Temporal Exploitability Scoring System (TESS)

Leonardo Sarra
Advisor: Prof. Silvia Bonomi

September 2020

1 Introduction

In recent years there is a strong and growing emphasis on IT governance and cybersecurity as technology-based information assets are becoming more common and relevant.
As technology advances, so too do the tools and methods employed by malicious entities to cause harm to an organization.
Risk assessment, in particular, plays a significant role in the security governance process because it helps in prioritizing changes to the risks based on their likelihood and consequences.
When we focus on the risk of a software vulnerability, we have that the ability to assess the risk of a software vulnerability relies on the basic characteristics of a vulnerability without taking into consideration security measures that are already in place.
This could lead to the wrong prioritization of the software patches, which wastes resources on fixes that could have been postponed and allow the malicious party to exploit vulnerability whose remediations were wrongly delayed.
Usually, the CVSS, the Common Vulnerability Scoring System, is used to prioritize the remediations of the risk of software vulnerabilities; this scoring system works by computing the severity of a vulnerability based on its characteristics (which may or may not change over time).
Because the CVSS base score does not take into consideration the vulnerability lifecycle and the security countermeasure of the affected system, this will lead to the problem previously described.
Our proposal is TESS, Temporal Exploitability Scoring System, which helps in fitting a model that predict the exploitability of a vulnerability at a given time t by leveraging the logs of an existing IPS/IDS in the system.
TESS, paired with CVSS, provides a solution to scale down the CVSS base score in relation to the environment and the lifecycle of the vulnerability.

2 Related Literature

For TESS, we took inspiration from EPSS, Exploit Prediction Scoring System, a system proposed by Jacob et al[1] that aims to predict the probability of exploitation of a vulnerability in the next 12 months by fitting a logistical regression model.
While TESS and EPSS share the same goal of improving the prioritization of the risk assessment process, they propose different metrics: exploitability at a given time in TESS and probability of exploitation in EPSS.
They also differ in the data source used to compute those metrics.
In the case of TESS, the data source from which vulnerability's information is obtained is public; hence, results can be easily reproduced.
EPSS, instead, uses a dataset which at the time of writing was not yet disclosed.
EPSS is not the only solution that tries to estimate the probability of exploitation of a vulnerability over its lifecycle; there are also proofs-of-concept by Edkrantz and Said[2], Bozorgi et al.[3], Almukaynizi et al.[4], of these one EPSS is the only work that uses real-world exploit data and keywords from the vulnerability's description as input features of the predictive model.
TESS is not the only work that focuses on computing a temporal exploitability.
For example, Abraham et al.[5] focus on computing the exploitability starting from the CVSS base metric and scaling it by a temporal weight factor $exploitability = cvss_i * temporal\ weight_i$.
The temporal weight factor used by Abraham et al. depends on the availability of the exploit; for example, an exploit for which no code can be found comes with a lower factor when compared to a highly available one.
Other approaches like the one proposed by Almukaynizi et al[4] use machine learning techniques to predict future cyber threats and the exploitability of software vulnerabilities through darkweb/deepweb discussions with hacking-related content.
TESS, unlike these approaches, is a proof of concept that works by using machine learning techniques with the historical data of previously used exploits (from an existing IPS/IDS) and the characteristics of the related software vulnerabilities to compute the exploitability.

3 Proposal

Our approach, named TESS, relies on the concept of temporal exploitability proposed by Abraham et al.[5], except that, it aims to compute the metric differently by changing the idea behind the temporal weight.
In TESS, we assume that the temporal weight depends on the lifecycle of the exploit, as described in Frei's work[6], and multiple factors that are environment-dependent which are ignored in the approaches that we previously described.
Those factors influence the temporal weight and depend on the system itself, which could adopt defensive measures, as described in the Pendleton et al.[7], aimed to mitigate attacks of a specific type, using a particular attack vector or different features.
The intuition behind TESS was that the exploitability of a vulnerability follows a gaussian distribution, in a similar way to the evolution of the risk during the lifecycle of a vulnerability described by Gorbenko et al[8] [Figure 1].

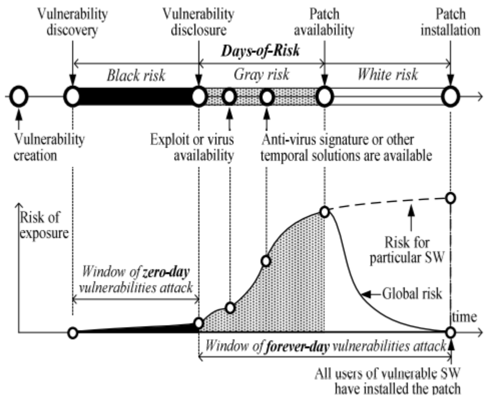
Vulnerability discovery
Vulnerability disclosure
Patch availability
Patch installation
Days-of-Risk
Black risk
Gray risk
White risk
Vulnerability creation
Exploit or virus availability
Anti-virus signature or other temporal solutions are available
Risk of exposure
Risk for particular SW
Window of zero-day vulnerabilities attack
Global risk
time
Window of forever-day vulnerabilities attack
All users of vulnerable SW have installed the patch

Figure 1: Risk lifecycle of a vulnerability

Because we expected exploitability to follow a risk-like trend, we opted to fit a non-linear model.
We believed the non-linear model to follow a specific trend:

- **Pre-Discovery**; when the vulnerability is not yet discovered, the exploitability is equal to 0
- **Between discovery and disclosure**; when an individual discovers the vulnerability, but it is still not disclosed to the public (e.g.: through NVD or vendor), we expect a low exploitability that slowly increases over time as become easier for the malicious entities to find relevant information about the vulnerability.
- **Between disclosure and patch availability**; when the vulnerability is public domain, but a patch is still not yet provided by the vendor is the moment in which exploitability is expected to rise rapidly over time, reaching its maximum just before the patch is made available.
- **Between patch availability and patch installation**; we expect the exploitability to slowly decay until it reaches zero once the patch is deployed.

The gaussian parameters supposedly change depending on the system we are trying to abstract, influenced by defensive measures, which are more or less effective based on the vulnerability features, and the cyber posture of the entity

maintaining the system.
Let us note that some vulnerabilities are an exception and do not follow this exploitability trend, like vulnerabilities related to discontinued software for which no patch will be ever made available or the ones that are unfixable because they are tied to the intrinsic characteristics of the software.
For these two cases, the exploitability is expected to slowly decay because either after the disclosure as mitigations are made available, or just because the affected product is discontinued and is getting replaced by unaffected alternatives.

Input dataset. The data that is used for fitting the model is the log of the IDS/IPS used in the system; from these logs, attack events and their outcomes are extrapolated.
In particular, TESS uses only data related to vulnerabilities that have reached the end-of-life in order to prevent the model from being fitted with vulnerabilities whose exploitability is still in flux.
The entries of the TESS dataset are supposed to contain the following attributes:

- CVE unique identifier
- Observation date
- Outcome of the attack from the adversary that used that particular vulnerability (Success/Failure)

This entry is later expanded by adding data from the NVD (National Vulnerability Database) like the disclosure date of the vulnerability, its description, the related CWE (Common Weakness Enumeration), and CAPEC (Common Attack Pattern Enumeration) entries.
To each entry is also assigned the value of the target function that will later be used to resolve the regression problem.

Target function. The function $f_{v,t}$ that we want to estimate depends on both the vulnerability and the observation time, the function is $f_{v,t} = \frac{st_{v,t-w,t}}{st_{v,t-w,t} + ft_{v,t-w,t}}$ where $st_{v,t-w,t}$ and $ft_{v,t-w,t}$ are respectively the number of success and failure tries that exploited a vulnerability v and that occurred at a specific instant $i \in [t-w, t)$.
When computing the number of successful tries/failures, we consider only the time window $[t-w, t)$, by doing so, the target function is computed using only a subset of the events. Therefore, it will adapt faster to the sudden changes of the exploitability trend.
The w value is arbitrarily chosen; it is crucial to choose a time frame that is not too small; otherwise, even small fluctuations in the success-failure ratio could lead to a skewed exploitability. Instead, if the time frame is too big, the exploitability will hardly change over time.
We use this particular target function in tandem with the starting CVSS exploitability to scale it down as the life of vulnerability goes by, the formula used for the exploitability is as follow: $tess\_expl_{v,t} = cvss\_expl_v * f_{v,t}$

Workflow. The workflow that is used to obtain the trained models is made of the following steps:

1. **Parsing of the input dataset**; the first step is to parse the dataset contains the entries <*CVE, Date, Outcome*> obtained from the underlying IDS/IPS.

2. **Entry expansion**; the dataset is expanded by including information about the CVE obtained by scraping the NVD website. The vulnerability's description is also processed by a "Rapid Automatic Keyword Extraction algorithm"[9] which will return a set of keywords that are later used as features.
   It is important to note that only a subset of the found keywords is considered; the keywords have a ranking that must be equal or greater than a threshold to be considered.

3. **Feature composition**; once all the information of the input event and the related vulnerability are obtained, they are used to create a feature vector composed of binary and integer values.
   The format used is <*keywords* $\|$ *capec _ids* $\|$ *cwe_ids* $\|$ *Ref. number* $\|$ *time*> where keywords, capec_ids, and cwe_ids are a set of binary values which models the presence or the absence of a specific feature/attack pattern/weakness. Reference number and time represent, respectively, the number of references obtained over the vulnerability life and the time at which the events occurred in the log.

4. **Feature selection**; after all the features for every data entry are computed, TESS provides the option to select a subset of the features based on their relevance. The selection is executed by training a GBM regressor built using LightGBM[10], a gradient boosting framework that uses tree-based learning algorithms to extrapolate the importance of each feature.
   In case the importance of a feature is below the arbitrarily chosen threshold, it is ignored and no longer taken into consideration.
   Gradient boosted machines (GBMs) are an extremely popular machine learning algorithm that has been proven successful across many domains. They work similar to a random forest approach, but while random forests build an ensemble of deep independent trees, GBMs build an ensemble of shallow and weak successive trees with each tree learning and improving on the previous.
   GBMs usually provide good predictive accuracy, do not require any preprocessing of the data, and comes with many tuning options.
   The main disadvantage of using GBMs is that they require many trees, which makes the execution both time and resource exhaustive.
   We chose LightGBM because it is very memory-efficient, which helps in relieve the toll on the system when the input dataset and the number of features are high.

5. **Feature reduction**; once features are computed, there is also the possibility to reduce the features to an arbitrary number of principal components by means of a PCA algorithm.
   PCA helps in removing correlated features hence improving the performance when fitting the model; it also helps in reducing overfitting that is mainly caused when there are too many features. On the other hand, PCA makes the independent variables less interpretable because the original features are turned into principal components that are not as readable and interpretable as original features. Moreover, if the number of principal components is not chosen carefully, it may miss some information compared to the original list of features.
   Because of its advantages and disadvantages, this step is also optional, and it is not supposed to be used together with the feature selection option unless the number of features selected is too high.

6. **Model training**; after the features that will be used for training are fixed, the model will be trained by passing as input the pair (features, target function value).
   TESS supports two regression models: an Epsilon-Support Vector Regression (SVR) model and a neural network.
   The SVR model is the easier model to use, but its fit time complexity is more than quadratic with the number of samples, which makes it hard to scale to datasets with more than 10000 samples.
   The neural network instead can be easily trained with a large dataset without incurring in an Out-Of-Memory state; the training time, on the other hand, is way higher when compared to the SVR model.
   The architecture of the NN is simple; it is made of 4 dense layers with 50/50/30/1 units and ReLU activation functions, except for the last dense layer that uses a linear activation.
   The network is optimized using a Root Mean Square Propagation algorithm, which uses mean square error as a loss function.

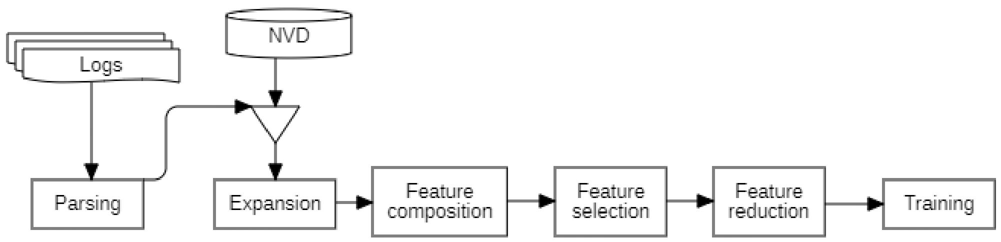

Figure 2: Workflow

Usecase of TESS. TESS is hard to adopt because it requires the system to have a high amount of historical data obtained through the use of an IPS/IDS, and not every company can justify the investment required to satisfy those requirements.
To tackle this problem, TESS's feature schema and models are made sharable by design; therefore, a company which cannot set up the log is still able to use a TESS model from another company with similar environmental factors and cyber posture to predict vulnerability exploitability.
The sharing of trained TESS models coming from different companies opens up multiple possibilities regarding the comparison of the effectiveness of the deployed security measures and the importance of the cyber posture.
For example, there could be multiple models for small, medium, and big enterprises but also models for companies that operate in the public sector.
Considering that the national security framework enforces specific security controls for companies that operate in such sector, it would be worthwhile comparing performances across the models to evaluate controls effectiveness.
To promote the sharing of the models, TESS comes with the ability to export the trained model and all the feature schema in a custom file format ".tess" ready to be distributed. This file format can be imported and allows to reproduce the same result obtained from the one who shared it.
Once the TESS file is imported, we can input a pair (vulnerability, time) to receive as output the exploitability of the vulnerability at the selected timestamp.

Model evaluation. Because of the multiple models, pre-processing options, and multiple configurable parameters, TESS needs to offer a way to evaluate the performance of the model.
In that regard, TESS provides a built-in option to perform a KFold or a shuffle split (75% training, 25% test) of the input dataset, then a part of the data is used to train the model while the rest is used to compute different metrics needed to evaluate the performance of the non-linear regression.
The metrics computed to validate the model are the following:

- Explained variance, which measures the proportion to which a mathematical model accounts for the variation of a given data set.
- Coefficient of determination (R2), the proportion of the variance in the dependent variable predictable from the independent variables.
- Max error.
- Mean absolute error (MAE).
- Mean squared error (MSE).
- Mean squared log error (MSLE).
- Median absolute deviation (MAD).

4 Tentative validation of TESS

Validating TESS is no simple task because of the lack of publicly available logs to construct the dataset entries.
So far, TESS has been tested with a dummy input dataset to verify that the workflow works correctly, but such dataset is biased and cannot be taken into consideration while validating TESS.
Therefore, to validate TESS, we try to verify the assumption we made regarding the vulnerabilities' lifecycle. To do so, we try to fit the SVR model to perform a regression trying to predict the timings and the extent of different relevant events of the vulnerability lifecycle:

- Date of publication of the first vendor advisory reference.
- Date of insertion of the CVSS v3 score.
- Changes during vulnerability's lifecycle of the CVSS v3 Temporal Score.
- Increases/decreases of the CVSS v3 Exploitability metric during vulnerability's lifecycle.

For this purpose, we built a dataset by scraping all the vulnerabilities published since 2016 by NVD.
For each of those vulnerabilities, we obtain, on top of the information that we already described in the TESS workflow, also the list of changes made over time to the reference entries and CVSS scores.
Let us note that for this validation, we used the same workflow explained before, except for the target function, which, in this case, express the timings and quantity of the changes to which the vulnerability was subject to.

Vendor advisory regression. The first task that we tried to resolve was to fit a regression model in order to predict the date of publication of the vendor advisory document.
The vendor advisory document contains information about the vulnerability in question, the available patch, and, eventually, the mitigations.
By predicting the distance in time between the day of the publication of the vulnerability and the date of publication of the vendor advisory, we would have proved the influence of the vulnerability features on the lifecycle.
For this task, we analyzed all the vulnerabilities published between 2018 and 2019 and disclosed at least one year ago whose vendor advisory was published after at least seven days from the disclosure.
Between 2018 and 2019, the vulnerabilities published were 27931.
However, only 6771, the filtering of the entries was required because most of the NVD entries have their vendor advisory published the same date the vulnerability was made public.
We suspect that this is a consequence of the vulnerabilities being mostly unknown during their lifecycle, which gave to the vendor enough time to work on a patch/mitigation for the issue. Therefore, the vulnerability is made public by

the vendor along with the release of the security countermeasures.
From the initial 27931 vulnerabilities, we filtered down to a set of 6771; then, through a KFold validation, we tested the performance of the SVR model, considering as input a feature vector with only CAPEC entries, only CWE entries, only keywords, and any composition of the three.
When keywords from the vulnerability description were used as features, we also applied a feature selection with threshold 8 to reduce the overall number of features, which otherwise would have been a problem when training the models.
The result were as follow:

Methodology	**exp var**	**max error**	**mae**	**mse**	**msle**	**mad**	**r2**
Only Capec	0.0042	233.02	11.06	536.11	0.196	9.06	-0.00179
Only CWE	-8.88	246.38	11.25	541.62	0.197	9.58	-0.0085
Only Keywords	0.047	241.31	9.77	500.45	0.158	7.62	0.04
Keywords+ Capec	0.046	246.38	9.84	500.25	0.15	7.79	0.0383
Keywords + CWE	0.046	248.69	9.77	500.16	0.158	7.65	0.0386
Keywords + CWE + CAPEC	0.041	242.77	10.011	505.77	0.174	7.94	0.033

Table 1: Performance in predicting vendor advisory date (Entries = 6771)

Results were disappointing, and this can be seen in particular from the R2 and explained variance metrics, in particular when only CAPEC or CWE ids features are used.
Results tend to improve when keywords are involved but are still far from satisfying.
Specifically, we have that the model struggles to identify a function that represents the task analyzed. This lead to the metrics having a considerably high error.

CVSS v3 date regression. The second regression task that we tried to solve was to predict the date of the release of the first CVSS score.
For this task, we took into consideration the vulnerability published from 2016 to 2019 disclosed at least one year ago (51069 entries); in particular, we filtered the entries to the ones whose scores were not added in the first seven days since the vulnerability disclosures.
The reason for this filtering is the same described before for the vendor advisory regression task.
The filtering led to only 228 entries being considered; therefore, we used a shuffle split validation to evaluate all the different setup.
The result were as follow:

Methodology	**exp_var**	**max_error**	**mae**	**mse**	**msle**	**mad**	**r2**
Only Capec	0.0055	566.34	140	75260.33	1.257	58.3	-0.12
Only CWE	-8.88	564.64	141.24	75580.03	1.28	58.16	-0.13
Only Keywords	0.0026	564	140.85	75283.66	1.27	58.13	-0.12
Keywords + Capec	0.0047	564.57	140.37	75140.85	1.26	58.37	-0.12
Keywords + CWE	0.0026	564.01	140.85	75283.65	1.27	58.13	-0.12
Keywords + CWE + CAPEC	0.0022	577.18	145.10	84021.19	1.4	49.32	-0.17

Table 2: Performance in predicting CVSS publication date (Entries = 228)

Here the results are the most disappointing, even when compared to the previous task. Both MSE and MSLE are extremely high, while R2 is negative. The fact that R2 is negative means that random values would have explained the variation in the response variable around its mean better than what is done with the date we provided.

CVSS v3 value regression. The last task we evaluated consisted of predicting the extent of the changes in the CVSS score.
This time we took into consideration vulnerabilities published between 2016 and 2019, disclosed at least one year ago, and who saw at least one change in their CVSS score; changes in the first week of the disclosure are ignored.
The choice of filtering out changes that happened in the first week has become necessary because multiple vulnerabilities were published with a placeholder CVSS score expressing a worst-case scenario, which was updated in the few days after the disclosure.
We opted to analyze only the subscore for exploitability and the temporal metrics because those are more subject to change over time.
Note that we also tested the changes in the CVSS exploitability separately, depending on if the subscore decreased or increased over time.
This was made necessary not to skew the results because we suspected that most of the vulnerabilities an overall increase did not undergo analysis in the late stage of their lifecycle.
Results were as follow:

Methodology	**exp_var**	**max_error**	**mae**	**mse**	**mad**	**r2**
Only Capec	-0.037	1.78	0.39	0.54	0.319	-0.12
Only CWE	0	1.8	0.406	0.58	0.3	-0.164
Only Keywords	0.116	1.494	0.35	0.38	0.26	0.09
Keywords + Capec	0.088	1.57	0.35	0.41	0.26	0.05
Keywords + CWE	0.11	1.49	0.35	0.38	0.26	0.09
Keywords + CWE + CAPEC	0.0038	1.86	0.31	0.38	0.14	0.007

Table 3: Performance in predicting the difference of temporal score from publish date and last change of CVE entry (Entries = 274)

Methodology	**exp_var**	**max_error**	**mae**	**mse**	**msle**	**mad**	**r2**
Only Capec	0.013	1.35	0.34	0.35	0.11	0.26	0.008
Only CWE	0.0	1.27	0.39	0.37	0.13	0.51	-0.031
Only Keywords	0.10	1.32	0.32	0.29	0.092	0.29	0.10
Keywords + Capec	0.1	1.35	0.31	0.29	0.089	0.25	0.10
Keywords + CWE	0.10	1.32	0.32	0.29	0.09	0.29	0.10
Keywords + CWE + CAPEC	0.11	1.35	0.30	0.30	0.086	0.22	0.096

Table 4: Performance when predicting increases for exploitability subscore (Entries = 336)

Methodology	**exp_var**	**max_error**	**mae**	**mse**	**mad**	**r2**
Only Capec	-0.11	1.26	0.30	0.25	0.24	-0.12
Only CWE	-4.44	1.09	0.32	0.21	0.26	-0.01
Only Keywords	0.10	1.08	0.26	0.17	0.24	0.10
Keywords + Capec	0.10	1.11	0.24	0.17	0.20	0.10
Keywords + CWE	0.10	1.08	0.26	0.17	0.24	0.10
Keywords + CWE + CAPEC	0.069	1.12	0.27	0.22	0.20	0.05

Table 5: Performance in predicting decreases for exploitability subscore (Entries = 301)

Results show better performances when compared to the other tasks. However, performances are still disappointing because of the low explained variance and R2 metrics. When keywords are involved, performances are better than the alternatives but still far from good.
Keywords paired with CWE seems to perform slightly better than the other combination; using keywords paired with CAPEC and CWE at the same time leads to generally even worse results.
As for the previous CVSS-related tasks, the number of entries is very low for an appropriate evaluation, and considering that CVSS v3 scores are generally not available for vulnerabilities published before 2016, it is hard to find more entries to consider.

Overall we cannot say that we are satisfied with the results of the various task when keywords were considered as results were better but still disappointing.
So we evaluated the events that occur in the entries of the NVD and their order, hoping that this could explain why results were disappointing.
For each event, we assigned a letter: "V" for the publication of the vendor advisory, "C" for the publication of the first CVSS score, "I" to express an increase in the overall CVSS score and "D" to express a decrease.
We started by analyzing the events of the vulnerabilities published between 2016 and 2019 disclosed at least one year ago; we considered even the CVSS events coming from a third party (usually the vendor of the affected software or NIST), and the result was the following:

```
{CV: 669, CVD: 19, VC: 11165, V: 13, VCC: 2, CDV: 3, CVI: 10,
CVDD: 1, CCVD: 10, VCD: 48, VCI: 31, CIV: 1, VCS: 2, VCID: 1,
CSV: 1}
```

Then we excluded CVSS third party updates and obtained the following results:

```
{V: 10793, VC: 1180, CV: 3}
```

This data clearly shows that NVD is a static dataset, CVSS are rarely present or updated and if they are is because a third-party reanalyzed the vulnerability.

On top of that, as seen before, most of the update happens when the patch is already available, this ends up providing a skewed lifecycle which cannot be used to evaluate the performance of TESS.
Therefore, the dataset is not suitable for evaluating TESS methodology, and we are left with no sources to evaluate vulnerability lifecycles.

5 Conclusions

TESS, while being an interesting idea that opens up the possibility of evaluating performances of different environments with different security controls, cannot be validated because of the lack of quality public data.
NVD dataset is not fit for validating TESS's assumptions because of its staticness; at the same time, there are no public logs of IPS/IDS to evaluate TESS's performance in relation with the baseline (CVSS's static exploitability score from NVD).
Because of these reasons, TESS has to be considered just a proof-of-concept until it can be appropriately tested.
On top of that, the difficulty in validating TESS's assumptions also raises questions regarding the validity of EPSS [1] that makes similar statements regarding the lifecycle of vulnerabilities. Those statements were proven by using a private custom made dataset by Jacobs et al.[1], and results cannot be audited.

Bibliography

[1] Jay Jacobs et al. "Exploit Prediction Scoring System (EPSS)". In: *arXiv preprint arXiv:1908.04856* (2019).

[2] Michel Edkrantz and Alan Said. "Predicting Cyber Vulnerability Exploits with Machine Learning." In: *SCAI*. 2015, pp. 48–57.

[3] Mehran Bozorgi et al. "Beyond heuristics: learning to classify vulnerabilities and predict exploits". In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010, pp. 105–114.

[4] Mohammed Almukaynizi et al. "Proactive identification of exploits in the wild through vulnerability mentions online". In: *2017 International Conference on Cyber Conflict (CyCon US)*. IEEE. 2017, pp. 82–88.

[5] Subil Abraham and Suku Nair. "A predictive framework for cyber security analytics using attack graphs". In: *arXiv preprint arXiv:1502.01240* (2015).

[6] Stefan Frei. *Security econometrics: The dynamics of (in) security*. Vol. 93. ETH Zurich, 2009.

[7] Marcus Pendleton et al. "A survey on systems security metrics". In: *ACM Computing Surveys (CSUR)* 49.4 (2016), pp. 1–35.

[8] Anatoliy Gorbenko et al. "Experience report: Study of vulnerabilities of enterprise operating systems". In: *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE. 2017, pp. 205–215.

[9] Stuart Rose et al. "Automatic keyword extraction from individual documents". In: *Text mining: applications and theory* 1 (2010), pp. 1–20.

[10] Guolin Ke et al. "Lightgbm: A highly efficient gradient boosting decision tree". In: *Advances in neural information processing systems*. 2017, pp. 3146–3154.