Detecting Cyber Security Vulnerabilities through Reactive Programming

Radmila Juric
University of Southeast Norway
radmila.juric@usn.no

Karoline Moholth McClenaghan
University of Southeast Norway
kmoholth@usn.no

Ole Christian Moholth
Kongsberggruppen, Norway
ole.christian.moholth@kongsberg.com

Abstract

We propose a software architectural model, which uses reactive programming for collecting and filtering live tweets and interpreting their potential correlation to software vulnerabilities and exploits. We aim to investigate if we could discover the existence of exploits for disclosed vulnerabilities in Twitter data streams. Reactive programming is used for performing filtering and querying of tweets to find potential exploits. The result of processing Twitter data streams with reactive programming could be broadcast, by pointing towards potential exploits, which might create a cyber-attack.

1. Introduction

We have been interested in analyzing information available in overt and Open Source Intelligence (OSINT) sources, which originate in live and user generated data, available through social media and Twitter in particular. Our aim is to detect possible cyber security threats in such sources, when disclosed vulnerabilities are found to be related to exploits which could materialize.

We have been using various contextual scenarios in which live and socially generated data play an important role in creating alerts in humanitarian catastrophes, addressing incidents which affect populations, and recognizing natural and health hazards [10, 14, 15, 17, 28]. There is significant evidence that Twitter data is one of the best available OSINT sources [6, 9, 21, 22]. The Twitter platform plays as important a role in detecting cyber security threats as it may have in addressing humanitarian crises and resilience [16, 28].

In this paper we look at software solutions which could detect vulnerabilities in Twitter data and associate them with potentially dangerous exploits.

The novelty of our work is threefold.

We first exploit the Reactive Programing Manifesto [29] and use reactive programming functions for processing live twitter data streams according to their content. In general, reactive programming has not been used for addressing problems in cyber security.

Second, we wanted to move away from predictions and sentiments, automatically extracted from collected tweets, when detecting vulnerabilities and exploits [5]. The Reactive Programming Manifesto enables us to create and run queries upon tweeter data streams, which may answer any questions we may have, at the time we collect tweets. We can come close to tweets, at the time when they are posted, and assess potential exploits even before vulnerabilities have been officially disclosed.

Third, we know that reactive programming supports constant changes in the way we collect and prepare tweets for further processing, if necessary. It is centered on the data model, i.e. collected data streams of tweets, and reacts to their content.

The results of using the Reactive Programming Manifesto upon twitter data streams could feed any type of OSINT. Tweets queried by reactive programing can also be processed further. We can use Big Data (BD) technologies, Artificial Intelligence (AI) algorithms, and Semantic Web Technologies (SWT), to achieve a high level of precision in interpreting and categorizing vulnerabilities [12, 26]. Therefore, manipulating tweets with reactive programming languages will create valuable data sets which should contain semantics related to vulnerabilities and exploits. We can streamline them to any other computational environment for further processing.

By avoiding immediate and automatic generation of Tweet sentiments, using BD technologies or Machine Learning (ML), we become independent in the way we wish to understand the semantic in tweets. We are also independent from various tools and algorithms, which collect tweets, because we created our own rules and modes of "listening" to twitter data streams. We filter and query them through the reactive

URI: https://hdl.handle.net/10125/60157
ISBN: 978-0-9981331-2-6

programing language's functions, chosen on an ad-hoc basis, while still "listening" to and observing incoming tweets. The output of our filtering and querying can be an input to AI algorithms and reasoning with SWT.

To demonstrate the efficiency of reactive programming in the manipulation of twitter data streams, we propose a software architecture, which accommodates computations based on more accurate but constantly changeable data. Computing with reactive programming platforms is an answer for addressing the temporal nature of detecting vulnerabilities and exploits in live and user generated data. The paper is organized as follows.

In section 2 we overview related work by looking at papers focusing on semantic co-relation between disclosed vulnerabilities and exploits, which may materialize. We also overview the role of live and socially generated data, from the Twitter platform in particular, which may play an important role in detecting vulnerabilities. There are no related works close to ours and therefore we touch a few publications on the application of reactive programming in cyber security. In section 3 we introduce the principles of the Reactive Programing Manifesto and define the software architectural model which allows us to process twitter data streams within the ReactiveX [30] platform. Section 4 gives the scenario in which we place our experiment and illustrates queries, run upon the collected tweets, and their results to detect vulnerabilities and exploits. This has been carried out through the software application generated from the proposed architectural model. In Conclusions we debate future pathways based on the results of this research.

2. Related work

Published research on cyber security is vast. However, there are no established threads, which illustrates what we have achieved in protecting networks, companies, government and private people, against constant production of continuous and evolving cyber-attacks of different nature and for different purposes. For this paper we systemize publications on cyber security into three groups. In the first, we look at sources of information (overt and covert) which deal with vulnerabilities of our software and networks and focus on exploit detections. In the second, we deal with the power of social media and online discussion for detecting and interpreting the possibilities of exploit detections, using various technologies including SWT. In the third group we talk about the role of the Reactive Programming Manifesto which has a slightly different approach in addressing cyber security threats.

2.1 Vulnerabilities and Exploits

There has been a steady interest in collecting and addressing the existence of numerous vulnerabilities [3,4,24] of software and networks, aiming at protecting us against exploits [1]. They are very important for generating shared Common Vulnerability and Exposure (CVE) databases [7]. The work of [33] explores the opportunity of early exploit detection using information available on Twitter, which existed before vulnerability disclosures. They identified features extracted automatically from twitter discourse and introduced a threat model, based on the earliest experience of using socially and live generated data for detecting exploits. In [8], the authors apply ML to make automatic predictions for unseen vulnerabilities based on previous exploit patterns. The authors of [39] overview various techniques used in vulnerability assessment and penetration testing (VAPT), aimed at increasing cyber security awareness in organizations. The authors of [23] use Wikitology, to extract concepts that describe specific vulnerabilities and attacks, map them to related concepts from DBpedia and generate machine understandable assertions. They can be added to already existing vulnerability descriptions.

2.2. Social Media and Cyber Threat Detections

We have been aware of increased interest in using the Twitter platform for detecting early warnings of cyber threats in various online discussions. They are not focused solely on vulnerabilities and exploits. In this scenario Twitter appears to be a winning platform where any information on potential, new or old cyber-attack are placed, debated and even accompanied with appropriate advice and potential plan for resilience. We have found two interesting papers in this domain. In [37] the authors introduce a lightweight framework that leverages social media sensors, such as Twitter and *darkweb* forums, and generates alerts of potential cyber threats. They scan tweets for vulnerabilities and other cyber threats and apply text mining techniques for identifying important terms, which in turn are juxtaposed to the terms identified when filtering discussions on *darkweb* forums. This approach may guarantee the credibility of information available on Twitter and provide semantic contexts essential for interpreting the meaning of collected tweets. The work available in [22] uses SWT and (Resource Description Framework (RDF) [31] with Semantic Web Rule Language (SWRL) [38] for creating the meaning of extracted intelligence from the twitter platform, relevant to cyber-attacks. Their CyberTwitter platform generates alerts, which can

feed various security systems. However, the most important outcome is a cyber security ontology which stores semantics of various cyber security threats and events. There are other works which created cyber security ontologies [20, 43, 44, 45], but we have found no evidence of their wider deployment in real life scenarios. They act as control vocabulary of terms based on Ontology Web Language (OWL) [25] and SWRL and do not use SWRL reasoning for adding ad-hoc meaning when creating cyber security intelligence and warning threats.

2.3. Reactive Programming for Detecting Vulnerabilities and Exploits

Reactive programming, as a declarative programming paradigm, is an ideal approach to deal with data streams and the propagation of changes they exhibit [2, 35, 41]. There are numerous software frameworks, which make it easier to create complex software applications by composition of asynchronous computing elements, including streams. This has led to a proliferation of new types of software applications, which are able to continuously process and interactively respond to various stimuli and are essential in the development of user interactive software [13, 34, 36, 46].

We could not find any published work which uses reactive programming to address the detection of exploits in twitter data streams. However, we would like to draw readers' attention to three papers, which advocate solutions close to our research and push forward the idea of computing real time event detection upon arriving data as continuous and changeable streams. In [18] the authors give an excellent overview of the application of the ReactiveX framework on the Android application for detecting specific events when driving cars. They capture data streams generated by sensors, but the way they process them with the ReactiveX framework does not differ from our own use of the platform. The implementation is lightweight because of the lower computational power of Android devices, but this is the same reason why our proposed software architecture generates an application, which is easy to create and run. We should agree that ReactiveX is very successful when creating event-based applications upon data streams but searching for vulnerabilities and exploits in incoming twitter data streams is not different (from the software engineering point of view) to detecting uncomfortable driving events in cars.

The authors of [19] and [32] are focusing on computer security events found within twitter data streams. Their preparation for listening to and finding events, which can compromise software security, is detailed and involves various technologies for classifying events and learning from them. We believe that their solutions could have benefitted from reactive programming to lift the burden of using ML and Natural Language Processing algorithms to reach the semantic buried in tweets.

3. Programming with ReactiveX and Tweet Manipulation

In ReactiveX [30] we can create event and data streams, combine and transform them with operators, subscribe to observable streams, and filter them according to our criteria. ReaciveX claims that they use "clean input/output functions over observable streams", which is sufficient to place functional programming in the heart of modern computations when we depend on an excessive amount of either sensor or live and user generated data. Streams are central to ReactiveX. They are cheap and ubiquitous, ideal for processing Twitter feeds and any other type of user and live generated data. Each stream may feed another stream, we can filter and merge them and map values from one stream to another. Therefore, ReactiveX has been a perfect framework in our experiments and efficient for collecting, filtering and querying tweets.

3.1. The Proposed Software Architectural Model

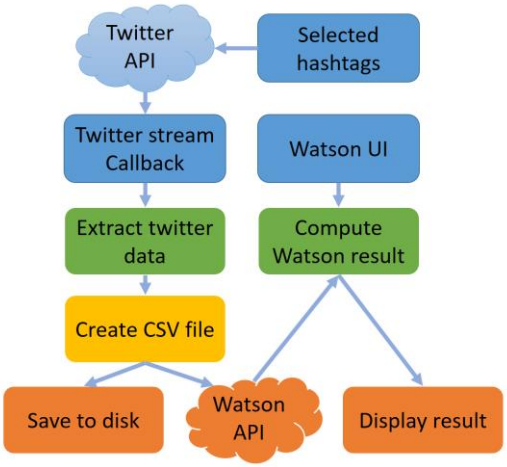

Figure 1: Architectural Model for Detecting and further Processing of Vulnerabilities and Exploits

Figure 1 shows a proposed software architectural model in which the left part of the Figure 1 deploys ReactiveX functions. We extract relevant twitter data streams, and store them in persistence. These streams

can be processed further with BD Technologies and Analytics, such as IBM Watson Studio. The model shows the simplicity of the architecture and synergy of applying ReactiveX functions and deploying BD analytics within the same software application.

In other words, through the selected # and Twitter API we can listen to twitter data streams and select them. The results can be either stored in a CSV repository or sent to the IBM Watson platform to perform further processing. We expect that CSV file will have sufficient semantics to inform us or IBM Watson if there are vulnerabilities discussed on the Twitter platform, and if so, which exploits might be materialised. Therefore, we leave open doors for any traditional processing or Twitter data streams as promoted in [19, 32, 33, 37].

However, we used the ReactiveX functions for getting involved in the content of incoming twitter data streams and Figure 2 shows exactly how ReactiveX was used in our architectural model.

In our experiment we wanted to observe if:

"there exist live tweets focusing on the semantic relationship between any software vulnerability(ies) and exploit(s)"

Through the twitter API, we subscribed to tweets matching our predefined filter: The filtering involves human intervention and therefore, keywords for filtering could have been picked from observing incoming twitter data streams using the Twitter API.

The subscription to tweets matching our filtering criteria results in a ***json*** object being generated and sent to our application when a status matches the filter criteria. This in turn triggers a call-back in the Java twitter API. At this stage, this ***json*** object is pushed into a Reactive Stream (publish subject), which allows us to use the status objects in a reactive stream. These streams can be immediately processed using ReactiveX functions. This means that semantics between vulnerabilities and exploits could be addressed further with ReactiveX.

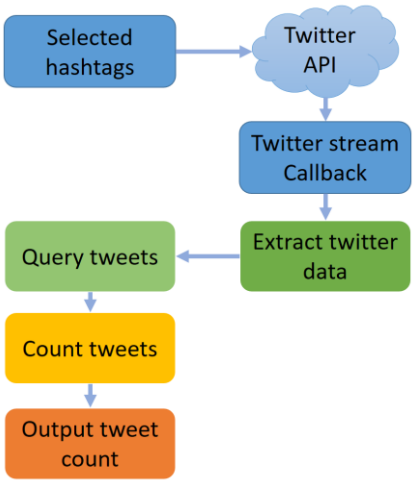

Figure 2: The Role of ReactiveX Framework in the Architectural Model of Vulnerabilities and Exploits

Figure 2 is an addendum to Figure 1. It shows the manipulation of tweets through ReactiveX. With ReactiveX functions, we can query reactive tweeter data streams and perform any type of tweet manipulation, which can help us to understand if the existence of a disclosed vulnerability is related to potential exploits.

Readers should note that computations in the left part of Figure 2 are constantly changeable. They depend on the choice of functions we pick from ReactiveX, which in turn is dictated by the content of incoming twitter data streams. Thus Figure 2 says that "we decided to query filtered tweets and count them for detecting vulnerabilities and exploits".

The architectural models from Figures 1 and 2 shows that we did NOT change the content of the incoming twitter data streams by processing them immediately with available ReactiveX functions. Their content is identical to the source, where they originate. Our filtered tweets are stored in the CSV repository as they were originally posted on the twitter platform. The content of CSV data files might not be the most suitable for getting an insight of "what exactly is tweeted about "#vulnerability and #exploits" using BD technology and Watson Analytics, but it keeps the semantics of original tweets intact. Therefore, these original tweets are queried and counted (in our experiment) in order to understand which semantics they bring within them.

We can also format them, at the same time when observing and querying them. Queries we ran would not change their content, i.e. the semantic of original tweets has not been processed. To summarise, Figure 2 says that our processing of tweets consisted of constant filtering, querying and counting of incoming tweeter data.

3.2. Filtering and Querying Tweeter Data Streams

It is obvious that the idea of using specific keywords in filtering and querying of tweets is one of the easiest approaches to extract more semantics from tweets, but it is not unreasonable:

Firstly, user input with specific keywords is a result of user's observations of tweets through Twitter API and ReactiveX. This brings more relevant semantics to the process of finding relationships between vulnerabilities and exploits in the collected tweets. It is important that user's intervention jumps in as a part of filtering and querying, because it also helps

to prepare tweets for either any further querying with ReactiveX or processing with BD technology.

Secondly, we know that we will be looking for "#vulnerability and #exploits" and therefore we (users) should supply more semantics in terms of more "keywords" which would help to understand which vulnerabilities and exploits are of interest to us and why. In this experiment we asked:

- Have we found any "#vulnerability and #exploits" while observing tweets?
- Has anything attracted our attention?
- Are there any new words which may describe vulnerability?
- Could we see any entry ID from CVE databases?
- Are some words, important for us, being repeated?
- Do we have too many retweets, and if so, what is going on?

This type of questions trigger more filtering and querying, which contribute towards discovering more semantics within collected tweets. This was essential for understanding if vulnerabilities currently tweeted are related to dangerous exploits or not [16, 26, 27]. The combination of user involvement through filtering and queries, and automated execution of ReactiveX functions can produce instant results, while we are still observing incoming tweeter data streams.

4. The Scenario and Experiment

In our Scenario, we had *proactive as opposed to reactive approach to cyber security* [40, 42] by detecting new or picking up old vulnerabilities in incoming tweeter data streams. The proactive approach to addressing cyber security means that we come very close to tweets at the time when they are posted. We are not waiting for someone to enter the semantic, relevant to vulnerabilities and exploits, within overt, OSINT and CVE database. We process this semantic from twitter data streams immediately, using tailored reactive programming environment as explained in Figure 2. We take a proactive approach to the detection of vulnerabilities and exploits but deploy ReactiveX framework as a part of the reactive programming environment. Thus we can understand and manage live and constantly changeable incoming twitter data streams, as they appear.

In April 2018 we observed the tweets from the Twitter platform through Twitter API/ReactiveX and included the keywords "*#vulnerability* and *#exploits*" to see if anything related to these two words was tweeted. We wanted to test if we could detect the existence of new vulnerability or exploits, which has not been documented in any OSINT, at the time when we detect them on Twitter. We also wanted to see if we can recognise tweets in the incoming data streams which talk about vulnerabilities already recorded in an OSINT, such as CVE database. Therefore, we approach the problem by defining a set of keywords, which would filter twitter data streams. We can then count them and draw conclusions on an important vulnerability. The result should show if any of them were discussed. The idea of filtering and counting tweets has been taken from our previous research in which the process of collecting and processing Twitter data was carried out by Hadoop Echo-system components, IBM Bluemix and the MapReduce algorithm interwoven within them [27]. Therefore, we were confident that "counting" tweets, according to our criteria through ReactiveX, would come very close to the results we obtained when using BD technology and addressing the extraction of sentiment in Twitter data [11].

Consequently, one of the best indicators for finding out if a vulnerability is being discussed or not, is if a high number of retweets, for a particular tweet, appears in a very short period of time, within our filtering criteria. Therefore, it is worthwhile to start with questions such as

1. How many tweets mention the word vulnerability?
2. How many tweets mentioned the word exploits?
3. How many tweets mentioned the word CVE?

There are numerous questions we could ask. Answers to them are instantly available through ReactiveX functions. They indicate that we either need to ask more questions (even perform more counting) or carefully examine the tweets which are tweeted the most. If there is no word CVE found in tweets, this would mean that tweets are talking about a particular vulnerability which has not been recorded in the CVE database. In that case, we should pay extra attention to the incoming twitter data streams and ask more questions, if we wish to learn more.

At the beginning of our experiment in April 2018, we did not know for how long we should "listen to twitter data streams" because we could not anticipate how quickly we could draw conclusions that *"something serious has been discussed"*. The questions above could have been asked at any time during the experiment, and further questions can be posed at any moment, on an ad-hoc basis. If the number of tweets we obtained from queries such as 1.-3. Above is high, considering the number of total tweets we collected, then it is prudent to increase the level of observation and create more queries.

When we started observing the incoming tweets, we did not find it necessary to run further queries because the tweets were not overwhelming us.

However, we quickly noticed the appearance of the CVE word, which had two implications:

a) the vulnerability discussed has already been reported and had its CVE database entry because its ID existed in tweets, and
b) we should be able to compare the semantics of this vulnerability between the content of the tweets and its description in the CVE database.

We performed a set of queries upon the collected tweets because we wanted to learn if the occurrence of this CVE entry would indicate how important the discussion about reported vulnerability was, and which exploits might materialise.

It is important to note that in our experiment, after finding out that there exists a particular entry to the CVE database in the content of incoming tweets, we started taking further keywords from both: filtered tweets, while observing them and scrolling them on the screen, and the description of the same vulnerability entered in the CVE database.

The timing of recording of this particular vulnerability in the CVE was not the same as when we conducted the experiment (CVE was recorded earlier). Also, the entry creation date of CVEs does not necessarily indicate when each vulnerability was discovered (this is the date when CVEID is allocated). Therefore, we had a gap of approximately 2 weeks between our experiment and the recorded date in the CVE database. However, we could quickly notice by scrolling incoming tweets (filtering only tweets which contained CVEID CVE-2018-1000136), without inspecting the CVE database, that our new keywords for running more queries should be "Signal Desktop", "Atom", "app", "remote code execution", because these words appeared very frequently, they were noticeable while scrolling the incoming tweeter data streams.

```
fileStream
    .map(String::toUpperCase)
    .filter(s -> s.contains("CVE-2018-1000136".toUpperCase()))
    .filter(s -> s.contains("Electron".toUpperCase()))
    .map(s -> 1)
          //Maps every tweet containing the words to 1
    .scan((previous, current) -> previous + current)
          //Adds one for every tweet
    .subscribe(count -> System.out.println("CVE-2018-1000136 AND Electron:\t"
                                + count));
          //Prints out tweet count for every tweet
```

Figure 3: A Sample Code of Filtering, Querying and Counting Incoming Twitter Data Streams

Figure 3 shows an excerpt of the code when using ReactiveX functions during filtering and querying tweets. It also shows the simplicity of coding and the way we "counted" relevant tweets.

4.1. Interpreting the Results

During our experiments we were able to react very quickly to any unusual number of tweeter data streams and analyse them while they were still being filtered and collected. As we mentioned earlier, the analysis had to have a certain level of human interaction, which is feasible to maintain through a simple web application. Figure 4 shows the prototype of our web application which hosts the software architectural models from Figures 1 and 2. We inspect tweeter data streams in user friendly environments, change the criteria for filtering, change keywords essential for our queries "as we go" and decide on the formatting of collected tweets. Black boxes in Figure 4 denote what is visible to the user. In Table 1 we give exactly what is visible within the OUPUT TWEET COUNT architectural component. The total number of tweets collected in 6 days in March 2018, initially filtered through *#vulnerability* and *#exploits* criteria was 6546, and the number of tweets which mention both words is relatively low, just 548 tweets. This is the main indicator that there was no excessive tweeting and retweeting about any particular vulnerability. Considering that there are 333 tweets which mention vulnerability CVE-2018-1000136, we could assume that the major vulnerability discussed here is CVE-2018-1000136. However, not all tweets used its CVEID. Some of them refer to it as "Electron" (Software, with vulnerability: *improper handling of values* in Webviews, which allows remote code execution (exploits)).

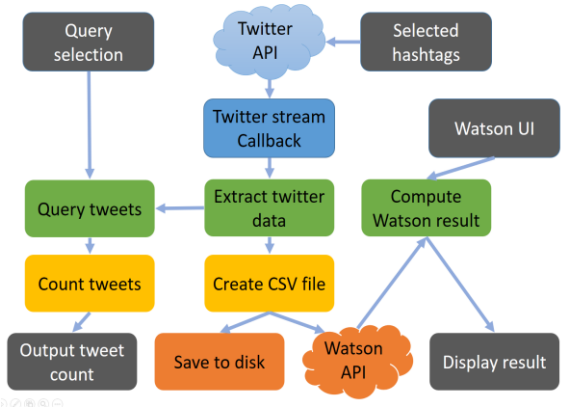

Figure 4: Architectural Model of the Implemented Web Application

We learned from Table 1 that

1) *Signal Desktop* and *Atom* are the applications affected and mentioned frequently (332 and 324 respectively) in the incoming tweets, but not recorded in the CVE database.

Table 1: Results of Queries with ReactiveX

Criteria	No of Tweets
Vulnerability	3172
Exploits	1985
CVE-2018-1000136	333
Remote code execution	81
Remote hacking	5
Signal Desktop	332
Electron	56
Atom	324
exploitable via an app	0
app	996
have been fixed	0
fixed	14
webview	0
CVE-2018-1000136 AND Remote code execution	0
CVE-2018-1000136 AND remote hacking	0
CVE-2018-1000136 AND Signal Desktop	316
CVE-2018-1000136 AND Atom	316
CVE-2018-1000136 AND fixed	0
CVE-2018-1000136 AND exploit	4
remote code injection AND electron	0
remote hacking AND electron	0
CVE-2018-1000136 AND Electron	19
CVE-2018-1000136 AND Signal Desktop AND Atom AND Remote code execution	0
CVE-2018-1000136 AND App	8
Vulnerability AND Exploit	548

2) Tweets are tweeting that either Atom or Signal Desktop are being affected. There are no tweets which include all: applications affected and the exploits (remote code execution).
3) Exploit "remote code execution" was not mentioned in tweets as much as we expected (just 81 times) which means that tweets focused directly on the vulnerability and applications affected by it. They probably categorised "remote code execution" because of the vulnerability, which does not have to be explicitly discussed.
4) Tweets are focused on either the Vulnerability or Exploits, and we very rarely find a complete explanation on the relationship between them, (only 548 tweets mentioned both words within the body of the tweet).
5) Tweets which give an overall description of the relationship between vulnerability, applications affected, and related exploits were not found.
6) Apart from connecting the vulnerability with affected applications (Atom and Signal Desktop) and associating the CVID of the vulnerability with Electron, we did not learn anything else from the tweets, except that the exploits are not to be ignored!
7) Finally, in the CVE database we learned that the attacks can be exploitable via an APP which allows the execution of third party code. However, the number of tweets which mention "app" and this vulnerability is extremely low (just 8). It is obvious that the applications affected are APPs and therefore there was no need to use the word in tweets.
8) The CVE database states that the vulnerability was fixed in some version of Electron, but there were no tweets on that.

To summarise: With constant observation of tweets, and their dynamic filtering and querying on an ad-hoc basis through ReactiveX we were able to learn what is happening with this particular vulnerability much quicker than by relying on the traditional OSINT, which store information in the persistence. We have found that there is more relevant and important information within the tweets than in the CVE database. The content of the tweets is not systemised and uniform, as entries to the CVE database are, but the information from Table 1 is more powerful than the text stored in the CVE database.

Constant observation of twitter streams, opens doors to various queries, which can change from one moment to another. This means that we learn about the potential vulnerabilities and exploits as we go, at the time when they are discussed on Twitter. Our set of keywords are changeable and influenced by the observation of incoming twitter data streams.

It is very important to note that our observation of live twitter data streams, at the beginning of the experiment, before we started running queries was noted as:

> **In the case of the CVE-2018-1000136 vulnerability, in tweets detected during the experiment, we could conclude quickly and without further processing that "*Signal desktop* app based on *Electron* might have dangerous exploit "*remote code execution*".**

This was unfolding in front of our eyes, it was an outcome of the queries available in Table 1, but it was also confirmed later, as the description of the **CVE-2018-1000136 vulnerability** in the CVE database.

5. Conclusions

In this research, we wanted to move away from popular and often effective ways of deriving sentiments from Twitter data to detect vulnerabilities, and potential dangerous exploits. It is not that we do not see the benefits of using BD technologies and AI algorithms, including ML when trying to identify and predict the content of tweets. There are so many effective solutions, which use powerful software technologies to address constant threats in our cyber spaces. Most of them use information available in live and user generated data. Twitter dominates in this market, by being a formal source of cyber security intelligence.

In our work, we wanted to create a software architectural solution, which comes close to twitter data streams and reacts to their content. This in turn invokes computations with ReactiveX functions around the data streams. Functional programming is an old but tried paradigm, which can be used here as a reactive programming platform, and thus allows us to focus on observing and manipulating incoming data streams.

These observations are crucial because they require user intervention and decision making. The User brings so much semantics to incoming twitter data streams and helps to identify if there is really something within these data streams which is of interest to us. ReactiveX functions, invoked on an ad-hoc basis, are illustrated as *filtering and counting* of data streams, through a carefully chosen set of key words. This idea has been used very often in BD technologies in which we perform counting by placing SQL queries upon NoSQL data, and the powerful MapReduce algorithm is not far away from it.

Therefore, we did not invent a new technology for exploring the semantic relationship between vulnerabilities and exploits, except placing reactive programing at the heart of the software solution. The answers to our question "*do we have any particular vulnerability discussed in tweets and if so, are there any exploits"* was unfolding in front of our eyes, supported by constant queries upon incoming data streams. We cannot ignore this. Without the architectures from Figures 1 and 2 and ReactiveX functions, we would not be able to filter and query tweets, while still observing them.

At the same time, we were also able to format tweets, which can easily enter BD and AI platforms for further processing. It is very likely that with formatting, using the framework from ReactiveX we could preserve as much semantic as possible, from the original tweets, and create data sets, which might be ready for BD analytics. The benefits are enormous: coming closer to the data reduces information overload, may clear data from unwanted semantics and embrace changes in data processing as we go. Having humans observing tweets and deciding on keywords for queries is a small price to pay for getting instant information on vulnerabilities, as they are tweeted.

Our prototype of web application from Figure 4 is operational but needs more work for commercializing it. We are currently exploring the automation of ReactiveX function selection, according to the incoming twitter data streams and formatting them to fit the processing required by Watson Analytics for Social Media. Preparing the CSV file from our architectural solutions, which can suit Watson Analytics can be performed with ReactiveX functions at any time during the process of collecting and filtering tweets.

Our future work includes the involvement of semantic technologies and SWT languages in particular which can create reasoning upon the content of CSV files from the architectural models. This will enable the creation of intelligent software solutions with synergies of the reactive programming, BD analytics and semantic technologies, housed within one software (web) application. The initial results we obtained are encouraging.

References

[1] L. Allodi, and F. Massacci, "Comparing Vulnerability Severity and Exploits Using Case-Control Studies ", ACM Transactions on Information and System Security, Vol. 17, No. 1, Article 1, August 2014.

[2] E. Bainomugisha, A. L. Carreton, T. van Cutsem, S. Mostinckx, and W. de Meuter, "A survey on reactive programming", ACM Comput. Surv. 45, 4, Article 52 (August 2013), 2013

[3] L. Bilge, and T. Dumitraş, "Before we knew it: an empirical study of zero-day attacks in the real world", In Proceedings of the 2012 ACM conference on Computer and communications security (CCS '12), New York, NY, USA, 2012.

[4] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits", In Proceedings of Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, Washington, DC, US, July 25–28, 2010, pp, 105-114.

[5] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection", IEEE Communications Surveys & Tutorials, vol. 18, no. 2, 2016, pp. 1153-1176.

[6] R. Campiolo, L.A. F. Santos, D. M. Batista and M. A. Gerosa "Evaluating the utilization of Twitter messages as a source of security alert", In Proceedings of the 28th Annual ACM Symposium on Applied Computing, Portugal, 2013, pp. 942-943.

[7] Common Vulnerabilities and Exploits, https://cve.mitre.org/.

[8] M. Edkrantz, S. Truvé, and A. Said, "Predicting Vulnerability Exploits in the Wild", In Proceedings of 2nd International Conference on Cyber Security and Cloud Computing, NY, US, 2015, pp. 513-514.

[9] Y. Erkal, M. Sezgin and S. Gunduz, "A New Cyber Security Alert System for Twitter", In Proceedings IEEE 14th International Conference on Machine Learning and Applications (ICMLA), FL, US, 2015, pp. 766-770.

[10] C. Everiss, J. Fenny, and R. Juric, "Ebola Crisis: An Investigation Into Levels Of Communication Following Vaccination", In Proceedings of the 20th International SDPS 2015 Conference, Texas, US, November 2015.

[11] J. D. Fotso, and R. Juric, "The Framework for Comparing Big Data Technologies in Processing Live and User Generated Data", In Proceedings of the 22nd International SDPS 2017 Conference, AL, US, Nov 2017.

[12] Y. Freeman, "Efficiency of Machine Learning in Cyber Security: A Study of Detection of Exploits for Disclosed Vulnerabilities within Open Source Intelligence", MSc Thesis, University of South East Norway, Department of Science and Industry systems, Kongsberg, Norway, 2018.

[13] T. Hartmann, A. Moawad, F. Fouquet, G. Nain, J. Klein, and Y. Le Traon, "Stream my models: Reactive peer-to-peer distributed models@run.time", In Proceedings of 18th International Conference on Model Driven Engineering Languages and Systems, Canada, 2015, pp. 80-89.

[14] R. Juric, and I. Kim, "Can Twitter Transform Communities Affected By E-BOLA", In Proceedings of the 20th International SDPS 2015 Conference, TX, US, 2015.

[15] R. Juric, and I. Kim, "Software Architectures For Smart Applications Which Merge Ontological Reasoning With Big Data Analytics", In Proceedings of the 22nd International SDPS 2017 conference, AL, US, 2017.

[16] R. Juric, I. Kim, Samsung, H. Panneerselvam, and I. Tesanovic, "Analysis Of Zika Virus Tweets: Could Hadoop Platform Help In Global Health Management", In Proceedings of the 50th HICSS Conference, HI, US, 2017.

[17] R. Juric, I. Kim, I. Tesanovic, and A. Sinkovic, "Raising Awareness of Dental Pain from Tweets", In Proceedings of the 19th International SDPS 2014 Conference, , Malaysia, 2014.

[18] Z. Jovanovic, R. Bacevic, R. Markovic and S. Randjic, "Android application for observing data streams from built-in sensors using RxJava," 2015 23rd Telecommunications Forum Telfor (TELFOR), Serbia, 2015, pp. 918-921.

[19] D. Kergl, "Enhancing network security by software vulnerability detection using social media analysis", 2015 International Conference on Data Mining Workshop (ICDMW), NJ, US, 2015, pp. 1532-1533.

[20] L. Leenen, and T. Meyer, "Semantic Technologies and Big Data Analytics for Cyber Defence", International Journal of Cyber Warfare and Terrorism (IJCWT) 6(3), 2016, pp.1-12.

[21] G. Lin, N. Sun, S. Nepal, J. Zhang, Y. Xiang, and H. Hassan, "Statistical Twitter Spam Detection Demystified: Performance, Stability and Scalability", IEEE Access, vol. 5, 2017, pp. 11142-11154.

[22] Mittal, P. K. Das, V. Mulwad, A. Joshi, and T. Finin, "CyberTwitter: Using Twitter to generate alerts for cybersecurity threats and vulnerabilities", In Proceedings IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), CA, US, 2016, pp. 860-867.

[23] V. Mulwad, W. Li, A. Joshi, T. Finin, and K. Viswanathan, "Extracting Information about Security Vulnerabilities from Web Text", In Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT '11), Vol. 3., Washington DC, US, 2011, pp: 257-260.

[24] K. Nayak, D. Marino, P. Efstathopoulos, and T. Dumitraş, "Some Vulnerabilities Are Different Than Others" Research in Attacks, Intrusions and Defences, Springer International Publishing, 2014, pp. 426-446.

[25] OWL available at https://www.w3.org/OWL/

[26] H. Panneerselvam, and R. Juric, "Evaluation Of Hadoop's Effiicency Through The Manipulation Of Live Data", In Proceedings of the 21st International SDPS 2016 conference, FL, US, 2016.

[27] H. Panneerselvam, R. Juric and J. D. Fotso", Asking Smart Questions About Your Big Data ", In Proceedings of the 21st International SDPS 2016 conference, FL, December 2016

[28] K. Pettai, R. Juric, and A. A. A. Bechina, "The Power of Microblogging in Disseminating Information in Humanitarian Crises: A Study of Nepalese Earthquake", In Proceedings of the 20th International SDPS 2015 Conference, TX, US, 2015.

[29] Reactive Programming Manifesto https://www.reactivemanifesto.org/ is available at https://www.reactivemanifesto.org/pdf/the-reactive-manifesto-2.0.pdf , Published on September 16 2014. (v2.0)

[30] ReactiveX Platform, http://reactivex.io/

[31] Resource Description Framework (RDF) available at https://www.w3.org/RDF/

[32] A. Ritter, E. Wright, W. Casey, and T. Mitchell, "Weakly Supervised Extraction of Computer Security Events from Twitter", in Proceedings of the 24th International Conference on World Wide Web, Italy, 2015, pp. 896–905.

[33] C. Sabottke, O. Suciu, and T. Dumitras, "Vulnerability disclosure in the age of social media: exploiting twitter for predicting real-world exploits", In Proceedings of the 24th USENIX Conference on Security Symposium, Washington, D.C., 2015.

[34] G. Salvaneschi, P. Eugster, and M. Mezini, "Programming with Implicit Flows", IEEE Software, vol. 31, no. 5, 2014, pp. 52-59.

[35] G. Salvaneschi, A. Margara, and G. Tamburrelli, "Reactive Programming: A Walkthrough", In Proceedings of the 2015 IEEE/ACM 37th International Conference on Software Engineering, Italy, 2015.

[36] G. Salvaneschi, S. Proksch, S. Amann, S. Nadi, and M. Mezini, "On the Positive Effect of Reactive Programming on Software Comprehension: An Empirical Study", IEEE Transactions on Software Engineering, vol. 43, no. 12, 1 Dec. 2017, pp. 1125-1143.

[37] A. Sapienza, A. Bessi, S. Damodaran, P. Shakarian, K. Lerman, and E. Ferrara, "Early Warnings of Cyber Threats in Online Discussions", In Proceedings of International Conference on Data Mining Workshops (ICDMW), New Orleans, LA, 2017, pp. 667-674.

[38] Semantic Web Reasoning Language (SWRL) available at https://www.w3.org/Submission/SWRL

[39] P. S. Shinde, and S. B. Ardhapurkar, "Cyber security analysis using vulnerability assessment and penetration testing", In Proceedings of World Conference on Futuristic Trends in Research and Innovation for Social Welfare, India, 2016, pp. 1-5.

[40] D. Solomon, "The End Of Reactive Security, And The Move To A Doctrine Of Cyber Defence", Cyber Security Review, Summer 2014 edition, Delta Business Media, 2014, available at http://www.cybersecurity-review.com/industry-perspective/the-end-of-reactive-security-and-the-move-to-a-doctrine-of-cyber-defence/

[41] A. Staltz, "The introduction to Reactive programming you have been missing", available at https://gist.github.com/staltz/868e7e9bc2a7b8c1f754

[42] R. Steinberger, "Proactive Versus Reactive Security", Computer Crime Research Centre, available at http://www.crime-research.org/library/Richard.html

[43] C. Strasburg, S. Basu, and J. S. Wong, "S-MAIDS: A Semantic Model for Automated Tuning, Correlation, and Response Selection in Intrusion Detection Systems", In Proceedings of IEEE 37th Annual Computer Software and Applications Conference, Japan, 2013, pp. 319-328.

[44] Z. Syed, A. Padia, T. Finin, L. Mathews, and A. Joshi, "UCO: A Unified Cybersecurity Ontology", The Workshops of the Thirtieth AAAI Conference on Artificial Intelligence Artificial Intelligence for Cyber Security: Technical Report WS-16-03, AZ, US, 2016.

[45] S. S. Tsneg, S.C. Lin, C.H. Mao, T. J. Lee, G. J. Qiu, and M. H. Lin, M.H. "An ontology guiding assessment framework for hacking competition", In Proceedings of the 10th International Conference on Ubi-media Computing and Workshops, Thailand, 2017.

[46] K. Vidyasanker, "On Continuous Queries in Stream", Procedia Computer Science Volume 109, ScienceDirect, 2017, pp. 640-647.