AJSE

American Journal of Science & Engineering

Volume 1 Issue 1

June 2019

American Journal of Science & Engineering (AJSE)

Society for Makers, Artists, Researchers and Technologists (SMART)

6408 Elizabeth Ave SE, Auburn 98092, Washington, USA

ISSN: 2687-9530 (Print) and 2687-9581 (Online)

IoT security threats analysis based on components, layers and devices

Izzat Alsmadi
Department of Computign and Cyber Security, Texas A&M University, San Antonio
ialsmadi@tamusa.edu

Fahad Mira
Department of Computer Science and Technology University of Bedfordshire.
Luton, LU1 3JU, UK
Fahad.Mira@beds.ac.uk

***Abstract*— Internet of Things (IoTs) continue to grow to cover different types of applications to connect us, our appliances, our gadgets, etc. with the Internet. Information uploaded from those devices or exchanged with them is very vital and can affect us significantly. As a result, security threats and attacks that can come through those devices impact us seriously. In this systematic literature review paper, we evaluated security threats and attacks on IoTs based on different categories such as: IoT vulnerabilities, threats and attacks based on IoT architecture layers, and IoT vulnerabilities, threats and attacks based on IoT components. We showed areas of open research based on those categories. Due to the large spectrum of applications for IoTs, we hope that this classification can help researchers in this area focus their research to target one specific domain, category or threats.**

Keywords*— *Internet of Things, Cyber attacks.

I. INTRODUCTION

Internet of Things (IoT) continues to grow as one of the major IT buzz-words in both the academia and the industry. It is a natural expansion for our Internet connectivity where things in our world are continuously joining the Internet and are connected as communication devices; sending and/or receiving data and instruction. Such connectivity made things around us "smart" or "intelligent". They can help us and support our everyday activities. They can also be autonomous and responsive; taking actions, without or with the least levels of human interventions, based on real time data or incidents.

However, similar to most technology advances and services, they will come with some costs, challenges or difficulties. For example, similar advances and challenges are seen in mobile smart devices, cloud computing, Online Social Networks (OSNs), etc. The cycles of advances and challenges are very natural and we just need to keep moving forward. In this scope, we will focus on security challenges in IoT. Security challenges are by far, the most significant challenges facing most of the IT cutting edge technology trends.

Unlike powerful computing devices (e.g. HPCs, computing servers, etc.), or even normal computing devices (e.g. Desktops, laptops, tablets, smart devices), most of IoT devices are much simpler than those previously mentioned in terms of computing power (e.g. processing, memory, storage, network, etc.).

OWASP (www.owasp.org) IoT project described the followings as the top 10 security issues/vulnerability categories in IoT devices/environments: Insecure Web Interfaces, Insufficient Authentication/Authorization, Insecure Network Services, Lack of Transport Encryption, Privacy Concerns, Insecure Cloud Interfaces, Insecure Mobile Interfaces, Insufficient Security Configurability, Insecure Software/Firmware, and Poor Physical Security.

In this paper, we will investigate IoT security issues based on the following models: IoT vulnerabilities, threats and attacks based IoT components, and IoT vulnerabilities, threats and attacks based on IoT architecture layers (OSI or other). The rest of the paper includes 2 sections based on those 2 models in addition to a summary and conclusion section.

II. IOT VULNERABILITIES, THREATS AND ATTACKS BASED ON IOT COMPONENTS

An IoT device is composed from different components. From one perspective, we can divide an IT device into: Hardware, middleware and presentation layers. Each layer needs to interface with the other layers. For example, hardware can communicate with the middleware through different interfaces such as RFID, WSNs, ZigBee, Bluetooth, WiFi, etc. Middlewares can be software components, applications, APIs, etc. The presentation is another layer with largely software components to interface with users (e.g. through online interfaces, web applications, services, etc.) or other devices.

From functional perspectives, IoT components can be divided into: Sensors, communication/networks, standards and protocols, interfaces, database, visualization, intelligent analysis and actions' components. Current IoT devices across the different domains and environments are not homogeneous and may vary widely in the level of details and complexities where some IoT systems are much more mature than others (e.g. to include, databases, visualization, automation and analytic functions). We will focus our security assessment in this section based on the following sub-components: IoT hardware devices, mobile clients, web clients, cloud clients, gateways, support services, 3rd party web services, and IoT interfaces.

A. *Attacks on IoT Hardware Devices*

With the continuous growth in IoT applications and domains security concerns and challenges are also growing (Jing et al., 2014, 3]}, Roman, R., Zhou, J., and Lopez, J. 2013.

Security controls require their own computing and power resources. The limited resources in IoT limit also the ability to implement many security control features (Chatzigiannakis, Vitaletti, and Pyrgelis 2016. IoT devices should learn from other environments as they evolve where some security vulnerabilities have been exposed and handled in other environments (e.g. default passwords, Wei 2016. Table 1 shows a sample of security issues based on hardware devices, (Alaba et al 2017}.

Table 1: IoT hardware security issues (Alaba al 2017)

Hardware	Threats	Vulnerability	Attacks
RFID	Tracking DoS, Repudiation, Spoofing	Alteration, Corrup-tion and Deletion	Eavesdropping, Counterfeiting
ZigBee	Packet manipula-tion	Hacking	Key exchange, KillerBee, and Scapy
Bluetooth	Eavesdropping, DoS	Bluesnarfing Bluejack-ing	Car Whisperer, Bluebugging
Sensors node	DoS, Exhaustion, Unfairness, Sybil	Flooding, Routing Protocols	Jamming, Tamper-ing, Collisions

B. *Attacks on IoT Mobile Client*

Insecure IoT mobile interfaces can lead attacking its access control (e.g. privilege escalation) and eventually claim control of the IoT device. Some of the IoT device vulnerabilities they can use include: Poor or insufficient

authentication, account or access control exploitation, weak or improper encryption methods, etc. (OWASP 2015).
In information systems, access controls are important security controls or mechanisms that are implements at different guard locations (e.g. routers, operating systems, servers, DBMSs, web servers). In current systems, access controls in those components are matures and have mechanisms to deal with several types of attacks on users and their credentials (e.g. username and password-based attacks, privilege escalations, accounts enumerations and locking, etc.). On the other hand, access control in IoT is less mature and many of those attacks are shown to be possible in IoT environments. One of the challenges of trying to implement protections methods from servers, DBMSs, etc. on IoT systems is that IoT has much limited resources and hence solutions should take this into consideration to offer practical solutions.
IoT on mobile devices face several categories of security threats (Spreitzer et al 2010, You et al 2010 Li et al 2014 8]}, HP 2015 8]}, Stout and Urias 2016 9]}, Shin et al 2017 10]}.
\subsection{Attacks on hardware, perception or physical components}
IoT devices vary in their architecture and details. However, some of the main generic sub-layers or components in IoT hardware layer include: sensors, sensor networks, RFID tags and readers, 12]}.

The perception layer have several two sub-parts: perception node (e.g. sensors or controllers), and
perception network that communicates with transportation network, (Jing et al., 2014, 3]}.
Abnormal sensor nodes can be injected by hackers in this layer. This is as a result of a compromised original node. Decentralized intrusion detection systems (IDS) can be used to detect such malicious nodes.
SVELTE is an example of a real time IoT IDS.
Physical components can be also exposed to encryption attacks. Public key management mechanisms are used to create, distribute and test access keys, (Jing et al., 2014, 3]}).

Low power public encryption algorithms provide realistic and reliable key management solutions, Gaubatz et al 2005, 14]}.

IoT components in general and low-level components in particular can be targeted by several types of Denial of Service (DoS) attacks, Mirai is a popular type of botnets that has recently caused large-scale DDoS attacks by exploiting IoT devices.

C. Attacks on middleware components

Middle layer works as a bridge that connects the hardware layer with application or presentation layer. Middle layer handles tasks such as: Object management, data filtering, data aggregation, access control.

We will describe examples of vulnerabilities/attacks in different IoT environments.

1) Mosquitto message broker

Mosquitto is a common messaging platform used in IoT. Mosquitto is a MQTT, MQ Telemetry Transport, which is a low overhead machine-to-machine protocol that is used for communication between various IoT device to "talk" to each other using a system of publish and subscribe messaging transport. This protocol has some active vulnerabilities and possible exploits, which affects multiple devices throughout different industries.

Devices affected by Mosquitto vulnerabilities include: home automated devices, smart devices such as thermostats, microwaves, lights, speakers, sensors and microcontrollers.

2) 6LowPAN IPv6 adapter or header compression protocol

6LowPAN protocol was designed by IETF as an adaptation layer of IPv6 for Low power and lossy networks. 6LoWPAN protocol enables IPv6 packets to be carried on top of low power wireless networks, specifically IEEE 802.15.4.

Several papers reported possible vulnerabilities in this protocol. 6LoWPAN devices are vulnerable to attacks that are inherited from both the wireless sensor networks and the Internet protocols.

3) Extensible messaging and presence protocol (XMPP)

XMPP is a communication protocol for message-oriented middleware based on XML
(extensible markup language).

XMPP has seen wide implementation in IoT applications with its lightweight versions such as: XMPP-IoT. Although XMPP specification possesses various security features, some vulnerabilities also exist that can be leveraged to compromise the IoT network).

D. Attacks on presentation or application components

Application layer handles delivery of different applications to different types of users. When it comes to application components, IoT devices and systems span a large spectrum of applications and environments which may vary significantly specially in their presentation layer components.

1) Modbus

Modbus is a popular application protocol for industrial control system communications. It provides master/slave communication in SCADA systems. Several studies discussed vulnerability issues in Modbus.

As it is known to have vulnerabilities, attackers, search for unique methods to identify Modbus (e.g. Modbus Version Scanner, PLC Modbus Mode Identification). to identify the existence or usage of Modbus protocol.
Modbus has no security elements. Any attacker who can reach a Modbus server will be able to read and write to the field device or reboot the device and run diagnostic commands.

2) Constrained Application Protocol (CoAP) web transfer protocol

CoAP is an IoT specialized web transfer protocol for constrained nodes (i.e. nodes that have limited memory and/or
processing power) and also in constrained networks (i.e. low power and lossy networks).

Several papers discussed vulnerability issues in CoAP.
CoAP is by default bound to unreliable transports such as UDP. Messages may arrive out of order, appear duplicated, or go missing without notice. As a result, CoAP implements a lightweight reliability mechanism, without trying to re-create the full feature set of a transport protocol such as: TCP.

3) Online Video Games

Online video games are very popular and used by millions of users around the world. They are also major attack targets and many serious attacks are reported in the last few years in online gaming websites and consoles.
For the Video Game Industry (or any industry) that blends user information and online services, they must be able to handle the protection of and, in the event of an attack or leak, that the integrity of both the systems and user's information will still be intact.
However, not all hacking within a video game is malicious/industry breaking. Most, if not all hacking that is done over a large amount of time can consist of in game glitches—exploits that only effect one's experience within a game. Rarely does a game, system, or service have an exploit that can result in large data breaches, and while they still can occur, it's not nearly on the scale at the rate of normal glitches occur. Cheng Ki describes how many systems are inherently flawed from their inception. Many Mobile gaming and Indie game developers do not have the same resources as the major gaming companies, such as Sony and Microsoft, and are more susceptible to the common issues and exploits that games and systems can be found and fixed had the resources been available.

III. IoT SECURITY THREATS ANALYSIS BASED ON LAYERS (THE DIFFERENT IOT LAYERS BASED ON OSI MODEL)

A. Perception Layer

This level comprises of altered sorts of learning sensors like RFID, Barcodes or the additional finder arrange [8]. The essential reason for this level is to recognize the particular items and battle with its gathered information acquired from the $64000 world through the assistance of its few sensors.

B. Network Layer

The motivation behind this level is to convey the assembled information acquired from the discernment level, to several express information preparing framework concluded present correspondence systems comparable Web, Portable System or the other very solid system [9].
Business Layer: This current layer's capacities cowl all of IoT applications and administrations the board. It will deliver functional diagrams, plans of action, stream outline, govt report, and so forth bolstered the quantity of right data got from the lower layer and compelling data investigation technique. bolstered the great investigation results, it'll encourage the intentional directors or administrators to make a ton of right choices concerning the business techniques and roadmaps.[1]

C. Middle-ware Layer

This level comprises of {data, knowledge} process frameworks that take programmed activities upheld the aftereffects of prepared information and connection the framework with the data that gives stockpiling abilities to the gathered learning. This layer is administration arranged that guarantees a similar administration sort between the associated gadgets [10].

D. Application Layer

This level acknowledges shifted reasonable uses of IoT upheld the necessities of clients and very surprising sorts of businesses like great Home, great setting, great Transportation and great Emergency clinic and so on [11].

E. Perception Layer Challenges

Observation level comprises of altered finder advances like RFID which territory unit presented to a few sorts of dangers that region unit referenced beneath:

1) Unauthorized Access to the Tags.

Owing to the absence of correct Confirmation instrument amid a sizable measure of RFID frameworks, labels can stand gotten to by someone while not approval. The assaulter can't just peruse the data anyway the information will be changed or maybe erased still [14].

2) Tag biological research.

Since labels zone unit sent on entirely unexpected articles which region unit noticeable and their insight will peruse and changed with some hacking systems hence will| they will be basically caught through some cybercriminal WHO container make a copy of the tag and in this way bargaining it amid a way that the user can't recognize the first and subsequently the traded off tag [15].

3) Eavesdropping.

since of the remote attributes of the RFID it turns out to be horribly direct for the assaulter to smell out the direction like passwords or the other information spilling out of tag-to-user or user to-label that makes it helpless because of the assaulter will manufacture it to use in wretched ways that [16].

4) Spoofing.

Ridiculing is before AN assaulter communicates imagining material to the RFID frameworks and influence it to accept its inventiveness erroneously that makes it appearing from the underlying supply [17]. Along these lines assaulter grows complete admission to the framework creation it powerless.

5) RF electronic jamming.

RFID labels likewise container stand undermined by sort of a DoS assault inside which correspondence through RF signals is upset with AN unquestionably more than clamor signals [18].

F. Network Layer Challenges:

Network level contains of the Wireless detector Network (WSN) which conveys the info after the detector toward the situation terminus with re-liableness. The connected safety problems area unit mentioned underneath:

1) Sybil Attack.

Sybil could be a very assault inside which the assaulter controls the hub to blessing various characters for one hub because of that a significant a piece of the framework will be undermined bringing about false information in regards to the repetition [19].

2) Depression Attack.

it's a caring of attack inside which the opponent brands the traded off hub look drawing in to the close hubs owing to that

all the material result any express hub is redirected near the bargained hub prompting packages droplet for example all the traffic is hushed though the framework is deceived to trust that the information has been gotten on the contrary aspect. Moreover, this assault winds up in a great deal of vitality utilization which may reason DoS assault [20].

3) Sleep Deprivation Attack.

The identifier hubs inside the Remote Sensor System region unit control driven with batteries with not subsequently reasonable timespan that the hubs region unit ensured to pursue the rest schedules to expand their timeframe. Lack of sleep is that the very assault that keeps the hubs conscious, prompting a great deal of battery utilization and subsequently battery timeframe is diminished that makes the hubs closed miserable [21].

4) Denial of Service (DoS) Attack.

The kind of assault inside which the system is overwhelmed with a pointless load of traffic by AN assaulter, bringing about asset depletion of the focused on framework in light of which the system ends up unprocurable to the clients [22].

5) Malicious code injection.

This can be an overwhelming very assault in which AN assaulter bargains a hub to infuse malevolent cipher hooked on the framework that might even finish in a whole finish of the system or inside the greatest pessimistic scenario; the assaulter will become full administration of the system [23].

6) Man-in-the-Middle Attack.

This can be a style of listening stealthily inside which the objective of the assault is that the imparting because of that the unapproved gathering will screen or deal with all the individual interchanges between the 2 parties gigantically. The unapproved gathering will even imagine the personality of the person in question and convey unremarkably to accomplish a great deal of material [24].

G. Middle-ware Layer Challenges

This level consists of knowledge storing machineries comparable cloud computing. The protection tests of this level area unit mentioned underneath:

1) Unauthorized Access.

Center product Level gives entirely unexpected borders to the requests and information storerooms. The assailant will basically aim mutilation toward the framework through disallowing the entrance toward the associated administrations of IoT or through erasing the predominant information. So AN unapproved access likely could be lethal aimed at the framework.

2) DoS Attack.

It's equivalent to the DoS assault referenced inside the past 2 layers for example it closes depressed the framework which finishes in the inaccessibility of the administrations.

4.3.3 Malicious business executive.

This generous of assault happens once somebody after the privileged alters the statistics for private favorable circumstances or the advantages of any outsider. The information will be essentially extricated so changed deliberately after the confidential.

H. Application Layer Challenges

The connected security problems with this level area unit represented underneath:

1) Malicious Code Injection.

AN assaulter will use the assault on the framework from end-client through certain riding methods that enable the assaulter to infuse any very pernicious code into the framework to take some very learning after the client.

2) Denial-of-Service (DoS) Attack.

DoS assaults today must turned out to be refined; it proposals a smoke shade toward hold available assaults to rupture the protective outline then therefore learning security of the customer, and though misleading the injured individual into an essential subjective procedure that the specific assault is going on in better places. These spots the non-encoded individual subtleties of the client on account of the programmer.

3) Spear-Phishing Attack.

It's AN email caricaturing assault wherein injured individual, a great positioning individual, is baited hooked on the hole the email finished that the rival accesses the qualifications of that unfortunate casualty so by a falsification recovers a great deal of touchy info.

4) Sniffing Attack.

AN assaulter will constrain AN assault on the framework by bringing an individual request hooked on the framework, which might pick up system information prompting defilement of the framework [25].

SECURITY AT totally different LAYERS

Around are a unit several researches being administrated to produce a dependable distinct security design which may offer privacy of the info safety and confidentiality. W. Zhang et al. [26] planned AN architecture for the protection in contradiction of the attainable threats

I. Perception Layer

Discernment Level is that the base level of the IoT plan that gives fluctuated safety efforts to the equipment. It serves four essential capacities that territory unit Verification, learning Security, Protection of touchy information and Hazard Appraisal that region unit referenced underneath:

1) Authentication.

Confirmation is done exploitation cryptanalytic Hash Calculations that gives computerized marks to the terminals that would look up to all the achievable commonplace assaults like Side-channel assault, Savage power assault, and Impact assault and so on.

2) Knowledge Privacy.

Defense of the info is reinforced through parallel and uneven cryptography controls like RSA, DSA, BLOWFISH and

DES, and so on that averts unapproved admission to the identifier learning though existence met or sent to the resulting level. Due to their little power utilization benefit, they will remain just authorized into the sensors.

3) Privacy of sensitive data.

Concerning action the delicate information, the lack of definition of the circumstance and appeal is become using K-Namelessness approach that assurances the security of the data like character and site, and so forth of the customer [27].

It's an important of IoT security that security ruptures and pivotal the least complex security ways. A case of it's the powerful Hazard Appraisal philosophy for IoT [28].

Certainly, smooth with such safety efforts, if AN interruption is identified inside the framework, a programmed Murder direction from the RFID user is dispatched to the RFID label that keeps AN unapproved access to the RFID label learning [29].

J. Network Layer

The network level that will rather be every wired or wireless is exposed to numerous styles of attacks. Owing to the directness of the wireless channels, infrastructures are checked just by some hackers. The network level security is any alienated into three varieties that square measure mentioned below:

1) Authentication.

With the help of a right validation procedure and reason to reason cryptography, underground market access to the finder hubs to unfurl imagine information likely could be avoided [30]. The most widely recognized very assault is that the DoS assault that impacts the system through heavy a lot of pointless traffic to it through an assortment of botnets oxyacetylene through the arrangement of unified gadgets.

2) Routing Security.

When the Authentication methodology, routing algorithms square measure enforced to verify the privacy of data exchange between the detector nodes and thus the method systems [31]. There are many sorts of analysis administrated for the routing ways in which beside provide Routing [32], at intervals that data to be transmitted is hold on within the type of packets that are then sent to the process system once being analyzed by the intermediate nodes, and thus the Hop-by Hop routing at intervals that exclusively address of the data destination is believed.

The safety of routing is safeguarded by as long as multiple ways for the info routing that recovers the aptitude of the system to search out miscalculation and keep arts upon some quite disappointment at intervals the system [33].

3) Knowledge Privacy.

The assurance the executives components screen the framework for any very interruption and finally information, trustworthiness ways are authorized to brand indisputable that the data got on the contrary end is that the equivalent due to the first one.

K. Middle-ware and Application Layer

This level merges the Middle-ware and Application level to form AN integrated security device. The protection classification is mentioned below:

1) Authentication.

Above all else, it experiences the validation procedure that anticipates access to any guilty party client through synchronized personality distinguishing pieces of proof. this can be explicitly equivalent to that of the distinguishing proof technique in both of the films aside from that this level authorizes confirmations through some beyond any doubt collaborating administrations which mean clients will even choose the related information to be imparted to the administrations. The real advances utilized in this layer zone unit Distributed computing and Virtualization, every one of that zone unit ready to fluctuated assaults. The cloud innovation will be essentially bargained; one in all the most noticeably awful risk is the business official danger. Similarly, Virtualization is presented to DOS and information robbery, and so forth a lot of investigation is required in every area to create a safe setting.

2) Intrusion Detection.

Its break detection methods offer responses for fluctuated security hazards by producing A carefulness on the pervasiveness of any doubtful action inside the framework in view of the nonstop watching and custody a log of the gate crasher's exercises which may encourage to follow the participant. There region unit entirely unexpected existing interruption recognition methods [34] together with the data mining method [35] and irregularity location.

3) Risk Assessment.

The danger valuation proposals defense for effective security ways and provides enhancements within the current security structure.

The typical philosophy to uphold the security necessities of a framework is to depend on cryptanalytic techniques. These methods go for muddling the information inside the broadcast, creation the recipient unfit to induce the substance of a got message, except if falling back on horrendously high procedure control. Data obscurity is normally performed at layer a couple of or over different ways attempt and shroud the broadcast at the physical level. This was generally accomplished with unequivocal strategies that unfurl the flag underneath the commotion limit of the illegal collector.

Be that as it may, as of late, the physical layer security scope has been stretched out to moreover grasp a ton of eye-catching properties. Well as of now succinctly characterize the chief significant segments for our examination.

2.2. Physical-Layer Security

Security at the physical level was mainly assumed inside the historical on the grounds that the utilization of an assortment range system (recurrence jumping, direct grouping mystery composing, and so on.) in order to abstain from listening in. These physical layer systems pointed toward action the unimportant presence of a hub or the undeniable reality that correspondence was notwithstanding going down. The most issue is that after the enemy knows about the central matters of the correspondence framework, the total security is damaged. As an issue of the real world, unfurl range methods aren't considered any more drawn out security frameworks anyway rather as debilitating countermeasures. It is acknowledged that traditional cryptography systems have exclusively on preliminary unpredictability based mystery

[5]. We tend to also perceive that solid data theoretic mystery or fantastic mystery is feasible by quantum cryptography bolstered some uncommon quantum impacts like interruption location and inconceivability of flag clone [6]. Unfortunately, the reliance on such impacts winds up in exceptionally low transmission power as consequences of feeble signs must be constrained to be utilized. Furthermore, elective restrictions, for example, revision in polarization, absence of advanced marks, might want of an intense channel, short separation and middle of the road blunders assemble these procedures not anyway speedily implementable [7]. One of the ongoing makes an endeavor to indicate mystery information rate is [8], wherever the MIMO (numerous sources of info different yields) mystery information rate is dissected underneath the conviction that the opponent does not perceive even his very own channel. Nonetheless, such strategies aren't clear, as a result of the established truth that they need a high scope of radio wires on either side of the connection to control. Existing physical layer security methodologies will be arranged upheld the physical trademark that is abused.

Mystery Limit: the most extreme rate possible between the real transmitter-recipient consolidate subject to the limitations on information gettable by the unapproved collector, i.e., the most extreme transmission rate at that the listener is unfit to unscramble relates to the refinement between the capacity of the authentic connection and in this way the listener interface. Channel Mark/Unique finger impression: security bolstered the misuse of 1 of the channel attributes. recurrence (RF) qualities of the genuine connection, e.g., the channel motivation reaction, territory unit want to turn out a common mystery. Utilization of different directional receiving wires to disarrange the transmitted information stream or to infuse clamor inside the course of the listener. Range Spreading of Flag Vitality: utilization of an assortment Range (SS) methods like Direct Grouping Spread Range (DSSS) and Recurrence Bouncing unfurl Range (FHSS). Participation: agreeable hubs send their signs towards the listener in order to break down its connection. As of late, elective methods that utilization the channel correspondence to give a mystery showed up. In [9,10], the debilitating undeniable by the channel between the 2 genuine clients is utilized to frame a mystery progressively (in fact, this framework isn't at the physical layer anyway at the connection layer). In [11], fake commotion is utilized to give mystery given a chose zone wherever security ought to be guaranteed. In [1214], commotion is utilized as a result of the transporter of the information amid a shut circle subject. elective ongoing works [15,16] utilize the collaboration of including well-disposed hubs to give a mystery rate amid a transmission connect between 2 hubs inside the system. The amicable hubs mainly soil the station of the adversary hubs. In [17], hubs furnished with various reception apparatuses utilize consistent rationale: they transmit fake Data 2016, 7, 49 4 of 17 clamor by choice inside the course of the adversary, constraining it inside the bearing of the companion/wanted client. Amusement hypothesis will be wont to ponder the improvement of reliability versus mystery for each genuine hubs and spies [18]. An audit of helpful procedures for upgrading the insurance will be found in [19]. A large number of the methodologies spoke to over region unit bolstered suspicions that fabricate them not just implementable amid a genuine world: some of those need that a run of the mill from the earlier mystery is shared by the authentic clients or switched inside the start-up part through uncertain channels, and some others accept to get a handle on that A listener is a blessing and wherever it's settled. In actuality, most existing outcomes on mystery information rate zone unit upheld a few sorts of suspicions that appear to be unreasonable [20,21]. It's been a test in logical hypothesis for a long time to search out reasonable approaches to acknowledge data theoretic mystery. Perfect secrecy is doable by victimization physical layer techniques subject to the disorder that the channels area unit unidentified to illegal users or the channel of the unauthorized users is a lot of noisy than that of the licensed users. Whereas the normal cryptography techniques bank heavily on the upper-layer operations, it's attention-grabbing to grasp whether or not the physical layer will have some built-in security to help the upper-layer security styles. Rather than victimization a further channel, the physical level ways also can be used to distribute secret keys, to provide location privacy and to supplement upper-layer security algorithms. The applying of physical layer security schemes makes it tougher for attackers to decipher the transmitted data.

In physical layer security for remote systems, the mystery rate is delineated on the grounds that the rate at that data will be transmitted on the Q.T. from a supply to its alleged goal. The most extreme attainable mystery rate is known as the mystery ability. For instance, amid a Gaussian channel, the mystery limit is laid out in view of the refinement of the (Shannon) ability of the channel between the supply and the goal and in this manner the capacity of the channel between the supply and a listener [5,22]. The mystery is sketched out as data theoretic mystery, i.e., the enemy got flag offers no a bigger number of information for spying than severe estimation. The execution of this sort of physical layer security procedures, in particular the data hypothetical mystery, isn't simple nor paltry. Beginning proposition battle with the abuse of the remote channel between real clients to extricate a key to be utilized for encoding the message [23]. The data hypothetical mystery guarantees that if the extraction is made underneath the conviction to have a reward over Eves channel, the key's not recoverable by Eve in any way. AN exhaustive survey of cross-layer systems for improving the security will be found in [23]. In [24], the security issues and arrangements zone unit assessed for what contemplations the IoT point zone. The physical-layer security in any case isn't mulled over as data hypothetical mystery. An outline of the difficulties confronting physical-layer security is as per [25]. This paper doesn't battle with key extraction, notwithstanding, the immediate utilization of the consequences of data hypothetical mystery to give a protected connection, i.e., underneath that conditions in reasonable applications, as IoT applications, the data hypothetical mystery will be straightforwardly connected, so the listener can't recuperate any information in regards to the message by attentive the channel.

3. State of affairs and Threat Analysis

As we tend to made open inside the Presentation, the IoT worldview is utilized amid a major choice of utilizations and circumstances, beginning from gifted (e.g., Boycott for e-Wellbeing) to recreational (e.g., WSN for games players following). These applications zone unit appallingly entirely

unexpected and everybody have its own necessities regarding information privacy, learning respectability, and so forth. Notwithstanding the varieties, with respect to every one of the gadgets utilized in IoT share some normal style highlights: they're modest, battery-worked, and come up short on a right info framework. The above-named restrictions, related to the necessity, to remain the one gadget value low, raise an assortment of issues in verifying the framework. assumptive that the data channel likely could be made secure by a right cryptanalytic topic (and moreover this presumption isn't to be underestimated), there is a unit 2 noteworthy focuses wherever IoT gadgets region unit subject to assaults that zone unit well entirely unexpected from the ordinary dangers to elective organized gadgets.

The principal shortcoming of IoT gadgets originates from their design (see [26,27]). Gadgets have a lifecycle (producing, arrangement, support, retirement). All through each progression, it's feasible that the client must reconfigure some of the gadget security properties (e.g., gadget organize affiliation, keys, and so forth.). This reconfiguration strategy is, obviously, a sensitive system. It must be performed on a safe channel, or AN assaulter may get need classification information. The second shortcoming emerges from the lack of an all-around characterized topology. Most IoT frameworks are work, impromptu, multi-bounce systems. This has the monstrous beneficial thing about expanding the framework strength and system timeframe; in any case, it moreover allows an assortment of assaults especially focused to the directing and multi-jump plans. These assaults territory unit was outstanding in writing and it's achievable to utilize a few countermeasures. All things considered, sleuthing and hindrance the assaults remains an open issue. The location is unpredictable, on the grounds that the world system information isn't feasible, and in this manner the check is troublesome still. As an issue of the real world, appropriated firewalls likely could be in fact conceivable; nonetheless, they'd expend valuable assets inside the gadgets.

Information encryption will upgrade organize security and protection. In any case, key administration is dependably AN open issue [28]. It ought to be focused on that key understanding (or key spread) could be a typical issue to all or any the system layers, from MAC to IP to Application.

At long last, as we tend to referenced previously, the MAC headers territory unit ordinarily not encoded, allowing assaults to the client's protection in view of connection assaults.

3.1. Scenario

In instruction to judge the IoT intimidations and attainable countermeasures, we'll target the professional setting, and, specially, happening e-Health applications. Single of the foremost promising use-cases of BANs is their application to reintegration and unceasing persevering condition watching. While not harm of generalization, we'll target the subsequent use-case:

A patient arrives a recovery territory, wherever a specialist puts some wellbeing watching gadgets. Amid the restoration assembly, the sensors assemble some information and spread them to a screen posting complete a door. When the conference closes, the specialist expels the sensors from the persevering. The specialist ought to have the capacity to introduce an apparatus (i.e., actuate it on a chose patient) and decommission the gadget (i.e., remove it from the patient). These tasks ought to be secure, quick, and idiot proof. So as to safeguard the patient's protection, his/her own insight ought to be scrambled. Besides, just the talented obligated for the gadget the board (specialist, nurture, and so on.) ought to have the capacity to deal with the gadgets, and he/she ought to be placeable by the framework, in order to stop botches. As referenced previously, these necessities likely could be fulfilled by exploitation pertinent cryptanalytic plans. The issue is the best approach to deal with the cryptanalytic material. The appropriate response will be to recharge all he cryptanalytic keys each time a specialist must utilize the gadgets. In any case, this could be easy to perform and verify method, ready to be performed also by untalented work force. Another situation likely could be one in all stock pursue and conveyance: a load likely could be outfitted with a stage identifier (conceivably estimation moreover elective information, similar to vibrations, temperature, and so forth.). The delivery staff should probably get to a bundle of gadgets to peruse as well as store information (e.g., the entry time to an area), apparently with very surprising access rights to the hang on learning steady with the job inside the association (straightforward driver, supervisor, and so on.).

3.2. Assaulter Capabilities

We expect that the assaulter is an audience, i.e., it's hypnotized by effort fragile information by prescribes that of dormant strikes. In the midst of an uninvolved attack, the hazard administrator doesn't modify or intrude with the ordinary exchanges between genuine customers. In the midst of along these lines, the ambush will go unnoticed for a sweeping time. Likewise, we will in general expect that the assaulter thinks about everything regarding the system defenseless, and, explicitly, its change and secret forming plans, the used traditions, the channels, etc. this can be relentless with the Kirchhoff's standard (or the indistinguishable Shannon saying) communicating that the enemy thinks about the structure, which security isn't to progress toward becoming weak by uncertain quality. Finally, we will in general limit the attacker's gear and code abilities to the most clear off-the-rack hardware and code realistically.

3.3. Risk Investigation Normally, the danger examination is predicated on the system levels, i.e., Macintosh/PHY (Physical Layer), Datalink, and so forth., or the assault assortments. Despite what might be expected, we might want to spotlight anyway very surprising gadget timespan occasions will be utilized by AN assaulter.

3.3.1. Device producing

An assault performed all through the gadget delivering will introduce a secondary passage or debilitate a cryptanalytic library, empowering the assaulter to perform shifted illegal activities. Amid this class, we additionally order the issues emerging from unsafe gadget creating, e.g., zero-day bugs, support floods, terrible utilization of libraries, and so forth. The over dangers should be eased by AN appropriate gadget creating cycle, together with an obligatory Weakness Appraisal (VA) strategy for gadgets conveying touchy information.

3.3.2. Device readying

Gadget preparing is ordinarily performed by guaranteed professionals. As an outcome, it shouldn't speak to a security

issue. In any case, a few parts of gadget the board could need to be constrained to be left to the clients. Amid this case, framework security likely could be undermined. For instance, a system may bet on the client character to collect the entrance stipends to certain administrations, and along these lines the client personality is checked through AN ID card. The framework is absolutely protected, assumptive that everybody the clients keep their cards individual and verified, which can't be ceaselessly the situation.

3.3.3. Device Maintenance

We name the gadget upkeep impartial intended for fulfilment. Every one of the tasks satisfaction to support (i.e., gadget microcode redesign, gadget substitutions, and so forth.) should be performed by confirmed experts. Be that as it may, the specific pattern is to allow Over the Air (OTA) framework updates and arrangement setup. Though this can be a dreadfully convenient component, it also allows AN assaulter to require the entire control of a framework just by impersonating the redesign technique. It'd seem intelligent that OTA and remote administration capacities should be remarkably durable against achievable assaults. Unfortunately, this can be not the situation. Indirect accesses are found in a few IoT and business net gadgets (e.g., home switches), and, in a few cases, that they had just been overlooked by the engineers.

3.3.4. Device Operations

A quantity of attacks will be done throughout the traditional device operations. Typically, they can be secret consistent with the attacked level:

Physical (L1), e.g., jamming.

MAC (L2), e.g., MAC spoofing, etc.

Network (L3), e.g., routing attacks, IP spoofing, etc.

Higher layers, e.g., man within the middle, etc.

Regardless of the variability of ambushes, an adequately unimaginable cryptography structure will shield the framework from the greater part of them. Regardless, it should be seen that the cryptography of the payload doesn't shield the headers, e.g., MAC-level cryptography won't figure the Mac headers. As a result, it is interminably profitable to move the cryptography to the base attainable layer, to stop ambushes reliant on the discovered customer's direct.

Cryptographic strategies have a procedure cost: the more sturdy a subject is, the more computationally genuine is. In opposition to data hypothetical methodologies, cryptography achieves its security by making it impracticable for the assaulter to rework a message. This can be accomplished by adjustment the cryptography healthiness and in this way the key legitimacy time. As an outcome, keys ought to be altered as regularly as achievable. In any case, this can be not a clear errand inside the instance of IoT gadgets.

3.3.5. Device Retirement

Gadget decommissioning shouldn't speak to a security risk, clearly. Undoubtedly, it tends to be a genuine downside if gadgets aren't appropriately deleted. Depending on the cryptanalytic plans, a client gadget may keep pertinent information in its memory. Particularly, the memory may contain some patient learning and a couple of cryptanalytic keys utilized by the system. This disadvantage isn't limited to customary gadget decommissioning (e.g., devolution, deficiencies, and so forth.): it applies furthermore to lost gadgets, i.e., stolen or not found any more.

3.3.6. Eavesdropping Effects on the System

As the communicated precursor, we'll concentrate on keeping A listener. In actuality, we tend to accept that the total framework is exploitation cryptanalytic systems, and in this manner, the exclusively frail segment is the arrangement part, wherever keys territory unit consulted between the gadgets. In the event that AN assaulter is prepared to with progress decode all the setup messages, it might (given enough computational power) revamp the resulting messages. As a result, the assaulter may utilize detached (disconnected) assaults to recoup touchy learning or dynamic (on the web) assaults, for instance, to change gadget microcode by putting in malevolent code. We will concentrate on the listening in of the setup part because we tend to trust this can be the premier essential half to verify.

We won't concentrate inside the blessing paper on the consequences of elective assault assortments which will be administrated by an uninvolved listener, similar to learning connection. For a discourse of feasible security improving strategies, the user will see [29].

Threats of Perception Layer

Sensor and intelligence entrenched technologies together with RFID readers, sensors or GPS area unit underneath threat as a result of varied security flaws. Key intimidations area unit mentioned underneath:

Ridiculing: it's started with an image communicating memo sent to locator organize by the assailants. It influences it to accept its creativity mistakenly that makes it appear from the underlying supply [29]. it's all the time that this situation fallouts in the assaulter getting full access to the framework making it defenseless. Flag/Radio Sticking: it's a kind of DoS assault that it possesses the conveying between the hubs and thwarts them from human activity with every supplementary [30]. Gadget altering/Hub catching: The assaulter catches the identifier hub physically replaces the hub with their vindictive hub. this sort of assault more often than not winds up in the assaulter increasing complete administration over the caught hub and damages the system [31]. A way-based DoS Assault (PDoS): amid this kind of DoS assault, the assaulter overwhelms indicator hubs an all-encompassing separation away by flooding a multi-hop start to finish correspondence way with either replayed parcels or infused false bundles [32]. Decreased framework handiness and depletion in batteries of hubs territory unit effects of this physical assault. Hub Blackout: The assault is connected sensibly or physically to the system and it stops the common sense of system parts. Hub administrations like perusing, gathering and starting activities region unit ceased because of this assault [31].

Listening stealthily: Remote attributes of RFID framework manufacture it achievable that aggressor sniffs out the direction like Arcanum or different information spilling out of tag-to-peruser or peruser to-tag making the framework defenseless [21] [33].

Various sorts of insight level attacks area unit listed below with connected risks on security mechanisms of IoT.

3.2.2. Threats of Network Layer

Network layer that is thought because the next-generation network area unit exposed to many sorts of threats. connected threats that come back from this layer area unit listed below:

Particular Sending: In such assaults, noxious hubs don't advance a few messages and by determination drop them, ensuring that they can't proliferate later on. The assaulter WHO is at risk for concealment or change of parcels beginning from a pick couple of hubs will commonly advance the rest of the traffic not to uncover her bad behavior. There is a unit of varying sorts of particular sending assaults. In one sort, the pernicious hub will by determination drop the parcels returning from a chose hub or a bundle of hubs. this case represents a danger of DoS assault for that hub or a pack of hubs. Another kind of specific sending assault is named Disregard and Avarice. amid this kind of assault, the subverted hub unpredictably skips directing a few messages [34]. Sybil Assault: it's handled as a malevolent gadget misguidedly usurping different personalities [35]. Sybil assault [36], AN assaulter will be in extra than one spot.

3.2.3. Threats of Support Layer

Target of threats in support layer area unit chiefly knowledge storage technologies. These threats area unit mentioned below:

Altering Information: The assault appears to be at one time a person from the inside alters the information for private points of interest or business focal points of any outsider firms the information will be separated and changed just intentionally after the private [17].

DoS Assault: Comparable impacts of DoS assaults that zone unit referenced in past layers are seen amid this layer, as well; for example, it closes down the framework which finishes in detachment of the administrations.

Unapproved Access: The assaulter will just invade into the framework and harm the framework by counteracting access to the associated administrations of IoT or erasing delicate learning. Henceforth, AN unapproved access will be lethal aimed at the basis [21].

3.2.4. Threats of Application Layer

The customized facilities supported the requirements of the users area unit enclosed within the application layer; e.g. the interface that user will management devices in IoT [4]. Threats in this layer chiefly board these facilities as stated underneath:

Sniffer/Lumberjacks: Assailants will present sniffer/lumberjack programs into the framework that take fundamental information from the system traffic. the most objective of the individual is to take passwords, documents (FTP records, Email records), and Email content. Numerous conventions zone unit was helpless against sniffing [40]. Infusion: Assailants could enter code straightforwardly into the applying that is dead on the server. this can be a terribly normal assault, clear to utilize, and can cause some perilous outcomes like learning misfortune, information debasement, and absence of obligation [30]. Session Commandeering: This assault uncovers individual personalities by abusing security defects invalidation and session the executives. this sort of assault is unfathomably normal and impacts of assault region unit fundamental. With the personality of another person, assaulter will accomplish something the $64000 client will do [30]. DDoS (Disseminated Disavowal of Administration): Its guideline is that the equivalent as a result of the customary Refusal of Administration assault. Be that as it may, it's dead by numerous assailants at a steady time [21] [30].

Social Engineering: an overwhelming danger for application level wherever assailants will get information from clients by means of talks, knowing each other, and so on [4].

IV. SHODAN SECURITY ASSESSMENT

Shodan is the most popular search engine for IoT devices that are visible to the Internet. All IoT systems facing the Internet, having IP addresses and running on TCP/UDP ports can be scanned and analyzed by Shodan.

using Shodan, we conducted a study on the most vulnerable aspects of IoT devices. Results showed that many reported IoT devices vulnerabilities are related to applications running on the same IoT servers. At the end, this will make IoT systems vulnerable. Many of those vulnerabilities are related to using weak passwords or manufacture default credentials. Examples of those popular vulnerabilities:

- Access control problems
- Operating systems credentials
- File transfer protocols
- Routers, switches, and firewalls
- Web servers and remote-control applications
- Web cameras

V. CONCLUSION

In this paper, we evaluated security threats and attacks on IoT based on several categories. As part of understanding security threats and attacks and understand how to deal with them, it is necessary to understand the different elements that can impact IoT systems and environments. We focused on two main categories: IoT vulnerabilities, threats and attacks based on IoT architecture layers, and IoT vulnerabilities, threats and attacks based on IoT components. We surveyed existing IoT threats and attacks based on those categories.

REFERENCES

[1] Gubbi, J., Buyya, R., Marusic, S. and Palaniswami, M., 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems, 29(7), pp.1645-1660.

[8] Zurich, E.T.H., 2007. Anti-counterfeiting Requirements Report. Kevin Ashton, That Internet of things thing, It can be accessed at: http://www.rfidjournal.com/articles/view?4986

[2] D. Singh, G. Tripathi, A.J. Jara, A survey of Internet-of Things: Future Vision, Architecture, Challenges and Services, in Internet of Things (WF-IoT), 2014

[3] Gartner, Inc. It can be accessed at: http://www.gartner.com/newsroom/id/2905717

[4] Rolf H.Weber, "Internet of Things - New security and privacy challenges," in Computer Law and Security Review (CLSR), 2010, pp. 23-30

[5] Rodrigo Roman, Pablo Najera and Javier Lopez, "Securing the Internet of Things," in IEEE Computer, Volume 44, Number 9, 2011, pp. 51-58

[6] Friedemann Mattern and Christian Floerkemeier, ”From the Internet of Computers to the Internet of Things,” in Lecture Notes In Computer Science (LNCS), Volume 6462, 2010, pp 242-259

[7] Hui Suo, Jiafu Wan, Caifeng Zou, Jianqi Liu, Security in the Internet of Things: A Review, in Computer Science and Electronics Engineering (ICCSEE), 2012, pp. 648-651

[8] Ying Zhang, Technology Framework of the Internet of Things and Its Application, in Electrical and Control Engineering (ICECE), pp. 4109-4112

[9] Xue Yang, Zhihua Li, Zhenmin Geng, Haitao Zhang, A Multilayer Security Model for Internet of Things, in Communications in Computer and Information Science, 2012, Volume 312, pp 388-393

[10] Rafiullah Khan, Sarmad Ullah Khan, R. Zaheer, S. Khan, Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges, in 10th International Conference on Frontiers of Information Technology (FIT 2012), 2012, pp. 257-260

[11] Shi Yan-rong, Hou Tao, Internet of Things key technologies and architectures research in information processing in Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE), 2013

[12] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini and Imrich Chlamtac, ”Internet of Things: Vision, applications and research challenges,” in Ad Hoc Networks, 2012, pp.1497-1516

[13] Luigi Atzori, Antonio Iera, Giacomo Morabito, ”The Internet of Things: A Survey,” in Computer Networks, pp. 2787-2805

[14] Mr. Ravi Uttarkar and Prof. Raj Kulkarni, ”Internet of Things: Architecture and Security,” in International Journal of Computer Application, Volume 3, Issue 4, 2014

[15] Mike Burmester and Breno de Medeiros, ”RFID Security: Attacks, Countermeasures and Challenges.”

[16] Benjamin Khoo, ”RFID as an Enabler of the Internet of Things: Issues of Security and Privacy,” in IEEE International Conferences on Internet of Things, and Cyber, Physical and Social Computing, 2011

[17] Aikaterini Mitrokotsa, Melanie R. Rieback and Andrew S. Tanenbaum, ”Classification of RFID Attacks.”

[18] Lan Li, ”Study on Security Architecture in the Internet of Things,” in International Conference on Measurement, Information and Control (MIC), 2012

[19] John R. Douceur, ”The Sybil Attack,” in Peer-to-Peer Systems - IPTPS, 2002, pp. 251-260

[20] Nadeem AHmed, Salil S. Kanhere and Sanjay Jha, ”The Holes Problem in Wireless Sensor Network: A Survey,” in Mobile Computing and Communications Review, Volume 1, Number 2

[21] Tapalina Bhattasali, Rituparna Chaki and Sugata Sanyal, ”Sleep Deprivation Attack Detection in Wireless Sensor Network,” in International Journal of Computer Applications, Volume 40, Number 15, 2012

[22] Dr. G. Padmavathi, Mrs. D. Shanmugapriya, ”A survey of ATtacks, Security Mechanisms and Challenges in Wireless Sensor Networks,” in International Journal of Computer Science and Information Security, Volume 4, Number 1, 2009

[23] Priyanka S. Fulare and Nikita Chavhan, ”False Data Detection in Wireless Sensor Network with Secure Communication,” in International Journal of Smart Sensors and AdHoc Networks (IJSSAN), Volume-1, Issue-1, 2011

[24] Rabi Prasad Padhy, Manas Ranjan Patra, Suresh Chandra Satapathy, ”Cloud Computing: Security Issues and Research Challenges,” in International Journal of Computer Science and Information Technology & Security (IJCSITS).

[25] Bhupendra Singh Thakur, Sapna Chaudhary, ”Content Sniffing Attack Detection in Client and Server Side: A Survey,” in International Journal of Advanced Computer Research, Volume 3, Number 2, 2013

[26] W. Zhang, B. Qu, Security Architecture of the Internet of Things Oriented to Perceptual Layer, in International Journal on Computer, Consumer and Control (IJ3C), Volume 2, No.2 (2013)

[27] K.E. Emam, F.K. Dankar, Protecting Privacy Using kAnonymity, in Journal of the American Medical Informatics Association, Volume 15, Number 5, 2008

[28] C. Liu, Y. Zhang, J. Zeng, L. Peng, R. Chen, Research on Dynamical Security Risk Assessment for the Internet of Things inspired by immunology, in Eighth International Conference on Natural Computation (ICNC), 2012

[29] T. Karygiannis, B. Eydt, G. Barber, L. Bunn, T. Phillips, Guidelines for Securing Radio Frequency Identification (RFID) Systems, in Recommendations of National Institute of Standards and Technology

[30] Yassine MALEH and Abdellah Ezzati, ”A Review of security attacks and Intrusion Detection Schemes inWireless Sensor Networks,” in International Journal of Wireless & Mobile Networks (IJWMN), Volume 5, Number 6, 2013

[31] Z. Xu, Y. Yin, J. Wang, A Density-based Energy-efficient Clustering Algorithm forWireless Sensor Networks, in International Journal of Future Generation Communication and Networking, Volume 6, Number 1, 2013

[32] Shashank Agrawal and Dario Vieira, A survey on Internet of Things.

[33] Chen Qiang, Guang-ri Quan, Bai Yu and Liu Yang, Research on Security Issues of the Internet of Things, in International Journal of Future Generation Communication and Networking, Volume 6, Number 6, 2013, pp. 1-10

[34] Animesh Patcha, Jung-Min Park, An overview of anomaly detection techniques: Existing solutions and latest technological trends, in Computer Networks, Volume 51, Issue 2, 2007

[35] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic and Marimuthu Palaniswami, ”Internet of Things (IoT): A vision, architectural elements, and future directions.”

[36] Akour, M., Alsmadi, I., & Alazab, M. (2017). The malware detection challenge of accuracy. In 2016 2nd International Conference on Open Source Software Computing, OSSCOM 2016 [07863750] Beirut, Lebanon: IEEE, Institute of Electrical and Electronics Engineers. DOI: 10.1109/OSSCOM.2016.7863676.

[37] Roman, R., Zhou, J., and Lopez, J. (2013). On the features and challenges of security and privacy in distributed internet of things. Computer Networks, 57(10), 2266–2279.

Modeling and Simulation of Data Centers to Predict Behavior

Moises Levy
Department of Computer & Electrical Engineering & Computer Science
Florida Atlantic University. Boca Raton, FL 33431, USA
mlevy2015@fau.edu

***Abstract* – Data center modeling and simulation help to estimate and predict key parameters under certain a-priori known conditions. Data center systems must adopt end-to-end resource management, covering both cyber and physical components. This paper describes a systematic, easy-to-follow approach for data center modeling and simulation using a cyber-physical systems lens. The methodology is aimed to facilitate the estimation of quality of service, airflow and power requirements, and key indicators to assess data centers, and to then compare them to one another, or compare different scenarios under which data centers operate. The results contribute to communicate and better understand data center behavior and areas of improvement.**

Keywords – Data centers, cyber-physical systems, modeling, simulation, airflow, energy, quality of service, workload.

I. INTRODUCTION

Data centers are comprised of information technology equipment (ITE) and supporting infrastructure (e.g., power, environmental control, telecommunications, fire protection, security, and automation). With today's increasing reliance on digital information, data centers' main task is to process and store information securely, and to provide users uninterrupted access to it.

Data centers must satisfy legislation, standards, best practices, and stringent technical requirements in order to guarantee reliability, availability, performance, security, and manage risks. They are frequently provisioned with additional resources to ensure operation under different scenarios. In a data center, numerous threats can cause failures, including human error and technical issues. The impact of failure on a data center is often significant, as lack of access to information entails high costs [1]. Intangible losses, such as business confidence, lost opportunities, and reputation are difficult to quantify.

Data centers are evolving from traditional projects to 'software-defined data centers' (SDDC), where ITE services are delivered as a service, enabled mainly by virtualization and the commoditization of ITE [2]. Another new trend includes 'software-defined power (SDP)' strategies to deliver a power-aware, optimized data center [3]. Modeling and simulation results provide a quantitative explanation of tradeoffs among the different data center components.

The motivation of this paper is to describe a comprehensive data center model, validate the model, and show simulations using a cyber-physical systems perspective, meaning the integration of computational and physical components, potentially including human interaction [4]. This approach is an iterative process, since equipment may be upgraded, added, or removed. Legacy and current generation systems may be in use simultaneously throughout the data center life cycle. A cyber physical system approach for data center modeling and simulation was introduced in papers originally presented at the *IEEE CCWC 2019, the 9th IEEE Annual Computing and Communication Workshop and Conferenc*e [5] [6], for which this work is an extension. The theoretical model serves as the foundation for simulations, to predict the behavior of a hypothetical system, or one where physical measurement and experimentation is not feasible. Results help to assess strategies for end-to-end resource management and key performance indicators improvement.

II. BACKGROUND

Since the deregulation of telecommunications in the U.S. (*Telecommunications Act of 1996* [7]), a number of standards and best practices related to telecommunications, and more recently to the data center industry have emerged to make general recommendations when designing, building or operating a data center. Numerous organizations worldwide have contributed to the creation of standards, white papers, best practices and other documents related to the data centers.

Some of the main publications include the *ANSI/TIA 942-A "Telecommunications Infrastructure Standard for Data Centers"* [8], the *ANSI/BICSI-002 "Data Center Design and Implementation Best Practices"* [9], the "*Data Center Site Infrastructure Standard*" [10] [11], the standard *ANSI/ASHRAE 90.4* "*Energy Standard for Data Centers*" [12], the "*Data Center Site Infrastructure Tier Standard: Operational Sustainability*" [13], the "*Data Centre Operations Standard*" (DCOS) [14], the *BICSI-009* "*Data Center Operations and Maintenance Best Practices"*[15], the *Singapore Standard SS 564* (Green Data Centers) [16], *and the ISO/IEC 22237 "Information technology - Data Centre facilities and infrastructures"* [17] [18] [19] [20] [21] [22] [23].

Previous work on data center modeling includes work on specific areas. Work focused on cooling system components include thermal properties [24], heat transfer [25], air temperature [26], air cooling with chillers [27], air conditioning systems [28], and transient thermal behavior [29]. The work in [30] shows a cyber-physical systems approach used for energy efficiency. The work in [31] and [32] present different formulation for energy consumption, and in [33] a feedback control scheduling is explained. The work in [34] and [35] analyze workloads using dynamic resource prediction and allocation, and in [36] workload scheduling with distributed

energy is examined. While the related work is of great relevance, it is very specific and does not study the data center's overall behavior. In addition, it does not reflect in the same model a comprehensive understanding of the data centers from workload through the prediction of different parameters (e.g., quality of service, power, airflow, and key indicators).

III. Data Center Modeling

To model a data center it is common practice to fragment it into numerous subsystems, given its complexity. The performance analysis should start with the analysis of each subsystem, and later evaluate the overall behavior of the complete system.

ITE requires power. It consumes energy and generates heat, both of which vary depending on the workload required to be processed. The cooling system cools the facility and extracts the heat. IT resource management systems actively manage the workloads. ITE, power, and cooling systems interact and are controlled by dynamic management of ITE resources.

The approach is comprised of three easy-to-follow steps: (1) modeling the cyber components, (2) modeling the physical components, and (3) identifying data center key indicators.

1. Modeling the cyber components

The first step consists of identifying and modeling the components responsible for processing the workloads. These components are defined as 'cyber' components or ITE. The *workload* is the rate and type of jobs processed by the data center as a whole, or by a given piece of ITE. The performance or *processing rate* of ITE is often given in terms of floating-point operations per second (FLOPS), and represents the maximum throughput or the maximum number of jobs that can be processed by the ITE in a unit of time [37].

Workload processed affects energy consumption, which directly impacts the physical environment. A data center that is not processing workload will still consume a fixed amount of energy to maintain the availability of all the resources. As workload increases, energy consumption increases, reaching a maximum where processing time may also rise. Parameters such as workload and ITE specifications can be used to estimate quality of service, power, airflow, energy, and key performance indicators.

1.1. Quality of service.

Quality of service includes variables such as queue length, waiting time, and processing time. The power required by the ITE depends on the workload and the quality of service. A data center with poor quality of service indicators may be unsuitable for the desired purpose.

The *workload arrival rate* (W_{in}) can be interpreted as the number of jobs or instructions arriving to the data center during a given period. Let *nodes* (N) be the total number of computational nodes. Only the active nodes (n) are available at a given time for processing a workload. The *idle nodes* (n_{idle}) represent the ITE which are not processing workload. ITE occupy physical space, require power and airflow, consume energy, generate heat, and require software and maintenance to operate, even when in idle mode.

The total workload arriving at the data center varies with time j and is represented by $W_{in,DC}(j)$. The total workload is distributed among the active nodes through a distribution factor represented by $S(i,j)$. An idle node is represented as a node with no workload to be processed, in other words $S(i,j) = 0$.

The *workload arrival rate* at any active node is the product of $W_{in,DC}(j)$ and $S(i,j)$. The workload departure rate depends on the *processing rate* $PR(i)$ and the workload waiting to be processed or *queue length* $L(i,j)$.

The workload evolution at an active node i at time j is shown in Figure 1 and can be described as follows in terms of *workload arrival rate* $W_{in}(i,j)$, *workload departure rate* $W_{out}(i,j)$, *and queue* $L(i,j)$.

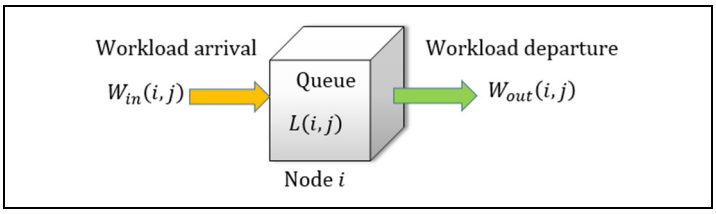


Fig. 1 Workload flow at an ITE node

The *workload arrival rate* at node i at time j considers the total workload arriving at the data center distributed among nodes:

$$W_{in}(i,j) = W_{in,DC}(j) * S(i,j) \quad (1)$$

If the ITE capacity is large enough to process the incoming workload all transactions are processed in real-time, then:

$$\big(W_{in}(i,j) + L(i,j-1)\big) < PR(i) \quad (2)$$

$$W_{out}(i,j) = W_{in}(i,j) + L(i,j-1) \quad (3)$$

$$L(i,j) = 0 \quad (4)$$

Otherwise, if the system is receiving more workload that it can process in real-time, the system is overloaded, and a queue is formed, generating congestion for the jobs to be processed, then:

$$\big(W_{in}(i,j) + L(i,j-1)\big) > PR(i) \quad (5)$$

$$W_{out}(i,j) = PR(i) \quad (6)$$

$$L(i,j) = W_{in}(i,j) + L(i,j-1) - PR(i) \quad (7)$$

The formulation represents the relationships among the *workload departure rate*, the state variable (*queue length*), and the controllable variables (*processing rate* and *workload allocation*) at node i at time j.

In summary, input workload arrives to the data center entering the queue, waits for service from a computational node, and eventually receives the service and then leaves, as shown in Figure 2. Quality of service variables are estimated permanently. The model considers open queuing and first-come-first–serve (FCFS) scheduling to simulate data center behavior, allowing external arrivals and departures of workloads.

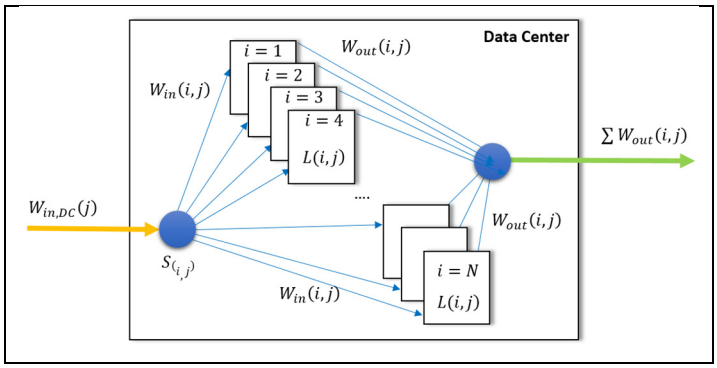


Fig. 2 Workload flow at the data center

Data center analysis is concerned with the way ITE capacity is managed over time. ITE *resource utilization* $U(i,j)$ represents the portion of ITE capacity used to process workload requests. It is estimated as the ratio of the workload processed to the processing rate, expressed in percentage, taking values between 0% and 100%:

$$U(i,j) = W_{out}(i,j)/PR(i) \quad (8)$$

The *average utilization* at time j considers utilization across all nodes.

$$U_{avg,N}(j) = (\sum_{i=1}^{N} U(i,j))/N \quad (9)$$

The idle nodes affect *average utilization* since they are not processing workloads. The *average utilization* considers only active nodes at time j:

$$U_{avg,n}(j) = (\sum_{i=1}^{n} U(i,j))/n \quad (10)$$

Likewise, the *maximum utilization* at time j considers utilization across all nodes:

$$U_{max}(j) = max(U(i,j)) \quad (11)$$

The *waiting time* or queuing time is a measurement of the workload remaining to be processed, and is estimated as the ratio of the length of the workload queue to the processing rate.

$$tw(i,j) = L(i,j)/PR(i) \quad (12)$$

The *average* and *maximum waiting time* are often related to the quality of service offered to the user. The *average waiting time* at time j across all active nodes is defined as:

$$tw_{avg}(j) = (\sum_{i=1}^{n} tw(i,j))/n \quad (13)$$

The *maximum waiting time* at time j across all active nodes is defined as:

$$tw_{max}(j) = max(tw(i,j)) \quad (14)$$

Response time may also be estimated, which is the total time that a workload spends in the system from arrival to service completion. Criteria may be established for workload migration, considering types of workloads, ITE requirements, migration policies, scheduling disciplines, and others. Further, migration of workloads to different data center sites may be considered.

Table 1 summarizes parameters for the cyber components of the model used to estimate quality of service parameters.

TABLE 1. PARAMETERS FOR CYBER COMPONENTS: QUALITY OF SERVICE

Parameter	Description
N	Number of ITE nodes available.
n	Number of active nodes available.
n_{idle}	Number of idle nodes.
$W_{in,DC}(j)$	Workload arrival rate at the data center at time$j$.
$S(i,j)$	Relative amount of workload allocated to the active node i at time j.
$W_{in}(i,j)$	Workload arrival rate at the active node i at time j.
$W_{out}(i,j)$	Workload departure rate at the active node i at time j.
$PR(i)$	Processing rate of node i.
$L(i,j)$	Queue length at node i at time j.
$U(i,j)$	Utilization of node i at time j.
$U_{avg,N}(j)$	Average utilization across all nodes at time j.
$U_{avg,n}(j)$	Average utilization across all active nodes at time j.
$U_{max}(j)$	Maximum utilization of node i at time j.
$t_w(i,j)$	Waiting time at node i at time j.
$tw_{avg}(j)$	Average waiting time at time j.
$tw_{max}(j)$	Maximum waiting time at time j.

1.2. Power

In general, processors and memory are the main power consumers for a data center server, followed by power supply loss, storage, PCI (peripheral component interconnect), motherboard, fans, and networking interconnects [38]. Processor power consumption varies by processor, workload, and power management technologies [38].

Most server workloads scale linearly from idle to maximum power. *CPU power consumption* can be estimated based on processor utilization. This approximation has shown to be true within a 5% margin of error [38].

$$P_{CPU}(i,j) = \left(P_{CPU\_max}(i) - P_{CPU\_idle}(i)\right) * U_{CPU}(i,j) + P_{CPU\_idle}(i) \quad (15)$$

Memory-related power consumption depends on the specific technology, the idle and active states, as wells as the workloads[38]. Processor and memory cooling is challenging, requiring thermal analysis, which includes power, airflow, temperature, and humidity.

Storage power consumption can be significant, depending on the quantity and technology of drives present. Average hard disk power consumption can be estimated as the weighted sum of the power required in idle, write, read, and seek states [38]. The variables w_1, w_2, w_3, and w_4 represent the respective weights, with values ranging between 0 and 1. The weights depend on the specific use, such as average office work or intensive storage operations (e.g., defragmentation, scanning the surface, copying).

$$P_S(i,j) = w_{1(i,j)} * P_{S_{idle}}(i) + w_{2(i,j)} * P_{S_{wr}}(i) + w_{3(i,j)} * P_{S_{read}}(i) + w_{4(i,j)} * P_{S_{seek}}(i) \quad (16)$$

Power supply efficiency depends on the current load. The optimal load is around 50% to 75%, efficiency drops dramatically for loads below 50%, and it does not really improve for loads higher than 75%. Power supply efficiency is typically profiled with high load factors (e.g., 80% to 90%), which may not be realistic. In a data center, *power supply efficiency loss* is considerable, since server workloads fluctuate, and servers do

not perform at full capacity most of the time [38].

Power consumption characterization for various workloads can be valuable. Studies with different servers and workloads have concluded that ITE power consumption closely follows processor utilization [38]. The *ITE power* requirement over time can be assumed linear from idle (P_{idle}) to maximum (P_{max}) power as workload increases. *ITE power* depends on the workload, and therefore is estimated based on its utilization *(U)*:

$$P_{ITE}(i,j) = \left(P_{max}(i) - P_{idle}(i)\right) * U(i,j) + P_{idle}(i) \quad (17)$$

The *power required* $P_{ITE}(j)$ by all nodes at time j is:

$$P_{ITE}(j) = \sum_{i=1}^{N} P_{ITE}(i,j) \quad (18)$$

The formulation supports the development of energy management strategies for various ITE. The *energy consumption* $E(i,j)$ at node i at time j is described in a discrete or continuous system as:

$$E(i,j) = \sum_{t=0}^{j} P_{ITE}(i,t) * \Delta t \text{ and} \quad (19)$$

$$E(i,j) = \int_{0}^{j} P_{ITE}(i,t) * dt \text{ , respectively.} \quad (20)$$

The *energy consumption* $E(j)$ by all nodes at time j is:

$$E(j) = \sum_{i=1}^{N} E(i,j) \quad (21)$$

Previously, maximum power consumption was around two times idle consumption. Newer technologies for processor designs improve this measurement to a factor of five, and are fast approaching ten, with further expected improvements [39]. This is a great achievement in terms of ITE energy efficiency, since less energy is consumed in the idle state.

Table 2 summarizes parameters for the power model required for the cyber components of a data center.

TABLE 2. PARAMETERS FOR CYBER COMPONENTS: POWER

Parameter	Description
$P_{CPU}(i,j)$	Power required by the CPU at node i at time j.
$P_{CPU\_idle}(i)$	Power required by the CPU at node i for the idle state.
$P_{CPU\_max}(i)$	Maximum power required by the CPU at node i.
$U_{CPU}(i,j)$	Utilization of the CPU at node i at time j.
$P_S(i,j)$	Power required by storage at node i at time j.
$P_{S_{idle}}(i)$	Power required by storage at node i for the idle state.
$P_{S_{wr}}(i)$	Power required by storage for writing tasks at node i.
$P_{S_{read}}(i)$	Power required by storage for reading tasks at node i.
$P_{S_{seek}}(i)$	Power required by storage for seeking tasks at node i.
$w_1(i,j)$	Weight at node i at time j for the idle state.
$w_2(i,j)$	Weight at node i at time j for writing tasks.
$w_3(i,j)$	Weight at node i at time j for reading tasks.
$w_4(i,j)$	Weight at node i at time j for seeking tasks.
$P_{ITE}(i,j)$	Power required by node i at time j.
$P_{idle}(i)$	Power required by node i for the idle state.
$P_{max}(i)$	Maximum power required by node i.
$U(i,j)$	Utilization of node i at time j.
$E_{ITE}(i,j)$	Energy consumption at node i until time j.

Variation in ITE power requirements related to the temperature of the air intake may be considered. Based on experimental results in [40], it is shown that power requirement of the server increases if the temperature of the air intake is increased. As an example, if the server inlet temperature is increased from 15 ºC to 35 ºC, an increase of the power required by the server is expected in a range of 7% to 20%.

2. Modeling the physical components

The second step consists of modeling the physical, mainly known as thermal components, including airflow and power requirements. 'Cyber' and thermal components are coupled through the energy consumption of ITE. Thermal behavior is affected by the power required by ITE, which depends on the workload processed and quality of service.

2.1. Airflow

ITE must satisfy operating conditions to guarantee their desired performance, reliability, and life expectancy. Airflow predictions involve thermodynamics concepts [41]. Most of the power drawn by ITE is dissipated as heat. In data centers, cold air from the cooling system absorbs heat generated mainly by ITE, and the warm air returns to the cooling system. The heat is then dissipated outside the facility. Figure 3 shows an airflow example, considering a data center with a raised floor and a cold/hot aisle configuration (cold air inlets of the cabinets face the same side, and hot air exhaust faces the same side).

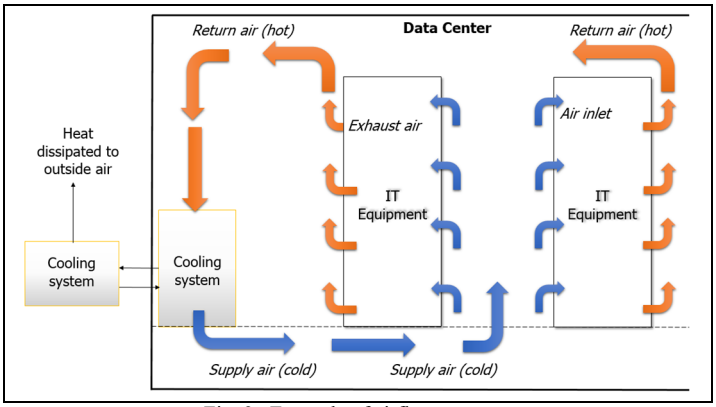


Fig. 3 Example of airflow management

The cooling system inside the data center (CRAC -*computer room air conditioner*- or CRAH -*computer room air handler*-) supplies cold air to the air inlets of the ITE cabinets through the raised floor. Exhaust (hot) air is returned to the cooling system unit for further heat exchange through the outside unit (e.g., condenser or chiller plant).

The airflow rate supplied by the cooling system usually exceeds the total airflow required by ITE. However, understanding airflow requirements for the ITE may optimize the airflow management from the cooling system. The airflow travels through the air inlet of the ITE to cool it down via convection and is exhausted as hot air. The convective heat transfer at the ITE can be described as [41]:

$$q = Cp * W * \Delta T \quad (22)$$

The *amount of heat transferred (q)* is estimated as the power drawn by the ITE. The *mass flow (W)* is the *airflow rate (Af)* multiplied by the *density of air (ρ)*. The *specific heat* (Cp) and *density of air* (ρ) are dependents of the temperature. The *temperature rise (ΔT)* is the difference between the intake air

and exhaust air temperatures. The previous equation can then be expressed at node i at time j as:

$$P_{ITE}(i,j) = Cp * Af(i,j) * \rho * \Delta T(i,j) \quad (23)$$

Therefore, manipulating the previous equation, the estimated *airflow* required is:

$$Af(i,j) = P_{ITE}(i,j)/\ (Cp * \rho * \Delta T(i,j)) \quad (24)$$

Considering a temperature of 25 ºC (or 77 ºF) and standard atmospheric pressure (1 atm.), the specific heat (Cp) is *1005 Joule/ (Kg·ºC)* and the density of air (ρ) is *1.184 Kg/m³*.

The airflow requirement (in cfm -cubic feet per minute-) is estimated as a constant multiplied by the ratio of the power of ITE and the temperature rise:

$$Af(i,j)_{\text{cfm}} = 1.78\ * P_{ITE}(i,j)\ /\ \Delta T(i,j)_{\circ\text{C}} = 3.20 * P_{ITE}(i,j)\ /\ \Delta T(i,j)_{\circ\text{F}} \quad (25)$$

If the temperature rise is unknown, it may be estimated at 20 ºC, to guarantee the reliable performance of the components [42], which results in approximately 9 cfm per 100 W of heat generated. The best practice to estimate the airflow is using real-time measurements for power and temperature rise.

The *total airflow* required by ITE in a data center must include all the nodes, and can be described as:

$$Af(j) = \sum_{i=1}^{N} P_{ITE}(i,j)/(Cp * \rho * \Delta T(i,j)) \quad (26)$$

$$Af(j)_{\text{cfm}} = \sum_{i=1}^{N} 1.78 * P_{ITE}(i,j)/\ \Delta T(i,j)_{\circ\text{C}} = \sum_{\text{i=1}}^{\text{n}} 3.20 * \text{P}_{\text{ITE}}(\text{i,j})/\ \Delta T(i,j)_{\circ\text{F}} \quad (27)$$

Table 3 summarizes the model parameters for airflow of the cooling system.

TABLE 3. PARAMETERS FOR THERMAL COMPONENTS: AIRFLOW

Parameter	Description
q	Amount of heat transferred (W).
Cp	Specific heat of air (Joule/ (Kg * ºC)).
ρ	Density of air (Kg / m³).
W	Mass flow (Kg / min).
ΔT	Temperature rise of air (°C or °F).
$\Delta T(i,j)$	Temperature rise of air at node i at time j (°C or °F).
$P_{ITE}(i,j)$	Power required (W) by node i at time j.
$Af(i,j)$	Airflow (cfm) through the air inlet of the ITE at node i at time j.
$Af(j)$	Airflow (cfm) required by all ITE at time j.

The formulation helps to identify opportunities to optimize airflow management and the cooling system. As the heat generated by ITE increases, the airflow rate may also increase to maintain temperature requirements. Most new ITE have variable-speed fans with control algorithms dependent on the utilization of the resource [43].

For the purpose of the model, it is assumed that there is no mixing between cold air and hot air, or that the mixing can be neglected, as it will not affect the temperature of the return air to the cooling system. In case the previous assumption does not hold, the temperature of the return air may be estimated considering the airflow management through computational fluid dynamics with the desired simplifications.

2.2. Power

The total cooling capacity of a cooling system can be expressed as the sum of sensible and latent heat removed. The cooling load in data centers is mainly sensible heat, generated by ITE, lights, and motors. Latent heat can be dismissed, as there are few people inside the data center and limited outside air. The sensible heat ratio is the ratio of sensible cooling to total cooling. On a scale from 0 to 1, it usually takes high values, ranging between 0.9 and 1. This is one of the main reasons for using precision cooling systems, designed for highly sensible heat ratios, as opposed to comfort cooling systems. In addition, precision cooling systems operate at higher airflow, satisfy strict temperature and humidity controls, and run continuously [44][45].

The majority of the power consumption in the cooling system stems from compressors (in the chillers or CRAC units), chilled and cooling tower water pumps, cooling tower or heat exchange blowers, and air handling unit blowers [46].

The *sensible coefficient of performance (SCOP)* is used to estimate the power requirements of the cooling system. The $SCOP$ is the ratio of net sensible cooling capacity to the power required to produce that cooling (excluding reheat and humidifier) [47] [48]. The $SCOP$ values for commercial precision cooling systems without economizers usually range from 1.8 to 3.8 [49].

The $SCOP$ is also dependent on the technical characteristics of the cooling system and the temperature of the cold air supplied. For example, for a specific water chilled CRAC unit where T_C is the temperature of the supplied air [50], it can be modeled as:

$$SCOP = 0.0068 * {T_C}^2 + 0.0008 * T_C + 0.458 \quad (28)$$

The amount of power required by the cooling system (P_{AC}) is the ratio of total heat loads to $SCOP$. The total heat loads are the sum of all power delivered to the ITE, lighting, and electrical distribution losses. For the sake of explanation, since power needs are time-dependent, we assume time j, although it is not indicated for each one of the terms in the following equations.

$$P_{AC} = (\sum P_{ITE} + \sum P_{Lighting} + \sum P_{Losses})/SCOP \quad (29)$$

Lighting power ($P_{Lighting}$) is dependent on the lighting power density, technology, and controls. It can be assumed as 1% of the total *power required by the ITE* (P_{ITE}) [51] [52].

$$\sum P_{Lighting} = 0.01 * \sum P_{ITE} \quad (30)$$

Electrical distribution systems are assumed to have a constant loss of 2% of the *total power required by ITE* (P_{ITE}). If a UPS is present, 2% is added to UPS losses [51]. The *UPS losses* are one minus the *UPS efficiency* (η_{UPS}), and depend on factors such as the UPS technology type, size, voltage, and load factor. UPS efficiency usually ranges from 65% to 97% [52]. *Electrical distribution system losses* (P_{Losses}) are defined as:

$$\sum P_{Losses} = (0.02 + (1 - \eta_{UPS})) * \sum P_{ITE} \quad (31)$$

Therefore, considering ITE, lighting, and electrical distribution losses, the amount of *power required by the cooling system* can be estimated as:

$$P_{AC} = (2.03 - \eta_{UPS}) * \sum P_{ITE} / SCOP \quad (32)$$

As a simplification, if the total heat load is considered only as the sum of all power required for the ITE, then:

$$P_{AC} = \sum P_{ITE} / SCOP \quad \text{or} \quad (33)$$

$$P_{AC}(j) = \sum_{i=1}^{N} P_{ITE}(i,j) / SCOP \quad (34)$$

In addition, the cooling system must consider the amount of airflow required by ITE based on workloads, utilization, and power requirements. The affinity laws for fans are used to estimate the power required by similar fans in relation to airflow [53]. The first one states that the airflow is proportional to fan speed:

$$Af_{FAN\_1}/Af_{FAN\_2} = N_{FAN\_1}/N_{FAN\_2} \quad (35)$$

And the second states that the power is proportional to the cube of the fan speed:

$$P_{FAN\_1}/P_{FAN\_2} = \left(N_{FAN\_1}/N_{FAN\_2}\right)^3 \quad (36)$$

Then, the fan power requirement is proportional to the cube of the airflow supplied:

$$P_{FAN\_1}/P_{FAN\_2} = \left(Af_{FAN\_1}/Af_{FAN\_2}\right)^3 \quad (37)$$

These estimations lead to energy management strategies when various fans are present in the data center. Consider a data center with just one CRAH unit. If the airflow required by the ITE is reduced by half, the power required by the CRAH unit is reduced by a factor of 8. Further, the data center usually has more than one CRAH unit, and the airflow required by the ITE may be supplied by multiple units, instead of just one unit.

Data center layout, cooling system, and ITE must be considered to understand airflow management requirements. Different strategies such as air economizers or free cooling may be implemented to reduce the power required for the cooling system [43].

Table 4 summarizes the model parameters for the cooling systems.

TABLE 4. PARAMETERS FOR THERMAL COMPONENTS: POWER

Parameter	Description
SCOP	Sensible Coefficient of Performance for the cooling system.
T_C	Temperature of the supplied air (or cold air) by the cooling system (°C).
P_{AC}	Power required by the cooling system (W).
P_{ITE}	Power required by ITE(W).
$P_{Lighting}$	Power required by lighting (W).
P_{Losses}	Electrical distribution system losses (W).
η_{UPS}	Efficiency of the UPS.
Af_{FAN}	Airflow capacity of the fan (cfm).
N_{FAN}	Speed of the fan (rpm).
P_{FAN}	Power required by the fan (W).

3. Identifying data center key indicators.

The last step consists of recognizing key indicators for the data center. Several metrics can help to examine efficiency, productivity, sustainability, operations, and risk, indicators for which the user must determine an acceptable level [54] [55]. For the sake of the explanation, the *Power Usage Effectiveness (PUE)* metric is used as an example of key indicator, since it is one of the most widely used energy efficiency metrics. Figure 4 shows the energy flow in a data center.

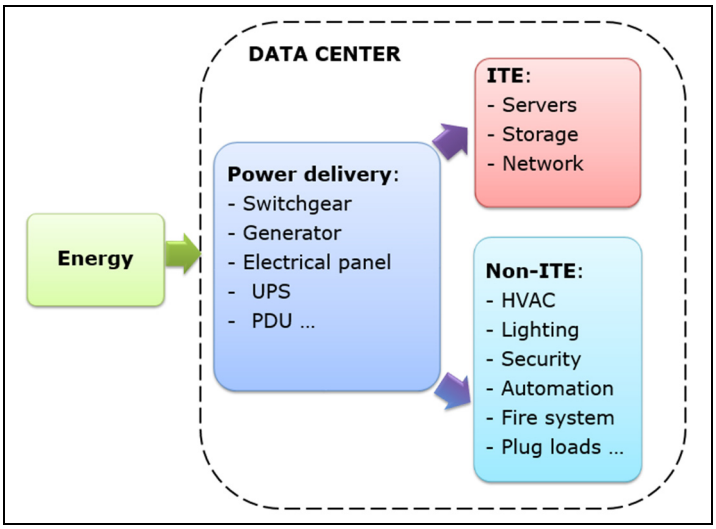


Fig. 4 Data center energy flow diagram

Right-sizing the data center based on modeling, simulation and projections of ITE utilization may not be accurate. Also, specific needs may change rapidly. Real-time monitoring of the data center may help to calibrate and validate models and simulations. This is made possible given that data center monitoring and management systems collect data (e.g., power delivered for ITE, power required by the total facility, airflow, ITE utilization) in real-time [56] [57]. Gathering the relevant data and understanding its nature is more important than simply collecting more data [58].

One of the most widely used energy efficiency metrics is *Power Usage Effectiveness (PUE)* [59], defined as the ratio of energy used in a facility to energy delivered to ITE. The ideal PUE is one. In terms of the average power required by the data center (P_{DC}) and the ITE (P_{ITE}), using the same measurement period, it can be expressed as:

$$PUE = P_{DC} / \sum P_{ITE} \quad (38)$$

The power required to operate a data center can be simplified as the summation of the power required by the ITE and the cooling system. Electrical distribution system losses and other systems (e.g., lighting) are neglected.

The $DCiE$ metric can be expressed in terms of power required by the ITE (P_{ITE}) and the cooling system (P_{AC}) as:

$$PUE = (\sum P_{ITE} + \sum P_{AC}) / \sum P_{ITE} \quad (39)$$

Considering a cooling system comprising CRAC units, the $DCiE$ metric can be expressed as a function of the *sensible coefficient of performance (SCOP)* of the cooling system (CRAC), the power required by the ITE P_{ITE} and the cooling system P_{CRAC}:

$$PUE = (\sum P_{ITE} + \sum P_{CRAC}) / \sum P_{ITE} \quad (40)$$
$$= 1 + 1 / SCOP$$

Another scenario may consider a cooling system comprising a chiller water plant and CRAH units, the PUE metric can be expressed as as a function of the *sensible coefficient of performance (SCOP)* of the cooling system (chiller), the power required by the ITE P_{ITE}, the chiller water plan $P_{Chiller}$ and the CRAH units (fans) P_{Fan}:

$$DCiE = \sum P_{ITE} / (\sum P_{ITE} + \sum P_{Chiller} + \sum P_{Fan}) \quad (41)$$
$$= \sum P_{ITE} / (\sum P_{ITE} + \sum P_{ITE} / SCOP + \sum P_{Fan})$$

The behavior of the $SCOP$ is nonlinear and usually decreases with lower temperatures. To supply colder air, the cooling system consumes more energy. Therefore, as we increase the temperature of the cold air supplied by the cooling system, the $SCOP$ increases, and the PUE decreases. There are limits imposed by climate analysis, cooling system type, ITE requirements (e.g., ASHRAE temperature range and maximum rate of change recommendations [40] [60] [61]), and airflow management.

IV. Model Validation

It is important to validate and understand the accuracy of the formulation for power and airflow requirements, which are relevant for data center design. Information from a computer manufacturer was used to validate the model formulation. It is worth noting that this information may not be available for all ITE from all manufacturers, in which case proper experiments should be developed if needed.

Two different rack servers are selected for comparison, the DELL PowerEdge R740 [62] with maximum power of 271 W and idle power of 104 W, and the DELL PowerEdge R940xa [63] with maximum power of 563 W and idle power of 165 W, both with the temperature set to 25 ºC. Data from the manufacturer for these rack servers was gathered through an infrastructure planning tool [64] [65].

Information provided by the manufacturer for the ITE includes maximum power (P_{max}), idle power (P_{idle}), ambient temperature, and for different values of utilization (U) the input power and airflow requirements, and the temperature rise (ΔT). The input power (P_{ITE}) and airflow (A_f) requirements are also estimated at different values of utilization (U) using model formulation (equations 17 and 27).

Figures 5 and 6 show the different parameters from the manufacturer and the formulation model. Data from the manufacturer is compared against the results from the formulation model, and their respective absolute margin of error is calculated. For the ITE mentioned, the model is accurate within a 20% margin of error, and with greater precision (less than 7% margin of error) if the utilization is greater than 50%.

Rack Server from DELL: PowerEdge R740
P max (W): 271 P idle (W): 104
Temp (oC): 25

	Data from maufacturer			Formulation from model			
% Utilization	Input Power (W)	Airflow (CFM)	Temp. rise (oC)	Input Power (W)	Error	Airflow (CFM)	Error
0%	104.00	33.10	5.7	104.00	0.0%	32.48	1.9%
10%	143.00	39.90	6.5	120.70	15.6%	33.05	17.2%
20%	166.00	43.00	7.0	137.40	17.2%	34.94	18.7%
30%	177.00	44.30	7.2	154.10	12.9%	38.10	14.0%
40%	190.00	45.80	7.5	170.80	10.1%	40.54	11.5%
50%	199.00	46.80	7.6	187.50	5.8%	43.91	6.2%
60%	207.00	47.80	7.8	204.20	1.4%	46.60	2.5%
70%	222.00	49.50	8.1	220.90	0.5%	48.54	1.9%
80%	237.00	51.10	8.3	237.60	0.3%	50.96	0.3%
90%	254.00	53.40	8.6	254.30	0.1%	52.63	1.4%
100%	271.00	55.30	8.8	271.00	0.0%	54.82	0.9%

Fig. 5 Parameters for DELL PowerEdge R740

Rack Server from DELL: PowerEdge R940xa
P max (W): 563 P idle (W): 165
Temp (oC): 25

	Data from maufacturer			Formulation from model			
% Utilization	Input Power (W)	Airflow (CFM)	Temp. rise (oC)	Input Power (W)	Error	Airflow (CFM)	Error
0%	165.00	106.10	2.8	165.00	0.0%	104.89	1.1%
10%	230.00	109.80	3.8	204.80	11.0%	95.93	12.6%
20%	288.00	122.30	4.2	244.60	15.1%	103.66	15.2%
30%	313.00	127.10	4.4	284.40	9.1%	115.05	9.5%
40%	344.00	132.50	4.7	324.20	5.8%	122.78	7.3%
50%	363.00	136.00	4.8	364.00	0.3%	134.98	0.7%
60%	383.00	139.20	5.0	403.80	5.4%	143.75	3.3%
70%	418.00	144.50	5.2	443.60	6.1%	151.85	5.1%
80%	453.00	149.40	5.5	483.40	6.7%	156.45	4.7%
90%	503.00	155.70	5.8	523.20	4.0%	160.57	3.1%
100%	563.00	162.30	6.2	563.00	0.0%	161.64	0.4%

Fig. 6 Parameters for DELL PowerEdge R940xa

V. Data Center Simulation

Simulation models help predict the behavior of the different components of a data center, and may assist operators to attain the desired data center performance. Simulations help understand tradeoffs between the data center's performance and costs.

After the formulation and implementation of the simulation model (e.g., MATLAB or Simulink tools) [6], the next steps include the design of experiments and analysis of simulation results. Simulations show the diverse quality of service parameters, utilization of ITE, power and airflow requirements, energy consumption, and as well other values as needed. The quality of service shows how the workload is being processed. In general, lower values represent higher performance. ITE specifications such as processing rate, power idle and power maximum, are usually related to the generation and the cost of ITE. The power requirements of ITE and airflow affect the cooling system performance. The energy consumption is mainly related to the operational expenses.

Depending on the needs, different hypothetical data center scenarios may be considered including various workload and ITE. The first challenge is to understand the type and the duration of the workload. Figure 7 shows examples of three different input signal types (normal, constant or random) to simulate the workload behavior.

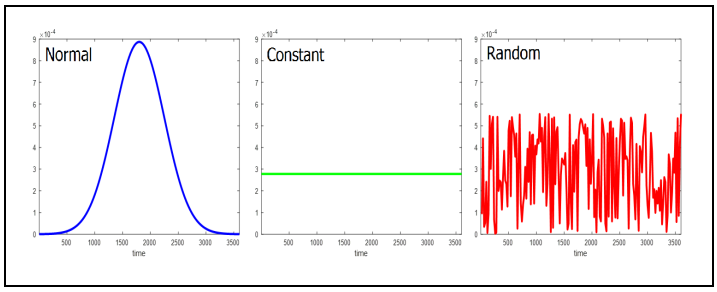


Fig. 7 Workload input signal types

The parameters of the workload input signal following a normal or Gaussian function are defined as:

$$f(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(t-\mu)^2}{2\sigma^2}} \quad (42)$$

Where $1/(\sigma\sqrt{2\pi})$ represents the height of the curve's peak, μ is the position of the center of the peak, and σ controls the width of the curve. The parameters μ and σ are user-defined. For the simulations values of $\mu = T_f/2$ and $\sigma = T_f/8$ are assumed, where T_f represents the duration of the workload input.

For the constant signal, the constant function with a value of $1/T_f$ can be used. And for the random signal, a uniform random number function between 0 and $2/T_f$ can be used.

The different coefficients for the workload input signals are established so the integral of these functions between 0 and Tf is approximately one. This is multiplied by a scale factor to achieve the desired total workload input. Consequently, the simulations can consider the same total workload for different workload input signal types.

As an example; consider a simulation with the following parameters: two nodes, workload input signal as a Gaussian function for one hour, sampling rate of one second, input scaler equal to 250, and uneven relative amount of workload among nodes (S = 30% , 70%). Identical power specifications for ITE resources (P_{idle}= 50 W , P_{max}= 200 W), but different processing rates (PR = 50 , 80 jobs/s). Different graphs can be obtained through the simulations to understand the behavior. Figure 8 shows the behavior of workload input versus workload output and queues. At the beginning, the ITE are able to process all the workload input in real-time, so the workload output follows the workload input. After, the system is overloaded, the ITE is receiving more workload than it can process in real-time, generating congestion for the jobs to be processed, and a queue is formed. The queue is different for each ITE, since they are processing 30% and 70% of the workload input respectively, and at different processing rates. Figure 9 shows the workload output and utilization per computational node. Figure 10 shows the waiting time per computational node, and the average and maximum waiting time system-wide. Lastly, Figure 11 shows the overall power and airflow requirements for ITE, and the energy consumption by ITE.

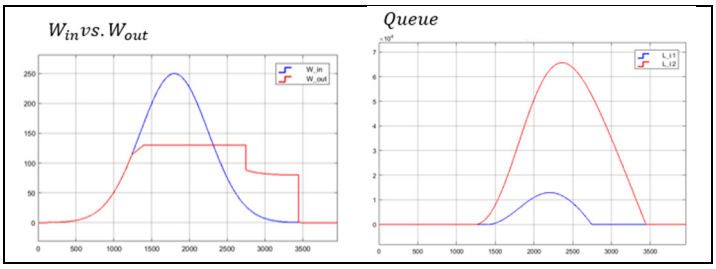


Fig. 8 Workload and queue graphs.

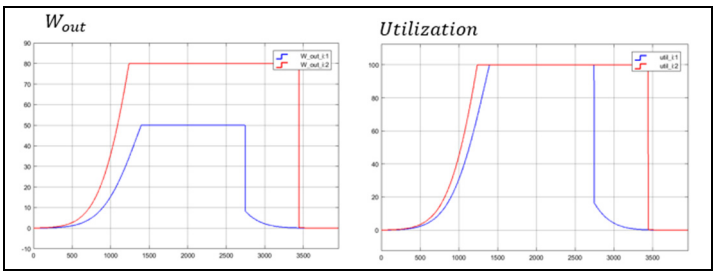


Fig. 9 Workload output and utilization graphs.

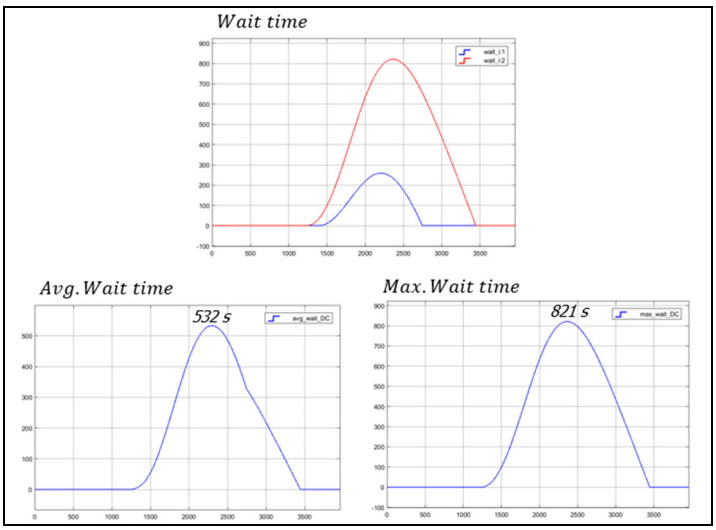


Fig. 10 Waiting time graphs.

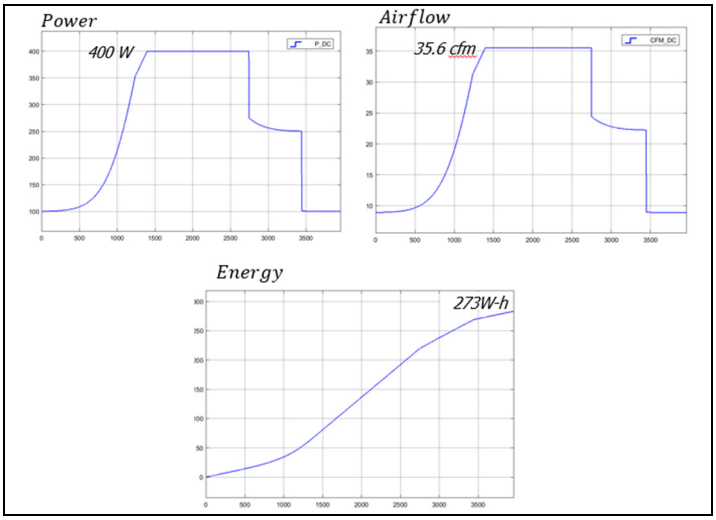


Fig. 11 Power, airflow and energy graphs.

To better understand the behavior of the data center under different conditions, multiple scenarios may be generated using the simulation model. For example, different number of nodes (1 to 50), different total workload input (10^5 to 10^7 jobs/s) can be considered, a workload input with the Gaussian function for one hour, and equal node distribution. Identical specifications for ITE resources: P_{idle}= 50 W, P_{max}= 200 W, and PR= 50 jobs/s. Figure 12 shows the results through 3D graphs. The different scenarios consider the total workload, the number of nodes, and the other axis, which is the total run time, the total energy consumption or the maximum waiting time.

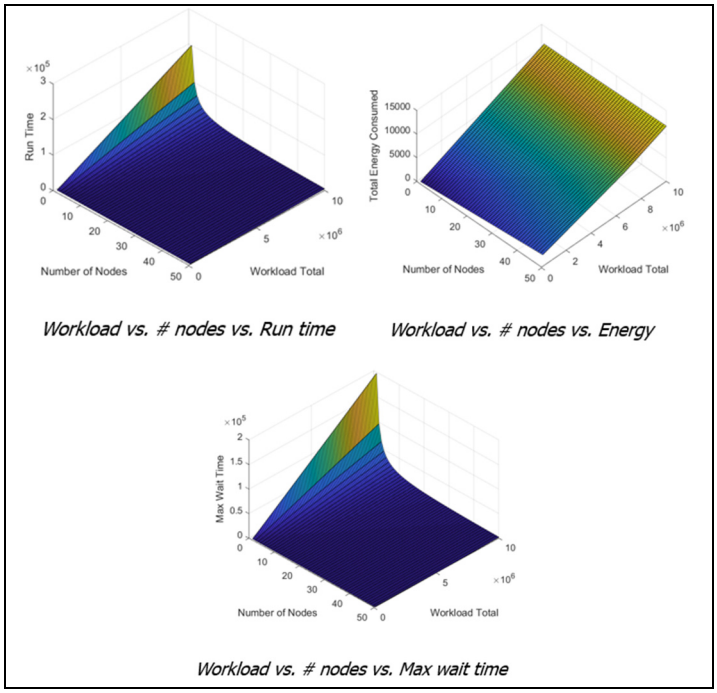


Fig. 12 Power, airflow and energy graphs.

High workloads with high number of ITE, result in good values for quality of service (e.g., low values for run time and maximum waiting time) but at higher costs (e.g., more ITE, as well as greater power and airflow requirements). High workloads and low number of ITE result in worse values for the quality of service parameters (e.g., high values for run time and maximum waiting time), at around the same operational costs (e.g., energy consumption) but with fewer ITE (lower capital expenditures).

The simulations described for the different hypothetical scenarios shows quantitatively that a data center at a certain time may be ideal given some premises, such as workloads, node distribution and ITE; however, when conditions change, performance is affected and the same data center may not be optimal.

VI. Conclusions

Modeling contributes to better understanding data center behavior, tradeoffs and impacts of under different assumptions. The model and simulations described, using a cyber-physical systems lens, involves a comprehensive view of a data center. Depending on each specific data center configuration and operations, the model assumptions may need to be revised. Results help to identify strategies to improve end-to-end resource management and key performance indicators.

Since right-sizing the data center based on modeling and simulations may not be accurate, and specific needs may change rapidly, real-time monitoring of the data center may help to calibrate and validate models.

VII. Future Research

Future work may consider calibrating and validating the theoretical model with real data from different manufacturers and experiments, for numerous ITE. It may also reflect different assumptions for the model.

Data center systems must include end-to-end resource management, covering both the cyber and the physical components. Multi-objective optimization, including quality of service and energy consumption is a desirable goal, since the requirements of data center components change continuously with time. There are currently no simulation solutions available that consider a cyber-physical approach, which also include security and risk assessment. New simulation tools are needed to support these capabilities.

References

[1] Ponemon Institute LLC, "Cost of data center outages. Data center performance benchmark series," 2016.
[2] Dell Inc., "The software-defined data center: a paradigm shift in data center deployment. White paper," 2015.
[3] BNP Media Inc, "The time is now for software defined power," *Mission Critical. Data center and mission-critical solutions*, vol. 11, no. 3. pp. 20–23, 2018.
[4] National Science Foundation, "Cyber-physical systems (CPS)." [Online]. Available: https://www.nsf.gov/publications/pub_summ.jsp?ods_key=nsf18538&org=NSF. [Accessed: 10-May-2019].
[5] M. Levy, D. Raviv, and J. O. Hallstrom, "Data center modeling using a cyber-physical systems lens," *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. Las Vegas, NV, 2019.
[6] M. Levy, D. Raviv, and J. Baker, "Data center simulations deployed in MATLAB and simulink using a cyber-physical systems lens," *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. Las Vegas, NV, 2019.
[7] The Senate of the United States. 104th Congress (1995-1996), *Telecommunications Act*. 1996.
[8] "Telecommunications infrastructure standard for data centers," ANSI/TIA-942-B, 2017.
[9] "Data center design and implementation best practices," ANSI/BICSI 002, 2019.
[10] P. Turner, J. H. Seader, V. Renaud, and K. G. Brill, "Tier classifications define site infrastructure performance," Uptime Institute, 2006.
[11] "Data center site infrastructure tier standard: topology," Uptime Institute, 2018.
[12] "Energy standard for data centers," ANSI/ASHRAE 90.4, 2016.
[13] "Data center site infrastructure tier standard: operational sustainability," Uptime Institute, 2014.
[14] "DCOS - Data Centre Operations Standard," EPI, 2016.
[15] "Data center operations and maintenance best practices," BICSI 009, 2019.
[16] "Green data centres – Part 1: energy and environmental management systems," Singapore Standards SS 564, 2013.
[17] "Information technology - Data centre facilities and infrastructures. Part 1: general concepts," ISO/IEC TS 22237-1, 2018.
[18] "Information technology - data centre facilities and infrastructures. Part 2-1: building construction," ISO/IEC TS 22237-2, 2018.
[19] "Information technology - Data centre facilities and infrastructures. Part 3: power distribution," ISO/IEC TS 22237-3, 2018.

[20] "Information technology - Data centre facilities and infrastructures. Part 4: environmental control," ISO/IEC TS 22237-4, 2018.

[21] "Information technology - Data centre facilities and infrastructures. Part 5: telecommunications cabling infrastructure," ISO/IEC TS 22237-5, 2018.

[22] "Information technology - Data centre facilities and infrastructures. Part 6: security systems," ISO/IEC TS 22237-6, 2018.

[23] "Information technology - Data centre facilities and infrastructures. Part 7: management and operational information," ISO/IEC TS 22237-7, 2018.

[24] S. U. R. Malik, K. Bilal, S. U. Khan, B. Veeravalli, K. Li, and A. Y. Zomaya, "Modeling and analysis of the thermal properties exhibited by cyberphysical data centers," *IEEE Syst. J.*, vol. 11, no. 1, pp. 163–172, 2017.

[25] V. López and H. F. Hamann, "Heat transfer modeling in data centers," *International Journal of Heat and Mass Transfer*, vol. 54, no. 25–26. pp. 5306–5318, 2011.

[26] R. Ghosh and Y. Joshi, "Dynamic reduced order thermal modeling of data center air temperatures," *Proceedings of the ASME InerPACK*. Portland, OR, pp. 423–432, 2011.

[27] T. J. Breen, E. J. Walsh, and J. Punch, "From chip to cooling tower data center modeling: part I. Influence of server inlet temperature and temperature rise across cabinet," 2010.

[28] R. Anton, H. Jonsson, and B. Palm, "Modeling of air conditioning systems for cooling of data centers," in *ITherm 2002. Eighth Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, 2002, pp. 552–558.

[29] G. Varsamopoulos, M. Jonas, J. Ferguson, J. Banerjee, and S. K. S. Gupta, "Using transient thermal models to predict cyberphysical phenomena in data centers," *Sustain. Comput. Informatics Syst.*, vol. 3, no. 3, pp. 132–147, 2013.

[30] L. Parolini, B. Sinopoli, B. H. Krogh, and Z. Wang, "A cyber–physical systems approach to data center modeling and control for energy efficiency," in *Proceedings of the IEEE*, 2012, vol. 100, no. 1, pp. 254–268.

[31] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: a survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1. pp. 732–794, 2016.

[32] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, *A taxonomy and survey of energy-efficient data centers and cloud computing systems*, vol. 82. Elsevier Inc., 2010.

[33] X. Zhao *et al.*, "Feedback control scheduling in energy-efficient and thermal-aware data centers," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 1. pp. 48–60, 2016.

[34] F. Tseng, X. Wang, L. Chou, H. Chao, S. Member, and V. C. M. Leung, "Dynamic resource prediction and allocation for cloud data center using the multiobjective genetic algorithm," *IEEE Systems Journal*, vol. PP, no. 99. pp. 1–12, 2017.

[35] A. M. Al-Qawasmeh, S. Pasricha, A. A. Maciejewski, and H. J. Siegel, "Power and thermal-aware workload allocation in heterogeneous data centers," *IEEE Transactions on Computers*, vol. 64, no. 2. pp. 477–491, 2015.

[36] H. T. Minh and M. Samejima, "An evaluation of job scheduling based on distributed energy generation in decentralized data centers," *2015 IEEE International Conference on Systems, Man, and Cybernetics*. Kowloon, pp. 1172–1177, 2015.

[37] H. Kobayashi and B. L. Mark, *System modeling and analysis: foundations of system performance evaluation*. Prentice Hall Press, 2009.

[38] L. Minas and B. Ellison, *Energy efficiency for information technology: how to reduce power consumption in servers and data centers*. Intel Press, 2009.

[39] *Datacom equipment power trends and cooling applications*, 2nd ed. Atlanta, GA: American Society of Heating Refrigerating and Air-Conditioning Engineers (ASHRAE), 2012.

[40] ASHRAE Technical Committee 9.9, "2011 thermal guidelines for data processing environments – Expanded data center classes and usage guidance," 2011.

[41] ASHRAE, "ASHRAE handbook: fundamentals (SI)," 2001.

[42] D. Moss, "Guidelines for assessing power and cooling requirements in the data center." Dell Power Solutions, pp. 62–65, 2005.

[43] ASHRAE Technical Committee 9.9, "Data center power equipment thermal guidelines and best practices. White paper," 2016.

[44] APC, "Why do I need precision air conditioning? White paper," 2001.

[45] Emerson Network Power, "Precision versus comfort cooling. Choosing a cooling system to support business-critical IT environments. White paper," 2010.

[46] C. D. Patel, R. K. Sharma, C. E. Bash, and M. Beitelmal, "Energy flow in the information technology stack: coefficient of performance of the ensemble and its impact on the total cost of ownership," Hewlett-Packard Development Company, 2006.

[47] "2016 standard for performance rating of computer and data processing room air conditioners," Air Conditioning Heating and Refrigeration Institute AHRI 1360, 2016.

[48] "Method of testing for rating computer and data processing room unitary air conditioners," ANSI/ASHRAE Standard 127-2012, 2012.

[49] U.S. Department of Energy, "DOE's Compliance certification database," 2018. [Online]. Available: https://www.regulations.doe.gov/certification-data/CCMS-4-Air_Conditioners_and_Heat_Pumps_-_Computer_Room_Air_Conditioners.html . [Accessed: 10-May-2019].

[50] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making scheduling 'cool': temperature-aware workload placement in data centers," *USENIX Annual Technical Conference*. Anaheim, CA, pp. 61–74, 2005.

[51] Ernest Orlando Lawrence Berkeley National Laboratory, "Data center profiler (DC Pro) tool: user's manual," 2016.

[52] Integral Group, "'DC Pro' data center on-line profiling tool. Calculation reference manual," 2014.

[53] "Laboratory methods of testing fans for aerodynamic performance rating," ANSI/AMCA Standard 210. ANSI/ASHRAE Standard 51, 1999.

[54] M. Levy and D. Raviv, "A novel framework for data center metrics using a multidimensional approach," *15th LACCEI International Multi-Conference for Engineering, Education, and Technology: Global Partnerships for Development and Engineering Education*. Boca Raton, FL, 2017.

[55] M. Levy and D. Raviv, "An overview of data center metrics and a novel approach for a new family of metrics," *Advances in Science, Technology and Engineering Systems Journal*, vol. 3, no. 2. pp. 238–251, 2018.

[56] M. Levy and J. O. Hallstrom, "A new approach to data center infrastructure monitoring and management (DCIMM)," *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*. Las Vegas, NV, 2017.

[57] M. Levy and J. O. Hallstrom, "A reliable, non-invasive approach to data center monitoring and management," *Advances in Science, Technology and Engineering Systems Journal*, vol. 2, no. 3. pp. 1577–1584, 2017.

[58] J. Koomey and J. Stanley, "The science of measurement: improving data center performance with continuous monitoring and measurement of site infrastructure," 2009.

[59] The Green Grid, "PUE™: A comprehensive examination of the metric. White paper # 49," 2012.

[60] ASHRAE Technical Committee 9.9, "Data center networking equipment – Issues and best practices," 2012.

[61] ASHRAE Technical Committee 9.9, "Data center storage equipment – Thermal guidelines, issues, and best practices," 2015.

[62] Dell Inc., "Dell EMC PowerEdge R740. Installation and service manual," 2018.

[63] Dell Inc., "Dell EMC PowerEdge R940xa. Installation and service manual," 2019.

[64] Dell Inc., "Dell Enterprise Infrastructure Planning Tool." [Online]. Available: http://dell-ui-eipt.azurewebsites.net/#/. [Accessed: 10-May-2019].

[65] Dell Inc., "Enterprise infrastructure planning tool user guide," 2018.

The Analysis of Sub-Communities Behavior in Social Networks

Izzat Alsmadi
Texas A&M, San Antonio, Department of Computing and Cyber Security, San Antonio, TX 78224
Email: ialsmadi@tamusa.edu

Chuck Easttom
Capitol Technology University
Email: wceasttom@captechu.edu

Abstract—Clique relations are useful in understanding the dynamics of a wide range of social interactions. One application of studying clique relations involves studying how such "detection of abnormal cliques' behaviors" can be used to detect sub-communities' behaviors based on information from Online Social Networks (OSNs).
In social networks, a clique represents a sub-group of the larger group in which every member in the clique is directly associated with every other member in the clique. Those cliques often possess a containment relation with each other where large cliques can contain small size cliques. Thus, finding the extent of the clique, or the maximum clique is an important research questions. In our approach, we evaluated adding the weight factor and integrating graph theory to clique algorithm in order to derive more data about the clique. In this regard clique activities are not like those in group discussions where an activity is posted by one user and is visible by all others. Our algorithm calculates the overall weight of the clique based on individual edges. Users post frequent activities. Their clique members, just like other entities, may or may not interact with all those activities.

Index Terms—OSN groups detection; Weighted cliques; Groups collaboration; Graph Theory and Social Media

I. Introduction

There are a diverse array of applications for tracking and measuring relationships in Online Social Networks (OSN's) (Stieglitz, et al., 2015; Papadopoulos, Kompatsiaris, Vakali, & Spyridonos, 2012). Marketing is a common application of social network analysis (Soares, et al., 2012). Another application is to study the influence of emotionally charged information such as social or political memes (Kramer, Guillory, & Hancock, 2014). However, social network relationships are also important for investigative applications (Duijn & Klerks, 2014). Whether the investigating agency is a law enforcement agency or an intelligence gathering agency the needs for an investigative application are identical. Counterterrorism is one area of investigation in which analysis of social media interactions can be very important and produce significant, actionable intelligence (Choudhary & Singh, 2015; Ishengoma, 2014; Kirby, 2007; Leistedt, 2013).

There are diverse approaches to analyzing social media interactions, regardless of the investigative intent (Alhajj & Rokne, 2014; Scott, 2017). Analytical approaches can be formulated based on diverse foundations. In some case large scale data scraping and mining is the preferred approach. This can be coupled with statistical analysis of the interactions. Graph theory provides a modality for studying a wide range of interactions (Easttom, 2018). Thus, it is appropriate to apply graph theory to analyzing social network relationships (Mitrou, Kandias, Stavrou, & Gritzalis, 2014; Zafarani, Abbasi, & Liu, 2014).

The approach we used in this current study was to focus on cliques in social media, and to apply graph theory to analyze those relationships. In the social sciences a clique is defined as a sub-group that has intergroup interaction coupled with some common interest (Scott, 2017). The specific nature of that interest does not affect the analytical approach. The extraction of clique relations in social networks can reveal information related the group those members belong to. In one pending application, we are showing how such "detection of abnormal cliques' behaviors" can be possibly used to detect terrorists' attacks based on information from OSNs.

In social networks, a clique is defined as a group of elements (people or subgroups) where every member in the clique is a friend with every other member in the clique. Several recent contributions tried have attempted to extract knowledge related to cliques in OSNs (e.g. Cotterell 2013, Comandur et al 2014, Hao et al 2014, Acemoglu et al 2014, Basuchowdhuri et al 2014, Hunter et al 2015, Hao et al 2016). The fact of so much research focused on this problem, is indicative of the importance of the problem.

II. Maximum Cliques, Weighted Cliques and Maximum Weighted Cliques

Clique analysis is effectively accomplished by the application of graph theory. A clique, C, (in an undirected graph G of (V, E), is a subset of the vertices where between each two vertices, there is an edge (i.e. a

relation). In the scope of groups collaborations, ω(G), is the size of a largest clique or maximal clique of G. For an arbitrary graph the clique can be defined as per the formula in figure 1.

$$\omega(G) \geq \sum_{i=1}^{n} \frac{1}{n - d_i},$$

Figure 1: Maximal cliques
where di is the degree of a graph vertex i

Several papers have indicated that maximum cliques' problem is an NP-complete problem (e.g. Butman et al 2007, Francis et al 2015, Paulusma et al 2016, Bart and Jansen 2016, etc.).

Weighting is commonly applied in graph theory in order to better model the phenomena being graphed. A weighted clique is the total weight of weighted maximum clique (Kumlander 2004). Based on this definition, calculating weighted clique takes two consecutive steps:

1. For a particular graph, calculate maximum clique in that graph, and then:
2. For that particular clique, calculate the overall weight of the edges in the clique.

In this formula, while for a particular graph, the maximum clique is dynamic and may change from one time to another, however, the weighted clique will always be tied to the maximum clique.

subset with 3 cliques (1,2,4,5,7,11), (1,5,6,8), and (3,9,10). In this graph and edges-weights shown in the right:

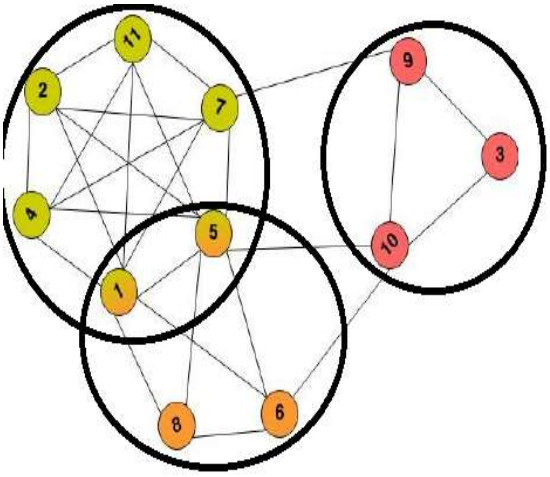
- Maximum clique is (1,2,4,5,7,11).
- Weighted clique is 2.84.

- Maximum weighted clique is 5.4 (for a clique that is not a maximum clique).

A couple of observations on our definition of: maximum weighted clique:

- This is a direct graph where edges are directional and for example value of edge-weight 1-5 is 0.1 and is different from the value of edge 5-1, (0.05). This is particularly true for relations in Online Social Networks (OSNs) where relations and not symmetric. A typical example is between a fanatic and their celebrity, edge will be very high only from one direction (which indicates high interactions from the fanatic side with celebrity activities).
- In our hypothetic example, that may look odd but true, a small size clique may have an overall stronger weighted clique than a larger clique.
- If we consider relations dynamics, maximum cliques can vary dynamically as so their maximum weighted cliques. We hypothesize that a significant increase of maximum weighted clique for a particular clique beyond their historical norms can be used to predict "abnormal behaviors).

Using this method detection of some "detection avoidance methods" is possible. For example, a clique may suddenly



Edge	Weight	Edge	Weight	Edge	Weight	Edge	Weight
1--2	0.01	4--5	0.01	7--1	0.01	6--8	0.15
1--5	0.1	4--7	0.1	11-1	0.1	6--5	0.04
1--7	0.03	4--1	0.03	11-2	0.03	6--1	0.05
1--11	0.05	5--1	0.05	11-4	0.05	6--8	0.15
1--4	0.12	5--2	0.12	11-5	0.12	8--1	0.23
2--1	0.13	5--4	0.13	11-7	0.13	8--5	0.04
2--5	0.2	5--7	0.2	1--6	0.2	8--6	0.05
2--7	0.09	5--11	0.09	1--8	0.09	9--3	0.9
2--11	0.15	7--1	0.15	5--6	0.15	9--10	0.88
2--4	0.23	7--2	0.23	5--8	0.23	3--9	0.95
4--1	0.04	7--4	0.04	6--5	0.04	3--10	0.78
4--2	0.05	7--5	0.05	6--1	0.05	10--9	0.97
						10--3	0.92
	Clique 1	2.84	Clique 2	1.38	Clique 3	5.4	

Figure 2: Three hypothetical cliques and their maximum weighted cliques

We introduced (Maximum weighted clique) to be always dynamic and not tied to the maximum clique. This means that for a particular graph, the maximum clique and the maximum weighted clique may refer to different subsets in the same graph. In order to show the difference, assume Figure 2 below indicates a hypothetical graph

delete an edge to avoid detection. However, as we

evaluated directed-edges, a single edge removal will not change the structure of the clique and will only cause that edge value to be zero.

III. METHODS AND APPROACHES

The focus of this study is on how to analyze information gathered from online social networks. The process of acquiring that information is not part of this study. There are a range of tools and techniques available for acquiring such data. One such tool that has been applied in many areas of social medial data gathering, is BuzzSumo (Dicerto , 2018). This tool has been used to analyze fake news and its relation to medical data (Waszak, Kasprzycka-Waszak, & Kubanek, 2018). ViralWoot is another tool that can be used to gather data from social media (Simmhan, 2017). The specific tool or technique for gathering the data is not relevant to this current study. Our focus is on analyzing the data once it has been collected.

For investigative purposes, particularly in intelligence gathering, the primary focus is on any deviation from established norms. For example, terrorists' networks will often have an increase in interactions prior to an operation. Therefore, we focused on an approach that identified such anomalous social interactions. We propose a method to detect "abnormal" clique behaviors through OSNs. We will collect cliques' networks from major OSNs such as: Facebook, Twitter and LinkedIn. We will use a weighted clique metric that we developed to be as a baseline for estimating average interactions between any OSN clique members. This weighted clique metric will be periodically (e.g. once a week), assessed and updated. A separate monitoring system will frequently (e.g. once every 8 hours) read the current weighted clique metric. The early alert will be triggered if this instance metrics is significantly (e.g. > 25%) higher than average weighted clique.

For example, say, for a Facebook clique of 6 individuals, weighted clique reading (as of the last month) is 13. While reading most recent weighted clique, if it shows a weighted clique of 20 (which is more than 25% increase from 13), an alert will be triggered to investigate the increase.

The usage of Cliques is popular in graph based analysis areas (e.g. social networks, bioinformatics, etc.) in order to understand connections and trust relations between graph node members.
As an algorithm, Clique calculates the maximum number of nodes in the graph in which every node in the Clique is a friend to all other nodes in the Clique.
Here are the major tasks toward this goal:

- We introduced a new "weighted clique" algorithm. We showed also how this algorithm can be used to extract different types or aspects of knowledge in Bioinformatics or OSNs. This was the first task to accomplish in this goal.
- As we acknowledged that different OSNs have different "models" of interactions between the different individuals, we plan to create several concrete models, based on the availability of time and resources for one or more of the following OSNs: Facebook, Twitter and LinkedIn.
- A major task, in terms of time and resources, in this goal is data collection and analysis for cliques' interactions. We started collecting data, based on our model from Facebook and Twitter. We need to extend the collection and analysis process to collect a "significantly" large enough dataset.

In this research we introduced the following three intuitive concepts:

1. Weighted edges (i.e. strength of bonds between nodes that interact with each other). Typically, such relation is considered only from a binary perspective (i.e. exist:1 or does not exist: zero). We showed that the mere existence of the relation can be misleading in some cases and a significant amount of information is needed to know the weight of level of such relation, if exists.
2. Weighted Friendships: Based on each node or user (e.g. in OSNs) edges or connections, weighted friendship can calculate a friendship value based on the overall values of weighted edges. We showed in our paper, some applications for such value.
3. Weighted Cliques: As described earlier the mere existence of relation between either one or a group of nodes, may not be enough to understand social interactions. Based on weighted edges, we introduced the concept of weighted cliques to show the strength of bond between clique members. Figure 3 shows the different between normal cliques count and our proposed weighted or normalized or weighted clique.

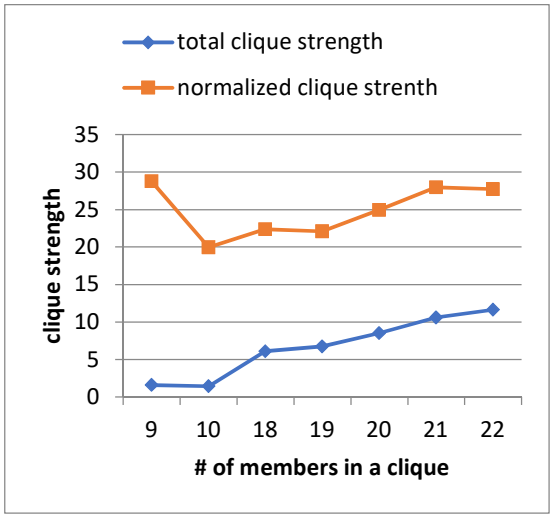


Figure 3: The comparison between total and normalized/weighted clique strengths

The extraction of clique relations in social networks can show information related to groups and how those groups are formed or interact with each other. In social networks, a clique represents a group of people where every member in the clique is a friend with every other member in the clique. Those cliques can have containment relation with

each other where large cliques can contain small size cliques. This is why most algorithms in this scope focus on finding the maximum clique. In our approach, we evaluated adding the weight factor to clique algorithm to show more insights about the level of involvement of users in the clique. In this regard clique activities are not like those in group discussions where an activity is posted by one user and is visible by all others. Our algorithm calculates the overall weight of the clique based on individual edges. Users post frequent activities. Their clique members just like other friends, may or may not interact with all those activities.

We showed in the Table 1 the largest cliques in the dataset. The first column includes IDs of node members in the clique. The second column includes the number of members in each clique. The third column includes the clique strength as the summation of all weighted edges in the clique. The last column represents the weighted clique strength which divides the total clique by the number of edges in the clique. Normal clique strength value is constantly increasing with the number of nodes or edges in the clique. However, the weighted clique value eliminates the dependency on the number of nodes or edges. As such, we can see for example that the smallest clique in the table has the highest weighted clique.

This is not a trend however as the next highest weighted clique is the one with 20 nodes. This indicates that trust or interaction between members in the first clique is the highest. When we study social networks at a larger context, highest and lowest weighted cliques can be of special interests. We can also look at variations in clique strength over a certain period of time. For example, a clique sudden increase of the weighted clique in a certain month over a period of time can trigger further investigations of what could cause such significant sudden increase (e.g. a clique collaborative activity or event).

Table 1: Total and weighted cliques

Clique members	NO	Total clique	Weighted clique %
221, 215, 190, 187, 177, 176, 171, 151	8	1.61	28.77
221, 215, 190, 187, 177, 175, 171, 170, 151	9	1.43	19.96
221, 215, 190, 181, 177, 176, 172, 171, 169, 165, 164, 162, 157, 156, 153, 152, 151	17	6.08	22.36
221, 215, 190, 181, 177, 175, 172, 171, 170, 169, 165, 164, 162, 157, 156, 153, 152, 151	18	6.75	22.08
221, 215, 181, 177, 175, 172, 171, 170, 169, 168, 166, 165, 164, 162, 157, 156, 153, 152, 151	19	8.52	24.94
216, 209, 206, 193, 192, 191, 180, 179, 175, 169, 167, 160, 158, 151, 98, 20, 18, 9, 4, 2	20	10.62	27.96
216, 209, 193, 192, 191, 180, 179, 175, 170, 169, 167, 163, 160, 158, 151, 98, 20, 18, 9, 4, 2	21	11.64	27.73

VI: Enhancing the Methodology

It should be readily apparent that the current methodology can be easily enhanced or modified if needed. For example, the current weighting threshold is set at a specific value. It would be relatively easy to adjust the method such that the weighting is dependent on specific statistical values that can vary over time. Such statistical values could include the variance in weighting over a period in time, expressed as with the following simple formula:

$$\frac{\sum_{i=1}^{n} s^2}{\triangle \mathrm{T}}$$

This measures the change in variance in relation to a change in time (delta T). This is one example of how statistical analysis can be applied to the clique analysis to provide either more information, or more granular information.

The current methodology can also be combined with graph theory to model a given clique. The clique would be represented as vertices in the graph, with the connections being edges (or arcs). Each edge would be weighted based on the strength of the connection. That strength could be measured via frequency of connectivity, number of messages per unit of time, or any similar measurement that is pertinent to the investigation in question. Then the variation in not only activity for the clique as a whole, but also for the individual edges could be monitored. This would allow the detection in variations of sub groups within the clique. These sub groups would be represented as sub graphs of the cliques' larger graph.

V: Model Evaluation

In order to evaluate our weighted clique model in OSNs and its our ability to detect "abnormal groups dynamics", we built a historical dataset based on the initial dataset described in Himel et al 2014. The volume of interactions between the friends in the network is calculated as totals for the given period of the collection process.
Following are our general steps to construct the model dataset.

- Original Facebook dataset described in Himel et al 2014 is used is the baseline dataset.
- The number of nodes or Facebook users as well as the number of edges between the members will be frozen through our one-week historical analysis. This is an assumption to fix the overall structure of the model where no changes on the nodes or relations will occur throughout the assessment. This

assumption is to simplify the model demonstration, but the model and algorithms do not exclude any frequent changes in the network structure.

- In our model, we will give to the different Facebook activities the same weight in the relation (e.g. a Like, Comment, Post, etc.). This is another assumption in the model to simplify its demonstration that can be adjusted based on the different OSNs and the nature of the different activities.
- We will randomly vary volumes of activities between network users between (25% of their baseline up to 300% of their baseline). The goal is to simulate users' actual behaviors in OSN and also demo cliques' dynamics.

Figure 4 shows a small sample of our model output on some selected cliques.

We can observe the followings based on Figure 3:

N-edges	N-nodes	list-edges...	Clique	Weight									
3	3	1	2	66.88889									
6	4	1	2	4	74.72222								
10	5	1	2	1	4	61.33333							
15	6	1	2	3	2	4	44.44444						
21	7	1	2	1	6	2	7	38.63492					
28	8	1	2	2	1	4	3	7	34.30952				
38	9	1	2	1	5	2	8	3	7	34.40741			
46	10	1	2	10	1	9	2	6	4	9	33.16296		
56	11	1	2	10	4	1	5	3	11	5	9	32.39394	
67	12	1	2	10	1	8	2	10	3	11	6	9	8

N-edges	N-nodes	list-edges...	Clique	Weight									
3	3	1	2	136.3333									
6	4	1	2	4	142.8333								
10	5	1	2	1	4	106.5							
15	6	1	2	3	2	4	74.8						
21	7	1	2	1	6	2	7	67.09524					
28	8	1	2	2	1	4	3	7	60.42857				
38	9	1	2	1	5	2	8	3	7	61.58333			
46	10	1	2	10	1	9	2	6	4	9	59.31111		
56	11	1	2	10	4	1	5	3	11	5	9	57.45455	
67	12	1	2	10	1	8	2	10	3	11	6	9	8

N-edges	N-nodes	list-edges...	Clique	Weight									
3	3	1	2	114.3333									
6	4	1	2	4	102.5								
10	5	1	2	1	4	86.3							
15	6	1	2	3	2	4	63.8						
21	7	1	2	1	6	2	7						
28	8	1	2	2	1	4	3	50.53571					
38	9	1	2	1	5	2	8	7	52.33333				
46	10	1	2	10	1	9	2	4	9	50.82222			
56	11	1	2	10	4	1	5	11	5	9	46.01818		
67	12	1	2	10	1	8	2	3	11	6	9	8	10

Figure 4: A sample output of our model

- As our model uses weighted clique values rather than clique or maximum clique values, the overall strength of the clique shows at the end of each record is independent from the clique size.
- Our model shows undirect edges/relations. This means that for two-friends, we are considering only one edge-value that can reflect one-way strength of the friendship. This was just an assumption in the model to simplify its demonstration and the model can be extended to consider or assume directed-edges or the two-way strength of the relation. The total number of the edges in the clique will then be doubles (N*N-1), rather than now (N*N-1)/2. As at the end, we are considering the overall weighted strength of the clique, this will not make a significant change.
- The 3 Figure parts indicate three consecutive days in the selected network. It can be used to monitor gradual or sudden increase in the overall weighted strength of communication of the clique. This can be used to trigger further deeper focused analysis of that particular clique.
- Our model is built purely on statistics; it does not require looking at the content of communication activities between clique members. This can alleviate issues related to privacy, legal concerns and performance.

IV: CONCLUSION

In this paper, we proposed a statistical approach to detect or alert for cliques' abnormal behaviors' using Online Social Networks (OSNs). This can be part of national security alert system that does not violate users' privacies as it does not need to look into users' contents (i.e. posted activities, friends, private messages, etc.). Due to its focus only on statistical assessments, the system also balances between security issues with performance and the impact that such systems may cause to OSNs.

As was noted in this paper clique analysis can be integrated with other methodologies. For example, the weighting can be represented with graph theory. The individuals are the nodes in the graph, the connections are the edges or arcs, and the weighting can be represented in a weighted graph. It is also possible to integrate additional statistical analysis into the clique analysis, as was discussed in this paper. Both of these enhancements to the currently described clique analysis technique are avenues for further research.

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] Acemoglu, D., Bimpikis, K., & Ozdaglar, A. (2014). Dynamics of information exchange in endogenous social networks. Theoretical Economics, 9(1), 41-97.

[3] Agarwal, V., & Bharadwaj, K. K. (2013). A collaborative filtering framework for friends recommendation in social networks based on interaction intensity and adaptive user similarity. Social Network Analysis and Mining, 3(3), 359-379.

[4] Alhajj, R., & Rokne, J. (2014). Encyclopedia of social network analysis and mining. Springer Publishing Company, Incorporated.

[5] Alsmadi, Izzat, Dianxiang Xu, and Jin-Hee Cho (2016). "Interaction-Based Reputation Model in Online Social Networks."

[6] Basuchowdhuri, P., Anand, S., Srivastava, D. R., Mishra, K., & Saha, S. K. (2014). Detection of communities in social networks using spanning tree. In Advanced Computing, Networking and Informatics-Volume 2 (pp. 589-597). Springer, Cham.

[7] Ayelet Butman, Danny Hermelin, Moshe Lewenstein, and Dror Rawitz. Optimization problems in multiple-interval graphs. In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, SODA '07, pages 268–277, 2007.

[8] Bart M. P. Jansen. Constrained bipartite vertex cover: The easy kernel is essentially tight. In Proc. 33rd STACS, volume 47 of LIPIcs, pages 45:1–45:13, 2016. doi:10.4230/LIPIcs. STACS.2016.45.

[9] Cho, Jin-Hee, Izzat Alsmadi, and Dianxiang Xu. "Privacy and Social Capital in Online Social Networks." Global Communications Conference (GLOBECOM), 2016 IEEE. IEEE, 2016.

[10] Choudhary, P., & Singh, U. (2015). A survey on social network analysis for counter-terrorism. International Journal of Computer Applications, 112(9).
[11] Comandur, S., Gupta, R., & Roughgarden, T. (2014). Counting small cliques in social networks via triangle-preserving decompositions (No. SAND2014-1516C). Sandia National Laboratories (SNL-CA), Livermore, CA (United States).
[12] Cotterell, J. (2013). Social networks in youth and adolescence. Routledge.
[13] Dicerto, S. (2018). A New Model for Source Text Analysis in Translation. In Multimodal Pragmatics and Translation (pp. 1-14). Palgrave Macmillan, Cham.
[14] Duijn, P. A., & Klerks, P. P. (2014). Social network analysis applied to criminal networks: Recent developments in dutch law enforcement. In Networks and network analysis for defence and security (pp. 121-159). Springer, Cham.
[15] Easttom, C. (2018). "A Systems Approach To Indicators Of Compromise Utilizing Graph Theory". 2018 IEEE International Symposium on Technologies for Homeland Security.
[16] M. C. Francis, D. Gon¸calves, and P. Ochem. The maximum clique problem in multiple interval graphs. Algorithmica, 71(4):812–836, 2015.
[17] Guo, L., Zhang, C., & Fang, Y. (2015). A trust-based privacy-preserving friend recommendation scheme for online social networks. IEEE Transactions on Dependable and Secure Computing, 12(4), 413-427
[18] Hao, F., Yau, S. S., Min, G., & Yang, L. T. (2014). Detecting k-balanced trusted cliques in signed social networks. IEEE Internet Computing, 18(2), 24-31.
[19] Hao, F., Park, D. S., Min, G., Jeong, Y. S., & Park, J. H. (2016). k-Cliques mining in dynamic social networks based on triadic formal concept analysis. Neurocomputing, 209, 57-66.
[20] Himel, D., Ali, M., Hashem, T. (2014). User interaction based community detection in online social networks, in: The 19th International Conference on Database Systems for Advanced Applications (DASFAA), Bali, Indonesia, 580 2014.
[21] Hunter, R. F., McAneney, H., Davis, M., Tully, M. A., Valente, T. W., & Kee, F. (2015). "Hidden" social networks in behavior change interventions. Journal Information, 105(3).
[22] Ishengoma, F. R. (2014). Online social networks and terrorism 2.0 in developing countries. arXiv preprint arXiv:1410.0531.
[23] Kirby, A. (2007). The London bombers as "self-starters": A case study in indigenous radicalization and the emergence of autonomous cliques. Studies in Conflict & Terrorism, 30(5), 415-428.
[24] Kramer, A. D., Guillory, J. E., & Hancock, J. T. (2014). Experimental evidence of massive-scale emotional contagion through social networks. Proceedings of the National Academy of Sciences, 201320040.
[25] Kumlander, D. (2004). A new exact algorithm for the maximum-weight clique problem based on a heuristic vertex-coloring and a backtrack search. In Proceedings of The Forth International Conference on Engineering Computational Technology, Civil-Comp Press, pp. 202–208.
[26] Leistedt, S. J. (2013). Behavioural aspects of terrorism. Forensic science international, 228(1-3), 21-27.
[27] Malhotra, A., Totti, L., Meira Jr, W., Kumaraguru, P., & Almeida, V. (2012). Studying user footprints in different online social networks. In Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012) (pp. 1065-1070). IEEE Computer Society.
[28] Mitrou, L., Kandias, M., Stavrou, V., & Gritzalis, D. (2014, April). Social media profiling: A Panopticon or Omniopticon tool?. In Proc. of the 6th Conference of the Surveillance Studies Network.
[29] Papadopoulos, S., Kompatsiaris, Y., Vakali, A., & Spyridonos, P. (2012). Community detection in social media. Data Mining and Knowledge Discovery, 24(3), 515-554.
[30] D. Paulusma, C. Picouleau and B. Ries, Reducing the clique and chromatic number via edge contractions and vertex deletions, Proc. ISCO 2016, LNCS 9849 (2016) 38-49.
[31] Scott, J. (2017). Social network analysis. Sage.
[32] Simmhan, Y. (2017). L1: Introduction (Doctoral dissertation, Department of Computational and Data Sciences© Department of Computational and Data Science, IISc).
[33] Soares, A. M., Pinho, J. C., & Nobre, H. (2012). From social to marketing interactions: The role of social networks. Journal of Transnational Management, 17(1), 45-62.
[34] Stieglitz, S., Dang-Xuan, L., Bruns, A., & Neuberger, C. (2014). Social media analytics. Business & Information Systems Engineering, 6(2), 89-96.
[35] Wang, Z., Liao, J., Cao, Q., Qi, H., & Wang, Z. (2015). Friendbook: a semantic-based friend recommendation system for social networks. IEEE Transactions on Mobile Computing, 14(3), 538-551.
[36] Waszak, P. M., Kasprzycka-Waszak, W., & Kubanek, A. (2018). The spread of medical fake news in social media–The pilot quantitative study. Health Policy and Technology.
[37] Yang, X., Guo, Y., & Liu, Y. (2013). Bayesian-inference-based recommendation in online social networks. IEEE Transactions on Parallel and Distributed Systems, 24(4), 642-651.
[38] Zafarani, R., Abbasi, M. A., & Liu, H. (2014). Social media mining: an introduction. Cambridge University Press.

Authors' Profiles

Izzat Alsmadi is an Assistant Professor in the department of computing and cyber security at the Texas A&M, San Antonio. He has his master and PhD in Software Engineering from North Dakota State University in 2006 and 2008. He has more than 100 conference and journal publications. His research interests include: Cyber intelligence, Cyber security, Software security, software engineering, software testing, social networks and software defined networking. He is lead author, editor in several books including: Springer The NICE Cyber Security Framework Cyber Security Intelligence and Analytics, 2019, Practical Information Security: A Competency-Based Education Course, 2018, Information Fusion for Cyber-Security Analytics (Studies in Computational Intelligence), 2016. The author is also a member of The National Initiative for Cybersecurity Education (NICE) group, which meets frequently to discuss enhancements on cyber security education at the national level.

Chuck Easttom. is an independent researcher and an adjunct professor for Capitol Technology University as well as the Director of the Quantum Computing and Cryptography Research Lab at Capitol Technology University. He is the author of 26 books, 50 researcha papers, and an inventor with 16 patents. He is a Senior Member of the IEEE, a Senior Member of the ACM, and a Distinguished Speaker of the ACM.

Developing Picosatellite Flight Software

Alex Antunes
Capitol Technology University, Laurel, 20708, USA
Email: aantunes@captechu.edu

Randy Powell
Capitol Technology University, Laurel, 20708, USA
Email: rkpowell@captechu.edu

***Abstract*—**We present a decision path for creating flight software for linux-based university-class picosatellites. We favor languages and frameworks that support modularity and strong exception handling, and add that languages enabling fewer lines of code are easier to validate. Heritage and use of existing frameworks are useful but human factors-- that are often team dependent-- are more crucial for undergraduate teams. Additionally, picosatellites can benefit from "pico Agile" development methods so as to maximize time available for testing. We include case studies including core Flight Software (cFS), our C-based TrapSat sounding rocket payload, and our Python-based Cactus-1 CubeSat.

***Index Terms*—**picosatellite, CubeSat, flight software, Python, cFS, Agile, linux

I. Introduction

University-class picosatellites, particularly CubeSats, are an active area of undergraduate engineering and science development. Also, most fail [1]. Factors that greatly contribute to picosatellite success include ensuring sufficient testing time, creating a valid concept of operations, and ensuring the software is robust.

Our missions at Capitol Technology University fit soundly into the university-class mission category, consisting of sounding rocket flights (2014-2019) and our 3U "Cactus-1" CubeSat (launching 2019). Our flight software runs on Linux and uses mixes of locally written C, the Core Flight System (cFS) C framework, and Python. We are bandwidth-limited, with no capability for post launch reprogramming. Our teams consist of engineering students with typically one faculty advisor and (rarely) one graduate student.

We found trying to get high heritage (flown before), well-documented, open-source components is a 'pick any two' problem. Building a core CubeSat bus using primarily open source hardware and software requires focusing on interface adherence over performance statistics and an expectation that refactoring will be needed. In that framework, software language choice and development environment decisions will strongly affect your delivered software reliability, and we document best steps towards creating picosatellite flight software.

Through our experience and survey of current CubeSat flight software systems, in terms of language choice, we argue that readability, modularity, and exception handling are more crucial than performance or heritage. Human factors in software development-- especially the changing roster of undergraduate teams on long-term projects and the career utility of languages the team must learn-- are also important.

We focus on Linux based development environments. A 2017 study [2] notes that non-real-time operating systems (RTOS) such as Linux have gained in favor with small spacecraft, typically using Ada, C, or C++. Likewise, that Linux is increasing in use; the same study cites its use in QuakeSat, UWE-1, UWE-2, IPEX, STRaND-1, PhoneSat, the Dove satellites, and LightSale-1 (ibid). Similarly, Commercial Off-the-Shelf (COTS) Linux-based CubeSat boards are a significant market for CubeSat developers [3].

Of the missions listed, only LightSail-1 had a major software-related mishap. The LightSail-1 mishap is interesting: a bug lead to a file growing too large, but the problem was fixable after a single-even effect (SEE) caused a reboot and provided a window for a bug fix to be applied [2].

The LightSail-1 case therefore can be considered a coding error rather than an operating system error. We agree with the argument that, as Linux is complex, testing a Linux-based flight system will be complex [2]. However, for university-class missions, we generally rely on any OS being stable and thus aim for clarity in design of the components we must develop. Additionally, testing should be carried out on the same system as will fly. OS testing thus 'piggybacks' on the flight software testing, and our primary error catching is to ensure our flight software is developed with sufficient time to robustly test.

With this, the burden is to determine the best underlying core bus and payload module development environment. Particularly for a university environment, where we must train up our student developers at roughly the same time we are using them to develop software, language choice is a key concern.

Ref. [4] is more blunt, noting “Many university CubeSat missions have failed due to software errors. This is not surprising considering that most flight software is written in C, a language that is difficult to use correctly.”

II. Language Suitability

When creating a flight software stack for a university-class mission (such as a CubeSat), two primary considerations are the strengths of the development environment and the software’s fault tolerance capability. A third, “flight heritage”, is often called for as an advantage for space hardware. However, since each university-class mission is either new or an evolution of a past mission, heritage is not often feasible for university-class missions.

In addition, our Cactus-1 work has shown that inheriting a working but unknown (to the new student developers) system can be detrimental due to time penalties incurred by the learning curve. To better choose a language, we offer several criteria.

The language choice must be well suited for spacecraft, it must be known or learnable by our students, and it should be a language useful to a student's future career. This follows a general rule for choosing a language in any situation (shown in Figure 1).Therefore we focus on development environment as a primary consideration, by suggesting three general reasons for considering which language to use. Put simply, you choose a language because it’s the best for the job, because you know it, and because it’s available/installed. Many times this is a ‘pick any two’ option, in particular the availability of other mission software solutions means code developed by one university is probably not available to others.

We do note efforts at code sharing and open source satellite solutions are increasing, and also that most flight software is developed in C, C++, Ada, or (increasingly) Python (as cited in the extended examples used in this paper). Since the software is being either developed or modified by students on a short time-scale, the three factors in play to consider are whether the students know the language (and conversely, whether the language is easy to learn), whether the language is well suited for the work, and the number of lines of code required to complete a task.

There is always an advantage of having programmers with deep experience in the language, and this can be increased by going with industry-standard languages. Space-X selected Linux and C++ because “there are many more Linux and C++ developers than, for example, VxWorks and Ada developers” [2]. By the same token, we assert that training students up in C/C++ or Python is a career-relevant skill that conveys. Esoteric or advanced languages like LISP and Haskell are therefore exempt from being chosen.

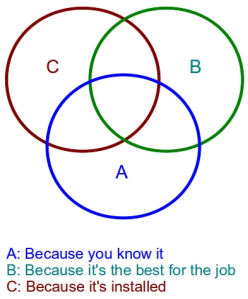


Fig. 1. Rules for creating software

This leaves the category of 'well suited to spacecraft' to consider. For the primary languages currently in use-- C, C++, Java, Ada, Python-- all fit the known/learnable and career-relevance concerns.

Knowledge of a language is a consideration, in that time spent learning a language is time lost to actually creating your software, and that novices will write buggier code than someone with experience in that language. The ability to have multiple ‘eyes’ on the project for debugging is also a factor; even one talented student programming in Rust is a risk if there are no other students nor faculty able to evaluate and debug their code. Therefore, spacecraft software writers should choose a language that is (a) openly available, (b) known to at least a subset of the team, and (c) learnable for those inexperienced with it.

We note two additional factors to choosing a spacecraft-specific language: ability of abstraction (and modularity), and ability to catch errors. And, we agree that, all else being equal, fewer lines of code leads to fewer errors [5].

III. Modularity and Exception Handling

Modularity is key in spacecraft software design. Individual hardware uses software drivers, often developed by component manufacturers (e.g. camera specifications, drivers from the vendor). Message based systems such as cFS or GMSEC use a central software handler (the bus) that polls for and pushes messages to individual software handlers. To add a new component, you write a handler that conforms to the interface or message spec, without needing tight integration.

If not using an existing bus, the team must develop their own bus. The likely designs use a primary loop that acts as an event handler, doing repeated tasks, scheduled tasks, and commanded or triggered actions. This handler can either have all functions written within it (typically as a library of team-created functions) or use sockets or files as a go-between to interact with separate software modules.

For a monolithic library-based approach, a simple looping handler might be (in pseudocode):

while Waiting:
do standard tasks
if (condition Q) do X
if (condition R) do Y
(etc)

Note that in this model, each possible activity is checked by the core routine to see if the conditions for it to be acted on is valid. The code design involves creating a set of rules that are stored within the core. Therefore, adding new routines requires modifying the core.

In contrast, a modular message bus uses an event handler to query autonomous modules for the existence of a task to be done:

while Waiting:
do standard tasks
check the queue for messages,
if so follow up on each

Under this model, the core contains a list of 'clients' to poll for messages, then if a message is received, either acts on that itself or calls a module. With this design, you can write new modules and simply register them to the core, requiring less modification to add modules but more overhead for each module (which must include the messaging API). Neither approach is 'better', but depends more crucially on your framework, your language choice, and your design philosophy.

For this paper, we suggest the advantage is in languages that make it easy to create modules that can be tested individually, then called by the flight software when needed. In a sequential language such as C or Ada, this means creating libraries of functions and subroutines so the code from the design stage is modular. For object-oriented such as C++, Java, and Python (or languages that support objects, like Ada), the objects themselves can provide modularity. The stronger the language support for modularity, the less likelihood of errors, in that a tested module that is validated can remain unchanged as development on further modules continues.

Any language can be misused or written poorly in, but some languages have stronger protection. Ada and Python tend to be 'stricter' in their coding, with strong typing, memory handling, and aggressive compiler (Ada) or interpreter (Python) pre-checking of the code.

Exception and error handling are needed due to the complexity factors in spacecraft. These include health and safety monitoring of nominal operations, communication uplinks and downlinks, the duty cycle required between routine and crisis operations before payload data is lost, and data issues (including completion, volume and timeliness, and calibration and validation).

The driver for requiring exception handling is that we can expect anomalies to occur in any of these aspects. A survey of 13 spacecraft over 2 years found 21 anomalies occurred [6], with a frequency of an anomaly about every 7 months of spacecraft operations. Of these 21, 4 had no mission impact, 11 resulted in loss of redundancy, 4 had degraded performance, and 2 lead to loss of mission. 16 of the 21 anomalies were in the data, payload, or attitude systems (with the remainder being launch, power, thermal or ground anomalies).

We assume that anomalies will occur, and that the flight software must catch errors and anomalies, ideally returning the spacecraft to a known, working baseline state. Two schools of exception handling exist. The 'failstop' principle argues code must stop once an exception occurs, because it indicates a bug and therefore the possibility of further problems [7], and notes the safest way is to stop the entire program. This may be a poor option for a spacecraft.

Anomalies are not always terminal; for example, a camera or sensor temporarily not being reached is a rare but tested event that our flight software must support. We prefer that unresponsive components fail gently, and remain in the polling loop for the next cycle of times or commanded events.

As CubeSat rapid development tends to be integration of multiply sourced drivers and components, a more viable approach is the controller must assume any component might fail. An exception that fails back to the main loop is therefore valid. Examples of good try/except use in spacecraft include, in order of effectiveness:

1) if error then skip
2) if error than call error function
3) if error than retry
4) if error than restart loop

The first two are preferable. Immediately and repeatably retrying a non-responsive component can lead to a blocking fault in the loop code. Similarly, restarting a loop due to an error can prevent subsequent necessary commands from being executed. Therefore, we adopt the philosophy that errors should be skipped, and crucial errors found in testing can be provided with error-handling code as testing and time allow.

A 2008 study noted that 1-5% of open source Java code is typically devoted to catching exceptions, and 3-46% of code then used to resolve exceptions (and that in the 80s, a similar survey indicated that up to 60% of code was devoted to exception handling) [8].

In terms of language support for exception handling, C is weak due to the lack of an internal try/except method.

Ada, Python, C++ and Java contain control structures to allow exception handling within the code. To implement error handling for Cactus-1, we use the exception handling built into Python. Because of its stronger exception handling using Try/Except clauses, our Python code can survive temporary or permanent errors in called modules.

The final factor we return to is lines of code required to complete a task, because it is easier to review a smaller code base than a larger one. The more code to create and test, the larger the chance of error. A 2013 report asserts Ada is preferable to C [5] in part because a task takes fewer lines in Ada than C, allowing for inspection as one test method. Some of this can be mitigated by modularizing your C code into libraries, however, effective object passing in C is itself a source of errors. Also, codes that emphasize readability and modularity (especially Ada and Python) are higher-level languages that require fewer lines to serve most spacecraft tasks.

For an example of readability, here is a a subset of a serial camera read on a Raspberry Pi using C versus Python (with error checking and comments removed), to illustrate the logic.

```
# In C (18 lines):
cam->frameptr = 0;
clearBuffer(cam);
serialPutchar(cam->fd, (char)0x56);
serialPutchar(cam->fd, (char)cam->serialNum);
serialPutchar(cam->fd, (char)FBUF_CTRL);
serialPutchar(cam->fd, (char)0x01);
serialPutchar(cam->fd,
 (char)STOPCURRENTFRAME);
unsigned int len;
len = serialGetchar(cam->fd);
len <<= 8;
len |= serialGetchar(cam->fd);
len <<= 8;
len |= serialGetchar(cam->fd);
len <<= 8;
len |= serialGetchar(cam->fd);
int32 pic_fd = OS_creat(file_path
 (int32)OS_READ_WRITE);
OS_write(pic_fd, (void *)image, imgIndex);
OS_close(pic_fd);

# In Python (12 lines):
ss = serial.Serial(PORT, baudrate=BAUD,
timeout=TIMEOUT)
reset(ss)
takephotocommand = [0x56, 0, 0x11, 0x00]
cmd = ''.join (map (chr, takephotocommand))
ss.write(cmd)
reply =  ss.read(5)
bytes = getbufferlength(ss)
photo = readbuffer(bytes,ss)
f = open(camfile, 'w')
photodata = ''.join(photo)
f.write(photodata)
f.close()
```

From a logic point of view, both are equivalent. Both examples also require non-intuitive documented driver calls (the 0x56, 0x11 and other byte patterns needed). The Python commands have higher clarity due to Python's typing, use of objects instead of pointers, and focus on readability. While the C version requires bit shifting logic (with “len |=” and “len << =8”), the only arcane Python commands are the uses of “join” and “map”. Python also more strongly separates data items (such as the byte specifications for the driver) from actions (generic items such as 'ss.write' and 'ss.read' to write/read from the serial port).

Readability is extremely crucial when considering software hand-offs and maintainability. As a university, we cannot assume continuity of developers. Each sounding rocket had a slightly different team, and Cactus-1 consisted of different phases of work spread out over 3 years (a very long time relative to an undergraduates' time availability).

In addition to readability, languages with a shorter development time leave more time available for testing and debugging. Code familiarity likewise speeds development time. Given these time factors, human factors in language choice again are more important than performance.

IV. Cactus-1 Use Case

Our Cactus-1 CubeSat system consists of health and safety sensors, two internally facing cameras, and a half duplex radio communications module. Our language-agnostic use case is on par with the needs of similar picosatellite missions. Our mission, being bandwidth-limited, with 1 ground station and no capability of changing on board programming, requires the ability to send command mnemonics, optionally overwrite stored mission parameters such as cadence and instrument modes, transmit health-and-safety packets plus image data, and comply with FCC shut-down requirements. A simple outline (in comment format) of our flight controller design is given as:

```
# “Cactus-I Flight Software”
# Initialize all systems
# have an OS-level monitor in case of crashes
# program eternally loops
        # get current time
        # CHECK FOR INCOMING COMMANDS
        # (file-based or listen-based?)
                # loop through what was received
                # check if it's valid, log errors
                # then carry out its function
        # REGULAR TIMED TASKS
        # HK data gather
        # payload data gather
        # optional sleep? if it saves power
```

We use the best practice of defining our code in comments, then adding code to the comments to build up the software, rather than writing code and commenting later. This allows the code to self-document and ensures the documentation matches the actual flight code.

Our command mnemonics are few. Table 1 defines our primary command mnemonics and their functions, which

also shows the low level of complexity required by our software. Most commands include optional arguments to set payload, power or communications parameters to values other than the pre-loaded defaults (that get reset to defaults when the base command is later issued).

The commands were defined by the mission and payload requirements, and all count as essential required items that the flight software must support. The use of optional arguments is a desired but not required feature set. In implementation we chose that overriding of default values does not persist across reboots.

V. Pico Agile Design

Regardless of language choice, the flight software must meet requirements and the payload must ship on time. Spacecraft launch times set a firm ship date, therefore software development cannot slide to later. Traditional “waterfall” development using Gantt charts and similar scheduling tools are one method for developing your software. However, given the rapid development time and small team size, we suggest Agile methods as being equally strong for managing your development.

Agile methods introduce a Burn Chart as a way to track the completion rate of a software project. In a Burn Chart, you start with 0% done and finish at 100% done. Agile typically creates “User Stories” as a task list of items that must be accomplished, and the Burn Chart counts down the number of User Stories left to implement.

For spacecraft software, you can consider the User Stories as the requirements and feature list, and that implementing each requirement and feature moves you closer to completion. In this fixed time environment, there are two approaches to completing software. If software development is running late, you can either add resources or reduce the feature set of the software, as shown in Figure 2.

Requirements are defined as items that must be complied with. Features are the set of items that would be useful but are not essential to flying. When designing flight software, 100% of requirements must be met, and then features added up until the software freeze date, after which no changes other than mission-critical bug fixes should be done.

We used an Agile approach named “Pico Agile” [9], which is an Agile method for project instantiation and control, but without formal Agile practices for day-to-day work. Pico Agile can act as an overlay with other Agile processes, serves as a work-flow version of a to-do list, and is highly suitable for solo projects or small teams with fixed resources on a set schedule.

Table 1. Cactus-1 Command Mnemonics

Core Commands		
Mnemonic	Functionality	Italic
downlink	high-power no-wait downlink of all data for up to 10 minutes	
beacon	sends only health and safety packets	
longwait	put into a low power mode for specified interval	
cadence	set alternative data modes and/or rates Subheading	
sporton/ sportoff	trigger mode for AMSAT outreach	
Anomaly Modes		
burn	re-trigger burn wire to re-try antenna deploy	
panic	start 1 day of beaconing	
fccoff	immediate cease of all transmissions	

Under Pico Agile, you create a 'to do' list of items in 4 categories: Prework/Infrastructure, Must Do Tasks, Should Do tasks, and Would Like tasks. Prework would be, for example, installing your development environment, databases, and outside packages and frameworks required.

The Must Do tasks are driven by the requirements and include basic functionality-- typically housekeeping, payload data gathering, and communications as a minimum set. Missions that have active power monitoring, attitude and/or maneuver control, and deployables would likewise add those to the Must Do category. FCC and launch provider requirements are also Must Do.

Should Do tasks include additional functionality as well as improvements to Must Do tasks. Should Do items often cross subsystems. For example, with Cactus-1, one Must Do requirement is that the system take an image of the Aerogel orbital debris capture substrate once per hour (minimum); a Should Do task is to be able to change that cadence to a ground-specified setting via commanding. Would Like tasks are the lowest priority items that, as the name states, are desirable but not core to the mission.

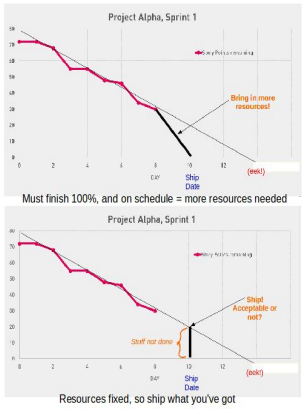


Fig. 2. Two burn approaches for fixed due date

Once you have your categories, the next step is to add Dependencies in order to track which items have other items that must be completed first. We suggest 3-digit #s, starting at 010, then 011, 012, 020, etc-- hopping in steps of 100 (categories) and 20-30 (feature sets)-- this leaves room to add items later and avoids the “number #1 must be most important” fallacy, unless it is actually relevant.

Use the numbers to build chains via increments for easier tracking. A sample roster of items for the Cactus-1 TrapSat payload system would be:

PREWORK:
- 001 install camera drivers

MUST DO:
- 100 take and save images
- 101 switch MUX to choose cameras
- 102 illuminate camera LEDs
- 200 take and save HK data

SHOULD DO:
- 130 set camera cadence rate
- 131 change camera resolutions (req 100)

WOULD LIKE:
- 170 add difference images (req 100)
- 270 trend HK data (req 200)

The last step is to ignore the Would Like items-- pragmatically, for most missions, the bulk of these will not be implementable in the time available. They are worth keeping for documentation purposes as well as in case someone wishes a side project, the mission launch is delayed, or if their implementation is trivial.

VI. Existing Frameworks and Heritage

Part of deciding on a language is evaluating the utility of existing frameworks. Frameworks can be previous versions of your flight software or external flight stacks provided by peers or the community. We briefly examine two community-provided frameworks: Core Flight System (cFS) and CubedOS, as well as our own multi-mission heritage with C and Python.

Core Flight System (cFS) is a C-based framework from the Flight Software Systems group at NASA/GSFC and is freely available. It is a bus that allows developers to write modules according to their Application Programming Interface (API) rather than have to work with the core code base [10. It is in C, and has a high level of abstraction through the API. In addition, user-written applications run as services and thus are added or removed during runtime.

The ADA language has the SPARK framework to enable good abstraction and better capture of errors [11]. One implementation is the CubedOS framework, consisting of 5991 lines of code-- of which 4095 are comments and SPARK test-case annotations [11]. The CubedOS team likewise maintain a C code base of 2239 lines. Their architecture is, like cFS, a message passing framework.

Heritage does not automatically guarantee success. We note the failure of the Ariane5 launch is attributed to code re-use, with Ariane 5 re-using the 10-year old Ariane 4 software and running into an integer conversion problem as a result [12]. We assert that you need to understand all of your code, even heritage code that has performed well but under a different hardware configuration or for differing data needs.

Internally at Capitol Technology University, we use a 'crawl, walk, run, fly' mission progression from lab through to high altitude balloons, to sounding rockets, culminating in the CubeSat [13]. This provides hardware continuity across projects. Similarly, we retain the software drivers for each component. However, the command and control needs for each step are vastly different, as are communication capabilities. Therefore, the core controller for our flight software stack is not guaranteed to retain heritage.

For our sounding rockets, the initial C based software was later merged into cFS. This two-step integration from a simple stand-alone C stack to the more robust cFS environment is equivalent to two complete software development efforts, as the integration is not trivial and required at least two developers for multiple semesters. For our Cactus-1 CubeSat, we found that prototyping a test Python-based flight stack was a task the co-author was able to achieve in under half a semester, and the author was able to then write the actual flight Python stack in a similar interval. This is due to the readability and modularity of Python as well as the small size of code our mission requires.

We therefore kept heritage for the hardware drivers and serial port protocols used, but did not retain heritage by continuing with cFS or revising the Python prototype. Instead, we used our Python payload and communications modules developed during building our flatsat, and integrated them under a controller program. Our flight controller model is simple enough that the redesign was preferable to refactoring.

We advocate Python as a strong language for flight software, and that is what we used for Cactus-1. Not counting drivers (which have to be written for hardware regardless of bus design), we can compare the development statistics for a C-based system to a Python-based system. We use as our cases our TrapSat Sounding Rocket (flown under NASA's RockSat-X program) as comparable to our Cactus-1 CubeSat (which flies the same detector).

Both use passive power monitoring (TrapSat via the rocket power line, Cactus-1 via a self-regulating solar-battery bus). TrapSat sent comms data via the parallel port; Cactus-1 operates a radio module via the serial port. Both have health and safety Housekeeping (HK) data. Cactus-1 can be commanded; TrapSat relies entirely on a pre-loaded schedule.

The comparison of the TrapSat sounding rocket,which used C and cFS, and the Cactus-1 CubeSat, which used in-house Python code, provides numerics on how language choice influences readability, code complexity, and the ability to transfer code across teams.

The TrapSat Sounding Rocket flight used a Raspberry Pi plus an Arduino, and included comms via the sounding rocket interface. TrapSat used cFS as its bus. Written in C, cFS itself can be considered stable and tested, so the code requiring creation and testing are the TrapSat applications.

The payload data interface consists of 6 C routines totaling 1127 lines. The comms/data handling interface is 7 C routines totaling 1394 lines. The health and safety monitoring is 5 C routines totaling 792 lines. This totals 3313 lines.

In contrast, Cactus-1 (shown in full in Figure 3) uses a Raspberry Pi. It has 1006 lines of Python code in 4 modules: a main Core controller program (147 lines), a library of core Bus functions (361 lines), a library of Payload-specific functions (463 lines), and a shared Config file (35 lines).

Half to two thirds of each routine is comments (e.g. only 57 of the 147 Core lines are actual code). Roughly half the code resides in the payload-specific library. The Core controller is small and easily tested. The Bus functions have high criticality, as they are responsible for communications and underlying bus functionality.

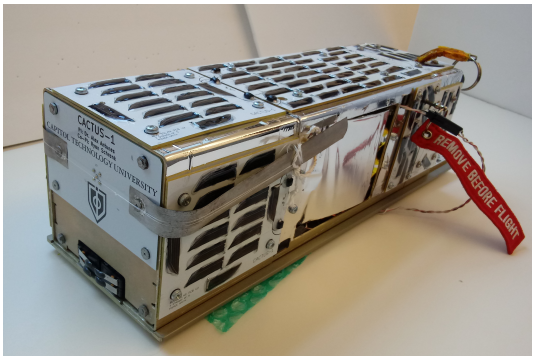


Fig. 3. The Cactus-1 3U (10cm x 10cm x 30cm) CubeSat

In terms of readability (for which Python is famous), the Cactus-1 code is highly readable, well commented, and has a high level of abstraction (very few lines of code per function). This makes it easier to visually debug, as well as making it straightforward to place try/except error handling checkpoints.

From a coding point of view, cFS requires more expertise to interface. The amount of new code required by TrapSat is roughly 3 times the amount of new code required by Cactus-1.

In practice, each subsystem hardware module from Cactus-1 during development was tested prior to integration. Therefore, the Cactus-1 flight controller is the least tested subsystem at the time of integration. Conversely, with TrapSat using cFS, the software and hardware were well tested but each cFS module was untested at the time of integration, because the prior lab bench code did not yet have the cFS bus protocols.

From a testing point of view, since Cactus-1 had to write their own bus, more testing time is required at the bus level before working at the Payload level.

In our cFS experience, its advantages is that it worked well for RockSat. cFS is also more scalable; once you've written one module, it is easier to write additional modules as you move to increasingly more complex spacecraft needs. cFS's primary disadvantage was its learning curve; we needed to learn its message passing API.

The advantage of the Cactus-1 Python system is the small code base, its simplicity, and that it is easy to find student Python developers. Its disadvantage is that the system interactions are more tightly coupled in the core function and thus you must modify the core bus when adding capabilities.

With this comparison, from a time resource stance, cFS is essentially overkill for a simple mission like Cactus-1, but could be useful if you have time and C developers. If you are not CS people (just engineers and scientists, like us), maybe not.

However, from a testing standpoint, the role-up of individual Python modules into our core scheduler and the great reduction in lines of code to track make the Python-based environment stronger for testing.

VII. Conclusions

When developing flight software, it is important to evaluate your language and framework choices not just on outside assessments of what is best or most commonly used, but more crucially based on your specific use case (especially regarding complexity and re-programmability) and the best use of your existing talent pool.

The choice of packages and languages for flight software should be made on the basis of team abilities and soft factors as much as on the language's capabilities. We favor continuity of developers over requiring high heritage for software. Agile processes and delineating which features are requirements and which are optional will be crucial, as spacecraft launch dates firmly set your development schedule.

Modular design that lets you inherit software components across prototypes and test builds will speed development and improve reliability. For language

choice, robust exception handling is a must, as anomalies should be expected to occur during a picosatellite mission. Additionally, when in doubt, going with known languages that are career-applicable is an appropriate decision in a university setting.

Ultimately, your pre-coding choices should aim for a faster development time producing readable code that will free time for more testing, as robust and frequent testing is a strong predictor of mission success. There are several existing frameworks and upcoming open source flight stacks available, and they should be considered within the use case of your specific mission. We suggest the use of Linux and Python, as with Cactus-1, as a good default choice for new teams looking at possibilities.

ACKNOWLEDGMENT

Support for Cactus-1 was provided by the Maryland Space Grant Consortium (MDSGC). The Cactus-1 launch opportunity is provided by the NASA CubeSat Launch Initiative (CSLI)..

REFERENCES

[1] Swartwout, M. (2018). Reliving 24 Years in the Next 12 Minutes: A Statistical and Personal History of University-Class Satellites. *32nd Annual AIAA/USU Conference on Small Satellites.*

[2] Leppinen, Hannu (2017). Current use of linux in space flight software. *IEEE Aerospace and Electronic Systems Magazine* 32(10), 4-13. DOI: 10.1109/MAES.2017.160182

[3] Kalman, A. (2012). Adapting Linux-based Computing to CubeSat. CubeSat Developers' Workshop/26th. *Annual AIAA/USU Conference on Small Satellites.*

[4] Chapin, P. (2019). Ten Years of Using SPARK to Build CubeSat Nano Satellites With Students. Retrieved from https://blog.adacore.com/ten-years-of-using-spark-to-build-cubesat-nano-satellites-with-students (Original work published 2019)

[5] Brandon, C., & Chapin, P. (2013). A SPARK/Ada CubeSat Control Program. Reliable Software Technologies – *Ada-Europe 2013*, 51-64.. DOI: 10.1007/978-3-642-38601-5_4

[6] Squibb, G, Boden, D., & Larson W. (2016). *Cost-Effective Space Mission Operations (2nd edition)*. McGraw-Hill.

[7] Weinreb, D. (2013). What Conditions (Exceptions) are ReallyAbout. Retrieved from https://web.archive.org/web/20130201124021/http://danweinreb.org/blog/what-conditions-exceptions-are-really-about (Original work published 2013).

[8] Weimer, W., & Necula, G. (2008). Exceptional Situations and Program Reliability. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 30(2). DOI: 10.1145/1330017.1330019

[9] Antunes, A. (2019), Pico Agile [PDF file]. Retrieved from http://ghostlibrary.com/ag (Original work published 2019)

[10] core Flight System (2019). Retrieved from https://cFS.gsfc.nasa.gov/Features.html (Original work published 2017)

[11] Brandon, C., & Chapin, P. (2017). CubedOS: A SPARK Message Passing Framework for CubeSat Flight Software. *SPARK/Frama-C Conference*.

[12] Jézéquel, J.M., & Meyer, B. (1997). Retrieved from http://www.irisa.fr/pampa/EPEE/Ariane5.html (Original work published 1997).

[13] Schrenk, R., Strittmatter, M., Walters, A., & Jagarnath, M. (2017). Crawl, Walk, Run, Fly. 2017 *Academy of Aerospace Quality Conference Proceedings*.

Authors' Profiles

Alex Antunes has a Ph.D. in computational sciences and informatics, specially computational astrophysics, from George Mason University, 2009, and a Masters of Science in astronomy from Pennsylvania State University, 1992. His field of study addresses open source satellite work, cheap deployable sensors, machine learning, and engineering education.

He is a professor of astronautical engineering at Capitol Technology University in Laurel, MD. Prior work includes spacecraft science operations and astrophysics work at GSFC, solar physics at NRL under an NRC fellowship, design of the DIY TubeSat "Project Calliope", and freelance science writing. In addition to scientific research on solar physics, X-ray binaries, and collisional remnants, he has written five "DIY" technical books for O'Reilly/Maker Media on amateur space and machine learning topics.

Dr. Antunes is on the advisory board for the NASA Academy of Aerospace Quality (AAQ) and is a member of American Geophysical Union/AGU, American Astronomical Society/AAS, AMSAT, National Association of Science Writers/NASW, DC Science Writers Association/DCSWA, International Game Designers Association/IGDA, American Society of Engineering Education/ASEE, American Institute of Aeronautics and Astronautics/AIAA.

Randy Powell graduated with a Bachelor of Science in Computer Engineering from Capitol Technology University in May 2019. He worked with the Cactus-1 CubeSat mission as an undergraduate. He works in Systems Engineering at General Dynamics Mission Systems. He is skilled in PHP, Python, Java, C, C++, and Perl.

Behavioral-based malware clustering and classification

1. Izzat Alsmadi
Texas A&M University, San Antonio
ialsmadi@tamusa.edu
2. Bilal Al-Ahmad
The University of Jordan
b.alahmad@ju.edu.jo
3. Iyad Alazzam
Yarmouk University
eyadh@yu.edu.jo

***Abstract*— Detection of malwares and security attacks is a complex process that can vary in its details, analysis activities, etc. As part of the detection process, malware scanners try to categorize a malware once it is detected under one of the known malware categories (e.g. worms, spywares, viruses, etc.). However, many studies and researches indicate problems with scanners categorizing or identifying a particular malware under different categories. There are different reasons for such challenges where different malware scanners, and sometime the same malware scanner, will categorize the same malware under different categories in different times or instances. In this paper, we evaluated this problem summarizing existing approaches on malware classification.**

Keywords—Malware: Detection, Classification, and Category

I. INTRODUCTION

Different methods are employed in the process of malwares' detection such as: signature/dictionary-, rule- and behavioral-based methods. Signature-based detection method (e.g. using hashes for known files or malwares) is widely used due to its efficiency and robustness in analyzing a large volume of files within a short amount of time. However, there are many obstacles in expanding this method to detect all types of malwares especially in unknown territories, new malwares, or old malwares with slightly different signatures.

Anomaly or behavioral-based detection methods can complement signature-based methods as they are slow but effective in unknown territories.

Collecting attributes to uniquely identify and distinguish malwares is not a trivial process. Attributes to define and distinguish malwares are many, not universal or widely agreed-upon, exist in different artifacts or locations, can be static or dynamic and can be interpreted by different scanners differently. Malware writers sometimes insert garbage calls in order to confuse the analyst with fake API calls or other useless features that can confuse the detection process and impact its decision and accuracy. They may also encrypt or package certain details in which collecting information about such details will be hard or impossible. Those are examples of the challenges to perform data analysis activities in malwares (e.g. clustering, classification, prediction, etc.).

There are many public malware scanners such as AutoShun, PhishLabs, Kaspersky, StopBadware, Sophos, or Netcraft. Testing output from the different malware scanners indicate that frequently they may have different decisions on the same files. It indicates that they employ also different detection techniques or methods as in (Akour et al., 2017).

Figure 1 shows a sample scanning result for a popular Trojan. Although it is a popular Trojan, yet (1) 9 malware scanners indicate this file as "clean", and (2) only 25 of the 67 malware scanners that identify this file as malware, clearly indicate that its type is a Trojan. Malware scanners have to complete scanning systems with usually a large number of files. They have to analyze each subject or suspect file with static and possibly dynamic methods. They have to be also accurate and avoid different cases of false positives and negatives. Those are also other examples of challenges facing malware scanners and all data analysis activities related to malware analysis and detection.

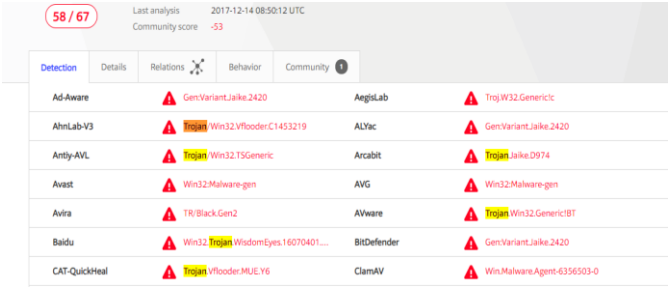


Fig. 1. A sample scanning result for a popular Trojan

Malwares can be classified under different major categories such as: viruses, worms, spywares, ransomwares, etc. Some of the main characteristics that can be used to distinguish those different groups from each other include:

- Payloads: By default, malwares contain a sort of harmful payload to achieve. For example, viruses cause files corruptions or destructions, worms consume machines bandwidths and resources, ransomwares encrypt victims' data, spywares spy on victims' activities, and so on. Some malwares may have different payloads at different stages. Some other large and complex malwares can have several payloads.
- Access or intrusion method: Different malwares have different mechanisms to make their first access to victim machines. Some malwares use other malware types only in their access stage. For example, a worm may use a Trojan method to reach victim machines and control them remotely.
- Propagation: Malwares perform different workflows from the moment of access of targets to the moment of payload deployment. (Saeed et al, 2013) paper shows a table of different categories of malwares and examples of unique attributes related to distinguishing criteria such as: Creation techniques, execution environment, propagation media and negative impacts.

In this paper, our main goal is to use data analysis to group or cluster malwares based on some attributes or behaviors and eventually compare such groups with existing groups related to known malware categories.

The rest of the paper is organized as the following: Section two focuses on research questions, goals and methodologies. In the third section a selection of relevant research publications is discussed. Section 4 focuses on experiments and analysis. Finally, paper is concluded in a small summary or conclusion section.

II. How do malware scanners decide a malware category?

The analysis of the malicious program is important to extract features, which describes the risk and the malware type; there are three types of detection and analysis method to identify the malware categories: (1) static analysis detection technique, (2) dynamic analysis detection, and (3) hybrid analysis detection technique.

First, Static analysis detection technique (Imtithal A.saeed, 2013) (Smita Ranveer, 2015) (Ekta Gandotra, 2014) (Dolly Uppal, 2014) (Balaji Baskaran, 2016) (Saba Arshad, 2017) (P. V. Shijoa, 2015), these studies described the static analysis that is analyzing software malicious and extract the features in the binary code or internal structure of the file without executing, the application is break down by some tools and techniques to rebuild the source code and algorithm of the application that created, it is done during program analyzer and debugger, this type is safe and fast. There are different static analysis techniques: Specific detection (e.g. signature or hash-based detection), and heuristic behavioral detection. Specific detection works by looking for known malware by a specific set of attributes (Imtithal A.saeed, 2013) (Smita Ranveer, 2015) (Dolly Uppal, 2014) (Saba Arshad, 2017), these approaches indicated that known malware have known signatures recorded in a database that can be applied on subject or tested files, but it cannot detect an unknown malware. While Heuristic behavioral detection: It called proactive technique, it is similar to signature based but it does not use searching for signature in code, it search for the instruction that is not appear in the application program (Imtithal A.saeed, 2013) (Dolly Uppal, 2014) they proposed that this process scans for previously unknown malware by looking for known abnormal or suspicious behaviors. Anomaly-based detection depends on monitoring system activities and classifying the subject as either normal or anomalous accordingly.

Heuristic detection technique (Dolly Uppal, 2014) have different types such: (1) File based heuristic analysis file based or file analysis Heuristic system, it analyzes the file completely and check if there is any command in the file can delete or harm other files, it will be considered as malicious. (2) Weight based heuristic analysis, this is the oldest technique, each application have a danger weight or value, and there is a threshold value, if the weight override the threshold value the application will contain a malicious code, (3) Rule based heuristic analysis at this type, the analyzer extract the rules of the application, and match it with the previously defined rules. Such if there is any mismatching then the application contains malware, and (4) Generic signature analysis, this process looks for malware by behaviors of known categories or that are variants of known categories. different behavior for the malware but belongs to the Same category used to discover new variant of malware, for example, statistical-based techniques apply statistical models on system activities such as network connections, bandwidth, memory usage, system calls, etc. which can be usually used by malware. Apparently, false positive cases are common in such scenarios where many “good” applications or system behaviors can be mistaken as malicious activities.

Second, dynamic analysis detection technique (Ekta Gandotra, 2014) (Dolly Uppal, 2014) (KIMBERLY TAM, 2017) (Saba Arshad, 2017) discuss that the dynamic analysis observes the result after executing the program by Analyzing the behavior or the Action of the application but it takes time as the executing time of applications. It interacts with the system while execution in VMware, Simulators and sandbox to find if the executable file is malware or not

Third, hybrid analysis detection technique (Ekta Gandotra, 2014) (Dolly Uppal, 2014) (Balaji Baskaran, 2016) (KIMBERLY TAM, 2017) they proposed that the hybrid analysis is combination of static and dynamic techniques by checking if there is any malware signature in the code. Then observe the code behavior.

In addition, detection can be also classified (Kyoung Han, 2014) into three categories: (1) Host based intrusion detection system, (2) network-based intrusion detection system, and (3) Hybrid intrusion Detection system. Host based intrusion observes and controls the dynamic behavior and the computer system state to check if there is any internal or external Activities that cheats the system policy. Network-based intrusion detection system analyzes all the packets in the network node. Host-based and network-based detection based on the sources of artifacts used in the analysis and detection processes. Hybrid intrusion Detection system: A combination of host- and network-based intrusion detection system is also possible especially with large and complex malware.

Dynamic detection methods run the suspect file in an isolated environment. The research study (Imtithal A.saeed, 2013) proposed isolated environment, sandboxes, where special APIs connect suspect file to the Virtual Machine (VM).From a data science perspective, malware detection process is a typical classification process of two stages: In the first stage, subject file, traffic, hash, etc. will be classified as either a malware or benign. Many malware scanners leave a third category: undecided if the subject fails to be allocated to either one of the malwares/benign categories. For example, many of the newly discovered malware may fall under this (unknown/undecided) category. In the second stage, if the subject is classified as a malware, different categories of malware are available and the process is to allocate this subject to one, or more of those categories.

III. What are the main features in attributes or categories of features that can be used to detect malwares and malware types?

Feature extraction (Mansour Ahmadi, 2016) (Yanfang Ye, 2017) Indicates that the malware detection and classification need to extract some features from (*. byte with Hexadecimal view) and (*. asm file form assembly view). The Portable executable (PE) header describes the basic information and it is rich of information for feature

extraction. It has two essential types: (1) Features from Hexadecimal files, and (2) features extraction from assembly files. There are several features used in Hexadecimal files such: (1) N-gram, (2) metadata, (3) Entropy, (4) image representation, and (5) string. N-gram is a sequence substring with N items, it is used in many fields with characterizing sequence. The study (Mansour Ahmadi, 2016) describes that N-gram means N-byte the malware sample represented as a sequence of hex values and described through n-gram analysis to give information about the malware type. The byte in the binary code contain (2^8) 256 value, by adding special symbol (??) it will become 257 different values, the"??" indicates that there is no mapping for the corresponding byte in the executable file, this value can be discarded. an example the 1 gram feature which represent the byte frequency that described in 256 dimensional vector, in (Meena, 2011) author proposed an approach to detect the Malicious code with n-gram that extracted from Portable execution of malware sample as a feature, (Zhang Fuyong, 2017) proposed classification method based On the similarly of n-grams attribute for malware detection. Also, Metadata it extracts and summarize the basic information of the file, such as file size, address, date modified or date created, (Mansour Ahmadi, 2016) describes that the PE header gives us a meta data information for the executable file. The study (Kun Wang, 2016) proposed a malware detection method depends on metadata of App that extracted from APK file and build vector space for datasets. In addition, the approach in (Mansour Ahmadi, 2016) explained that the entropy technique based on describing the measurement of disorder, it is a numerical measure, the study (Jared Lee, 2015) used the entropy information to interact with metamorphic detection problem, computing Entropy become on the byte level representation of the Malware sample, to measure the disorder in the byte distribution in byte code.

Moreover, image representation (Mansour Ahmadi, 2016) show that the malware sample can be visualized the byte code and explain each byte as gray-level of pixel in the image, this technique used in visual signature By patterns similarity (Kyoung Han, 2014) proposed a method for classification using image representation by converting binary files into images and entropy graph. As in the studies (Mansour Ahmadi, 2016) (Yanfang Ye, 2017), the string transfers the Hex byte file to ASCII string from PE because some strings are not clean enough, so taking the string length better than String and counted as features for malware classification. In terms of the feature's extraction from assembly files, there are several studies used various features such: (1) metadata, (2) symbol, (3) operation code, (4) registers, (5) application program interface, (6) data define instruction, (7) section, and (8) miscellaneous.

Metadata is identical to metadata in hexadecimal file depend on (Mansour Ahmadi, 2016) the technique done by computing and extracting the information about the file as assembly file features, and the size of the binary code file as features but after being disassembled, the two sizes will be different but both of them are recorded. It used symbol to calculate the frequency of the symbols and check the high frequent that used to prevent and avoid malware detection. Also, the operation code it is a representation of machine code, also called instruction syllable and it is a part of machine language instruction that identify the operation to be executed and symbolize assembly instruction which uses 93 common operation code that utilized as features. In addition, there is another approach which assigns a malware sample to a specific family by the frequencies of registers in processors; the registers are small storage in processors with high speed. Application programming interface, many studies focused on detecting malware and their types based on the nature of API calls (Dong-Jie Wu1, 2012) (Zhu, 2013). Also, the study in (Mansour Ahmadi, 2016) identified 794 most frequent APIs in malware that obtained from 500K malicious samples, the type, number and frequency of the different APIs can be also used to uniquely classify the malware. This can vary by the platform (e.g. Windows, Apple, Linux, Android, etc.)

Data define instruction is another important feature, there is no API calls in some malware sample, it contains an operation code like 'db' (defining byte), 'dw' (defining word) and 'dd' (defining double word) that used for packing of malware detection. For each line in the assembly file starts with dot and the assembly name like '.text', '.data', '. bss', '. rdata', '. rsrc, the section feature is used. The executable file has some sections. The study (Mansour Ahmadi, 2016) pointed out that the modification of some sections and generation can be applied on the unknown section names, then counting the common sections and calculating the recorded property of the unknown sections. Miscellaneous is also an important feature that has been used in the literature. It is an assembly code that taken as a keyword and extract the frequency of 95 chosen keywords for feature category, it some part of them indicates the number of blocks or the number of loaded DLL header as in (Mansour Ahmadi, 2016).

Also, Metadata it extracts and summarize the basic information of the file, such as file size, address, date modified or date created, (Mansour Ahmadi, 2016) describes that the PE header gives us a meta data information for the executable file. The study (Kun Wang, 2016) proposed a malware detection method depends on metadata of App that extracted from APK file and build vector space for datasets. In addition, the approach in (Mansour Ahmadi, 2016) explained that the entropy technique based on describing the measurement of disorder, it is a numerical measure, the study (Jared Lee, 2015) used the entropy information to interact with metamorphic detection problem, computing Entropy become on the byte level representation of the Malware sample, to measure the disorder in the byte distribution in byte code.

Moreover, image representation (Mansour Ahmadi, 2016) showed that the malware sample can be visualized the byte code and explain each byte as gray-level of pixel in the image, this technique used in visual signature By patterns similarity, (Kyoung Han, 2014) proposed a method for classification using image representation by converting binary files into images and entropy graph. As in the studies (Mansour Ahmadi, 2016) (Yanfang Ye, 2017), the string transfer the Hex byte file to ASCII string from PE because some strings are not clean enough, so taking the string length better than the string and counted as features for malware classification.

In terms of the features extraction from assembly files, there are several studies used various features such: (1) metadata, (2) symbol, (3) operation code, (4) registers, (5) application program interface, (6) data define instruction, (7) section, and (8) miscellaneous.

Metadata is identical to metadata in hexadecimal file depend on (Mansour Ahmadi, 2016) the technique done by computing and extracting the information about the file as assembly file features, and the size of the binary code file as features but after being disassembled , the two sizes will be different but both of them are recorded.

The study (Mansour Ahmadi, 2016) used symbol to calculate the frequency of the symbols and check the high frequent that used to prevent and avoid malware detection. Also, the operation code it is a representation of machine code, also called instruction syllable and it is a part of machine language instruction that identify the operation to be executed and symbolize assembly instruction which uses 93 common operation code that utilized as features.

In addition, there is another approach which assigns a malware sample to a specific family by the frequencies of registers in processors; the registers are small storage in processors with high speed. Application programming interface, many studies focused on detecting malware and their types based on the nature of API calls (Dong-Jie Wu1, 2012) (Zhu, 2013). Also, the study in (Mansour Ahmadi, 2016) identified 794 most frequent APIs in malware that obtained from 500K malicious samples, the type, number and frequency of the different APIs can be also used to uniquely classify the malware. This can vary by the platform (e.g. Windows, Apple, Linux, Android, etc.). Data define instruction is another important features, there is no API calls in some malware sample, it contains an operation code like ‘db’ (defining byte), ‘dw’ (defining word) and ‘dd’ (defining double word) that used for packing of malware detection.

For each line in the assembly file starts with dot and the assembly name like ‘.text’, ‘.data’, ‘.bss’, ‘.rdata’, ‘.rsrc, the section feature is used The executable file have some sections,. The study (Mansour Ahmadi, 2016) pointed out that the modification of some sections and generation can be applied on the unknown section names, then counting the common sections and calculating the recorded property of the unknown sections. Furthermore, miscellaneous is also an important feature that has been used in the literature (Mansour Ahmadi, 2016). It is an assembly code that taken as a keyword and extract the frequency of 95 chosen keywords for feature category, it some part of them indicates the number of blocks or the number of loaded DLL header.

A. Detection based on API calls

Many research papers focused on evaluating the ability to detect malwares and their types based on the nature of API calls used by the malware such as: (Wu et al., 2012), (Ahmadi et al., 2016), (Peiravian and Zhu, 2013), (Aafer et al., 2013), and (Sami et al.,2010).

(Ahmadi et al., 2016) identified 794 most frequent APIs in malwares are taken into account, which are obtained from approximate 500K malicious samples based on Microsoft Kaggle challenge in 2016. The type, number and frequency of the different APIs can be also used to uniquely classify the malware. This can vary by the platform (e.g. Windows, Apple, Linux, Android, etc.). Authors classified extracted features into: Hex dump-based features and features extracted from disassembled files. Figure 4 shows a sample of their results and some of the important features they found to detect malwares. Figure 5 shows an example of DLL imports for a popular Trojan.

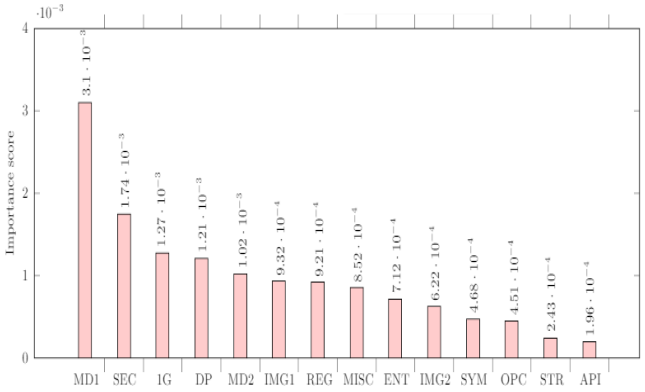


Fig. 2. Importance of features based on mean-decrease impurity

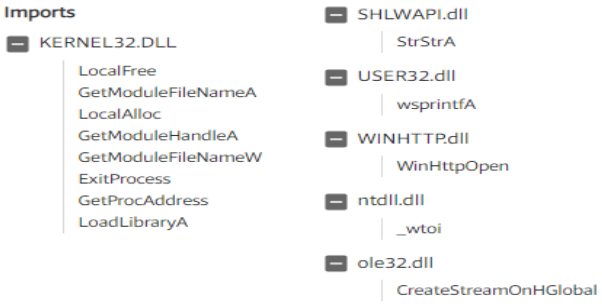


Fig. 3. DLL imports for a popular Trojan

In some cases, the frequency of usage of certain APIs can be an indicator of a malware behavior. Table 1 below shows (in a small dataset of 50 malwares and 150 Benign files) the top APIs used by malwares in comparison of Benign files.

TABLE I. Top APIs used by malwares in comparison of Benign files

API Name	Freq_W_%	Freq Ben%
exitprocess	0.96	0.49
closehandle	0.92	0.76
getmodulefilenamea	0.92	0.43
getmodulehandlea	0.88	0.78
writefile	0.86	0.67
getprocaddress	0.84	0.72
getstartupinfoa	0.82	0.53
createfilea	0.80	0.39
regclosekey	0.76	0.67
rtlunwind	0.76	0.45
virtualalloc	0.76	0.43
getcommandlinea	0.76	0.37

B. Entropy level Detection and Structural Features

Entropy is extracted from *.byte files and calculated at byte level. The sliding window method is adopted with the size of each window as 10,000 bytes According to the Shannon’s formula.

IV. Given a new malware, how can we classify this malware?

There are many learning techniques that applied to classify the malware in the literature (Mansour Ahmadi, 2016) & (George E. Dahl, 2013) such: (1) K-nearest neighbors, (2) Support vector machine, (3) Boosted Trees–XGBoost, (4) Neural Network, and (5) Naive Bayes. First, K-nearest neighbors is one of the most important machine learning technique that used for classification and regression, the basic idea is to predict a point by the neighbors set by measuring the distances, different distance measurements method used for finding the closest neighbors and the most used method is the Euclidean Distance and it is good for the features from the same type, for different type it is better to use Manhattan distance. Second, support vector machine is one of the supervised learning methods for classification, the main idea is to create a hyperplane that separate the classes in some way and to find the hyperplane with the maximum margin, the distance between the support vector and the hyperplane is referred to as margins. Third, Extreme Gradient Boosting (Tianqi Chen, 2016) is based on the model of Gradient Boosting, the XGBoots is a tree ensemble which incorporates a batch of classification and regression trees (CART), used to achieve state of the art results on many machine learning challenges.

The study (Tianqi Chen, 2016) used XGBoosting as a classification method and the analysis of the training set (20,000 samples) and then test the performance against testing set (2,000 unseen samples) shows that XGBoost Model lead to a better accuracy compared to the other models in the research. Fourth, Multi-Layer Perceptron (MLP), it is another supervised learning algorithm based on training dataset by a function and input layer and output layer and a several non-linear hidden layers, the first layer is the input layer made up of a set of neuron which represent the set of features, each neuron from the previous layer in an output value and each value contains two parts, a weighted linear summation and a non-linear activation function. The approach (Joshua Saxe, 2015) introduces a deep neural network based on malware detection system by using an experimental dataset on 400,000 software binaries with a detection rate of 95% and a false positive rate of 0.1%. Fifth, Naive Bayes is probabilistic classifier algorithm based on Bayes theorem, this method based on treating and evaluate the probability of each feature independently and make the prediction based on Bayes theorem, Naive Bayes classifier (Nikola Milosevic, 2017) used to detect malicious android application but the Naïve Bayes have the worst performance in that study.

V. If a malware has common features from different known malware categories, to which category such malware is usually allocated?

Automatic malware categorization plays an essential role in identifying the big size of malware. Different malware detection uses common features in different categories. Some of the malicious files belong to the same family with the same malicious behavior and features, there are some techniques to deal with these files and detect the malicious correctly. In (Jiang Y. Z.-C.-Q.-Y.-Z.-J., 2017) the study proposed a method based on a mixture model clustering ensemble to make an effective malware clustering analysis and system by combining some different features and clustering algorithms.

Another study (Jiang Y. Y.-T.-Y.-Q., 2010) proposed a principled cluster ensemble framework by automatic malware categorization system to group the malware samples into families with a common characteristic, by combining individual clustering solutions. Also, the study (Xin Hu, 2013) improves the approach that used previously in (Jiang Y. Y.-T.-Y.-Q., 2010) by making a combination between static and dynamic features and integrates the results with a similarity metrics. Furthermore, the study (Blake Anderson, 2012) proposed a new technique using kernels for the similarity metric on each special view and multiple kernels learning to get the best classification accuracy by the support vector machine, it used all the information about available execution to perform classification not just by a single data source.

VI. How do malware scanners decide a malware category?

Malware scanners employ one of the following 3 major categories of detection methods: (1) Specific Detection (e.g. signature or hash-based detection): This works by looking for known malwares by a specific set of attributes. Known malwares have known signatures recorded in a database that can be applied on subject or tested files, (2) Generic Detection: This process looks for malwares by behaviors of known categories or that are variants of known categories. For example, statistical-based techniques apply statistical models on system activities such as network connections, bandwidth, memory usage, system calls, etc. which can be usually used by malwares. Apparently, false positive cases are common in such scenarios where many “good” applications or system behaviors can be mistaken as malicious activities and (3) Heuristic, anomaly or behavioral detection: This process scans for previously unknown malwares by looking for known abnormal or suspicious behaviors. Anomaly-based detection depends on monitoring system activities and classifying the subject as either normal or anomalous accordingly.

Detection can be also classified into host-based and network-based detection based on the sources of artifacts used in the analysis and detection processes. Hybrid (host- and network-based) is also possible especially with large and complex malwares.

Dynamic detection methods run the suspect file in an isolated environment. Three types of isolated environments are used: Sandboxes where special APIs connect suspect file to the Virtual Machine (VM). Alternatively, emulation or Virtual Machine Monitors (VMMs) are used to run the suspect file. From a data science perspective, malware detection process is a typical classification process of two stages:

1. In the first stage, subject file, traffic, hash, etc. will be classified as either a malware or benign. Many malware scanners leave a third category: undecided if the subject fails to be allocated to either one of the malwares/benign categories. For example,

many of the newly discovered malwares may fall under this (unknown/undecided) category.

2. In the second stage, if the subject is classified as a malware, different categories of malwares are available and the process is to allocate this subject to one, or more of those categories.

The analysis of the malicious program is important to extract features, which describes the risk and the malware type; there are three types of detection and analysis method to identify the malware categories: (1) static analysis detection technique, (2) dynamic analysis detection, and (3) hybrid analysis detection technique.

First, Static analysis detection technique (Imtithal A.saeed, 2013) (Smita Ranveer, 2015) (Ekta Gandotra, 2014) (Dolly Uppal, 2014) (Balaji Baskaran, 2016) (Saba Arshad, 2017) (P. V. Shijoa, 2015), these studies described the static analysis that is analyzing software malicious and extract the features in the binary code or internal structure of the file without executing, the application is break down by some tools and techniques to rebuild the source code and algorithm of the application that created, it is done during program analyzer and debugger, this type is safe and fast. There are different static analysis techniques: Specific detection (e.g. signature or hash-based detection), and heuristic behavioral detection.

Specific detection works by looking for known malware by a specific set of attributes (Imtithal A.saeed, 2013) (Smita Ranveer, 2015) (Dolly Uppal, 2014) (Saba Arshad, 2017),these approaches indicated that known malware have known signatures recorded in a database that can be applied on subject or tested files, but it cannot detect an unknown malware. While Heuristic behavioral detection: It called proactive technique, it is similar to signature based but it does not use searching for signature in code, it search for the instruction that is not appear in the application program (Imtithal A.saeed, 2013) (Dolly Uppal, 2014) they proposed that this process scans for previously unknown malware by looking for known abnormal or suspicious behaviors. Anomaly-based detection depends on monitoring system activities and classifying the subject as either normal or anomalous accordingly.

Heuristic detection technique (Dolly Uppal, 2014) have different types such: (1) File based heuristic analysis file based or file analysis Heuristic system, it analyzes the file completely and check if there is any command in the file can delete or harm other files, it will be considered as malicious. (2) Weight based heuristic analysis, this is the oldest technique, each application have a danger weight or value, and there is a threshold value, if the weight override the threshold value the application will contain a malicious code, (3) Rule based heuristic analysis at this type, the analyzer extract the rules of the application, and match it with the previously defined rules. Such if there is any mismatching then the application contains malware, and (4) Generic signature analysis, this process looks for malware by behaviors of known categories or that are variants of known categories. different behavior for the malware but belongs to the Same category used to discover new variant of malware, for example, statistical-based techniques apply statistical models on system activities such as network connections, bandwidth, memory usage, system calls, etc. which can be usually used by malware. Apparently, false positive cases are common in such scenarios where many “good” applications or system behaviors can be mistaken as malicious activities.

Second, Dynamic analysis detection technique (Ekta Gandotra, 2014) (Dolly Uppal, 2014) (KIMBERLY TAM, 2017) (Saba Arshad, 2017) discuss that the dynamic analysis observes the result after executing the program by Analyzing the behavior or the Action of the application but it takes time as the executing time of applications. It interacts with the system while execution in VMware, Simulators and sandbox to find if the executable file is malware or not.

Third, Hybrid analysis detection technique (Ekta Gandotra, 2014) (Dolly Uppal, 2014) (Balaji Baskaran, 2016) (KIMBERLY TAM, 2017) they proposed that the hybrid analysis is combination of static and dynamic techniques by checking if there is any malware signature in the code. Then observe the code behavior. In addition, detection can be also classified (Kyoung Han, 2014) into three categories: (1) Host based intrusion detection system, (2) network-based intrusion detection system, and (3) Hybrid intrusion Detection system.

Host based intrusion observes and controls the dynamic behavior and the computer system state to check if there is any internal or external Activities that cheats the system policy. Network-based intrusion detection system analyzes all the packets in the network node. Host-based and network-based detection based on the sources of artifacts used in the analysis and detection processes. Hybrid intrusion Detection system: A combination of host- and network-based intrusion detection system is also possible especially with large and complex malware.

Dynamic detection methods run the suspect file in an isolated environment. The research study (Imtithal A.saeed, 2013) proposed isolated environment, sandboxes, where special APIs connect suspect file to the Virtual Machine (VM).From a data science perspective, malware detection process is a typical classification process of two stages: In the first stage, subject file, traffic, hash, etc. will be classified as either a malware or benign. Many malware scanners leave a third category: undecided if the subject fails to be allocated to either one of the malware/benign categories. For example, many of the newly discovered malware may fall under this (unknown/undecided) category. In the second stage, if the subject is classified as a malware, different categories of malware are available and the process is to allocate this subject to one, or more of those categories.

VII. Given a new malware, how can we classify this malware?

There are many learning techniques that applied to classify the malware in the literature (Mansour Ahmadi, 2016) & (George E. Dahl, 2013) such: (1) K-nearest neighbors, (2) Support vector machine, (3) Boosted Trees–XGBoost, (4) Neural Network, and (5) Naive Bayes.

First, K-nearest neighbors is one of the most important machine learning technique that used for classification and regression, the basic idea is to predict a point by the neighbors set by measuring the distances, different distance measurements method used for finding the closest neighbors and the most used method is the Euclidean Distance and it is

good for the features from the same type, for different type it is better to use Manhattan distance.

Second, support vector machine is one of the supervised learning method for classification, the main idea is to create a hyperplane that separate the classes in some way and to find the hyperplane with the maximum margin, the distance between the support vector and the hyperplane is referred to as margins.

Third, Extreme Gradient Boosting (Zhu, 2013) is based on the model of Gradient Boosting, the XGBoots is a tree ensembles which incorporates a batch of classification and regression trees (CART), used to achieve state of the art results on many machine learning challenges . The study (Tianqi Chen, 2016) used XGBoosting as a classification method and the analysis of the training set (20,000 samples) and then test the performance against testing set (2,000 unseen samples) shows that XGBoost Model lead to a better accuracy compared to the other models in the research.

Fourth, Multi-Layer Perceptron (MLP), it is another supervised learning algorithm based on training dataset by a function and input layer and output layer and a several non-linear hidden layers, the first layer is the input layer made up of a set of neuron which represent the set of features, each neuron from the previous layer in an output value and each value contains two parts, a weighted linear summation and a non-linear activation function. The approach (Joshua Saxe, 2015) introduced a deep neural network based on malware detection system by using an experimental dataset on 400,000 software binaries with a detection rate of 95% and a false positive rate of 0.1%. Fifth, Naive Bayes is probabilistic classifier algorithm based on Bayes theorem, this method based on treating and evaluate the probability of each feature independently and make the prediction based on Bayes theorem, Naive Bayes classifier (Nikola Milosevic, 2017) used to detect malicious android application but the Naïve Bayes have the worst performance in that study.

VIII. Dealing with Malware Detection Differences or Discrepancies

If malware has common features from different known malware categories, to which category such malware is usually allocated?

Automatic malware categorization plays an essential role in identifying the big size of malware. Different malware detection uses common features in different categories. Some of the malicious files belong to the same family with the same malicious behavior and features, there are some techniques to deal with these files and detect the malicious correctly. In (Jiang Y. Z.-C.-Q.-Y.-Z.-J., 2017), the study proposed a method based on a mixture model clustering ensemble to make an effective malware clustering analysis and system by combining some different features and clustering algorithms.

Another study (Jiang Y. Y.-T.-Y.-Q., 2010) proposed a principled cluster ensemble framework by automatic malware categorization system to group the malware samples into families with a common characteristic, by combining individual clustering solutions. Also, the study (Xin Hu, 2013) improves the approach that used previously in (Jiang Y. Y.-T.-Y.-Q., 2010) by making a combination between static and dynamic features and integrates the results with a similarity metrics. Furthermore, the study (Blake Anderson, 2012) proposed a new technique using kernels for the similarity metric on each special view and multiple kernels learning to get the best classification accuracy by the support vector machine, it used all the information about available execution to perform classification not just by a single data source.

Different approaches and features have used by many researchers in malware detection area to detect malwares. In (Idika et al., 2007) malware detection techniques categorized based on two general main categories: Signature based detection and anomaly-based detection. Signature based detection technique utilizes the known malicious software characteristics in deciding the malicious of the software under inspection. Anomaly based detection uses the software awareness that creates ordinary behavior to detect the maliciousness of the software under examination. Specification based technique is a special type of anomaly-based detection where some rule set and specification of legitimate behavior are influenced in order to detect the maliciousness of the software under inspection.

There are three different approaches are employed in every detection technique namely: dynamic, static and hybrid. Each approach is determined through specifying the information gathering technique in order to detect malicious software. Static approach usually is used to detect the malicious software before execution, whereas the dynamic approach is used to detect the malicious software during software execution (Idika et al., 2007).

Signature-based methods are the utmost popular methods in malware detection (Gutmann, 2007). Signature is like a pattern of an executable file and represents a distinctive feature for every file. Signature based method uses the extracted features from many malwares to classify them and is considered faster and efficient than other approaches. Signatures are extracted primarily with distinctive sensitivity for remaining sole, therefor signature based methods have lesser error rate (Gutmann, 2007). On the other hand, Signature based methods have not the ability to detect unidentified malware variations and need great effort of time and money to extract distinctive signatures. Moreover, incapability to meet malware that mutates its code in every infection like polymorphic and metamorphic is considered extra disadvantage (Gutmann, 2007).

Behavior based methods monitor program behavior to determine whether the software is malicious or not (KALPA, 2011). Behavior based mechanisms have the ability to detect malwares that keep on creating new mutants because always they will use the system services and resources in the same way (Jacob et al., 2008). The main disadvantage of the behavior-based malware detection methods is huge volume of scanning time. The behavior-based detector consists from three main components: Data collector, Interpreter, and matcher. The major improvement of the behavior-based malware detection technique is the capability to determine the type of malware where the variant of malware is polymorphic or unknown (Ahmed et al., 2012).

IX. Conclusion

There are several factors that make the process of malware detection and classification complex. Malwares evolve rapidly and hackers continuously employee new methods to avoid detection or manipulate the analysis and classification activities. The malware categories themselves are not clearly identified in such form that make the process of detection and distinction between the different categories straightforward. In this scope, we conducted this survey paper to raise and answer questions related to malware detection and classification issues. In addition to previous challenges, malware detection should be quick and close to real time detection and eradication. It should also be accurate to minimize false positive and negative cases.

References

Akour, M., Alsmadi, I., & Alazab, M. (2017). The malware detection challenge of accuracy. In 2016 2nd International Conference on Open Source Software Computing, OSSCOM 2016 [07863750] Beirut, Lebanon: IEEE, Institute of Electrical and Electronics Engineers. DOI: 10.1109/OSSCOM.2016.7863676.

Kyoung. Soo Han (2014). Malware analysis using visualized images and entropy graphs. International Journal of Information Security, 14(1), 1-14.

Balaji Baskaran, A. R. (2016). A Study of Android Malware Detection Techniques and Machine Learning. MAICS, 15-23.

Blake Anderson, C. S. (2012). Improving Malware Classification: Bridging the. Proceedings of the 5th ACM workshop on Security and artificial intelligence - AISec 12.

Dolly Uppal, V. (2014). Basic survey on Malware Analysis, Tools and Techniques. International Journal on Computational Science & Applications, 4(1), 103-112.

Dong-Jie Wu1, C.-H. M.-E.-M.-P. (2012). DroidMat: Android Malware Detection through Manifest and API Calls Tracing. Seventh Asia Joint Conference on Information Security.

Ekta Gandotra, D. B. (2014). Malware Analysis and Classification: A Survey. Journal of Information Security, 5(2), 56-64.

George E. Dahl, J. W. (2013). LARGE-SCALE MALWARE CLASSIFICATION USING RANDOM PROJECTIONS AND NEURAL NETWORKS. IEEE International Conference on Acoustics, Speech and Signal Processing.

Hossein Fereidooni, M. C. (2016). ANASTASIA: ANdroid mAlware detection using STatic analySIs of Applications. 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS).

Imtithal A. saeed, A. S. (2013). A Survey on Malware and Malware Detection Systems. International Journal of Computer Applications, 67(16), 25-31.

Jared Lee, T. H. (2015). Compression-based analysis of metamorphic malware. Int. J. Security and Networks, 10(2), 124-136.

Jiang, Y. Y.-T.-Y.-Q. (2010). Automatic Malware Categorization Using Cluster Ensemble. Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 10.

Jiang, Y. Z.-C.-Q.-Y.-Z.-J. (2017). Based on Multi-features and Clustering Ensemble Method for Automatic Malware Categorization. IEEE Trustcom/BigDataSE/ICESS.

Joshua Saxe, K. B. (2015). Deep Neural Network Based Malware Detection Using Two-Dimensional Binary. 10th International Conference on Malicious and Unwanted Software (MALWARE).

KIMBERLY TAM, A. F. (2017). The Evolution of Android Malware and Android Analysis Techniques. ACM Computing Surveys, 49(4), 1-41.

Kun Wang, T. S. (2016). Mmda: Metadata based Malware Detection on Android. International Conference on Computational Intelligence and Security.

Mansour Ahmadi, D. U. (2016). Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification. Proceedings of the Sixth ACM on Conference on Data and Application Security and Privacy - CODASPY 16.

Meena, S. J. (2011). Byte Level n–Gram Analysis for Malware. Communications in Computer and Information Science Computer Networks and Intelligent Computing, 51-59.

Nikola Milosevic aMilosevic, A. D.-K. (2017). Machine learning aided Android malware classification. Computers & Electrical Engineering, 61, 266-274.

P. V. Shijoa, A. S. (2015). Integrated static and dynamic analysis for malware detection. Procedia Computer Science, 46, 804 – 811.

Saba Arshad, A. K. (2017). Android Malware Detection & Protection: A Survey. International Journal of Advanced Computer Science and Applications, 7(2), 2013-2017.

Smita Ranveer, S. H. (2015). Comparative Analysis of Feature Extraction Methods of Malware Detection. International Journal of Computer Applications, 120(5), 1-7.

Tianqi Chen, C. G. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 16.

Xin Hu, K. G. (2013). DUET: Integration of Dynamic and Static Analyses for Malware Clustering with Cluster

Ensembles. Proceedings of the 29th Annual Computer Security Applications Conference on - ACSAC 13.

Yanfang Ye, T. L. (2017). A Survey on Malware Detection Using Data Mining Techniques. ACM Computing Surveys, 50(3).

Zhang Fuyong, Z. (2017). Malware Detection and Classification Based on ngrams Attribute Similarity. IEEE International Conference on Computational Science and Engineering.

Zhu, N. P. (2013). Machine Learning for Android Malware Detection Using Permission and API Calls. IEEE 25th International Conference on Tools with Artificial Intelligence.

Imtithal A. Saeed, Ali Selamat, and Ali Abuagoub, A Survey on Malware and Malware Detection Systems, International Journal of Computer Applications (0975 – 8887), Volume 67– No.16, April 2013.

P. Gutmann. "The Commercial Malware Industry.", In DEFCON conference, 2007.

Idika, Nwokedi & Mathur, Aditya. (2007). A survey of malware detection techniques. Purdue University.

KALPA, "Introduction to Malware", "http://securityresearch.in/ index.php/projects/malware_lab/introduction-to-malware/8/", 2011.

G. Jacob, H. Debar, and E. Filiol, "Behavioral detection of malware: from a survey towards an established taxonomy," Journal in Computer Virology, pp. 251–266, 2008.

A. Ahmed, E. Elhadi, M. A. Maarof and A. H. Osman, "Malware Detection Based on Hybrid Signature Behaviour Application Programming Interface Call Graph Information Assurance and Security Research Group." Journal, A., Sciences, A., & Publications, S., Faculty of Computer Science and Information Systems, 9(3), 283–288,2012