of random number".
  - Denial of Service (DoS) can be a direct consequence.
  - KMN as in finding private keys from public keys using a common factor attack.
  - VRF as of using a predictable random number with a signature algorithm, such as DSA.
  - IEX could be of the exact value of a generated random number or a small range of values from which the generated random number is easy to figure out.

TRN: Example – CVE-2008-0141

CVE-2008-0141:
“actions.php in WebPortal CMS 0.6-beta generates predictable passwords containing only the time of day, which makes it easier for remote attackers to obtain access to any account via a lostpass action.“ [1]

[1] The MITRE Corporation, CVE Common Vulnerabilities and Exposures, CVE-2008-0141.

TRN: Example – CVE-2008-0141

CVE-2008-0141 → using TRN taxonomy

Cause: Inadequate Entropy Sources (current date/time and user name)
Attributes:
- Function: Mixing
- Algorithm: Concatenation
- Used for: Generation (of password)
- Randomness Requirement: Non-Inferable (time known from password reset time, name known from user register)

Consequences: IEX (of password), leading to ATN (Authentication Fault)

BF: Pseudo-Random Number Bugs (PRN)

- We define Pseudo-Random Number Bugs (PRN) as: :
  The software generated output does not satisfy all use-specific pseudo-randomness requirements.

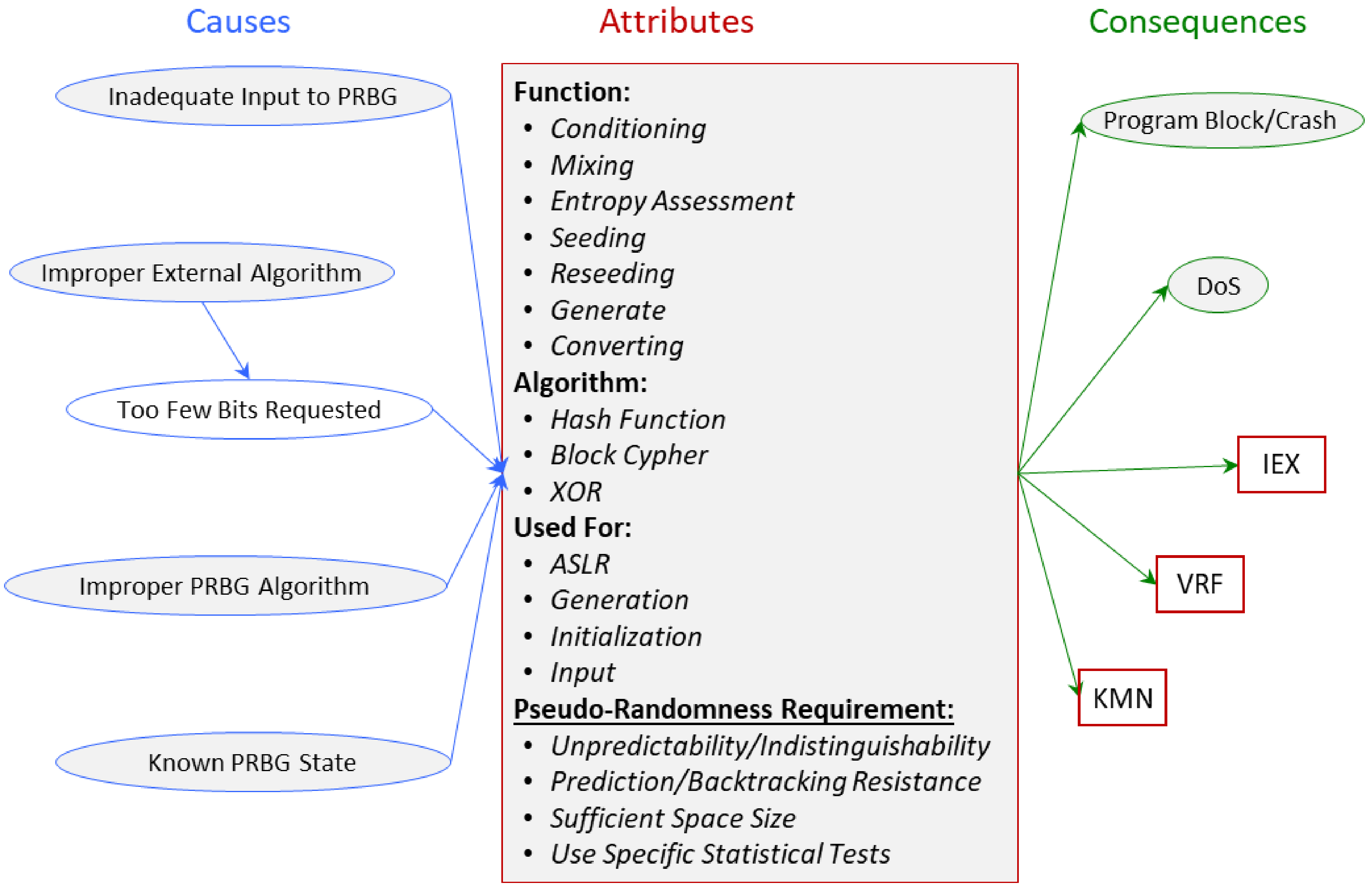
  Note that the output sequence is of random bits or numbers from a PRNG.

  PRN is related to: TRN, ENC, VRF, KMN, IEX.

Related CWEs, SFPs:

- SFP Primary Cluster: Predictability, which is CWE-905 with members CWE 330 to 344.
- Among them, PRN CWEs: 330, 331, 332, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343.

PRN: Causes, Attributes, and Consequences



Causes

- Inadequate Input to PRBG
- Improper External Algorithm
- Too Few Bits Requested
- Improper PRBG Algorithm
- Known PRBG State

Attributes

Function:

- *Conditioning*
- *Mixing*
- *Entropy Assessment*
- *Seeding*
- *Reseeding*
- *Generate*
- *Converting*

Algorithm:

- *Hash Function*
- *Block Cypher*
- *XOR*

Used For:

- *ASLR*
- *Generation*
- *Initialization*
- *Input*

Pseudo-Randomness Requirement:

- *Unpredictability/Indistinguishability*
- *Prediction/Backtracking Resistance*
- *Sufficient Space Size*
- *Use Specific Statistical Tests*

Consequences

- Program Block/Crash
- DoS
- IEX
- VRF
- KMN

PRN: Attributes

- **Function:**
  - ✓ Conditioning
  - ✓ Mixing
  - ✓ Entropy Assessment
  - ✓ Seeding
  - ✓ Reseeding
  - ✓ Generate
  - ✓ Converting.
- **Algorithm:**
  - ✓ Concatenation
    – the usual mixing of output from IID sources
  - ✓ Hash Function
  - ✓ Block Cipher
  - ✓ XOR.

PRN: Attributes

- **Used For** (what the output sequence is used for)**:**
  - ✓ ASLR (Address Space Layout Randomization)
  - ✓ Generation
    – of passwords or cryptographic keying material (keys, nonces)
  - ✓ Initialization
    – of cryptographic primitives (e.g., an initialization vector for cipher block chaining mode of encryption; or a salt for hashing)
  - ✓ Input to Algorithm
    – simulation, statistics, mathematics (e.g., Monte Carlo integration), or general algorithms.

PRN: Attributes

- **Pseudo-Randomness Requirement** (this is the failed requirement)**:**
  - ✓ Unpredictability/ Indistinguishability
  - ✓ Prediction/Backtracking Resistance
  - ✓ Sufficient Space Size
  - ✓ Use Specific Statistical Tests.

The pseudo-random output sequence should be statistically independent and unbiased. It should pass the use-specific statistical tests for randomness.

- ➢ Unpredictability – not possible to predict next generated output from previous output.
- ➢ Prediction Resistance –not possible to predict future output bits even if past or present state is known. (Prediction resistance is not possible without a live entropy source.)
- ➢ Backtracking Resistance – not possible to recover (backtrack) past output bits based on knowledge of the state at a given point in time.
- ➢ Indistinguishability for a PRNG – its output is computationally indistinguishable (i.e., by any probabilistic polynomial time algorithm) from a truly random sequence.
- ➢ Space size – the number of elements of the space of possible outputs.

PRN: Attributes

PRNGs used for cryptography/security must satisfy the following requirements:

- ✓ Unpredictability/Indistinguishability
- ✓ Backtracking Resistance
- ✓ Sufficient Space Size.

Prediction Resistance however is not always required – e.g. for PIV cards.

→ PRN is a high level class, so sites do not apply.

PRN: Example 1 – CVE-2001-1141

CVE-2001-1141:
“The Pseudo-Random Number Generator (PRNG) in SSLeay and OpenSSL before 0.9.6b allows attackers to use the output of small PRNG requests to determine the internal state information, which could be used by attackers to predict future pseudo-random numbers.“ [1]

[1] The MITRE Corporation, CVE-2001-1141, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-1141.

PRN: Example 1 – CVE-2001-1141

CVE-2001-1141 description using PRN taxonomy:

Cause: Improper PRNG Algorithm (C md_rand – the secret PRNG state is updated with portion, as small as one byte, of the PRNG’s previous output, which is not secret)

Attributes:

Function: Mixing (back into entropy pool)

Algorithm: Hash Function (SHA-1 – used for PRNG output and to update its internal secret state)

Used For: Generation (of cryptographic keying material – nonces, cryptographic keys)

Pseudo-Randomness Requirements: Sufficient Space Size and Unpredictability
(can be predicted from previous value through brute force)

Consequences: KMN>Generate with IEX of future keying material and ENC>IEX of sensitive data

PRN: Example 2 – CVE-2008-4107

CVE-2008-4107: “The (1) rand and (2) mt_rand functions in PHP 5.2.6 do not produce cryptographically strong random numbers, which allows attackers to leverage exposures in products that rely on these functions for security-relevant functionality, as demonstrated by the password-reset functionality in Joomla! 1.5.x and WordPress before 2.6.2, a different vulnerability than CVE-2008-2107, CVE-2008-2108, and CVE-2008-4102.” [1]

[1] The MITRE Corporation, CVE-2008-4107, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4107

PRN: Example 2 – CVE-2008-4107

CVE-2008-4107 description using PRN taxonomy:

Cause: Improper PRNG Algorithms (not cryptographically strong PHP 5 `rand` and `mt_rand`)
Attributes:
- Function: Generate (pseudo-random numbers)
- Algorithms: e.g., LCG or LFSR, Mersenne Twister
- Used For: Generation (of passwords)
- Pseudo-Randomness Requirements: Unpredictability/ Indistinguishability and Prediction Resistance

Consequence: IEX (of password), leading to ATN

PRN: Example 3 – CVE-2009-3238

CVE-2009-3238: “The get_random_int function in drivers/char/random.c in the Linux kernel before 2.6.30 produces insufficiently random numbers, which allows attackers to predict the return value, and possibly defeat protection mechanisms based on randomization, via vectors that leverage the function's tendency to "return the same value over and over again for long stretches of time.“ [1]

[1] The MITRE Corporation, CVE-2009-3238, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-3238

PRN: Example 3 – CVE-2009-3238

CVE-2009-3238 description using TRN and PRN taxonomy:

An TRN leads to a PRN.

TRN

Cause: Improper RNG Algorithm (same area is used for the hash array, allowing to repeatedly start from the same seed)

Attributes:

Functions: Mixing (of pid and jiffies), Conditioning

Algorithms: Concatenation, Hash Function (MD4)

Used For: Seeding

Randomness Requirements: Sufficient Entropy

Consequence: Inadequate Input to PRNG (repeating seed)

PRN

Cause: Inadequate Input to PRNG (repeating seed)

Attributes:

Function: Generate (pseudo-random numbers)

Used For: ASLR

Pseudo-Randomness Requirement: Unpredictability

Consequences: IEX of addresses

PRN: Example 4 – CVE-2017-15361 (ROCA – Return Of the Coppersmith Attack)

CVE-2017-15361:
“The Infineon RSA library 1.02.013 in Infineon Trusted Platform Module (TPM) firmware, such as versions before 0000000000000422 - 4.34, before 000000000000062b - 6.43, and before 0000000000008521 - 133.33, mishandles RSA key generation, which makes it easier for attackers to defeat various cryptographic protection mechanisms via targeted attacks, aka ROCA. Examples of affected technologies include BitLocker with TPM 1.2, YubiKey 4 (before 4.3.5) PGP key generation, and the Cached User Data encryption feature in Chrome OS.” [1]

[1] The MITRE Corporation, CVE-2017-15361, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-15361

PRN: Example 4 – CVE-2017-15361 (ROCA)

CVE-2017-15361 description using KMN, and PRN taxonomy:

A KMN with inner PRN**.**

KMN:

- **Cause**: Improper Algorithm Step (for generation of primes for RSA keys) leads to inner PRN
- **Attributes:**
  - Cryptographic Data: Keying Material (keys)
  - Algorithm: RSA (key generation from two primes)
  - Operation: Generate (pair of public and private keys)
- **Consequences:** Weak Public Key, which leads to IEX of Private Key.

Inner PRN:

- **Cause:** Improper External Algorithm (generation of primes for RSA keys from random numbers and a constant related to keys size) leads to Too Few Bits Requested
- **Attributes:**
  - Functions: Converting, Seeding (low entropy requested)
  - Used for: Generation (of secret prime numbers)
  - Randomness Requirement: Sufficient Space Size (e.g., one random number is only 37 bits for 512-bit RSA keys)
- **Consequence:** IEX of generated primes (which format allows keys fingerprinting, factorization with Coppersmith algorithm, and finding random numbers and primes).

Cryptography Cluster (CRY):

- Key Management Bugs (KMN)
- Encryption Bugs (ENC)
- Verification Bugs (VRF)

Cryptography

- Broad, complex, and subtle area.
- Incorporates many clearly separate cryptographic processes, such as:
  - ✓ Encryption/ Decryption
  - ✓ Verification of data or source
  - ✓ Key management.
- Each cryptographic process
  → uses particular algorithms
  (e.g. symmetric/ asymmetric encryption, MAC, digital-signature)
  → to achieve particular security service.
  (e.g. confidentiality, integrity authentication, identity authentication, origin non-repudiation)

Cryptography Bugs

There are bugs if the software does not properly:

- Transform data into unintelligible form
  - ✓ Some transformations require keys – e.g. encryption and decryption
  - ✓ While others do not require keys – e.g. secret sharing.
- Verify:
  - ✓ Authenticity – data integrity, data source identity, origin for non-repudiation, content of secret sharing
  - ✓ Correctness – for uses such as zero-knowledge proofs.
- Manage keys
- Perform other operations.

Cryptography Attacks

Examples of attacks are:

- ✓ Spoofing messages
- ✓ Brute force attack
- ✓ Replaying instructions
- ✓ Timing attack
- ✓ Chosen plaintext attack (CPA)
- ✓ Chosen ciphertext attack (CCA)
- ✓ Exploiting use of weak or insecure keys
  (e.g. factorization of public key to obtain private key).

Cryptographic Store or Transfer

We use cryptographic store or transfer to illustrate the BF Cryptography Bugs (CRY) Classes:

- Encryption Bugs (ENC)
- Verification Bugs (VRF)
- Key Management Bugs (KMN)

Note: These classes may appear in many other situations such as:

- Self-sovereign identities
- Block ciphers
- Threshold cryptography.

We focus on transfer (or store) because it is:

- ✓ Well known a
- ✓ What most people think of when “cryptography” is mentioned.

Bugs in Cryptographic Store or Transfer

We define bugs in cryptographic store or transfer as:

> The software does not properly encrypt/decrypt, verify, or manage keys for data to be securely stored or transferred.

Bugs in Cryptographic Store or Transfer

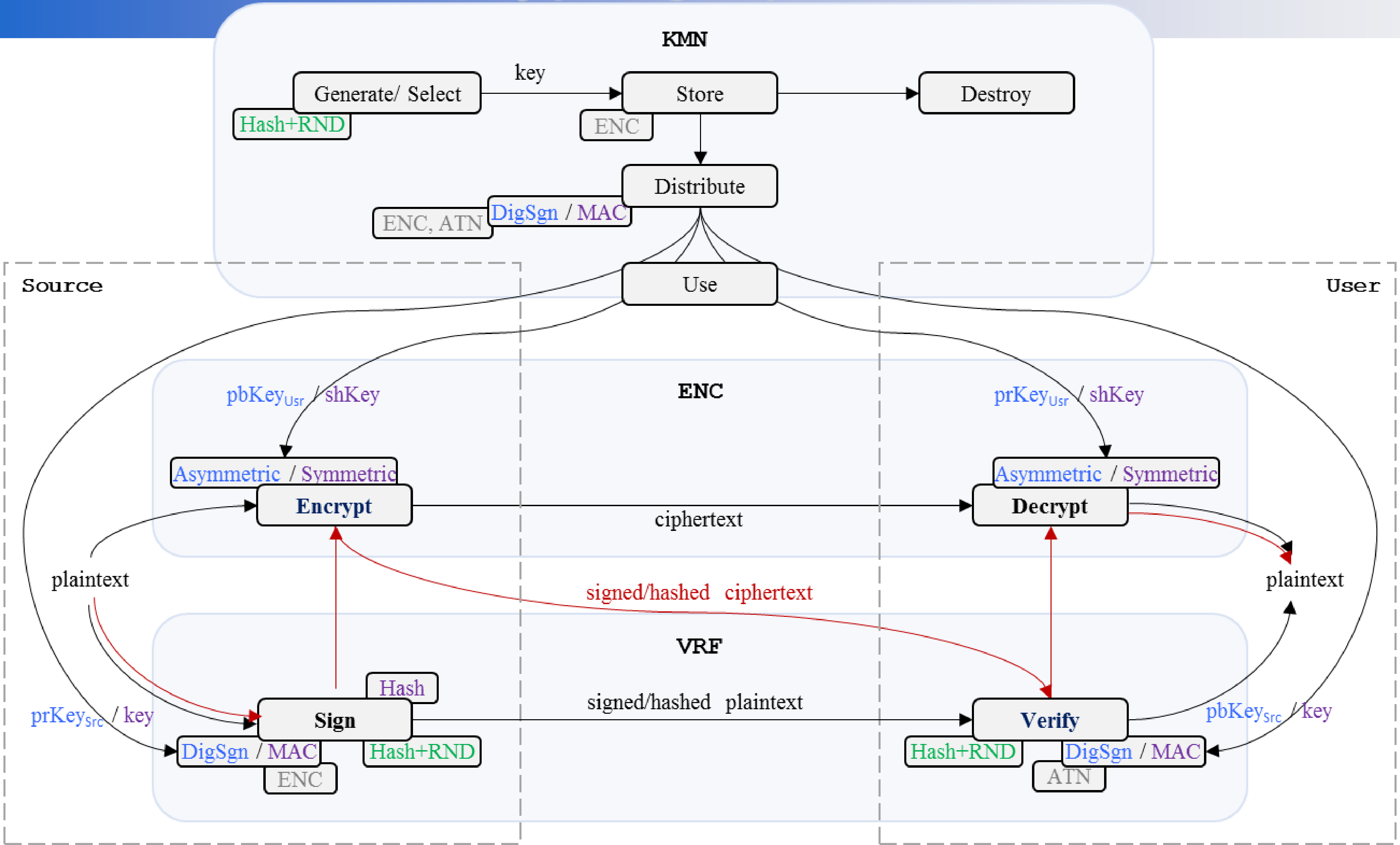
A modern, secure, flexible cryptographic storage or transfer protocol:

- Likely involves subtle interaction between following processes:
  - ✓ Encryption
  - ✓ Verification
  - ✓ Key Management
- May involve multiple stages of:
  - ✓ Agreeing on encryption algorithms
  - ✓ Establishing public and private keys
  - ✓ Creating session keys
  - ✓ Digitally signing texts for verification.

Thus, Encryption may use Key Management, which itself uses Encryption and Verification.

→ Need of a Model of these recursive interactions and
where potentially ENC, VRF, KMN, and other BF bugs could happen.

BF: Model of Cryptographic Store or Transfer



BF: CRY Model – Key Management Bugs

- KMN is a class of bugs related to key management.
- Key management comprises:
  - ✓ Key generation
  - ✓ Key selection
  - ✓ Key storage
  - ✓ Key retrieval and distribution
  - ✓ Determining and signaling when keys should be abandoned or replaced.

A particular protocol may use any or all of these operations.

BF: CRY Model – Key Management Bugs

- Key Management could be by:
  - ✓ a third party certificate authority (CA) – distributes public keys in signed certificates
  - ✓ the source
  - ✓ the user

Thus the Key Management area intersects the Source and User areas.

Key Management often uses a recursive round of encryption and decryption, and verification to establish a shared secret key or session key before the actual plaintext is handled.

BF: CRY Model – Encryption Bugs

- ENC is a class of bugs related to encryption.
- Encryption comprises:
  - ✓ Encryption by the source
  - ✓ Decryption by the user.

- Encryption/ decryption algorithms may be:
  - ✓ Symmetric – uses same key for both
  - ✓ Asymmetric – uses pairs of keys: one to encrypt, other to decrypt.

Public key cryptosystems are asymmetric.

The ciphertext may be sent directly to the user, and verification accompanies it separately.
The red line is a case where plaintext is signed or hashed and then encrypted.

BF: CRY Model – Verification Bugs

- VRF is a class of bugs related to verification.
- Verification:
  - Takes a key and either the plaintext or the ciphertext
    signs or hashes it then passes the result to the user.
  - User uses the same key or the other member of the key pair to verify source.

BF: CRY Model – Keys Usage

- Symmetric encryption – one secretly shared key (shKey) is used:
  - ✓ Source encrypts with shKey
  - ✓ User decrypts with shKey, too.

- Asymmetric encryption – pairs of mathematically related keys are used, source pair: ($pbKey_{Src}$, $prKey_{Src}$), user pair: ($pbKey_{Usr}$ and $prKey_{Usr}$):
  - ✓ Source:
    - ○ encrypts with $pbKey_{Usr}$
    - ○ signs with $prKey_{Src}$
  - ✓ User:
    - ○ decrypts with $prKey_{Usr}$
    - ○ verifies with $pbKey_{Src}$.

BF: Encryption Bugs (ENC)

- We define Encryption Bugs (ENC) as:

  The software does not properly transform sensitive data (plaintext) into unintelligible form (ciphertext) using cryptographic algorithm and key(s).
- We define also the Decryption Bugs as:

  The software does not properly transform ciphertext into plaintext using cryptographic algorithm and key(s).

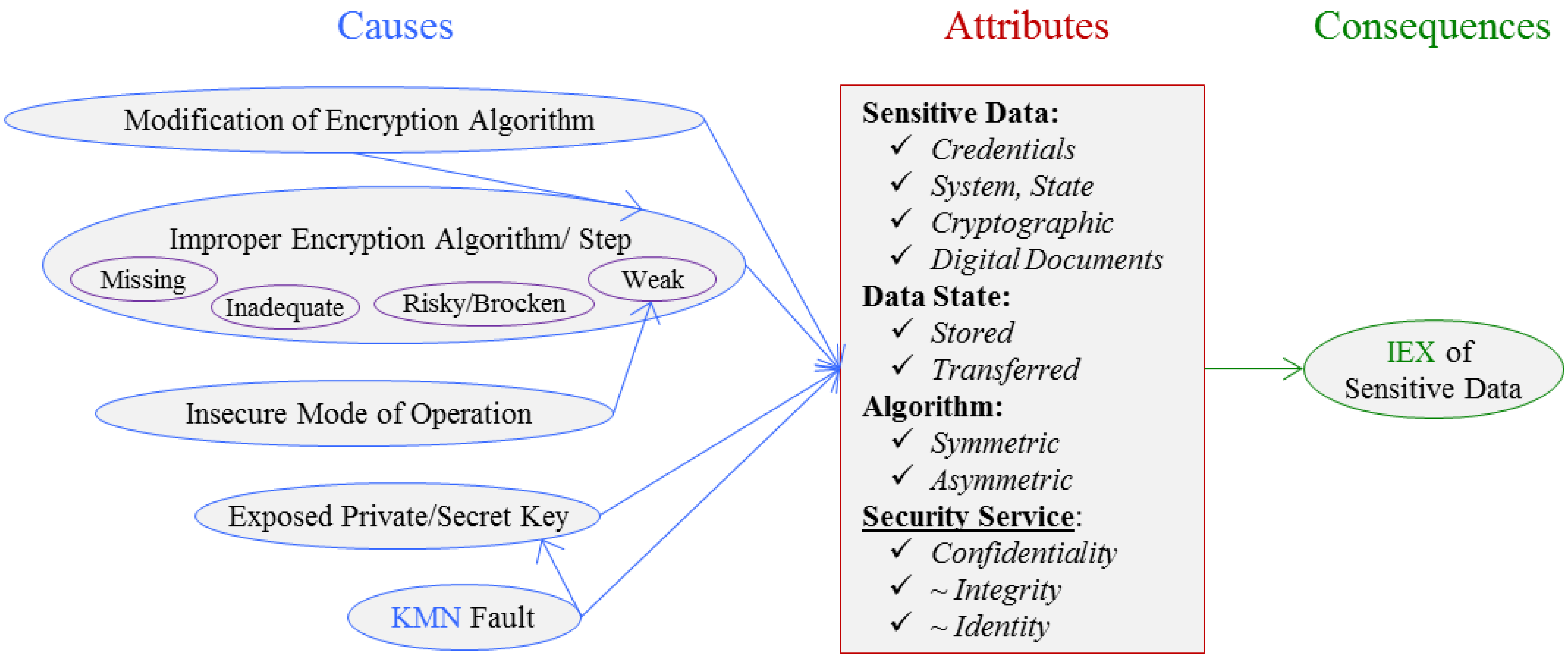
Note that "transform" is for confidentiality.

ENC is related to KMN, TRN, PRN, and IEX.

Related CWEs, SFPs and ST:

- ✓ CWEs: 256, 257, 261, 311-318, 325, 326, 327, 329, 780.
- ✓ SFP clusters: SPF 17.1 Broken Cryptography and SFP 17.2 Weak Cryptography under Primary Cluster: Cryptography.

ENC: Causes, Attributes, and Consequences



Causes

Modification of Encryption Algorithm

Improper Encryption Algorithm/ Step

Missing

Inadequate

Risky/Brocken

Weak

Insecure Mode of Operation

Exposed Private/Secret Key

KMN Fault

Attributes

Sensitive Data:
- ✓ *Credentials*
- ✓ *System, State*
- ✓ *Cryptographic*
- ✓ *Digital Documents*

Data State:
- ✓ *Stored*
- ✓ *Transferred*

Algorithm:
- ✓ *Symmetric*
- ✓ *Asymmetric*

Security Service:
- ✓ *Confidentiality*
- ✓ *~ Integrity*
- ✓ *~ Identity*

Consequences

IEX of Sensitive Data

ENC: Attributes

- **Sensitive Data** – This is secret (confidential) data.
  - ✓ Credentials: Password, Token, Smart Card, Digital Certificate, Biometrics (fingerprint, hand configuration, retina, iris, voice.)
  - ✓ System Data: Configurations, Logs, Web usage, etc.
  - ✓ State Data
  - ✓ Cryptographic Data: hashes, keys, and other keying material
  - ✓ Digital Documents.
- **Data State –** This reflects if data is in rest or use, or if data is in transit.
  - ✓ Stored: data in rest or use from files (e.g. ini, temp, configuration, log server, debug, cleanup, email attachment, login buffer, executable, backup, core dump, access control list, private data index), directories (Web root, FTP root, CVS repository), registry, cookies, source code & comments, GUI, environmental variables.
  - ✓ Transferred: data in transit between processes or over a network.

ENC: Attributes

- **Algorithm** –the key encryption scheme used to securely store/transfer sensitive data.
  - ✓ Symmetric (secret) key algorithms (e.g. Serpent, Blowfish)
    use one shared key.
  - ✓ Asymmetric (public) key algorithms (e.g. Diffie-Hellman, RSA)
    use two keys (public, private).
- **<u>Security Service(s)</u>** – that was failed by the encryption process
  - ✓ Confidentiality – the main security service provided by encryption.
  - ✓ ~Integrity, ~Identity Authentication – in some specific modes of encryption.

→ ENC is a high level class, so sites do not apply.

ENC: Example 1 – CVE-2002-1946

CVE-2002-1946 → using **ENC** taxonomy

Cause: Weak Encryption Algorithm (one-to-one mapping)

Attributes:

Sensitive Data: Credentials (passwords)

Data State: Stored (in registry)

Algorithm: Symmetric (that allows obtaining shared key and decryption)

Security Service: Confidentiality

Consequence: IEX of Sensitive Data (credentials)

CVE-2002-1946: “Videsh Sanchar Nigam Limited (VSNL) Integrated Dialer Software 1.2.000, when the "Save Password" option is used, stores the password with a weak encryption scheme (one-to-one mapping) in a registry key, which allows local users to obtain and decrypt the password.“ [1]

[1] The MITRE Corporation, CVE Common Vulnerabilities and Exposures, CVE-2002-1946.

ENC: Example 2 – CVE-2002-1697

CVE-2002-1697 → using **ENC** taxonomy

Causes: Insecure Mode of Operation (ECB) leads to Weak Encryption Algorithm (for same shared key produces same ciphertext from same plaintext)

Attributes:

Sensitive Data: Any (Credentials, Cryptographic, …)

Data State: Transferred (over network)

Algorithm: Symmetric (that allows identifying patterns and data recovery)

Security Service: Confidentiality

Consequence: IEX of Sensitive Data

CVE-2002-1697: "Electronic Code Book (ECB) mode in VTun 2.0 through 2.5 uses a weak encryption algorithm that produces the same ciphertext from the same plaintext blocks, which could allow remote attackers to gain sensitive information." [1]

[1] The MITRE Corporation, CVE-2002-1697, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-1697.

BF: Verification Bugs (VRF)

- Our Definition:
  The software does not properly sign data, check and prove source, or assure data is not altered.

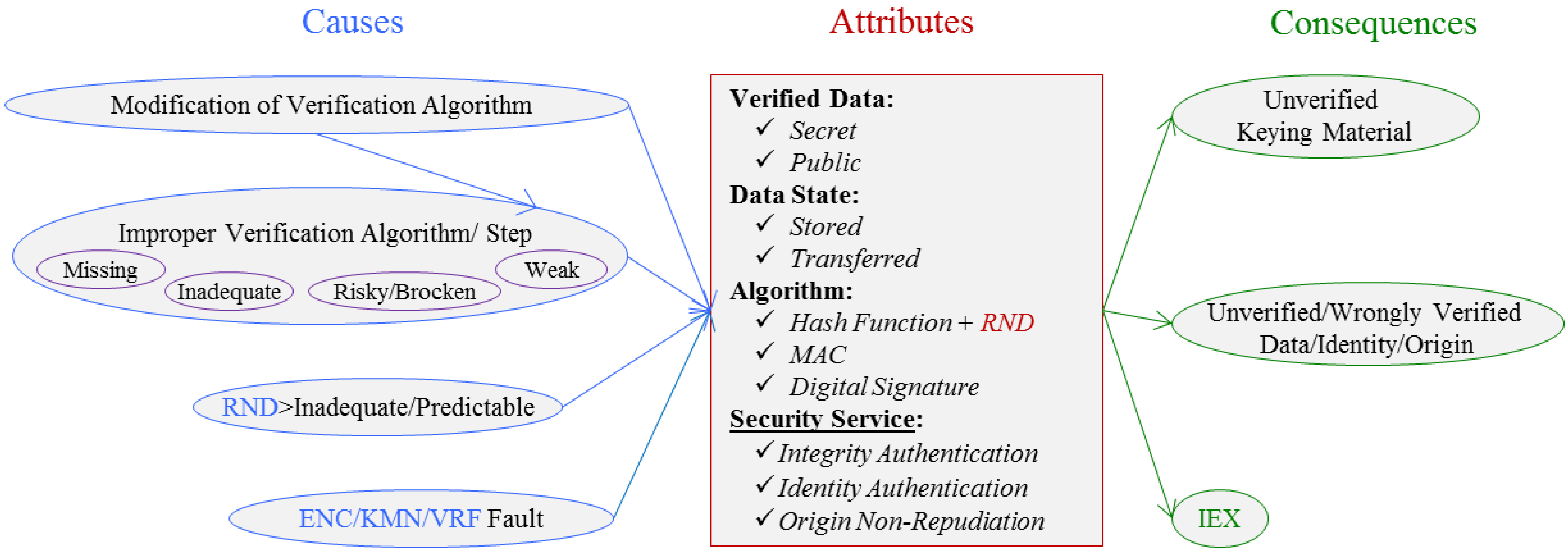
  Note that “check” is for identity authentication, “prove” is for origin (signer) non-repudiation, and “not altered” is for integrity authentication.

  VRF is related to KMN, TRN, PRN, ENC, ATN, IEX.

Related CWEs, SFPs and ST:

- ✓ CWEs: 295, 296, 347.
- ✓ SFP cluster: SFP 17.2 Weak Cryptography under Primary Cluster: Cryptography.

VRF: Causes, Attributes, and Consequences



Causes

Modification of Verification Algorithm

Improper Verification Algorithm/ Step

Missing

Inadequate

Risky/Brocken

Weak

RND>Inadequate/Predictable

ENC/KMN/VRF Fault

Attributes

Verified Data:

- ✓ *Secret*
- ✓ *Public*

Data State:

- ✓ *Stored*
- ✓ *Transferred*

Algorithm:

- ✓ *Hash Function + RND*
- ✓ *MAC*
- ✓ *Digital Signature*

Security Service:

- ✓ *Integrity Authentication*
- ✓ *Identity Authentication*
- ✓ *Origin Non-Repudiation*

Consequences

Unverified Keying Material

Unverified/Wrongly Verified Data/Identity/Origin

IEX

VRF: Attributes

- **Verified Data** – This is the data that needs verification. It may be confidential or public.
  - Secret (confidential) Data: cryptographic hashes, secret keys, or keying material.
  - Public Data: signed contract, documents, or public keys.
- **Data State –** This reflects if data is in rest or use, or if data is in transit.
- **Algorithm** – Hash Function + RND, Message Authentication Code (MAC), Digital Signature.
  - Hash functions are used for integrity authentication. They use RND.
  - MAC are symmetric key algorithms (one secret key per source/user), used for integrity authentication, identity authentication. It needs authentication code generation, source signs data, user gets tag for key and data, and verifies data by tag and key.
  - Digital Signature is an asymmetric key algorithm (two keys), used for integrity and identity authentication, and origin (signer) non-repudiation. It needs key generation, signature generation, and signature verification.

  MAC and Digital Signature use KMN and recursively VRF.

VRF: Attributes

- **Security Service** – This is the security service the verification process failed.
  - Data Integrity Authentication – for data and keys
  - Identity Authentication – for source authentication
  - Origin (Signer) Non-Repudiation – for source authentication.

→ VRF is a high level class, so sites do not apply.

VRF: Example 1 – CVE-2001-1585

CVE-2001-1585: “SSH protocol 2 (aka SSH-2) public key authentication in the development snapshot of OpenSSH 2.3.1, available from 2001-01-18 through 2001-02-08, does not perform a challenge-response step to ensure that the client has the proper private key, which allows remote attackers to bypass authentication as other users by supplying a public key from that user's authorized_keys file.“ [1]

[1] The MITRE Corporation, CVE Common Vulnerabilities and Exposures, CVE- 2001-1585.

VRF: Example 1 – CVE-2001-1585

CVE-2001-1585 description using VRF taxonomy:

CVE 2001-1585 → VRF

Cause Missing Verification Step (challenge-response) in public key authentication

Attributes:

Verified Data: Any (Secret/ Public)

Data State: Transferred (over network)

Algorithm: Digital Signature (not using such allows private key not to be verified by public key)

Security Service: Identity Authentication

Consequence: IEX

VRF: Example 2 – CVE-CVE-2015-2141

CVE-2015-2141: “The InvertibleRWFunction::CalculateInverse function in rw.cpp in libcrypt++ 5.6.2 does not properly blind private key operations for the Rabin-Williams digital signature algorithm, which allows remote attackers to obtain private keys via a timing attack. .” [1]

[1] The MITRE Corporation, CVE-2141, http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=2015-2141

VRF: Example 2 – CVE-CVE-2015-2141

CVE-2015-2141 description using VRF taxonomy:

CVE 2015-2141 → VRF

Cause: Modification of Verification Algorithm by adding a step (blinding)

Attributes:

Verified Data: Any (Secret/ Public)

Data State: Transferred (over network)

Algorithm: Digital Signature (Rabin-Williams) (that allows obtaining the private key in cases of incorrect unblinding)

Security Service: Identity Authentication

Consequence: IEX

BF: Key Management Bugs (KMN)

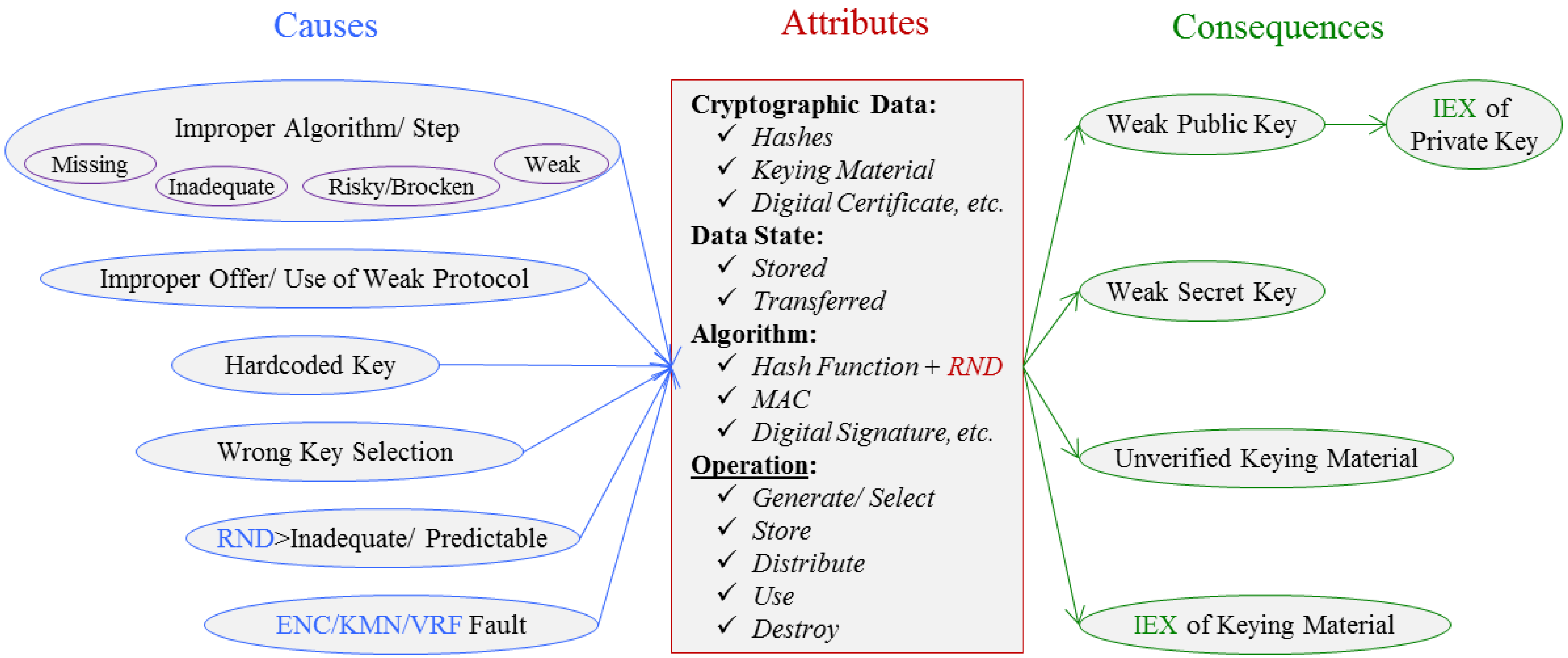
- Our Definition:
  The software does not properly generate, store, distribute, use, or destroy cryptographic keys and other keying material.

  KMN is related to ENC, TRN, PRN, VRF, IEX.

Related CWEs, SFPs and ST:

- ✓ CWEs: 321, 322, 323, 324.
- ✓ SFP clusters: SFP 17.2 Weak Cryptography under Primary Cluster: Cryptography and SFP 4.13 Digital Certificate under Primary Cluster: Authentication .

KMN: Causes, Attributes, and Consequences



Causes

Improper Algorithm/ Step

Missing

Inadequate

Risky/Brocken

Weak

Improper Offer/ Use of Weak Protocol

Hardcoded Key

Wrong Key Selection

RND>Inadequate/ Predictable

ENC/KMN/VRF Fault

Attributes

Cryptographic Data:

✓ *Hashes*

✓ *Keying Material*

✓ *Digital Certificate, etc.*

Data State:

✓ *Stored*

✓ *Transferred*

Algorithm:

✓ *Hash Function + RND*

✓ *MAC*

✓ *Digital Signature, etc.*

Operation:

✓ *Generate/ Select*

✓ *Store*

✓ *Distribute*

✓ *Use*

✓ *Destroy*

Consequences

Weak Public Key

IEX of Private Key

Weak Secret Key

Unverified Keying Material

IEX of Keying Material

KMN: Causes, Attributes, and Consequences

- **Cryptographic Data** – Hashes, Keying Material, Digital Certificate.
- **Data State –** This reflects if data is in rest or use, or if data is in transit.
- **Algorithm** – Hash Function + RND, MAC, Digital Signature.
- **<u>Operation</u>** –This is the failed operation: Generate uses RND.
  - Store – includes update and recover
  - Distribute – includes key establishment, transport, agreement, wrapping, encapsulation, derivation, confirmation, shared secret creation; uses ENC and KMN (reclusively)
  - Use
  - Destroy.

→ KMN is a high level class, so sites do not apply.

BF: KMN Example (FREAK)– CVE-2015-0204, CVE-2015-1637, CVE-2015-1067

FREAK – Factoring attack on RSA-ExportKeys

CVE-2015-0204: “The ssl3_get_key_exchange function in s3_clnt.c in OpenSSL before 0.9.8zd, 1.0.0 before 1.0.0p, and 1.0.1 before 1.0.1k allows remote SSL servers to conduct RSA-to-EXPORT_RSA downgrade attacks and facilitate brute-force decryption by offering a weak ephemeral RSA key in a noncompliant role, related to the "FREAK" issue. NOTE: the scope of this CVE is only client code based on OpenSSL, not EXPORT_RSA issues associated with servers or other TLS implementations.” [1]

CVE-2015-1637: “Schannel (aka Secure Channel) in Microsoft Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows 8.1, Windows Server 2012 Gold and R2, and Windows RT Gold and 8.1 does not properly restrict TLS state transitions, which makes it easier for remote attackers to conduct cipher-downgrade attacks to EXPORT_RSA ciphers via crafted TLS traffic, related to the "FREAK" issue, a different vulnerability than CVE-2015-0204 and CVE-2015-1067.” [2]

CVE-2015-1067: “Secure Transport in Apple iOS before 8.2, Apple OS X through 10.10.2, and Apple TV before 7.1 does not properly restrict TLS state transitions, which makes it easier for remote attackers to conduct cipher-downgrade attacks to EXPORT_RSA ciphers via crafted TLS traffic, related to the "FREAK" issue, a different vulnerability than CVE-2015-0204 and CVE-2015-1637.” [3]

[1] The MITRE Corporation, CVE--2015-0204, https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2015-0204

[2] The MITRE Corporation, CVE--2015-1637, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-1637.

[3] The MITRE Corporation, CVE--2015-1067, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-1067.

BF: KMN Example (FREAK)

FREAK description using VRF taxonomy: An inner KMN leads to an inner ENC, which leads to an outer ENC.

Inner KMN:

Cause: Improper Offer of Weak Protocol (Export RSA – offered from MITM-tricked server and accepted by client)

Attributes:

Cryptographic Data: Keying Material (pair of private and public keys)

Data State: Transferred (over network)

Algorithm: Export RSA (512-bits key generation based on prime numbers,
such that private key can be obtained from public key through factorization)

Operation: Generate

Consequence: IEX Keying Material (private key)

Inner ENC:

Causes: KMN Fault leads to Exposed Private Key

Attributes:

Sensitive Data: Cryptographic (Pre-Master Secret)

Data State: Transferred (over network)

Algorithm: Asymmetric (RSA) (that allows decryption of Pre-Master Secret using exposed private key
and computation of Master Secret)

Security Service: Confidentiality

Consequence: IEX of Sensitive Data (Master Secret)

BF: KMN Example (FREAK)

Inner KMN and inner ENC only set up the secret key. Outer ENC is the actual general data transfer.

Outer ENC:

Causes: KMN Fault leads to Exposed Secret Key (Master Secret)

Attributes:

- Sensitive Data: Credentials (passwords, credit cards)
- Data State: Transferred (over network)
- Algorithm: Symmetric (key is known)
- Security Service: Confidentiality

Consequence: IEX of Sensitive Data (credentials)

BF: KMN Example (FREAK)

Interestingly in this example the consequence from the first bug (inner KMN) causes the second bug (inner ENC), whose consequences cause the third bug (outer ENC).

Inner KMN is:

- A server bug – sending a weak key (that the client did not ask for) intended for KMN use by client (encrypting Pre-Master Secret).
- And also a client bug – as the client accepted the offer of using the insecure method, and therefore the server proceeded. The client could have refused that offer.

Inner ENC is:

- A client bug – using that weak key to encrypt the Pre-Master Secret, and then transmitting that weakly encrypted Pre-Master Secret over a network that is not secure.

BF: KMN Example (FREAK) – Source Code

Client

`#ifndef OPENSSL_NO_RSA`	
`if (alg_k & SSL_kRSA) {`	`if (alg_k & SSL_kRSA) {`
	`  if (!SSL_C_IS_EXPORT(s->s3->tmp.new_cipher)) {`
	`    al=SSL_AD_UNEXPECTED_MESSAGE;`
	`SSLerr(SSL_F_SSL3_GET_SERVER_CERTIFICATE,SSL_R_UNEXPECTED_MESSAGE);`
	`    goto f_err;`
	`  }`
`if ((rsa=RSA_new()) == NULL) {`	`if ((rsa=RSA_new()) == NULL) {`
`SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_MALLOC_FAILURE);`	`SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_MALLOC_FAILURE);`

Server

`case SSL3_ST_SW_KEY_EXCH_B:`	`case SSL3_ST_SW_KEY_EXCH_B:`
`alg_k = s->s3->tmp.new_cipher->algorithm_mkey;`	`alg_k = s->s3->tmp.new_cipher->algorithm_mkey;`
`if ((s->options & SSL_OP_EPHEMERAL_RSA)`	
`#ifndef OPENSSL_NO_KRB5`	
`   && !(alg_k & SSL_kKRB5)`	
`#endif )`	
`    s->s3->tmp.use_rsa_tmp=1;`	
`else`	
`    s->s3->tmp.use_rsa_tmp=0;`	`s->s3->tmp.use_rsa_tmp=0;`
`if (s->s3->tmp.use_rsa_tmp`	`if (`

If client ciphersuit is non-export then returned by server RSA keys should be also non-export.

Therefore, handshake that offers export RSA key (512 bits, which is weak) should be abandoned by client.

The buggy code includes a handshake that enables accepting a 512-bit RSA key.

The fix is adding code that checks whether client ciphersuit is non-export and for abandoning the handshake if this is the case.

Access Control Cluster (ACC):

- Authentication Bugs (ATN)
- Authorization Bugs (AUT)

BF Model of Access Control

<<still under development>>

…

Refined Classes

Buffer Overflow (BOF)

BF: Buffer Overflow (BOF)

- Our Definition:

The software accesses through an array a memory location that is outside the boundaries of that array.

This definition is clearer than CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer: *“The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer.”*

✓ clarifies that access is through the same buffer to which the intended boundary pertains.

✓ accurately, precisely, and concisely describes violation of memory safety.

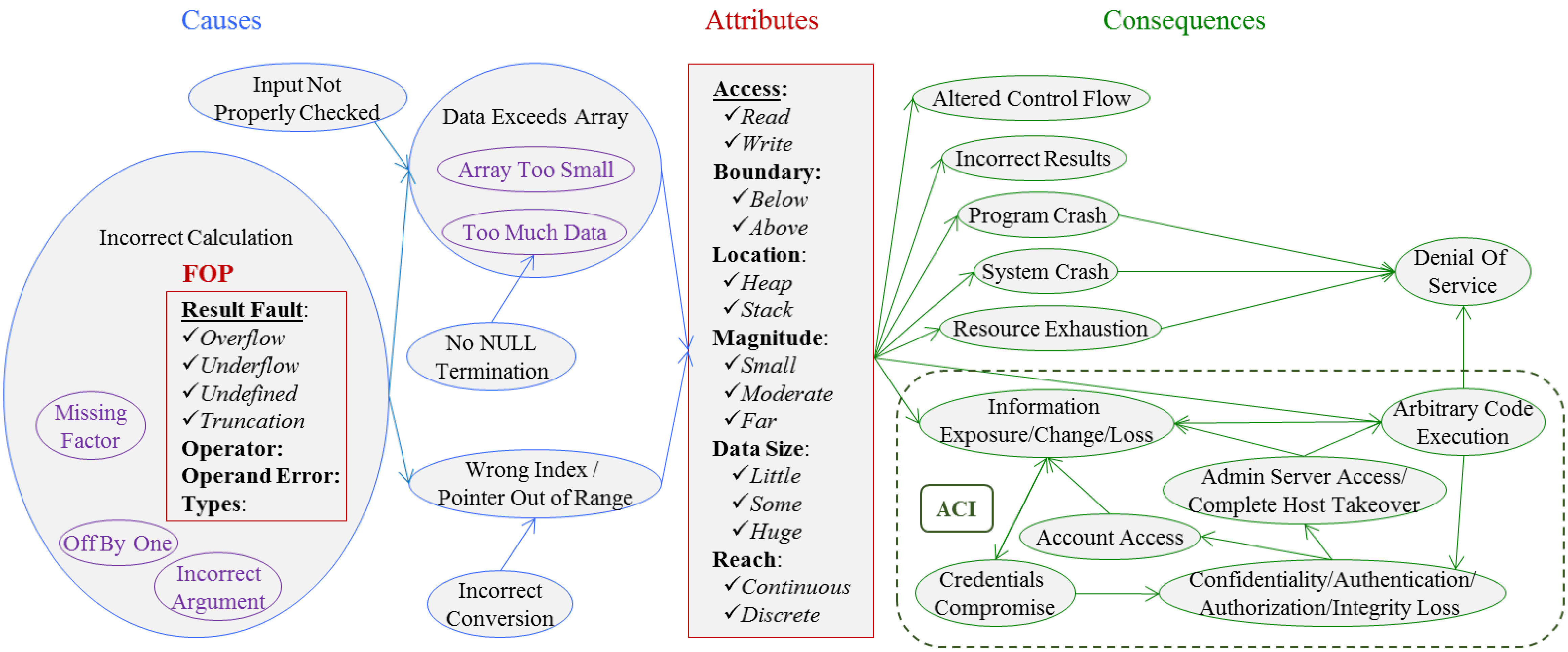
Related CWEs, SFP and ST

CWEs are 119, 120, 121, 122, 123, 124, 125, 126, 127, 786, 787, 788, 805, 806, 823.

SFP cluster is SFP8 Faulty Buffer Access under Primary Cluster: Memory Access.

ST is the Buffer Overflow Semantic Template.

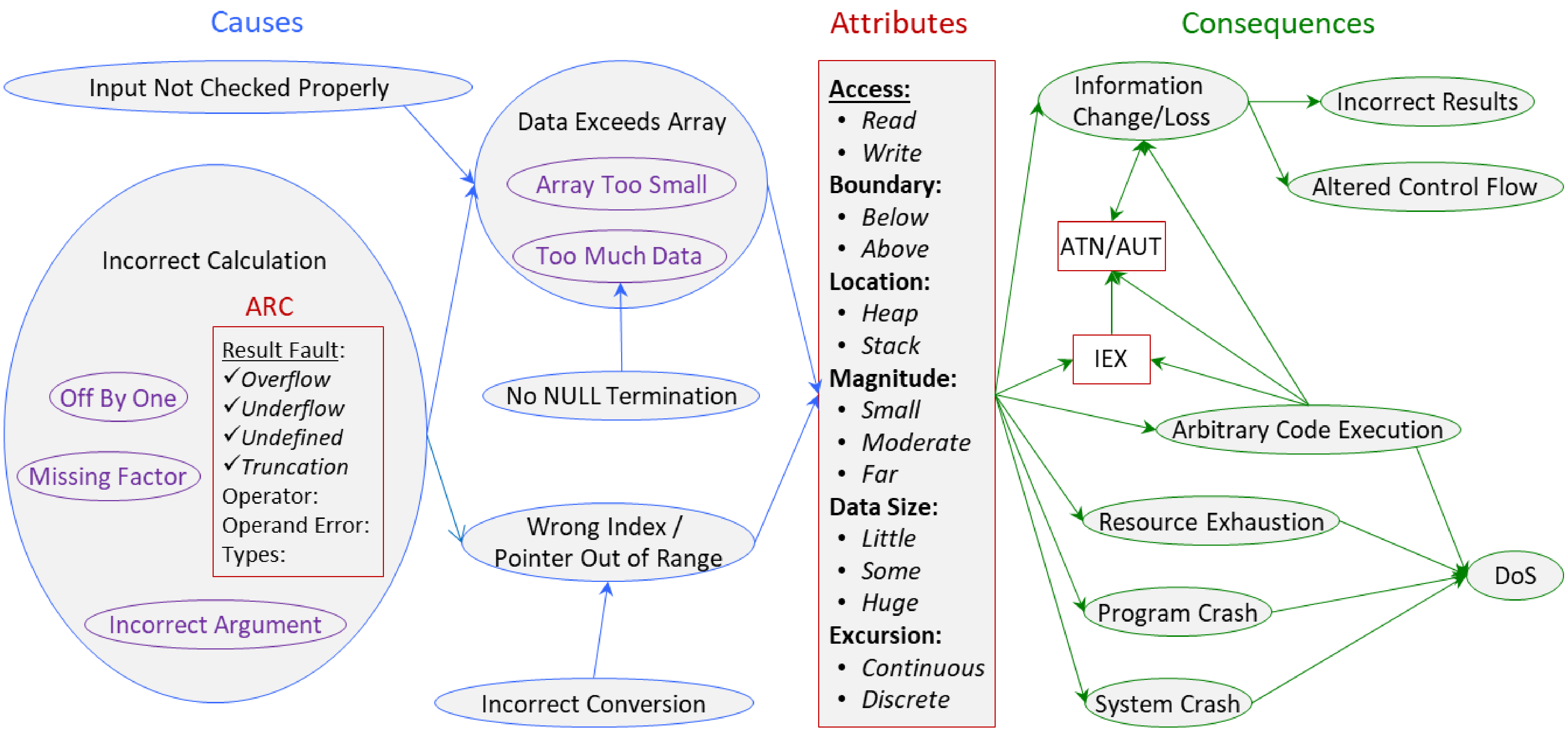
<<OLD>>BOF: Causes, Attributes, and Consequences



Result Fault:
✓*Overflow*
✓*Underflow*
✓*Undefined*
✓*Truncation*
Operator:
Operand Error:
Types:

Access:
✓*Read*
✓*Write*
Boundary:
✓*Below*
✓*Above*
Location:
✓*Heap*
✓*Stack*
Magnitude:
✓*Small*
✓*Moderate*
✓*Far*
Data Size:
✓*Little*
✓*Some*
✓*Huge*
Reach:
✓*Continuous*
✓*Discrete*

BOF: Causes, Attributes, and Consequences



Causes

Input Not Checked Properly

Data Exceeds Array

Array Too Small

Too Much Data

Incorrect Calculation

ARC

Result Fault:
✓*Overflow*
✓*Underflow*
✓*Undefined*
✓*Truncation*
Operator:
Operand Error:
Types:

Off By One

Missing Factor

Incorrect Argument

No NULL Termination

Wrong Index / Pointer Out of Range

Incorrect Conversion

Attributes

Access:
- *Read*
- *Write*

Boundary:
- *Below*
- *Above*

Location:
- *Heap*
- *Stack*

Magnitude:
- *Small*
- *Moderate*
- *Far*

Data Size:
- *Little*
- *Some*
- *Huge*

Excursion:
- *Continuous*
- *Discrete*

Consequences

Information Change/Loss

Incorrect Results

Altered Control Flow

ATN/AUT

IEX

Arbitrary Code Execution

Resource Exhaustion

Program Crash

System Crash

DoS

Injection (INJ)

BF: Injection (INJ)

- Our Definition:
  Due to input with language-specific special elements, the software assembles a command string that is parsed into an invalid construct.

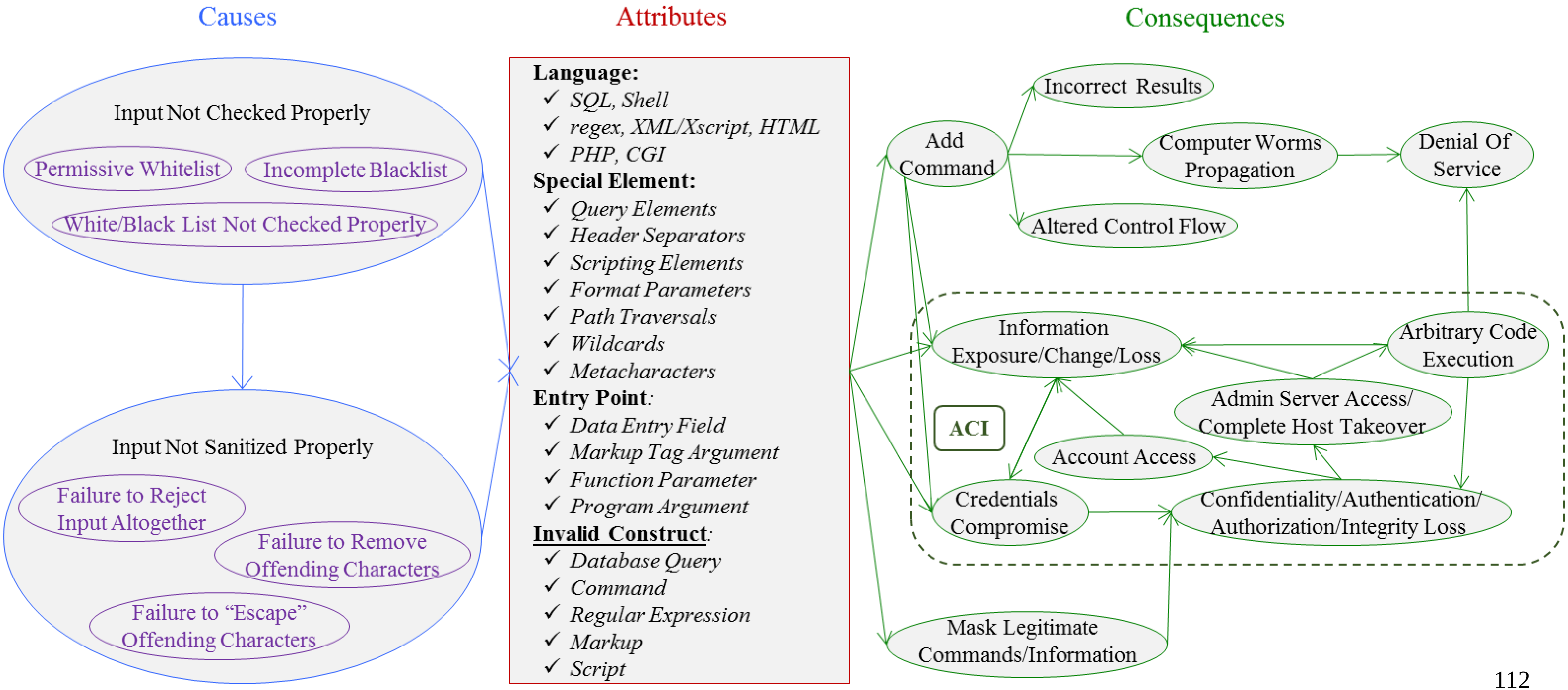
  In other words, the command string is interpreted to have unintended commands, elements or other structures.

  INJ is related to: ATN, ...

Related CWEs, SFPs and ST:

- ✓ CWEs related to INJ are 74, 75, 77, 78, 80, 85, 87, 88, 89, 90, 93,94, 243, 564, 619, 643, 652.
- ✓ Related SFPs are SFP24 and SFP27 under Primary Cluster: Tainted Input, and SFP17 under Primary Cluster: Path Resolution.
- ✓ The corresponding ST is the Injection Semantic Template.

<<OLD>>INJ: Causes, Attributes, and Consequences



Causes

Input Not Checked Properly

- Permissive Whitelist
- Incomplete Blacklist
- White/Black List Not Checked Properly

Input Not Sanitized Properly

- Failure to Reject Input Altogether
- Failure to Remove Offending Characters
- Failure to “Escape” Offending Characters

Attributes

Language:

- ✓ *SQL, Shell*
- ✓ *regex, XML/Xscript, HTML*
- ✓ *PHP, CGI*

Special Element:

- ✓ *Query Elements*
- ✓ *Header Separators*
- ✓ *Scripting Elements*
- ✓ *Format Parameters*
- ✓ *Path Traversals*
- ✓ *Wildcards*
- ✓ *Metacharacters*

Entry Point*:*

- ✓ *Data Entry Field*
- ✓ *Markup Tag Argument*
- ✓ *Function Parameter*
- ✓ *Program Argument*

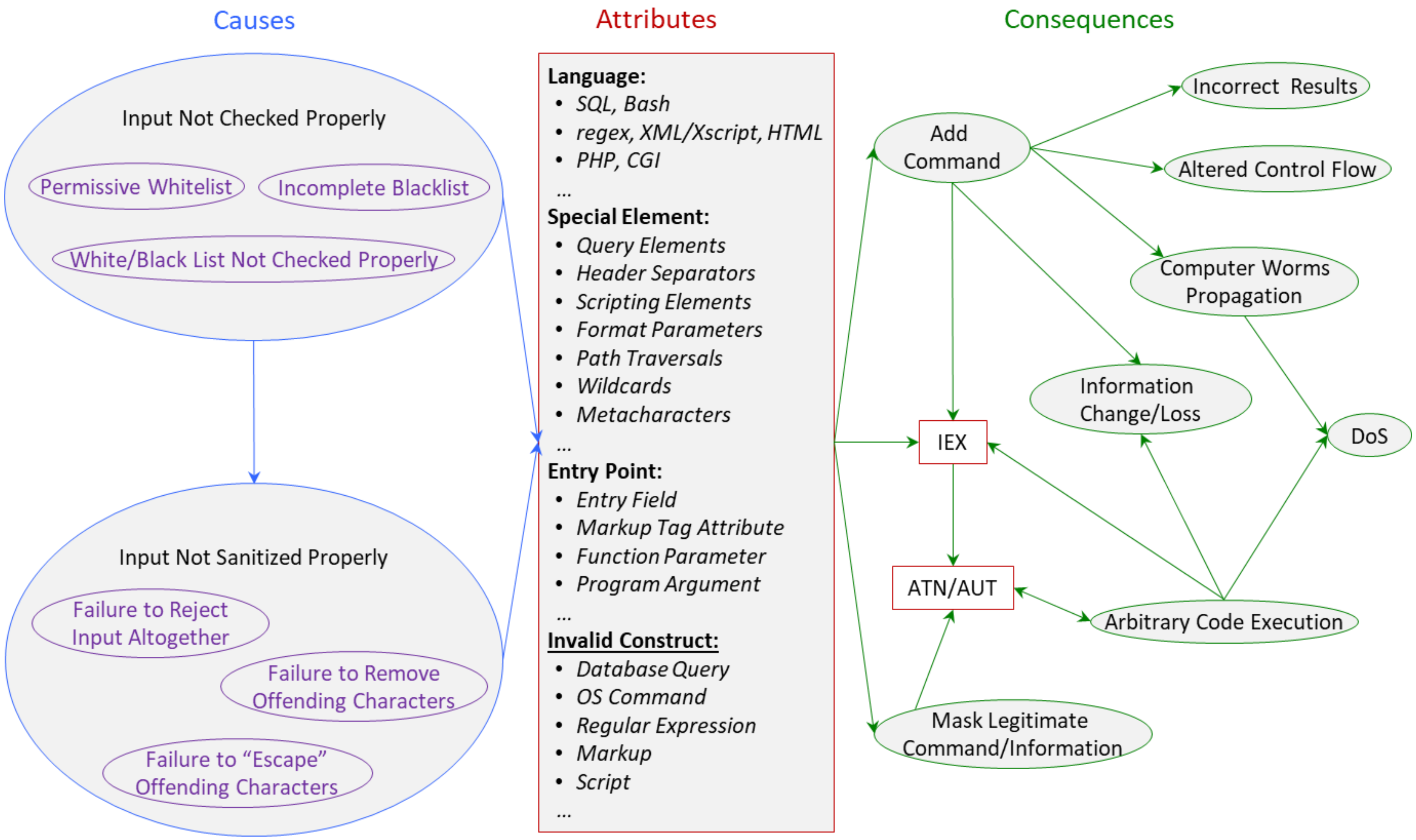
Invalid Construct*:*

- ✓ *Database Query*
- ✓ *Command*
- ✓ *Regular Expression*
- ✓ *Markup*
- ✓ *Script*

Consequences

Add Command
Incorrect Results
Computer Worms Propagation
Denial Of Service
Altered Control Flow
Information Exposure/Change/Loss
Arbitrary Code Execution
ACI
Admin Server Access/ Complete Host Takeover
Account Access
Credentials Compromise
Confidentiality/Authentication/ Authorization/Integrity Loss
Mask Legitimate Commands/Information

INJ: Causes, Attributes, and Consequences



Causes

Input Not Checked Properly

Permissive Whitelist

Incomplete Blacklist

White/Black List Not Checked Properly

Input Not Sanitized Properly

Failure to Reject Input Altogether

Failure to Remove Offending Characters

Failure to "Escape" Offending Characters

Attributes

Language:

- *SQL, Bash*
- *regex, XML/Xscript, HTML*
- *PHP, CGI*

…

Special Element:

- *Query Elements*
- *Header Separators*
- *Scripting Elements*
- *Format Parameters*
- *Path Traversals*
- *Wildcards*
- *Metacharacters*

…

Entry Point:

- *Entry Field*
- *Markup Tag Attribute*
- *Function Parameter*
- *Program Argument*

…

Invalid Construct:

- *Database Query*
- *OS Command*
- *Regular Expression*
- *Markup*
- *Script*

…

Consequences

Add Command

Incorrect Results

Altered Control Flow

Computer Worms Propagation

Information Change/Loss

DoS

IEX

ATN/AUT

Arbitrary Code Execution

Mask Legitimate Command/Information

Control of Interaction Frequency (CIF)

BF: Control of Interaction Frequency (CIF)

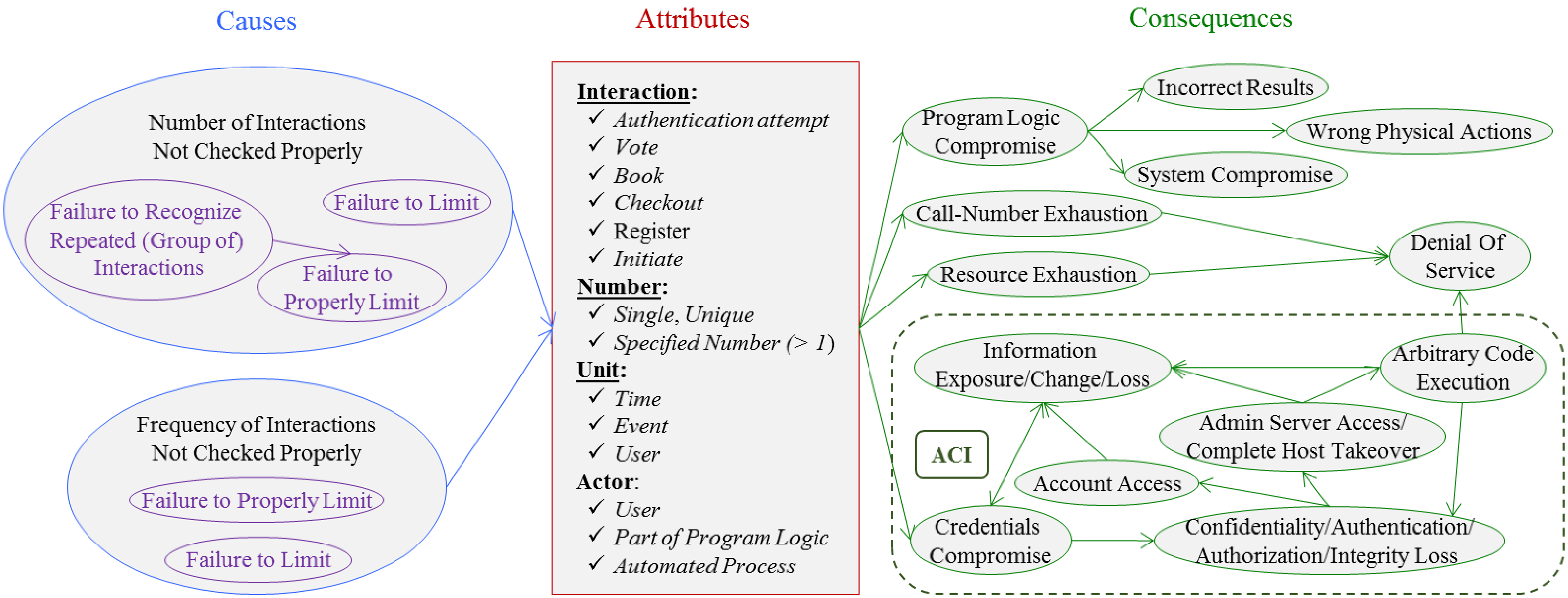
- Our Definition:
  The software does not properly limit the number of repeating interactions per specified unit.

  E.g. failed logins per day, one vote per voter per election (more for certain races!), maximum number of books checked out at once, etc. Interactions in software could be per event or per user.

  This class shows that we must acknowledge outside or local "policies".

- Related CWEs, SFPs and ST:
  - ✓ CWEs related to CIF are 799, 307, 837.
  - ✓ The related SFP cluster is SFP34 Unrestricted Authentication under the Primary Cluster: Authentication.

<<OLD>>CIF: Causes, Attributes, and Consequences



Causes

Number of Interactions Not Checked Properly

- Failure to Recognize Repeated (Group of) Interactions
- Failure to Limit
- Failure to Properly Limit

Frequency of Interactions Not Checked Properly

- Failure to Properly Limit
- Failure to Limit

Attributes

Interaction:

- ✓ *Authentication attempt*
- ✓ *Vote*
- ✓ *Book*
- ✓ *Checkout*
- ✓ Register
- ✓ *Initiate*

Number:

- ✓ *Single, Unique*
- ✓ *Specified Number (> 1)*

Unit:

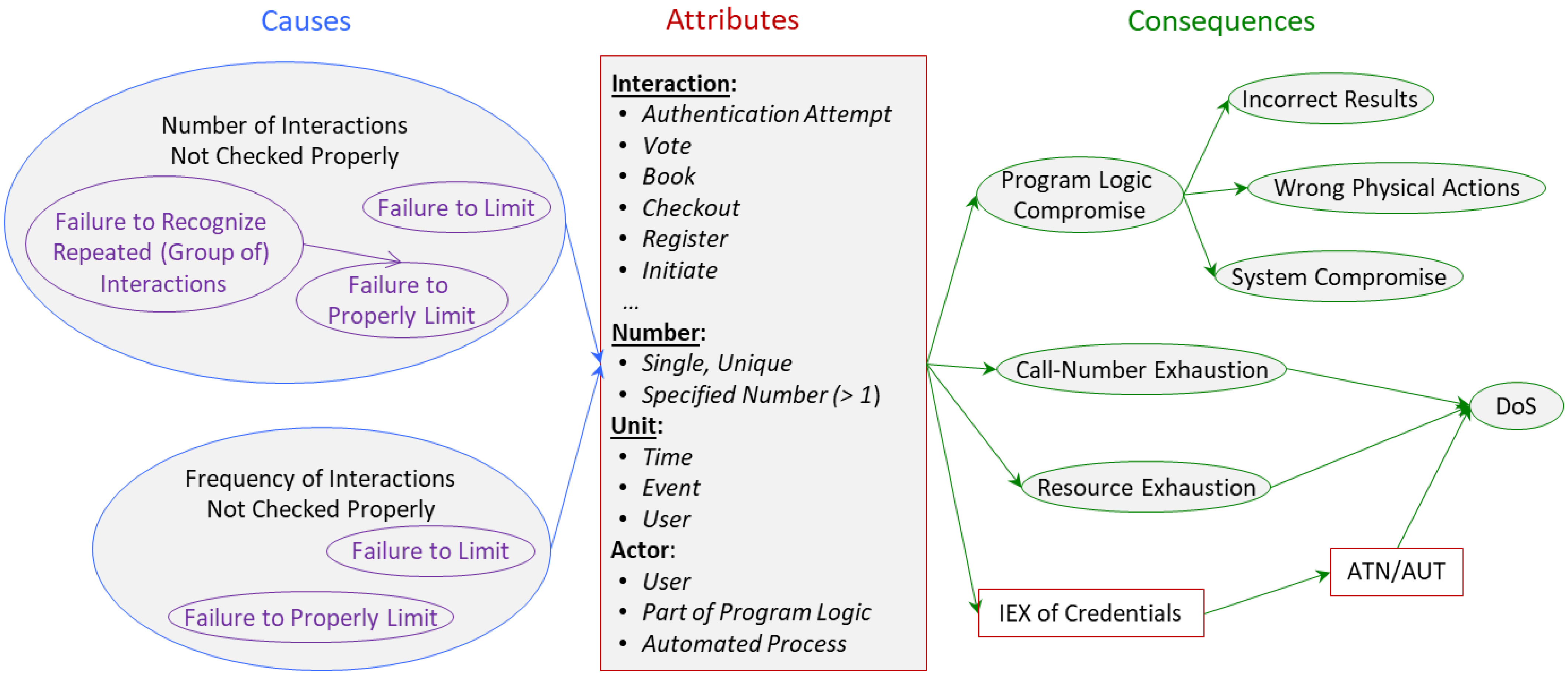
- ✓ *Time*
- ✓ *Event*
- ✓ *User*

Actor:

- ✓ *User*
- ✓ *Part of Program Logic*
- ✓ *Automated Process*

Consequences

CIF: Causes, Attributes, and Consequences



Causes

Number of Interactions Not Checked Properly

Failure to Recognize Repeated (Group of) Interactions

Failure to Limit

Failure to Properly Limit

Frequency of Interactions Not Checked Properly

Failure to Limit

Failure to Properly Limit

Attributes

Interaction:

- *Authentication Attempt*
- *Vote*
- *Book*
- *Checkout*
- *Register*
- *Initiate*

…

Number:

- *Single, Unique*
- *Specified Number (> 1)*

Unit:

- *Time*
- *Event*
- *User*

Actor:

- *User*
- *Part of Program Logic*
- *Automated Process*

Consequences

Program Logic Compromise

Incorrect Results

Wrong Physical Actions

System Compromise

Call-Number Exhaustion

Resource Exhaustion

DoS

IEX of Credentials

ATN/AUT

4. Benefits of Using BF

Benefits of Using BF

BF provides a superior, unified approach that allows us to:

- Precisely and unambiguously express software bugs or vulnerabilities.
- Explain clearly applicability and utility of different software quality or assurance techniques or approaches.
- More formally reason about assurance techniques or mitigation approaches that may work for a fault with certain attributes (but not for the same fault with other attributes).

Benefits of Using BF

With BF practitioners and researchers can more accurately, precisely and clearly:

- Describe problems in software.
- Clearly document the classes of bugs that a tool does and does not report.
- Explain what vulnerabilities the proposed techniques prevent.

➢ Those concerned with software quality, reliability of programs and digital systems, or cybersecurity
→ will be able to make more rapid progress by more clearly labeling the results of errors in software.

➢ Those responsible for designing, operating and maintaining computer complexes
→ can communicate with more exactness about threats, attacks, patches and exposures.

BF: Future Work

- Explain more vulnerabilities using the developed BF classes.
- Develop more and more BF classes.

→ Our goal is for BF to become the software developers’ and testers’ “*Best Friend*.”

Questions

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