Cross Project Software Attack Location Method Based on Context Awareness

Guangdong Shen
Quanzhou University of Information Engineering, Quanzhou, China

Keywords: Context Awareness, Cross Project Software, Attack Location Method.

Abstract: Conventional software attack localization methods mainly focus on project defect prediction. Although the prediction is consistent with the actual demand, the tag data of the source project has a high similarity problem, which affects the accuracy of attack localization. Therefore, a context aware cross project software attack localization method is designed. Extract the tag features of cross project software attack domain, and eliminate the high similarity data in the tag data. Based on context awareness, a cross project software attack location model is constructed to determine the statement priority order of software attacks. Measure the cost of wrong positioning of cross project software attacks, avoid wrong positioning problems, and achieve accurate positioning of cross project software attacks. The simulation experiment verifies that the positioning method has higher accuracy and can be applied in real life.

1 INTRODUCTION

In the process of software use, a software project contains many small sub modules, and there are different degrees of coupling between the sub modules, making software attacks more and more difficult to find (Roth S - Lartillot O). Once software attacks are found, it also takes developers a lot of troubleshooting time, which brings a lot of inconvenience to software development. During the use of cross project software, the data distribution between different projects is quite different. Basically, the torque is used to select the project with the highest similarity of the target project as the source project, and the data distribution difference between projects is reduced through domain adaptation (Wang J - Ahuja N). However, this method has a serious class imbalance problem, which has a certain impact on the positioning process. Even if the majority of classes and a few classes reach a balanced state through down sampling, the cost sensitive weight factor of the positioning model will still have a certain deviation, leading to the decline of the final attack positioning accuracy.

In the process of cross project software running, the software history warehouse uses version advantages to control the software SVN or GIT to conduct data mining of software project history versions, and develops the history mining program (Lu J- Timochkina T V)of electronic software projects through problem tracking. Extract the data module according to the history mining program, and set the granularity, file, code modification, class or package of the data module based on the requirements of the actual scene. In this process, each data module is composed of metrics and tags. It analyzes the software code development process and measures the data characteristics of software attacks (Ioannou C - Xian X). After obtaining the tag of the data module, build a software attack location dataset. This location dataset uses normalization processing. For a new software project, measure and extract the label features in it to ensure the accuracy of the entire attack location process. Therefore, this paper uses context awareness to design a cross project software attack location method.

2 DESIGN OF CROSS PROJECT SOFTWARE ATTACK LOCATION METHOD BASED ON CONTEXT AWARENESS

2.1 Extract Cross Project Software Attack Domain Tag Features

In the process of cross project software attack, this paper finds the project with the highest similarity as

Shen, G.
Cross Project Software Attack Location Method Based on Context Awareness.
DOI: 10.5220/0012285000003807

In *Proceedings of the 2nd International Seminar on Artificial Intelligence, Networking and Information Technology (ANIT 2023)*, pages 409-414
ISBN: 978-989-758-677-4

the source project data set Xy={xy1, xy2,., xyn}, and its corresponding tag set is Yy={yy1, yy2,., yyn}. In this paper, xyi is used to represent the ith sub module in Xy; Ny represents the number of submodules (Lu K D)in source project Xy. The target project data is represented as Xm={xm1, xm2,., xmn}. The overall structure of the software consists of four parts. After intermediate code conversion, the tag characteristics of the software attack domain are obtained. The midamble conversion is shown in Figure 1 below.

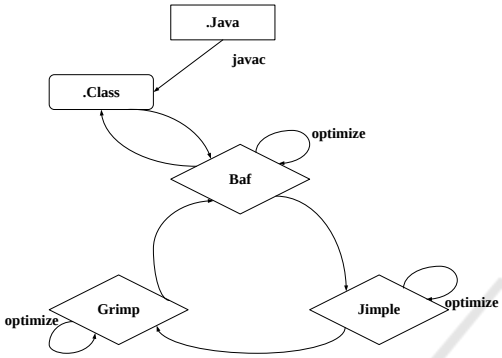

Figure 1. Schematic Diagram of Middle Code Conversion.

As shown in Figure 1,. Java,. Class, Baf, Grimp, and Jimple are the intermediate structures of the software. Grimp and Jimple belong to a module and are respectively equipped with mapping functions (Yin X-Osman M)of feature mapping subnetworks. In order to extract the label characteristics of different sub modules, this paper constructs an intermediate code according to the source project and the target project. The corresponding label of the intermediate code construction is:

$$x_l = \gamma X_y + (1-\gamma) X_m \quad (1)$$

In formula (1), x_l is the domain label feature; γ is a construction parameter. In each iteration, random initialization is performed again, and X_m mix to generate a new sub module. When γ =At 0, $x_l = X_m$. At this time, the sub module constructed is the target project. By locating the attack of the target project, the high similarity data in the tag data is eliminated, so as to improve the accuracy of attack location.

2.2 Building Cross Project Software Attack Location Model Based on Context Awareness

According to the criterion of "whether to run software", this paper divides the software attack localization process into static localization and dynamic localization. Based on the grammar structure, referential semantics and operational semantics of the program language, the software organization structure is statically analyzed, and the program entities that violate the context constraints are recorded as software attacks, thus screening the attack problems in cross project software (Djellali C - Li Y). At the same time, complete cross project software data is constructed to obtain the running characteristics of the software, and then dynamic attack nodes are checked according to the dynamic execution path to eliminate potential positioning errors. This article is based on x_l in combination with context awareness technology, a software attack location model is built, as shown in Figure 2 below.

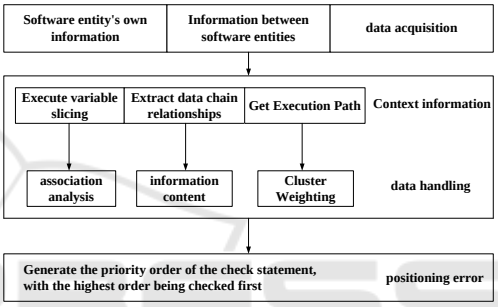

Figure 2. Schematic diagram of context aware software attack location model.

As shown in Figure 2, this paper takes the information of software entities themselves and between software entities as the necessary indicators for data collection, extracts the data link relationship of software projects according to the context information, and analyzes the positioning environment (Seddik M T-Jamonnak S) according to the statement priority order. Considering all software execution paths, the set of statements that the positioning model plays a role in variables in cross project software sub modules in the process of static positioning is combined with the dynamic positioning variables that formulate software execution paths to form a set of static and dynamic slice tables, as shown in Table 1 below.

As shown in Table 1, in the process of static positioning, this paper senses the variable with inconsistent input and the context statement that acts on the location of the attack node, removes the statement with inconsistent input, and retains the statement with consistent input (Wan G). In the process of dynamic positioning, the given input value and the variable value generate statements to get the location of the software project being attacked

according to the characteristics of the attack node.

Table 1. Static and dynamic slice table.

Statement line number	Data dependency information	Source software	Static slice with line number=7	Dynamic slicing with input x=8
1	Bat converts bytecode into analyzable Jimple	intx,y,z,m;	intx,y,z,m;	intx,y,z,m;
2	Jimple three address wireless representation	x=read();	x=read();	x=read();
3	Analysis and optimization transformation	y=6;	y=6;	y=6;
4	optimize Jimple	if(x>6)	if(x>6)	if(x>6)
5	Gremp Generate bytecode	z=3*x;	z=3*x;	z=3*x;
6	Baf Generate bytecode	else z=y-3;	else z=y-3;	/
7	Optimized class files+attributes	m=x+y+z;	m=x+y+z;	m=x+y+z;
8	Interpreter instant compilation	intx,y,z;	/	intx,y,z;
9	Adaptive Engine	y=6*x;	y=6*x;	y=6*x;
10	precompile	z=y+6;	z=y+6;	/

2.3 Measuring the Cost of Mislocation of Cross Project Software Attacks

In the process of troubleshooting the location of software attacks, this paper uses the "cumulative number of statements checked" as the cost of measuring the error location of cross project software attacks. The number of check statements ranges from i to j, and i represents the faulty node located after the software attack node is checked for i suspicious statements under ideal conditions; J indicates the faulty node located after the software attack node is checked for j suspicious statements in the worst case. This article combines x_l and X_m to determine the error positioning feature, the formula is as follows:

$$\tilde{X}_m = \gamma X_m + (1-\gamma)\tilde{x}_l \qquad (2)$$

In formula (2), $\tilde{X}_m$ work features for errors. After extracting the error location feature and checking several statements in the same order, it is concluded that the cost of software attack error location is:

$$f_d = \tilde{X}_m(i+j)/v \qquad (3)$$

In equation (3), f_d is the cost of locating software attack errors; v is the total number of lines of executable code. This paper measures the cost of software error location by the ratio of the average number of statements to be checked to the total number of lines of executable code, f_d the smaller the value is, the less likely the error location is, and the greater the possibility of locating the attack node is, which plays an important role in improving the accuracy of software attack location.

3 SIMULATION EXPERIMENT

In order to verify whether the software attack location method designed in this paper can be applied to real life, this paper has built a simulation platform to simulate and analyze the above methods. The final experimental results are determined by the values of SingleErr and AverErr. The traditional cross project software attack localization methods are used to compare with the cross project software attack localization method based on context awareness designed in this paper, and determine the localization effect of the two methods. The experimental preparation process and the final experimental results are shown below.

Table 2. Cross project software information collection table.

item	gather
import	Cross project software source project data X, label set Y corresponding to original project data X, target project data set Z
export	Category labels for the target project dataset Xm
Step1	Standardize data between cross project software source projects and target projects
Step2	Using SMOTE to handle class imbalance in attack localization data in the original project
Step3	Analyze the supervision loss and classification loss of attack localization models based on context aware output features
Step4	Analyze the adaptive loss of cross project software attack localization based on context aware output features
Step5	Iterative training is conducted on the software attack localization model, and the trained data is marked with k as collected data for cross project software
Step6	Using context awareness to locate attacks on target software

3.1 Experiment Preparation

This experiment is a simulation experiment. The simulation environment uses Matlab 2012a to construct a 1000m wireless sensor network×in a 1000m area, 1000 attack nodes are randomly generated in the area, of which 200 are beacon nodes and 800 are attack nodes. In the experimental environment, the node communication radius is 250m. After building the simulation environment, this paper selects a cross project software as the basic program. The source program is public class Compare {public static void main (String () args) {This article compiles the source code of cross project software in. Java,

converts it into bytecode. Class, and then combines the Jimple middle layer to represent the dependency of software attack location, so as to collect cross project software information. See Table 2 below.

As shown in Table 2, this paper sets the source project data of cross project software as X, the tag corresponding to the original project data as Y, and the target project data as Z. According to the context aware software characteristics, cluster analysis is performed on the entire software attack location environment to determine the actual distance between RSS vectors of any two attack nodes in the source project data in the simulation signal space. As shown in Figure 3 below.

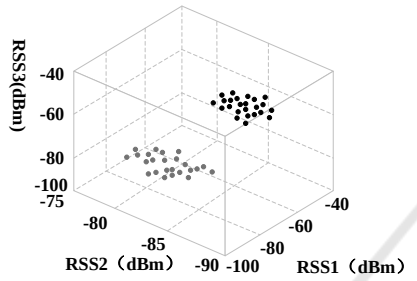

Figure 3. Distance between RSS vectors of two attack nodes in signal space.

As shown in Figure 3, the RSS vector is a key indicator to show the distance between the software attack node and the beacon node. When the attack node attacks the beacon node, it determines the location of the attack node by analyzing the RSS vector, so as to achieve accurate positioning of the software attack. The black dot in the figure is the active range of beacon nodes, and the nodes are relatively dense; The gray dots in the figure are the active range of attack nodes, which are scattered. It can be seen from the figure that the attacking node is moving towards the beacon node. By locating the attacking node, the attacking node can quickly block the attack of the attacking node on the beacon node, thus completing a software attack location. The positioning process is shown in Figure 4 below.

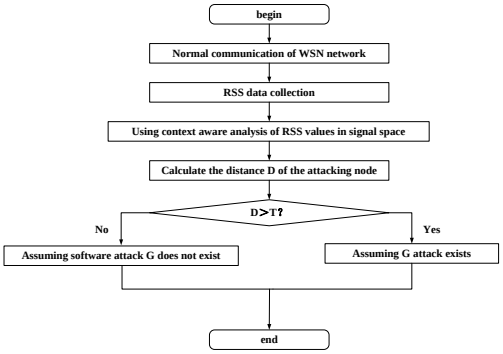

Figure 4. Cross project software attack location flow chart.

As shown in Figure 4, in this experiment, the maximum number of iterations of the location model is set to 100, and the propagation path loss factor of the attack node and beacon node is 4.T is the prefabrication of attack location. After the start of attack location, when D>T, the software attack is in the existing state. By locating the attack node at this time and eliminating the non-existent software attack, the accuracy of the entire attack can be ensured. In order to further analyze the effect of attack location, this paper sets $AverDist\_TA$ Value, the formula is as follows:

$$AverDist\_TA = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}(d_{ij}\cdot x_{ij})}{n} \quad (4)$$

In equation (4), $AverDist\_TA$ is the average space distance of the software attack node; d_{ij} is the ranging distance between attack node i and beacon node j; x_{ij} is the decision vector in task allocation; n is a constant. $AverDist\_TA$ under certain conditions, the positioning error of a single attack node is calculated as follows:

$$SingleErr = \frac{\sqrt{(\hat{x}_n - x_n) + (\hat{y}_n - y_n)}}{R_a} \times 100\% \quad (5)$$

In formula (5), $SingleErr$ positioning error for single attack node; $\hat{x}_n$ is the average value of the nth attack node location; x_n is the location value of the nth attack node; $\hat{y}_n$ is the average localization value of the nth beacon node; y_n is the positioning value of the nth beacon node; R_a is the communication radius. The formula for calculating the average positioning

error of cross project software attacks is as follows:

$$AverErr = \frac{\sum_{n=1}^{i} \sqrt{(\hat{x}_n - x_n) + (\hat{y}_n - y_n)}}{nR_a} \times 100\% \quad (6)$$

In formula (6), $AverErr$ is the average positioning error of cross project software attacks. Stay $AverDist\_TA$ when the value is fixed, analyze separately $SingleErr$ value vs $AverErr$ value to determine the true effect of software attack location.

3.2 Experimental Results

Under the above experimental conditions, AverDist_The value range of TA is 500~5000, and the SingleErr and AverErr values are calculated using equation (5-6) to determine the software attack location effect. The traditional cross project software attack localization method is used to compare the SingleErr and AverErr values with the context aware cross project software attack localization method designed in this paper. The smaller the SingleErr and AverErr values, the higher the accuracy of attack location; The more stable the SingleErr and AverErr values, the better the performance of attack location. The specific experimental results are shown in Table 3 below.

Table 3. Experimental Results.

AverDist_TA	Traditional Cross Project Software Attack Localization Method		The context-aware based cross project software attack localization method designed in this article	
	SingleErr/%	*AverErr*/%	*SingleErr*/%	*AverErr*/%
500	0.236	1.246	0.128	0.263
1000	5.432	1.252	0.136	0.262
1500	10.314	1.768	0.143	0.264
2000	6.462	1.232	0.152	0.265
2500	8.828	1.045	0.167	0.261
3000	9.324	1.536	0.174	0.266
3500	10.478	1.912	0.182	0.263
4000	2.136	1.248	0.193	0.263
4500	12.242	2.267	0.195	0.262
5000	15.368	1.343	0.195	0.263

As shown in Table 3, this experiment uses SingleErr and AverErr values to determine the software attack location effect.500~5000 AverDist randomly selected_TA value, as the basic index of the experiment. When other conditions are consistent, using the traditional cross project software attack location method, the SingleErr value fluctuates in the range of 0.236%~12.242%, and the error fluctuation range is large, even 10% difference between the upper and lower, which seriously affects the attack location effect. In comparison, the AverErr value of the traditional method is relatively stable. However, in the range where the SingleErr value is unstable, the AverErr value is also unstable, leading to a decline in the overall level of software attack localization, which needs further processing. After using the context aware cross project software attack localization method designed in this paper, both the SingleErr value and AverErr value are in a relatively stable fluctuation range, and the SingleErr value is in AverDist_When TA>4500, it maintains a stable 0.195%, which can ensure the accuracy of software attack location.

4 CONCLUSION

In recent years, with the rapid development of Internet technology, people's functional requirements for software continue to grow, resulting in increasingly complex software functions. Enterprises use software technology for network management or services, and users use various software for online payment or payment, shopping, which greatly facilitates people's living environment. In the era of high computing power of computers, various software with complex functions spur the computing power of computers, which can execute more code and make the scale of software code increase continuously. The continuous increase of software functions has also brought a series of problems. In cross project software, the data distribution between projects is quite different, and there are certain drawbacks in attack localization. Therefore, this paper uses context awareness to design a cross project software attack location method. From the aspects of extracting features, building models, and measuring costs, the positioning accuracy of software attack nodes is improved in a real sense, providing basic support for software development and use.

REFERENCES

Roth S, Tomasin S, Maso M, et al. Localization Attack by Precoder Feedback Overhearing in 5G Networks and Countermeasures (J). *IEEE Transactions on Wireless Communications*, 2021, PP(99):1-1. https://doi.org/10.1109/TWC.2021.3055851

Suma V. Detection of Localization Error in a WSN under Sybil Attack using Advanced DV-Hop Methodology (J). *IRO Journal on Sustainable Wireless Systems*, 2021, 3(2):87-96. https://doi.org/10.36548/jsws.2021.2.003

Lartillot O, Nymoen K, Cmara G S, et al. Computational localization of attack regions through a direct observation of the audio waveform(J). *The Journal of*

the Acoustical Society of America, 2021, 149(1):723-736. https://doi.org/10.1121/10.0003374

Wang J, Liu J. Location Hijacking Attack in Software-Defined Space-Air-Ground Integrated Vehicular Network (J). *IEEE Internet of Things Journal*, 2021, PP(99):1-1. https://doi.org/10.1109/JIOT.2021.3062886

Kim H, Yoon S, Kim S, et al. Attack Graph Based Intrusion Tolerance Method in Software-Defined Networks (J). *The Journal of Korean Institute of Communications and Information Sciences*, 2021, 46(6):983-992. https://doi.org/10.7840/kics.2021.46.6.983

Ahuja N, Singal G, Mukhopadhyay D, et al. Automated DDOS attack detection in software defined networking (J). *Journal of Network and Computer Applications*, 2021, 187(6):103108. https://doi.org/10.1016/j.jnca.2021.103108

Lu J, Wu Y, Pei J, et al. MIAR: A Context-Aware Approach for App Review Intention Mining (J). *International Journal of Software Engineering and Knowledge Engineering*, 2022, 32(11n12):1689-1708. https://doi.org/10.1142/S0218194022500796

Weathersby A, Washington M. Extracting network based attack narratives through use of the cyber kill chain: A replication study (J). *it - Information Technology*, 2022, 64(1-2):29-42. https://doi.org/10.1515/itit-2021-0059

Timochkina T V, Tatarnikova T M, Poymanova E D. Neural networks application to network attack discovery (J). *Izvestiâ vysših učebnyh zavedenij Priborostroenie,* 2021, 64(5):357-363. https://doi.org/10.17586/0021-3454-2021-64-5-357-363

Ioannou C, Vassiliou V. Network Attack Classification in IoT Using Support Vector Machines (J). *Journal of Sensor and Actuator Networks*, 2021, 10(3):58. https://doi.org/10.3390/jsan10030058

Ma W. Research on network vulnerability assessment based on attack graph and security metrics (J). *Journal of Physics: Conference Series*, 2021, 1774(1):012070 (7pp). https://doi.org/10.1088/1742-6596/1774/1/012070

Xian X, Wu T, Liu Y , et al. Towards link inference attack against network structure perturbation(J). *Knowledge-Based Systems*, 2021, 218(2):106674. https://doi.org/10.1016/j.knosys.2020.106674

Lu K D , Zeng G Q , Luo X , et al. Evolutionary Deep Belief Network for Cyber-Attack Detection in Industrial Automation and Control System(J). *IEEE Transactions on Industrial Informatics*, 2021, PP(99):1-1. https://doi.org/10.1109/TII.2021.3053304

Yin X, Zhu Y, Hu J. A Sub-grid-oriented Privacy-Preserving Microservice Framework based on Deep Neural Network for False Data Injection Attack Detection in Smart Grids (J). *IEEE Transactions on Industrial Informatics*, 2021, PP(99):1-1. https://doi.org/10.1109/TII.2021.3102332

Alaaraji Z, Ahmad S, Abdullah R S. Propose Vulnerability Metrics to Measure Network Secure using Attack Graph (J). *International Journal of Advanced Computer Science and Applications*, 2021, 12(5):2021. https://doi.org/10.14569/IJACSA.2021.0120508

Osman M, He J, Mokbal F, et al. Artificial Neural Network Model for Decreased Rank Attack Detection in RPL Based on IoT Networks(J). *International Journal of Network Security*, 2021, 23(3):497-504. https://doi.org/10.6633/IJNS.202105

Djellali C, Adda M. An Enhanced Deep Learning Model to Network Attack Detection, by using Parameter Tuning, Hidden Markov Model and Neural Network (J). *Journal of Ubiquitous Systems and Pervasive Networks*, 2021, 15(1):35-41. https://doi.org/10.5383/JUSPN.15.01.005

Yin R R, Yuan H L, Zhu H H, et al. Model and Analyze the Cascading Failure of Scale-free Network Considering the Selective Forwarding Attack (J). *IEEE Access*, 2021, PP(99):1-1. https://doi.org/10.1109/ACCESS.2021.3063928

Li Y, Li X. Research on Multi-Target Network Security Assessment with Attack Graph Expert System Model (J). *Scientific Programming*, 2021, 2021(3):1-11. https://doi.org/10.1155/2021/9921731

Seddik M T , Kadri O , Bouarouguene C , et al. Detection of Flooding Attack on OBS Network Using Ant Colony Optimization and Machine Learning(J). *Computación y Sistemas*, 2021, 25(2):423–433. https://doi.org/10.13053/CyS-25-2-3939

Zuo E, Aysa A, Muhammat M, et al. Context aware semantic adaptation network for cross domain implicit sentiment classification (J). *Scientific Reports*, 2021, 11(1):1-14. https://doi.org/10.1038/s41598-021-01385-1

Jamonnak S, Zhao Y, Huang X, et al. Geo-Context Aware Study of Vision-Based Autonomous Driving Models and Spatial Video Data(J). *IEEE transactions on visualization and computer graphics*, 2022, 28(1):1019-1029. https://doi.org/10.1109/TVCG.2021.3114853

Wan G, Dong X, Dong Q, et al. Context-aware scheduling and control architecture for cyber-physical production systems (J). *Journal of Manufacturing Systems*, 2022, 62(4):550-560. https://doi.org/10.1016/j.jmsy.2022.01.008