Information Exposure (IEX): A New Class in the Bugs Framework (BF)

Irena Bojanova
SSD, ITL, NIST
Gaithersburg, MD, USA
irena.bojanova@nist.gov

Yaacov Yesha
CSEE, UMBC
Baltimore, MD, USA;
SSD, ITL, NIST
Gaithersburg, MD, USA
yaacov.yesha@nist.gov

Paul E. Black
SSD, ITL, NIST
Gaithersburg, MD, USA
paul.black@nist.gov

Yan Wu
CSD, CAS, BGSU
Bowling Green, OH, USA
yanwu@bgsu.edu

***Abstract*—Exposure of sensitive information can be harmful on its own. In addition, it could enable further attacks. A rigorous and unambiguous definition of information exposure faults can help researchers and practitioners identify them, thus avoiding security failures. This paper describes Information Exposure (IEX), a new class in the Bugs Framework (BF). The IEX class comprises a rigorous definition and (static) attributes of the class, along with their related dynamic properties, such as proximate and secondary causes, consequences and sites. We use the IEX class to analyze specific vulnerabilities and provide clear descriptions. We also discuss lessons we learned that will help create additional BF classes.**

Keywords—sensitive information, information exposure, information leakage, software weaknesses, bug taxonomy, attacks.

I. Introduction

The software profession is in need of a structured framework allowing us to unambiguously discuss software faults, failures, attacks and vulnerabilities. Some analogous organizational structures in science are the Periodic Table of Elements, the Tree of Life, the Geographic Coordinate System, and the Dewey Decimal Classification System.

Common Weakness Enumeration (CWE) [1], Common Vulnerabilities and Exposures (CVE) [2] are widely used compilations. However, for very formal, exacting work, the definitions are often inaccurate, imprecise or ambiguous. Each CWE bundles many stages, such as likely attacks, resources affected and consequences. The coverage is uneven, with some combinations of attributes well represented and others not appearing at all. Software Fault Patterns (SFP) [3] is an excellent advance but does not tie fault clusters to causes or chains of fault patterns nor to consequences of a particular vulnerability. We present a more detailed discussion on related efforts in [4], as the focus of this paper is on one of our newly developed classes in the Bugs Framework (BF) [4,5].

The BF allows to more accurately and precisely define software bugs or vulnerabilities. Just as the structure of the periodic table reflects the underlying atomic structure, we are developing a taxonomy dictated by the "natural" organization of software bugs, while using as stepping stones known bugs enumerations, compendia and collections. BF organizes bugs and faults into distinct classes. Each BF class comprises level, causes, attributes, consequences, and sites of bugs.

In this paper, we present our brand-new BF class Information Exposure (IEX) – including the BF information exposure model, examples of IEX descriptions of CVE vulnerabilities, and lessons learned. Previously developed BF classes are presented in the Publications page in [5].

II. Information Exposure

Information and data can be stored, transferred, and used by digital systems. Information exposure, or information leaks, occurs when the system inadvertently reveals sensitive information inappropriately. [6]

Through information exposure bugs, the software may reveal login credentials, private keys, state and system data, as well as personal, financial, health, or business data. 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 and one new BF class.

In this section we discuss related terms and our BF model of information exposure.

A. Information and Data

The terms “data” and “information” are often used interchangeably. *Data* is “a set of values of qualitative or quantitative variables” [7]. *Information* is “any entity or form that provides the answer to a question of some kind or resolves uncertainty" [8]. To what extent data is informative to someone depends on how unexpected it is to that person. A difference between data and information is that 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, an 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. Typically, a person’s age is considered information, whereas “temperature” may be considered data, since it may be yesterday’s high temperature (where?) or the current temperature in a furnace. Without further meaning, it is unclear whether the temperature is in Centigrade, Fahrenheit, Kelvin, or something else!

In software, information is generated by processing data [9]. We distinguish between information that is sensitive and information that is not. Certain kinds of information can be

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

indirectly sensitive: when revealed can lead to harmful consequences.

If sensitive information at rest in a file or database is properly encrypted (see BF class ENC [5]), then information cannot be exposed, assuming a secure decryption key. If information is communicated outside the system via a secure channel, it cannot be exposed, either.

Although "sensitive information" may seem to need no exposition, we elaborate to be more concrete. Sensitive information includes credentials, system data, state data, cryptographic data, digital documents, and personally identifiable and business data.

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.

Personally identifiable data corresponds to personally identifiable information (PII) and personally identifiable financial information (PIFI). PII is any information that could be used to distinguish people, e.g. social security number (SSN), driver's license number, and identification card number. PIFI includes financial account numbers with security codes/ access codes/ passwords. There is also protected health information, which includes a patient's medical record or payment history, and payment card information, such as cardholder name, expiration date, card verification value (CVV2 for Visa), card validation code (CVC2 for MasterCard), personal identification number (PIN) or PIN block, content of magnetic stripe, etc.

Business data covers intellectual property and trade secrets, operational and inventory data, and industry-specific data, in addition to customer and employee data.

B. *Information Exposure Model*

To understand information exposure, we developed a general model. Fig. 1 presents our BF information exposure model, showing 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.

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. [10] A program is comprised of functions' invocations. Most functions process input data into output information or data. However, some functions receive only control flow (e.g. to display information, error messages, dialogs; or to act on a global data value or when a state variable has changed or needs to be assessed) [11]. A process is a program in execution and holds resources such as central processing unit (CPU), memory, disk, and input/output. A program can involve more than one process [12]. A session is a temporary, interactive information interchange between two or more devices, or between a computer and a user (e.g. login session) [10].

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.

Information could leak through *legitimate channels* during normal use of software. For example, via information display, via queries (including query strings in SQL queries and GET requests), from hardcoded information (password, cryptographic key, etc.), class cloning, serializable classes,

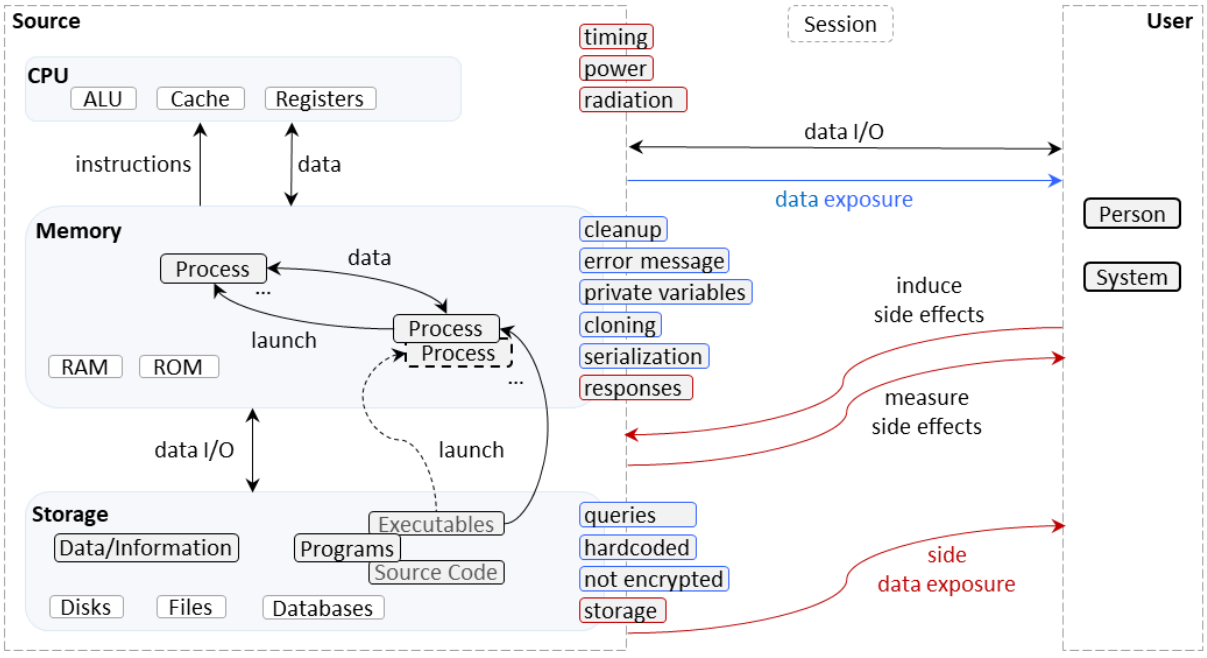

Fig. 1. The BF Model of Information Exposure. Legitimate channels are in blue. Side channels are in red.
(CPU – central processing unit, ALU – arithmetic logic unit, RAM – random-access memory, ROM – read-only memory)

includes removing previously used information, buffer cleanup (dead store removal), and use of `realloc()`. Sessions may leak information via session-ID length, sessions state boundaries, caching, and improper cleanup.

A *diagnostic channel*, or 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, or other responses to erroneous inputs or erroneous data processing. If an attacker forces an internal fault, it may divulge sensitive information, including details on software implementation logic. [6]

A *side channel* is not intended to transmit information; however, it does transmit information [13]. Information may be revealed or deduced due to discrepancies or behavioral inconsistencies – for example, conveying different responses (e.g. an operation is successful or not), taking different time (e.g. CPU timing), consuming different power, using different storage, or emitting different electromagnetic radiation. Behavioral inconsistency could be internal or external.

A *covert channel* is a side channel that is created deliberately as a hidden communication channel [14]. Unfortunately, a covert channel may be created by optimization techniques, such as compiler optimizations [15] and speculative executions [16].

In a side channel, it is common for an attacker to control both the part inducing the side effect and the part measuring it [17,18,19]. 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. [20] describes creating covert channels using transmission control protocol/Internet protocol (TCP/IP). Examples of side/covert attacks are Meltdown and Spectre [21], as well as the inference attacks [22].

III. Information Exposure Class – IEX

With that background, we now define the new BF IEX class.

A. Definition

We define Information Exposure Faults (IEX) as:

Information is leaked through legitimate or side channels.

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

IEX is related to the following current BF classes: BOF, INJ, CIF, ENC, VRF, KMN, TRN, PRN.

B. Taxonomy

Fig. 2 depicts IEX causes, attributes and consequences.

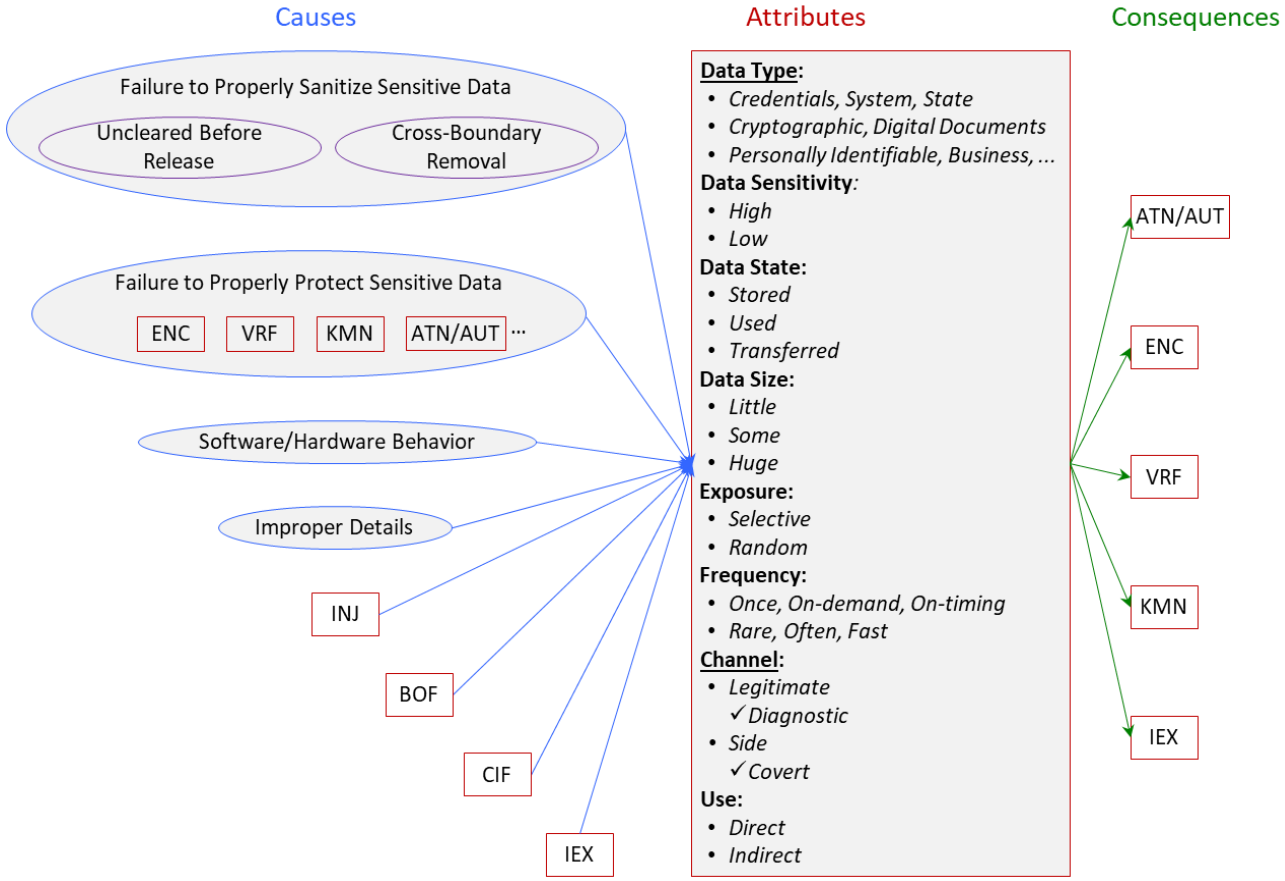

Fig. 2. Information Exposure (IEX) class.

The attributes of IEX are:

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 includes public records, phone books, or online directories.

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, file transfer protocol (FTP) root, concurrent versions system (CVS) repository), or on discs. Information can be in use by functions/programs -- source code (incl. comments); threads, registries, cookies, graphical user interface (GUI), environmental variables. Data can be also in transit between processes or over a network.

Data Size – Little, Some, Huge. This indicates how much data/information is leaked.

These distinctions are important in some cases. For instance, Heartbleed [23] 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 [23]).

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 that: `Frequency * Size = Rate`.

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.

C. *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 [1].

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.

ENC includes failure to encrypt (cleartext storage, recoverable format storage, cleartext transmission) and 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 is control of interaction frequency, including limiting the number of failed log in attempts. If there is no limit on the number of attempts, account names or passwords may be discovered by brute force attacks.

One *Information Exposure* (IEX) fault may lead to another information exposure. For instance, an information exposure of all client credit cards may have been caused by earlier obtaining the password for a privileged account.

D. *Related CWEs and SFP*

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, and 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 [3].

E. *Examples*

1) CVE-2007-5172

This vulnerability is listed in [2] and discussed in [1] (CWE-209).

The BF IEX taxonomy for this vulnerability is:

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.

Analysis (based on [2] – CVE 2007-5172 and CVE-2008-2049; and [1] – CWE-209):

CVE-2007-5172 Description: "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." [2].

2) CVE-2004-0243
This vulnerability is listed in [2] and discussed in [1] (CWE-203) and [24].

The BF IEX taxonomy for this vulnerability is:

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.

Analysis (based on [2] – CVE-2004-0243):

CVE-2004-0243 Description: "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." [2]. See also [1] (CWE-203), and [24].

3) CVE-2017-5753 and CVE-2017-5715 (Spectre)
This vulnerability is listed in [2] and discussed in [25, 26].

The BF IEX taxonomy for this vulnerability is:

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.

Analysis (based on [3,25,26,27,28]]): Caching results in reduced access time, and this outcome of speculative execution based on the wrong branch is not undone. The attacker uses a cache timing channel using Flush+Reload [27] and Evict+Reload [28] Mis-training of the processor by the attacker results in the processor's speculative execution based on a wrong branch. Timing can be used to recover information about the cache state, which exposes user's data.

4) CVE-2017-5754 (Meltdown)
This vulnerability was introduced in [17], and listed in [2].

The BF IEX taxonomy for this vulnerability is:

Cause: Hardware Behavior (CPU out-of-order execution)
Attributes:
Data Type: Any (passwords in password manager or browser, photos, emails, 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.

Analysis (based on information in [17]):

A timing covert channel is established using the Flush+Reload technique in [27]. It is based on the observation that access time is lower to an address that was cached. Therefore, access time measurements are used to determine which addresses were cached. The cached addresses are used to compute a secret value.

IV. Conclusion

A. Summary

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.

This joins other rigorously-defined BF classes, such as Encryption/Decryption Bugs (ENC), Verification Bugs (VRF), Key Management Bugs (KMN), True-Random Number Bugs (TRN), and Pseudo-Random Number Bugs (PRN). We presented the attributes of IEX, along with the its causes and consequences.

We analyzed particular vulnerabilities related to the IEX class and used IEX to provide clear descriptions. We showed that the BF-structured descriptions are quite concise, while still far clearer than unstructured explanations that we have found.

B. 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."

V. References

[1] The MITRE Corporation, Common Weakness Enumeration (CWE), https://cwe.mitre.org.

[2] The MITRE Corporation, Common Vulnerabilities and Exposures (CVE), https://www.cve.mitre.org.

[3] N. Mansourov, "DoD Software Fault Patterns," KDM Analytics, Inc., 2011. http://www.dtic.mil/cgi-bin/GetTRDoc? AD=ADB381215/.

[4] I. Bojanova, P. E. Black, Y. Yesha, and Y. Wu, "The Bugs Framework (BF): A Structured approach to express bugs", Proceedings of IEEE International Conference on Software Quality, Reliability and Security (QRS), 2016, pp. 175-182.

[5] The Bugs Framework, https://samate.nist.gov/BF.

[6] P. Anderson, "Common Embedded Vulnerabilities, Part 2: Information Leaks", https://www.electronicdesign.com/embedded/common-embedded-vulnerabilities-part-2-information-leaks [Accessed: Jan. 30, 2019].

[7] Wikipedia. Data. https://en.wikipedia.org/wiki/Data.

[8] Wikipedia. Information. https://en.wikipedia.org/wiki/Information.

[9] TechDifference. "Difference Between Data and Information," [Online]. Available: https://techdifferences.com/difference-between-data-and-information.html . [Accessed: Feb. 5, 2019].

[10] Wikipedia. Session (computer science). https://en.m.wikipedia.org/wiki/Session_(computer_science) .

[11] ScienceDirect. "Software Function," [Online]. Available: https://www.sciencedirect.com/topics/computer-science/software-function . [Accessed: Feb. 5, 2019].

[12] TechDifference. "Difference Between Program and Process," [Online]. Available: https://techdifferences.com/difference-between-program-and-process.html . [Accessed: Feb. 5, 2019].

[13] "Side Channels and Covert Channels". [Online]. Available: http://ver.miun.se/courses/security/lectures/sidechannels.pdf. [Accessed: Jan. 27, 2019].

[14] Stack Exchange. "Covert, Overt, and side channels." [Online]. Available: https://security.stackexchange.com/questions/113332/covert-overt-and-side-channels. [Accessed: Jan. 27, 2019].

[15] Wikipedia. Category:Compiler optimizations. https://en.wikipedia.org/wiki/Category:Compiler_optimizations .

[16] Wikipedia. Speculative execution. https://en.wikipedia.org/wiki/Speculative_execution .

[17] M. Lipp, et al. "Meltdown: Reading kernel memory from user space," *27th USENIX Security Symposium (USENIX Security 18)*. Aug. 15-17, 2018, Baltimore, MD, United States. pp. 973-990.

[18] Wikipedia. Side-channel attack. https://en.wikipedia.org/wiki/Side-channel_attack.

[19] D. Brumley and D. Boneh. "Remote timing attacks are practical." *Computer Networks* 48, no. 5 (2005). Pp. 701-716.

[20] "Covert Channels: How Insiders Abuse TCP/IP to Create Covert Channels," Mar. 1, 2018. [Online]. Available: https://thecybersecurityman.com/2018/03/01/covert-channels-how-insiders-abuse-tcp-ip-to-create-covert-channels/. [Accessed: Jan. 27, 2019].

[21] "Meltdown and Spectre." [Online]. Available: https://meltdownattack.com, https://spectreattack.com. [Accessed: Jan. 27, 2019].

[22] Wikipedia. Inference attack. https://en.wikipedia.org/wiki/Inference_attack.

[23] "The Heartbleed Bug." [Online]. Available: http://heartbleed.com/. [Accessed: Jan. 27, 2019].

[24] "Re: sqwebmail web login." [Online]. Available: https://marc.info/?l=bugtraq&m=107583269206044&w=2. [Accessed: Jan. 27, 2019].

[25] P. Kocher, et al. "Spectre attacks: Exploiting speculative execution." *arXiv preprint arXiv:1801.01203* (2018).

[26] D. Gruss. "Software-based Microarchitectural Attacks." [Online]. Available: https://gruss.cc/files/oecg_2018.pdf . Pdf. [Accessed: Jan. 27, 2019].

[27] YAROM, Y., AND FALKNER, K. FLUSH+RELOAD: A high resolution, low noise, L3 cache side-channel attack. In USENIX Security Symposium 2014 (2014), USENIX Association, pp. 719–732.

[28] LIU, F., YAROM, Y., GE, Q., HEISER, G., AND LEE, R. B. Last-level cache side-channel attacks are practical. In IEEE Symposium on Security and Privacy (S&P) 2015 (2015), IEEE.