2022 IEEE International Conference on Software Maintenance and Evolution (ICSME) | 978-1-6654-7956-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICSME55016.2022.00055

"*When the Code becomes a Crime Scene*" Towards Dark Web Threat Intelligence with Software Quality Metrics

G. Cascavilla*, G. Catolino†, F. Ebert† D.A. Tamburri*, WJ van den Heuvel†
*JADS - TU/e Eindhoven University, The Netherlands
†JADS - Tilburg University, The Netherlands
{n(ame).surname}@jads.nl

***Abstract*—The increasing growth of illegal online activities in the so-called dark web—that is, the hidden collective of internet sites only accessible by a specialized web browsers—has challenged law enforcement agencies in recent years with sparse research efforts to help. For example, research has been devoted to supporting law enforcement by employing Natural Language Processing (NLP) to detect illegal activities on the dark web and build models for their classification. However, current approaches strongly rely upon the linguistic characteristics used to train the models, e.g., language semantics, which threatens their generalizability. To overcome this limitation, we tackle the problem of predicting illegal and criminal activities—a process defined as threat intelligence—on the dark web from a complementary perspective—that of dark web code maintenance and evolution—and propose a novel approach that uses software quality metrics and dark website appearance parameters instead of linguistic characteristics. We performed a preliminary empirical study on 10.367 web pages and collected more than 40 code metrics and website parameters using sonarqube. Results show an accuracy of up to 82% for predicting the three types of illegal activities (i.e., suspicious, normal, and unknown) and 66% for detecting 26 specific illegal activities, such as drugs or weapons trafficking. We deem our results can influence the current trends in detecting illegal activities on the dark web and put forward a completely novel research avenue toward dealing with this problem from a software maintenance and evolution perspective.**

***Index Terms*—Software Code metrics, Dark Web, Software Code Quality, Machine Learning**

I. Introduction

The dark web is the well-known hidden, fishy, and dangerous side of the internet where illegal trades take place daily [1]: it hosts almost every type of illegal material, from weapons trafficking, to exchange of every type of drug up until pedo-pornography [2]. To access such a hidden area of the internet, users can employ The Onion Routing engine (TOR), a specialised browser[1] that creates a privacy-preserving network where websites are not indexed, and where users can keep their identities anonymous. On the one hand, the threat intelligence research community is collaborating with law enforcement agencies to help detect and automatically categorize illicit activities from the dark web [3]–[7]. On the other hand, most of the proposed approaches harness Natural Language Processing techniques, which are connected to language characteristics, implying poor generalizability, e.g., one approach cannot be suitable for every context.

At this point, we argue a simple fact concerning dark web pages: they constitute a theatre wherefore software becomes the stage of illicit activity *by-design*, but remain primarily software, nonetheless. For this reason, dark web pages must evolve and must be developed/maintained like typical software systems. Like other software systems (e.g., other web portals) dark web pages can be analyzed from a software engineering (SE) maintenance and evolution perspective [8], e.g., exploiting website characteristics, source code metrics and bugs, software quality indicators and more [9], [10]. Our conjecture regards exploiting metrics related to measuring the quality and the evolution of software, *e.g.*, LOC for classifying illegal activities on dark web pages together with website appearance parameters *e.g.*, min words in a sentence.

The contribution of this work is twofold: i) perform a seminal study that investigates how source code maintenance and evolution metrics can be used as predictors for illicit activities of dark web applications, ii) provide the first approach that gathers dark web threat intelligence page using source code quality metrics and website appearance parameters regardless of its language.

We perform an empirical study on 10.367 web pages and collect 40+ code metrics and website parameters using *sonarqube*. Then, we build two prediction models for detecting illicit activities of dark web pages considering two different target variables. The first one consists of 26 categories corresponding to various illicit activities, *e.g.*, drugs. The latter reflects 3 categories, namely *suspicious*, *unknown* and *normal*.

To address the aforementioned study, we formulate the following research questions:

> **RQ1:** To what extent can online illegal activities be classified based on source code metrics and website appearance parameters?

To address **RQ1**, we analyzed whether source code metrics and website appearance parameters are important for the classification of illicit online activities, analyzing which features are more relevant compared to others.

[1]https://www.torproject.org/

2576-3148/22/$31.00 ©2022 IEEE
DOI 10.1109/ICSME55016.2022.00055

RQ2: Which classifier outperform when classifying illicit online activities?

To address **RQ2**, we analyzed the performance of the 4 main classifiers in Machine-Learning (ML), i.e., SVMs, decision-tree learning, regression modelling; subsequently we compare their prediction accuracy.

Our preliminary results show that our approach reaches an accuracy of 82% for predicting high granularity illegal activities, *e.g.*, suspicious, while an accuracy of 66% in detecting 26 specific illegal activities, such as drugs or weapons.

These findings open a completely novel path of inquiry, reflecting threat intelligence research over dark web pages where code quality metrics—usually used in SE to address maintenance and evolution—become first-class citizens to be exploited in a completely different context, i.e., *supporting cyber-law enforcement*.

II. Related work

The deep and dark web have already been extensively studied from different perspectives. On the one side, some studies analyze the quality and health of the dark web open-source software community [2], [8]. For instance, Chen *et al.* and Onyango *et al.* performed an investigation of dark web open-source components from both software and community perspectives but focusing on code vulnerabilities, dark web content sentiment, and developers' community health. On the other side, most studies focus on profiling dark web markets or provide new approaches for detecting illegal activities [3], [11]. Celestini *et al.* [11] detected and analyzed the criminal mechanisms representing the baseline of the illicit drugs trade phenomenon. Ericsson *et al.* [12] investigated how products are grouped in a dark web market. Nurmi *et al.* [13] studied the seller's reputation on the Finnish version of Silk Road dark market. Differently, Al Nabki *et al.* [3] proposed a classification approach to classify dark web pages into different classes (*e.g.*, violence or pornography) using text mining techniques. Finally, Sabbah *et al.* [14] used a hybridized term-weighting method to improve classification based on text mining techniques on the dark web. The use of textual content is the *fil rouge* that connects the mentioned approaches. Conversely, *Our approach aims at classifying web pages without any text mining—and thus independent from websites' language—and using code quality indicators, paving the way towards using software maintenance and evolution metrics and indicators as basic dark web threat intelligence analytics, e.g., to aid cyber-law enforcement.*

III. Methodology

The goal of this study is to classify illicit activities of a dark webpage using software quality metrics as well as website appearance parameters. In this section, we describe the methodology used for conducting our study[2].

[2]Replication package of the study is available in the appendix [15].

1) Dataset: In the context of our study, we relied on a dataset publicly available, *i.e.*, Darknet Usage Text Address (DUTA) [4]. DUTA contains the web pages and textual context of 10.367 dark web pages manually-labelled. DUTA classifies dark web pages in 26 individual threat and illegality classes, *e.g.*, human-trafficking, violence, drugs, abuse, etc.

Parameter	Mean	Std.	Median
Blocker Violations	0	0	0
Bugs	52.0	960.3	1.0
Code Smells	36.2	756.0	0.0
Cognitive Complexity	0.0	0.0	0.0
Comment Lines	59.8	2174.4	0.0
Comment Lines Density	4.6	14.0	0.0
Confirmed Issues	0.0	0.0	0.0
File Complexity	1.0	0.011	1.0
Complexity	1.0	0.011	1.0
Critical Violations	0.0	0.0	0.0
Development Cost	31467.3	334226.3	2280.0
Duplicated Blocks	10.4	76.7	1.0
False Positive Issues	0.0	0.0	0.0
Files	1.0	0.0	1.0
Generated Lines	0.0	0.0	0.0
Generated Ncloc	0.0	0.0	0.0
Info Violations	0.026	1.64	0.0
Lines	1267.4	13887.7	94.0
Ncloc	1048.9	11140.9	76
Sqale Rating	1.09	0.46	1.0
Violations	88.1	1342.9	2.0
Open Issues	88.1	1342.9	2.0
Public Docm. API Density	100.0	0.0	100.0
Public Undocumented API	0.0	0.0	0.0
Reliability Rating	1.84	0.84	2.0
Reliability R.Effort	335.4	5555.9	5.0
Reopened Issued	0.0	0.0	0.0
Security Rating	0.0	0.0	0.0
Security R. Effort	0.0	0.0	0.0
Skipped Tests	0.0	0.0	0.0
Sqale Debt Ratio	145.2	12478.1	0.0
Sqale Index	253.7	5709.1	0.0
Test Errors	0.0	0.0	0.0
Duplicated Files	0.50	0.49	1.0
Duplicated Lines	780.5	6515.2	11.0
Duplicated Lines Density	43.8	47.3	1.5
Effort to Reach Maint. A	116.2	4588.7	0.0
Major Violations	47.3	812.2	0.0
Minor Violations	40.8	917.8	0.0
Number of Links	269.2	3470.8	6.0
Number of Words	2449.3	29956.8	96.0
Minimum Words in Sentence	5.86	65.5	1.0
Maximum Words in Sentence	56.8	152.7	32.0
Bitcoin	0.26	0.44	0.0

TABLE I: Statistics of numerical parameters.

2) Independent Variables: As mentioned in the introduction, we relied on source code metrics and website appearance parameters; to extract them, we used Sonarqube[3], which is well known in the software maintenance and evolution research and practice community. The tool extracts code metrics *e.g.*, LOC, technical debt information, *e.g.*, code smell, management and community metrics, *e.g.*, effort, and webpage parameters, *e.g.*, number of words, based on raw HTML files. We extracted 30 software code metrics, such as *lines of code*, *violations*, and *bugs* and 16 website appearances like *amount of links found*, the *number of words* and whether or not there are *bitcoin transactions* on the webpage. These metrics provide insights without being language-website dependent. In total, we extracted 46 metrics for each dark web page. They are shown in Table II. Table I shows some descriptive statistics instead. However, some features, *e.g.*, *test errors*, have a value of 0 for every data point— see the mean and median—thus,

[3]https://www.sonarqube.org

leading to being useless for a prediction. Consequently, these features have been discarded.

3) Target Variables: Our target variable concerns the prediction of illicit activities. As mentioned in III-1, DUTA dataset is already labelled. Indeed, it contains 26 labels corresponding to different illicit activities, *e.g.*, drugs. Although the dataset is widely used, we know that it is always challenging to face a multi-class prediction [4], [11], [14], for this reason, we decided to build two different models with a different granularity level of the target variable. The first model has 26 categories as the target variable. The second one, the clustering result of the 26 labels in 3 main categories, namely *suspicious*, *unknown* and *normal*. *Suspicious* are web pages that might contain illegal activities. *Normal* are just ordinary web pages, and *unknown* is a web page that is either down, empty or locked. Table III shows the 26 categories and the corresponded high granularity categorization.

4) Feature Engineering: After extracting the data, we applied some feature engineering steps. First, we analyzed outliers since they can exert a negative effect on the classification process [16]. We used PyOd KNN's model to detect them, which determines the outlier scores based upon a selected point nearest neighbor. The step is fundamental since could improve data quality and decrease the chance of the classifiers being influenced by possible outliers [16]. After removing the outliers, we considered scaling our data (between 0 and 1) since we noticed that software code metrics and website appearance parameters had different distributions.

5) Machine-Learning Techniques and Experimental Settings: We used four classifiers, *i.e.*, K-Nearest Neighbor (KNN), Logistic Regression (LR), Support Vector Machines (SVM), and Random Forest (RF). These classifiers have been selected since they come from different ML families [17], thus being a representable sample in the literature. We used the 80-20 validation strategy, and the validation set was used only to check the best-performing model. The training data have been splitted using stratified k-fold cross-validation. We tested k=5 and k=10 based on previous studies [18], [19]. Furthermore, we applied hyper-parameter tuning using gridsearch as shown in Table IV); All the values are combined into a Cartesian product. This means that every possible combination of hyper-parameter values has been tested. Moreover, we applied an iterative feature engineering approach, which means that we experimented with and without balancing the data—applying oversampling technique—and with and without normalizing the data to optimize the result of the experiments. All of these different models have been tested using the same training data.

A. Evaluation Metrics

To evaluate the performance, we relied on four well known metrics, namely: *accuracy*, *recall*, *precision*, and *F1-score*.

IV. Results and Findings

In this section, we discuss the results of our study, reporting only the best models obtained.

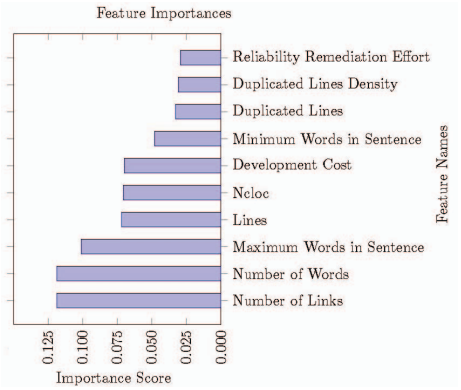

Fig. 1: The 10 most important features.

1) Feature Importance: Figure 1 shows the 10 most important features using Random Forest as a classifier. It is interesting to note that both website appearance (Number of Lines, Number of Words, Maximum Words in Sentence and Minimum Words in Sentence) and software metrics (Lines, Ncloc, Development Cost, Duplicated Lines, Duplicated Lines Density and Reliability Remediation Effort) appear in this list of the 10 most relevant features. Furthermore, we can notice that the features are not decreasing evenly. Some features have roughly the same importance instead. Finally, this figure shows how a dark web page cannot be classified solely on 1 or 2 features since much valuable information would go missing, possibly hurting the classification accuracy.

2) Model Comparison: The resulting models are shown in Table V and VI. Table V represents the results where the target variable consists of the 26 different illicit activities, while Table VI the high granularity categorization. Table represent the best models obtained from testing 1.768 different setups (variation in hyper-parameters, pre-processing, and target variables), as discussed in section III.

We notice that the performance varies in terms of scores between each classifier. However, we are not surprised about it due to the different nature of the classifiers. Furthermore, we see that the performance sightly improves when scaling the data. Netherverless, logistic regression is the worst classifier to predict illicit activities—with different levels of granularity—while the Random Forest algorithm outperformed in both. A possible explanation behind these results can be attributed to the nature of the Random Forest classifier; it is an ensemble technique that allows the construction of several decision trees with a subset of features, and the combination of the prediction of the decision trees is performed by using majority voting. Moreover, Random Forest implicitly applied feature selection, thus possibly improving the results. In the following, the resulted configuration given by gridsearch:

- **26 Categories:** K(5), Max Depth(50), Number of Estimation(1000), Scaling(Yes);
- **3 Categories:** K(5), Max Depth(50), Number of Estimation(200), Scaling(No).

Feature Name	Description
skipped tests	Number of skipped unit tests.
sqale dept ratio	Ratio between the cost to develop the software and the cost to fix it. The Technical Debt Ratio formula is: Remediation cost / Development cost.
sqale index	Effort to fix all maintainability issues. The measure is stored in minutes in the DB. An 8-hour day is assumed when values are shown in days.
test errors	Number of unit tests that have failed.
duplicated files	Number of files involved in duplications.
duplicated lines	Number of lines involved in duplications.
duplicated lines density	Density of duplication = Duplicated lines / Lines * 100
effort to reach maint. rating a	Effort to fix all maintainability issues.
major violations	A violation that might have a substantial impact on productivity. Ex: too complex methods, package cycles, etc.
minor violations	A violation that might have a potential and minor impact on productivity. Ex: naming conventions, Finalizer does nothing but call superclass finalizer, etc.
number links	The amount of outbounded links on a webpage.
number of words	The amount of words on a webpage.
min words in sentence	The minimum amount of words in a sentence on a webpage.
max words in sentence	The maximum amount of words in a setence on a webpage.
bitcoin	Boolean if there are bitcoin transaction available on a webpage.
dark web	Check if it is an .onion webpage.
blocker violations	A violation that is operational/security risk: This issue might make the whole application unstable in production. Ex: calling garbage collector, not closing a socket, etc.
bugs	Number of bugs.
code smells	Number of code smells.
cognitive complexity	How hard it is to understand the code's control flow.
comment lines	Number of lines containing either comment or commented-out code.
comment lines density	Density of comment lines = Comment lines / (Lines of code + Comment lines) * 100
confirmed issues	Number of issues whose status is Confirmed
file complexity	It is the Cyclomatic Complexity calculated based on the number of paths through the code for a specific file.
complexity	It is the complexity calculated based on the number of paths through the code. Whenever the control flow of a function splits, the complexity counter gets incremented by one. Each function has a minimum complexity of 1. This calculation varies slightly by language because keywords and functionalities do.
critical violations	A violation that is Operational/security risk: This issue might lead to an unexpected behavior in production without impacting the integrity of the whole application. Ex: NullPointerException, badly caught exceptions, lack of unit tests, etc.
development cost	The development cost to create the code.
duplicated blocks	Number of duplicated blocks of lines. (at lest 100 successive and duplicated tokens spread over at least 10 lines of code)
false positive issues	Number of false positive issues
files	Number of files.
last commit date	Date of last commit.
generated lines	Number of lines generated.
generated ncloc	Number of lines of code generated.
info violations	A violations that is unknown or not yet well defined security risk or impact on productivity.
lines	Number of physical lines (number of carriage returns).
ncloc	Number of physical lines that contain at least one character which is neither a whitespace nor a tabulation nor part of a comment.
sqale rating	Rating given to your project related to the value of your Technical Debt Ratio. The default Maintainability Rating grid is: A=0-0.05, B=0.06-0.1, C=0.11-0.20, D=0.21-0.5, E=0.51-1
violations	Number of violations.
open issues	Number of issues whose status is Open
public documented api density	Density of public documented API = (Public API - Public undocumented API) / Public API * 100
public undocumented api	Public API without comments header.
reliability rating	Rating for reliability.
reliability remediation effort	Effort to fix all bug issues. The measure is stored in minutes in the DB. An 8-hour day is assumed when values are shown in days.
reopened issues	Number of issues whose status is Reopened
security rating	Rating for security.
security remediation effort	Effort to fix all vulnerability issues. The measure is stored in minutes in the DB. An 8-hour day is assumed when values are shown in days.

TABLE II: A description of all features that have been extracted.

Class	Amount	Activity Type
Hosting	2096	Normal
Empty	1509	Unknown
Cryptocurrency	818	Normal
Down	785	Unknown
Locked	759	Unknown
Personal	545	Normal
Drugs	427	Suspicious
Counterfeit Credit-Cards	370	Suspicious
Social-Network	357	Normal
Services	319	Suspicious
Marketplace	244	Suspicious
Pornography	236	Suspicious
Forum	201	Suspicious
Hacking	197	Suspicious
Cryptolocker	170	Suspicious
Violence	85	Suspicious
Counterfeit Money	73	Suspicious
Counterfeit Personal-ID	56	Suspicious
Library	41	Normal
Casino	25	Normal
Religion	20	Normal
Leaked-Data	20	Normal
Fraud	19	Suspicious
Art	19	Normal
Politics	12	Normal
Human-Trafficking	4	Suspicious

TABLE III: Distribution of classes.

Model	KNN	LR		SVM		RF	
Param	# of K	Solver	Max iter	Gamma	C	Max depth	N estimators
Value	1,2,3,4 5,6,7 8,9,10 12,14,16 18,20,25 30,40,50	'newton-cg' 'lbfgs' 'sag'	50, 100 300, 500, 1000	0.00000001 0.0000001 0.000001 0.00001 0.0001, 0.001 0.01, 0.1, 1, 10, 100, 1,000 10,000	0.001, 0.01 0.1, 1, 10, 100,1000, 10,000, 100,000 1,000,000 10,000,000 100,000,000 1,000,000,000	3,10, 50 None	10,20,30 50, 100, 200, 500,1000 2000

TABLE IV: Hyper-parameter values.

Classifier	Scaling	Recall	Precision	F1-score	Accuracy
KNN	Yes	0.532	0.529	0.529	53.171%
LR	Yes	0.109	0.211	0.109	10.929%
SVM	Yes	0.503	0.563	0.504	50.340%
RF	Yes	0.660	0.671	0.661	65.968%
KNN	No	0.519	0.527	0.522	51.925%
LR	No	0.240	0.103	0.103	10.780%
SVM	No	0.297	0.380	0.380	29.728%
RF	No	0.663	0.673	0.664	**66.251%**

TABLE V: Results classifiers with 26 classes as target variable.

Classifier	Scaling	Recall	Precision	F1-score	Accuracy
KNN	Yes	0.702	0.714	0.706	70.232%
LR	Yes	0.435	0.503	0.393	43.520%
SVM	Yes	0.701	0.695	0.694	70.062%
RF	Yes	0.817	0.818	0.822	**81.664%**
KNN	No	0.681	0.688	0.683	68.081%
LR	No	0.443	0.603	0.391	44.312%
SVM	No	0.556	0.723	0.574	55.631%
RF	No	0.802	0.807	0.804	80.192%

TABLE VI: Results classifiers with 3 classes as target variable.

Finally, there is a significant difference between the accuracy for the model with 26 classes and the one with 3 classes. Indeed, while comparing the different scores for the various classes, it is wise to keep the accuracy baseline in mind. The baseline accuracy represents the accuracy got when a classifier always predicts the majority class. For example, the baseline accuracy of the first model (with 26 classes) is 23%, while the

other 50%. Consequently, we noticed that some classifiers are valid, but Random Forest outperformed the baseline accuracy by approximately 43% (26 classes) and 32% (three classes), thus adding value as a predictor.

A. Findings

This study aimed to classify illicit activities of a dark webpage based on software code metrics and website appearance parameters. In the following, the findings achieved:

> **Finding of RQ1:** The analysis showed how software code metrics and website appearance parameters are crucial for predicting a dark web page, thus leading our approach to improving generalizability since it is not language-website dependent. Both types of features have predicting power and should therefore be considered when classifying a dark web page.

> **Finding of RQ2:** The **Random Forest** algorithm outperforms the other classifiers. Indeed it performs better for both the target variables predicted, *i.e.*, the 26 and the 3 classes, respectively.

We can conclude that besides textual analysis, information about source code, like code metrics, technical debt, and website appearance, is novel and promising in detecting illicit activities on the dark web with a reasonable accuracy.

B. Threats to Validity

Some threats can affect our study. The first one is related to the extraction of the metrics. However, we relied on Sonarqube, a well-known tool considered reliable by the research community. To ensure the reliability and reproducibility of the results, we used the SKLEARN library for constructing our models. Finally, our results are based on the DUTA dataset, possibly avoiding the generalizability. We plan in the future to extend the study by labeling new dark web pages.

V. Conclusion

This study provides a novel approach to classifying illicit activities of dark web pages using software code metrics and website appearance parameters. Results show that both are crucial factors for predicting illicit activities. Thus, both have predicting power and should be considered when classifying a dark web page. Our approach showed how illegal online activities could be classified with an accuracy of 81.664% using Random Forest. In future work, we plan to improve the dataset's quality by crawling and labeling new pages and analyzing additional characteristics of the source code.

VI. Acknowledgement

We thank Martijn Keizer for the work done during his master thesis. The work is supported by EU Twining DESTINI project (857420), and, the Dutch Ministry of Justice and Safety through the Regional Table Human Trafficking Region East Brabant sponsored the project SENTINEL.

References

[1] J. Sanchez and G. Griffin, "Who's afraid of the dark? hype versus reality on the dark web," 2019. [Online]. Available: https://bit.ly/38gzdsk

[2] H. Chen, *Dark web: Exploring and data mining the dark side of the web*. Springer Science & Business Media, 2011, vol. 30.

[3] M. W. Al Nabki, E. Fidalgo, E. Alegre, and I. de Paz, "Classifying illegal activities on tor network based on web textual contents," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, 2017, pp. 35–43.

[4] M. W. Al-Nabki, E. Fidalgo, E. Alegre, and L. Fernández-Robles, "Torank: Identifying the most influential suspicious domains in the tor network," *Expert Systems with Applications*, pp. 212–226, 2019.

[5] S. He, Y. He, and M. Li, "Classification of illegal activities on the dark web," ser. ICISS 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 73–78.

[6] N. Tavabi, N. Bartley, A. Abeliuk, S. Soni, E. Ferrara, and K. Lerman, "Characterizing activity on the deep and dark web," in *Companion Proceedings of The 2019 World Wide Web Conference*. Association for Computing Machinery, 2019, p. 206–213.

[7] A. H. M. Alaidi, R. M. Al_airaji, H. T. Alrikabi, I. A. Aljazaery, and S. H. Abbood, "Dark web illegal activities crawling and classifying using data mining techniques," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 16, no. 10, p. pp. 122–139, May 2022.

[8] S. Onyango, E. Steenvoorden, J. Scholten, S. Jansen, P. Gregory, P. Kruchten *et al.*, "Assessing the health of the dark web: An analysis of dark web open source software projects," in *Agile Processes in Software Engineering and Extreme Programming–Workshops*, vol. 426, no. 1. Springer, Cham, 2021, pp. 125–134.

[9] A. Suali, S. Fauzi, M. Nasir, W. Sobri, and I. Raharjana, "Software quality measurement in software engineering project: A systematic literature review," *Journal of Theoretical and Applied Information Technology*, vol. 97, no. 3, pp. 918–929, 2019.

[10] F. N. Colakoglu, A. Yazici, and A. Mishra, "Software product quality metrics: A systematic mapping study," *IEEE Access*, vol. 9, pp. 44 647–44 670, 2021.

[11] A. Celestini, G. Me, and M. Mignone, "Tor marketplaces exploratory data analysis: the drugs case," in *International Conference on Global Security, Safety, and Sustainability*. Springer, 2017, pp. 218–229.

[12] E. Marin, A. Diab, and P. Shakarian, "Product offerings in malicious hacker markets," in *2016 IEEE conference on intelligence and security informatics (ISI)*. IEEE, 2016, pp. 187–189.

[13] J. Nurmi, T. Kaskela, J. Perälä, and A. Oksanen, "Seller's reputation and capacity on the illicit drug markets: 11-month study on the finnish version of the silk road," *Drug and alcohol dependence*, vol. 178, pp. 201–207, 2017.

[14] T. Sabbah, A. Selamat, M. H. Selamat, R. Ibrahim, and H. Fujita, "Hybridized term-weighting method for dark web classification," *Neurocomputing*, vol. 173, pp. 1908–1926, 2016.

[15] anonymous. (2022) Appendix for paper "when the code is the crime scene". [Online]. Available: https://figshare.com/s/45e98a78d2aa9f6e7f0f

[16] K. Kadota, D. Tominaga, Y. Akiyama, and K. Takahashi, "Detecting outlying samples in microarray data: A critical assessment of the effect of outliers on sample classification," *Chem-Bio Informatics Journal*, vol. 3, no. 1, pp. 30–45, 2003.

[17] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.

[18] Y. Bengio and Y. Grandvalet, "No unbiased estimator of the variance of k-fold cross-validation," *Journal of machine learning research*, vol. 5, no. Sep, pp. 1089–1105, 2004.

[19] J. D. Rodriguez, A. Perez, and J. A. Lozano, "Sensitivity analysis of k-fold cross validation in prediction error estimation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 3, pp. 569–575, 2009.