From Team Dynamics to Security Smells

Anonymous Author(s)

ABSTRACT

The goal of this research is to help software practitioners in understanding how does their team dynamics influence the security smells in a code base they produce through an empirical study of open source teams, interactions therein, and the code base they produce. NSA: Working goal.

ACM Reference format:
Anonymous Author(s). 2020. From Team Dynamics to Security Smells. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 3 pages.
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Code smells are poor implementation practices (or choices) that software developers apply during software development [3]. These smells result in critical flaws in the code base. *Security smells* — a specialized class of code smells — are indicative of security weaknesses in code base [7]. Code smells (or security smells) occur not only because of lack of technical know-how but also because of social debt that occur in software development teams [6].

Social debt or traits in a software development team such as lack of interaction within team members can further lead to additional project overruns in terms of time and cost. Previous work refers to social debt in teams as community smells [6]. We generalize the concept of community smells as traits — both good or bad — of a team, and refer to that as *team traits*. Team traits refer to characteristics of a team and its members individually as well as collectively which are observed when the members collaborate.

Existing studies on factors affecting team performance have identified *communication* between team members as an important contributing factor. Daim et al. [2] explored the communication breakdown in Global Virtual Teams (GVTs) and found effective communication as integral to efficient collaboration. Cheng et al. [1] investigated the importance of trust in semi-virtual collaboration among students in multi-culture and uni-culture teams. They identified three aspects, namely, language, values (e.g., attitude, perception) and habitual behaviors, that contributes to the trust difference between multicultural groups and uni-cultural groups. Palomba et al. [6] investigated correlation between community smells and code smells and identified organization smells as an important factor in code refactoring decisions and the way developers act on code smells. They discovered *community smells* through interviewing developers.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

The goal of this research is to help software practitioners in understanding how does their team dynamics influence the security smells in a code base they produce through an empirical study of open source teams, interactions therein, and the code base they produce.

In this research, we intend to study sociotechnical aspects of software development and software communities, and understand the relation between the two. Specifically, we investigate the following research questions:

RQ 1 (traits): What are these traits that an open source team has? Which of these are specific only to open source teams?
NSA: To answer RQ1, we do a literature survey in social psych and software engineering. We may identify more traits from our analysis of repositories.

RQ 2 (detection): How can we automate the detection of the team traits identified in RQ1?

RQ 3 (influence): How does the identified team traits influence the security of the code base?

Contributions. NSA: Our contributions include …

Organization. Section 2 describe the relevant related works. Section 3 describes the dataset we analyze. Section 4 details the team traits and how we extract those. Section 5 lists the security smells and how we identify each. Section 6 analyzes how team traits influence security smells. Section 7 describes our results. Section 8 concludes with discussion of future directions.

2 RELATED WORKS

Previous work from Palomba et al. [6] aimed at empirically exploring the relation between social and technical debt, by investigating the connection between two noticeable symptoms behind community and code smells. Palomba et al.'s research method based on qualitative investigation features a survey of developers. Their findings highlighted that the persistence of code smells not only depends on technical factors studied by past literature but also on additional aspects related to the social debt occurring in software communities.

3 DATASET

We now describe our dataset. To identify the relevant repositories, we define the following inclusion and exclusion criteria.

Inclusion criteria.

The Github repositories were included in the research collection if they complied with all the inclusion criteria. An explanation of the criteria follows:

(1) number of organization members are larger than 5
(2) project languages are either Python or JavaScript
(3) active commits lifetime of the projects are longer than 1 year

Exclusion criteria. NSA: revise; these are the repositories which meet inclusion criteria but are discarded for other reasons.

The repositories which meet the following exclusion criteria are discarded:

(1) number of organization members are too small to extract communication between organization members
(2) project language are not Python or JavaScript
(3) lifetime of a project too short to obtain sufficient amount of information

After identifying the relevant Github repositories to include in the analysis, based on the exclusion and inclusion criteria, information related to project languages, organization members, active commits lifetime of the project were extracted for each repository.

Data collection.

We collect 160 repositories that we use to determine the team traits and security smells. We collect these 160 repositories from GHTorrent because GHTorrent monitors the Github public event time line. For each event, it retrieves its contents and their dependencies, exhaustively. For the repositories which meet the inclusion and exclusion criteria, we extract the following attributes:

Table 1: Dataset attributes.

Attribute	Source	Criteria
Code base	GitHub	TBD
Collaborators	GitHub	>= 5
Commit Messages	local copy of GHTorrent	>= 1 year
Issues	local copy of GHTorrent	TBD
Pull requests	local copy of GHTorrent	TBD
Programming and scripting language	local copy of GHTorrent	Python/JavaScript

NSA: From the above mentioned attributes, we could compute other attributes.

4 TEAM TRAITS

Team traits refer to characteristics of a team and its members individually as well as collectively which are observed when the members collaborate. The traits include

- Communication quotient: NSA: each member's contribution to information sharing within the team
- Team cohesion: NSA: whether team members agree with each other; is there tension in the team?
- Social awareness: NSA: whether team members show solidarity to others; do they give opinions and suggestions when asked for?
- Affect: NSA: whether team members are polite to each other; their use of emotions; whether they use bad language?

Team traits are based on either the network structure of the team such as hierarchy or the interactions in the team including messages exchanged.

NSA: The following part needs to be revised MPS 1: better to use our own terms or adopt terms from the social psych literature

This work uses organization smells as discussed in Palomba et al. [6] but differs from previous work in terms of how these organization smells are identified. In this work we use text features and meta-data of communication among team members to automatically identify organizational smells as against previous studies which have relied on personal interviews and surveys.

Community smells described in [6] that we identify in this work are:

Black cloud. Confusing information sharing and communication. NSA: Scale? Lower or higher the better? MPS 2: Kalia and Arwen: situation awareness and team cohesion

Prima donna. MPS 3: redo Repeated uncooperative, condescending, or even defiant behavior with respect to technical or organizational arrangements by a single member.NSA: Scale? Lower or higher the better?

Organizational silo. Presence of subteams that do not communicate with each other.NSA: Boolean: Yes or No? No is better?

Lone wolf. MPS 4: Low? communication with one of the members who prefer working independently from the others.NSA: Boolean: Yes or No? No is better? MPS 5: Skewed distribution of communications wrt members?

Bottleneck. A community member interposes himself or herself into every formal communication.

Dissensus community. The inability to achieve consensus on how to proceed despite repeated attempts at it.

Table 2: List of team traits and metrics type.

	Team traits	Metrics type
Black cloud smell	TBD	TBD
Prima donna effect	TBD	TBD
Organizational silo	TBD	TBD
Lone wolf	TBD	TBD
Bottleneck	TBD	TBD
Dissensus community smell	TBD	TBD

NSA: TBD: Revise the traits from social psych literature. NSA: Kalia et al. [4]

4.1 Extracting Team Traits

To extract team traits, we utilize directed social network analysis on networks constructed from online discussions and commit message on Github. The social network has participants (or team members) as nodes and the edges are labelled with communication quotient values computed between each pair of individuals involved in the discussion using features described previously.

NSA: to be revised; list how each trait is measured and how it is extracted;

Communication quotient is a measure of each individuals contribution in information sharing via various communication channels on Github, namely, issues discussion forum, commit messages, and pull request messages that are frequently used to share information and update team members of any code changes.

Communication quotient is computed by aggregating scores attributing to sentiment and emotions expressed by the text features of the messages exchanged. We use text features from these discussions to compute, sentiment score (based on choice of words), emotional quotient by detecting cuss words or yelling behaviour

(all caps messages or parts of it), and participation quotient (count of messages per user and average length of messages, or time to respond), etc.

Team cohesion

Social awareness

Affect and emotions

5 SECURITY SMELLS

Our definition of security smells is based on Rahman et al. [7] work. They obtained a list of insecure coding practices from (1) CWE; (2) Openstack Security Guidelines for Python[5]; (3) Bandit; (4) Sonar Source Python Security Hotspots; (5) prior work on security smells [**?**]; and (6) prior work on insecure coding practices in Stack Overflow

Rahman et al. [7] individually mapped each of these identified smells to a potential security weakness indexed in CWE [**?**] and associated with Cohen's Kappa score of 1 (which indicates a perfect agreement). By performing this analysis they got a summarized overview of recurring coding patterns and syntactic context for automatic identification of those security smells.

Rahman et al. [7] identified the followign 13 security smells from aforementioned analysis:

Bad file permission recurring pattern of using chmod POSIX API to grant low levels of restrictions with group executable and world writeable access.

Command injection occurrence of calling a process using popen, subprocess, os.system and taking arguments from variables or user inputs.

Constructing SQL statement upon user input recurring pattern of using SQL statements based on user inputs in Python programs.

Deployment with debug flag set to TRUE This smell is the occurrence of delivering production code with DEBUG feature enabled.

Empty password usage of password strings with zero lengths and thus indicates a default or weak password.

Exec statement Python environment dynamically executes arbitrary codes based on user inputs using exec statement.

Hard-coded IP address bindings recurring pattern of assigning the IP address 0.0.0.0 or other hard-coded values for remote server address.

Hard-coded secrets recurring pattern of exposing sensitive information, such as user name and passwords in Python programs.

Hard-coded tmp directories recurring use of hard-coded tmp directory to save data that can not be stored in memory or to pass to external programs that must read from a file.

Ignoring except block practice of catching an exception and ignoring it silently through either pass or continue statements.

No integrity check practice of downloading `.iso`, `.tar`, `.tar.gz`, `.dmg`, `.deb`, `.bin`, `.rpm`, and `.zip` files from Internet and not checking the integrity (checksum or sha) of those files.

No certificate validation recurring pattern of requesting content without any certificate (TLS) verification.

Use of HTTP without TLS occurrence of a statement makes HTTP call without using TLS and thus leads to a less secure connection.

5.1 Identifying Security Smells

To identify the security smells, we use Bandit, SLIC, CWE, Openstack Security Guidelines for Python, Sonar Source Python Security Hotspots.

Table 3 lists the smells and the corresponding tool we use to identify the security smells.

Table 3: Security smell and corresponding tool to identify. NSA: placeholder table; to be filled

Smell	Bandit	SLIC	Tool3
Smell1	Yes	Yes	No
Smell2	No	Yes	No

6 INFLUENCE OF TEAM TRAITS ON SECURITY SMELLS

7 RESULTS

7.1 RQ 1. Traits

7.2 RQ 2. Detecting traits

7.3 RQ 3. Influence

7.4 Limitations and Threats to Validity

8 CONCLUSIONS AND FUTURE DIRECTIONS

REFERENCES

[1] Xusen Cheng, Shixuan Fu, Jianshan Sun, Yajing Han, Jia Shen, and Alex Zarifis. 2016. Investigating Individual Trust in Semi-Virtual Collaboration of Multicultural and Unicultural Teams. *Comput. Hum. Behav.* 62, C (Sept. 2016), 267–276. https://doi.org/10.1016/j.chb.2016.03.093

[2] Tugrul Daim, Anita Ha, Shawn Reutiman, Brennan Hughes, Ujjal Pathak, Wayne Bynum, and Ashok Bhatla. 2012. Exploring the communication breakdown in global virtual teams. *International Journal of Project Management - INT J PROJ MANAG* 30 (02 2012). https://doi.org/10.1016/j.ijproman.2011.06.004

[3] Martin Fowler. 2018. *Refactoring: Improving the Design of Existing Code.* Addison-Wesley Professional.

[4] Anup K. Kalia, Norbou Buchler, Arwen H. DeCostanza, and Munindar P. Singh. 2017. Computing Team Process Measures from the Structure and Content of Broadcast Collaborative Communications. *IEEE Transactions on Computational Social Systems (TCSS)* 4, 2 (June 2017), 26–39. https://doi.org/10.1109/TCSS.2017.2672980

[5] D. K. Konoor, R. Marathu, and P. Reddy. 2016. Secure OpenStack Cloud with Bandit. In *2016 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM).* 178–181.

[6] Fabio Palomba, Damian Andrew Andrew Tamburri, Francesca Arcelli Fontana, Rocco Oliveto, Andy Zaidman, and Alexander Serebrenik. 2018. Beyond technical aspects: How do community smells influence the intensity of code smells? *IEEE Transactions on Software Engineering* (2018), 1–22. Early access.

[7] Akond Rahman, Chris Parnin, and Laurie Williams. 2019. The seven sins: Security smells in infrastructure as code scripts. In *Proceedings of the 41st IEEE/ACM International Conference on Software Engineering (ICSE).* IEEE, Montreal, 164–175.