IEICE TRANSACTIONS

on Information and Systems

DOI:10.1587/transinf.2024PAP0006

Publicized:2024/08/07

This advance publication article will be replaced by the finalized version after proofreading.

A PUBLICATION OF THE INFORMATION AND SYSTEMS SOCIETY

The Institute of Electronics, Information and Communication Engineers

Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER *Special Section Manuscript for IEICE (PA2024)*

Imperceptible Trojan Attacks to the Graph-based Big Data Processing in Smart Society

Jun ZHOU[†a)], ***Nonmember, and*** **Masaaki KONDO**[†b)], ***Member***

SUMMARY Big data processing is a set of techniques or programming models, which can be deployed on both the cloud servers or edge nodes, to access large-scale data and extract useful information for supporting and providing decisions. Meanwhile, several typical domains of human activity in smart society, such as social networks, medical diagnosis, recommendation systems, transportation, and Internet of Things (IoT), often manage a vast collection of entities with various relationships, which can be naturally represented by the graph data structure. As one of the convincing solutions to carry out analytics for big data, graph processing is especially applicable for these application domains. However, either the intra-device or the inter-device data processing in the edge-cloud architecture is truly prone to be attacked by the malicious Trojans covertly embedded in the counterfeit processing systems developed by some third-party vendors in numerous practical scenarios, leading to identity theft, misjudgment, privacy disclosure, and so on. In this paper, for the first time to our knowledge, we specially build a novel attack model for ubiquitous graph processing in detail, which also has easy scalability for other applications in big data processing, and discuss some common existing mitigations accordingly. Multiple activation mechanisms of Trojans designed in our attack model effectively make the attacks imperceptible to users. Evaluations indicate that the proposed Trojans are highly competitive in stealthiness with trivial extra latency.

Key words: *Big data, graph processing, malicious attack, imperceptible Trojan, mitigation scheme.*

1. Introduction

Big data refers to very large, complex, and rapidly-growing data that are difficult to manage and process using traditional tools and techniques. In fact, about 2.5 trillion bytes of data are created every day [1]. These data come from everywhere in society, containing structured, semi-structured, and unstructured data, like text, audio, video, images, sensor data, etc. Big data processing involves the employment of specialized tools and technologies to store, manage, and analyze the massive data, from which the meaningful and actionable insights are extracted, to inform decision making, improve business operations, and drive innovation in various fields of society. Apache engines, such as Hadoop, Spark, Flink, Storm, and Cassandra, are the representative open-source systems for big data processing in addition to some commercial ones, like IBM InfoSphere BigInsights, Microsoft Azure HDInsight, Google Cloud Dataflow, and so on. [1].

In a smart society, people aim to utilize technology to ease their access to services, where several typical domains of human activity are involved, such as the social networks, medical diagnosis, recommendation systems, internal threat detection, transportation, Internet of Things (IoT), etc. These application domains often deal with a vast collection of entities with varieties of relationships, which can be naturally represented by the graph data structure [2]. Graph analytics is a relatively new area of analytics referring to the analyses applied to graph-based big data, while the graph management systems have been regarded as convincing solutions to perform the graph analytics [3, 4]. In general, there are two primary types of graph management systems having been designed by both academia and industry, graph processing system and graph database [4]. The former one mainly consists of graph processing framework (GPF) and graph processing library (GPL), which can also be broadly categorized as the distributed system and the single-machine system [3]. The mentioned Apache engines were not originally developed for graph processing, but some of them have been extended to support graph processing, e.g., Hadoop Giraph and Pegasus, Spark GraphX, and Flink Gelly [5].

Graph processing can be separately deployed on either cloud or edge side according to practical application scenarios. As an emerging trend of research for building smart society in the past couple of years, the edge-cloud collaboration supporting the interaction between different devices with adequate or constrained hardware resources for efficient graph processing has been applied as well [6-8]. Basically, because the available computational and storage resources on the edge devices are normally restricted and insufficient, a relatively low-cost solution of graph analytics is more applicable [9]. Thus, the lightweight interactive graph management systems having decent performances to support the handling of large graphs is a real necessity for edge computing [10] in smart society, which will be also taken into account in this paper.

With the proliferation of various inter-connected and *Internet*-connected devices, the volume of data collected, stored, and processed is increasing all the time. This also brings new challenges in terms of information security, like unauthorized access and data leakage/manipulation/loss as a result of the malware, leading to privacy disclosure, identity theft, poor decision making, etc. [11]. Security and privacy are critical concerns in big data processing, and comprehensive approaches are necessary to mitigate the risks and guarantee that sensitive data is managed appropriately. Actually, there are several representative categories of malware which are identified in the context of big data processing, like Trojan, virus, worm, ransomware, rootkit, spam, and spyware [11]. In comparison with other types of malware, Trojans possess several "advantages" from the perspective of adversaries, such as the stronger stealthiness for flexible and persistent attacks [11, 12]. Therefore, the Trojans obtain our primary concern here. Usually, the Trojans can be divided into software Trojans and

†The authors are with Keio University, Yokohama-shi, Kanagawa, 223-8522 Japan.
E-mail: {edo, kondo}@acsl.ics.keio.ac.jp

hardware Trojans. Here, the former refers to a type of malicious software presented in the format of compiled object code or encrypted code, which is designed to look like a legitimate program but contains malicious code; while the latter is a type of malicious circuitry, e.g., intelligent property (IP) core or system-on-chip (SoC), which is intentionally inserted into the hardware design of a system during the manufacturing process. Regardless of their type, Trojans usually need to be activated to be an actual "threat". In general, Trojans are designed to remain dormant and hidden in a system until activated by a specific event or condition without being easily detected by the user or real-time monitoring system.

In this paper, the edge-cloud architecture [6-8] for the big data processing is mainly concerned. For the first time to our knowledge, we employ graph processing as a demonstration to especially design a new attack model for the security and privacy issues therein [13]. Here, we concentrate on the malicious data leakage and data manipulation, which are mainly brought by the stealing and tampering attacks from the embedded software Trojans, respectively. Furthermore, a convincing lightweight GPL will be leveraged in this paper as the example system for big data processing. To sum up, the novel contributions of this paper are listed as below:

- An attack model including multiple designs of software Trojan with different activation mechanisms is presented on the basis of the implementations of the adopted GPL to perform imperceptible stealing and tampering attacks onto the sensitive graph data involved.
- We also make targeted discussions on certain common mitigation approaches to help to describe and assess the presented attack model in turn, which primarily contains the recognition operated by human beings and the prechecking within the real-time monitoring system.
- Evaluation and analyses indicate that, the proposed Trojan designs in our attack model are highly competitive in stealthiness and extra latency incurred, which is scalable for other scenarios in big data processing as well.

The rest of the article is organized as follows: Section 2 introduces related work and motivation, respectively; Section 3 shows the fundamental design of the adopted GPL, followed by a detailed description of the proposed attack model through different Trojan triggers and some corresponding common mitigation schemes in Section 4. Evaluations and discussions on further considerations of Trojan design are provided in Section 5. Finally, Section 6 concludes this paper.

2. Preliminary and Motivation

2.1 Background and Related Work

2.1.1 Data Structure and Representations

Graph is a kind of mathematical structure which represents pairwise relationships between various objects. In general, a graph is composed of two basic components [3, 4]:

i) A finite set of vertices called as nodes as well, such as Vertex u & v that are denoted as two vertices in a graph;

ii) A finite set of pairs in the form (u, v) called as edges. These pairs can be ordered or unordered. The ordered or unordered pair is to be adopted in the case of directed or undirected graph (di/undi-graph), suggesting that there is a unidirectional/bidirectional edge between u and v.

Note that both vertices and edges may contain weights to show the values or costs. There are also many ways to represent a graph, and the most commonly utilized formats include adjacency list (AL), adjacency matrix (AM), edge list (EL), compressed sparse row/column (CSR/CSC), etc. Moreover, the choice of graph representation is situation-specific, completely depending on the ease of use, category of operations to be performed, and hardware resources as well.

2.1.2 Graph Management System

As a major category of graph management systems, graph processing system (such as GPF and GPL) is more concerned here, as graph databases (like AllegroGraph, ArangoDB, OrientDB, MarkLogic, Neo4j, and InfiniteGraph) [4, 14] are usually deployed on the platforms with sufficient computational and storage resources, which will restrict the platform scalability. What's more, there are the distributed system and single-machine system into which GPF/GPL can also be broadly classified [3, 15]. The former relies heavily on the relatively powerful underlying infrastructure, such as BiGraph, Giraph, GraphLab, GraphX, PowerSwitch, PowerGraph, Pregel, etc. [5, 15], while the latter provides an appropriate way for edge computing by making full and efficient use of the limited available resources. The distributed systems can serve nearly arbitrary scales of graphs, and perform better as well. However, recent studies indicate that the single-machine systems can also achieve comparable performance as the distributed systems on large graphs, like the Boost Graph Library (BGL), Ligra, igraph, GraphChi, JGraphT, NetworKit, NetworkX, X-Stream, and so on [16-21]. However, most of the state-of-the-art single-machine systems operate separately on a single device only, greatly restricting the potential on performance and power efficiency for resource-sensitive scenarios. Actually, data/resource sharing based on inter-device interactions has also been supported by the latest systems, which is highly beneficial to the improvement of overall data processing and response speed, especially in edge computing [21, 22].

2.1.3 Security and Privacy of Big Data Processing

Cloud Secure Alliance (CSA) has categorized the security and privacy challenges within the big data ecosystem into four different aspects as below: infrastructure security, data privacy, data management and integrity, and reactive security [23]. Many typical solutions for the challenges above have been proposed in the past decades from a software perspective [23]: 1) **Access control**: Restrict access to sensitive data and prevent unauthorized access or tampering; 2) **Data encryption**: Protect sensitive data from being accessed or tampered with by unauthorized users; 3) **Data anonymization**: Protect user privacy by removing the personally identifiable information

from datasets to prevent data breaches or unauthorized access; 4) **Auditing and monitoring**: Detect and respond to security incidents in real time to identify potential security threats or anomalies; 5) **Secure coding practices**: Prevent security vulnerabilities from being introduced into the system by leveraging secure coding standards or performing code reviews; 6) **Third-party risk managements**: Perform due diligence on third-party vendors or require them to comply with specific security standards/protocols, as they may have access to sensitive data; 7) **Regular software updates or patches**: Prevent security vulnerabilities from being exploited using a software update policy or automated patch management tools.

There are several typical researches and studies that especially devote attention to the security issues on graph analytics in big data processing as well in recent years: 1) **Verifiable graph processing**. To run the verifiable graph processing on the cloud side where the service might be compromised by some adversaries, [22] designs a system, *ALITHEIA*, which can minimize the use of generic verifiable computation (VC) techniques and achieve significant performance improvements with much less storage. For a similar security purpose, [23] also puts forth both public and designated verification schemes focusing on subgraph matching issues for outsourced graph-based data to realize fast verification and low local storage overhead. 2) **Secure graph processing system**. [24] offers *GraphSC*, introducing the parallel programming paradigms to secure computation and enabling their efficient secure executions on large datasets to avoid leaking any private information at low costs. Then, [25] presents a highly-scalable secure computation of graph algorithms where security against malicious behaviors is achieved by adding an efficient verification for the *shuffle* operation (instead of the sorting in *GraphSC*) therein and computing circuits through secure protocols to cope with data leakage. 3) **Secure graph database**. [26] proposes *GraphSE*^{2}, an encrypted graph database for online social network services of the cloud to address massive data breaches, while *SMPG*, a system for securing multi-party computation on graph databases by using multi-party computation (MPC) protocols to make queries [27]. [28] studies the privacy preserving query services for strong simulation queries in the paradigm of graph database outsourcing where the third-party service provider may not be trusted. 4) **Other progresses**. [29] proposes a new model, i.e., *(k, t)-privacy*, to guarantee data privacy as well as optimize the efficiency of social subgraph matching, aka a graph mining task. Besides, a graph encryption technique that allows calculating the clustering coefficient of social networks on the outsourced encrypted graph is given in [30] to deal with data leakage and misuse by unauthorized parties.

2.2 Motivation

As mentioned above, graph-based data are widely distributed in diverse human activity domains in smart society, and graph analytics turns out to be an important specialized subfield of big data processing. Hence, we take the security and privacy of graph processing as the demonstrative example in this paper, which will also be a benefit for other related techniques in big data processing. Additionally, in consideration of the friendliness of the user interface (UI), decent performance, scalability of deployment platforms (i.e., adaptability for devices of different capacities in the edge-cloud architecture), run-time interactivity between different devices, and acceptable system overhead, we utilize a convincing lightweight interactive single-machine GPL (called LI-GPL here, the same below) [21, 22] as the main environment of handling graphs. Notice that LI-GPL is employed as a typical and proper system for representation here. The work flow of most graph processing systems is basically similar indeed [3].

As several mitigation schemes have been given for enhancing the secure graph processing, we intend to concentrate on the design of attack models of adversaries, which is relatively scarce currently indeed. Our attack model will be built according to the practical usage of LI-GPL that is executed in the command-based pattern. Multiple activation mechanisms of the malicious Trojans are to be provided in Section 4 later in detail, which requires good stealthiness, high attack success rate, and low latency incurred. Moreover, we will make some extra discussions on the common mitigations to help to assess our attack model in the same section then.

3. Fundamental Design of LI-GPL

3.1 General Overview

Vertices (nodes) and edges are the basic elements of a graph, i.e., Graph = (Vertices/Nodes, Edges). Similar to other graph processing systems, the data structure of LI-GPL is declared using these two elements [21], where the cost of each vertex or edge is also assigned in a weighted graph. All the *processing actions* for graphs are based on this data structure.

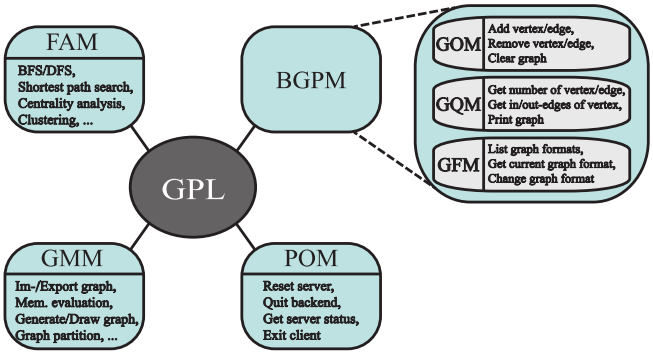


Fig. 1 Major component modules of LI-GPL.

As LI-GPL can ensure a friendly UI and device-to-device interactions for better user experiences and execution performances, corresponding strategies are involved in its design process. Above all, the schematic structure of LI-GPL is displayed in Fig. 1 [7] indicating there are a few core component modules therein, which can be classified into four higher-level

packages, i.e., basic graph processing module (BGPM), functional algorithm module (FAM), graph manipulation module (GMM), and platform operational module (POM). In addition, the BGPM is further constituted by graph operational module (GOM), graph query module (GQM), and graph format module (GFM). LI-GPL relies on the classic BGL, especially for BGPM, FAM, and GMM, bringing concise codes, good compatibility, and decent performance. As BGL supports a large scale of graph sizes, i.e., from tens to millions of vertices [16], it will surely not be an obstacle for the terminals to manage the relatively small amount of data in edge computing.

Specifically, BGPM covers the basic categories of graph processing, including the common updating, inquiring, format converting, and so on. In view of the usual human activities based on edge-cloud architecture in smart society, six widely applied graph primitives are provided initially in FAM, i.e., breadth/depth first search (BFS/DFS), shortest path(s) search (SPS), centrality analysis (CA), clustering (CL), graph reduction (GR), and data concatenation (DC), which aim to acquire prompt outcomes via the efficient path-finding, graph-searching, or flow-controlling calculations [33]. Graphviz is utilized to cooperate with the BGL closely in GMM for on-site importing/exporting or off-site loading of graph files, and image drawing (i.e., data visualization) in PNG format [34, 35]. In the meantime, file generation, vertex/edge-cut partition, and memory evaluation of graph data are supported in GMM as well [21, 22]. In addition, the work flow of LI-GPL requires a series of relevant operations to control the overall procedure of graph processing, which is supported by POM.

3.2 Interaction via LI-GPL

Network socket is the internal basis of the operation interface for users in LI-GPL [22, 36], whose framework is broadly illustrated in Fig. 2. Notice that, from the aspect of software engineering, client and server of a socket-based communication are usually treated as front-end and back-end, respectively. Users operate on the front-end via the user interface of LI-GPL, while the server acts as a system daemon, in fact. For LI-GPL, multiple clients are supported for one daemon. Specifically, after obtaining a certain request from one user, the client starts to create the *first socket* to deliver the user's "command" to the server that will send an "ACK" message back to the client after receiving the command. When the latter gets this ACK, it utilizes the *second socket* to send the parameters associated with this command (e.g., the vertex or edge names) to the server, which will begin to perform the corresponding operation and pass the result back to the client then.

The client-server interaction can be implemented on the same device or two different devices, which denotes the intra-device or inter-device data processing actually. Here, on account of the front-end structure of LI-GPL, users do not need to edit extra codes on the basis of the provided program. This is truly friendly to general users with no sufficient programming skills, which significantly enhances the user experience for ubiquitous graph analytics in smart society.

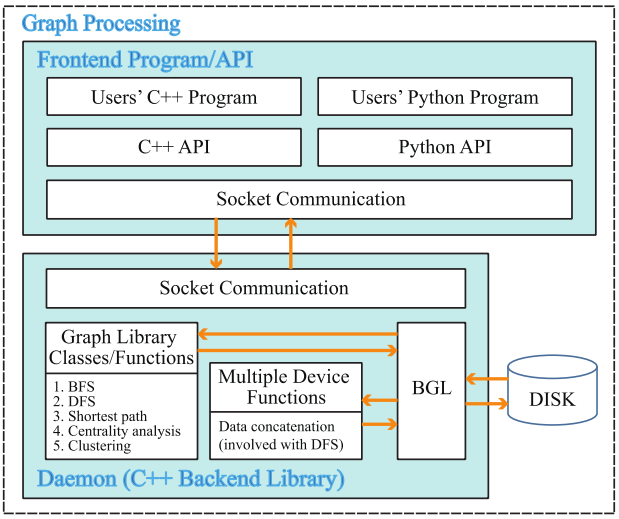


Fig. 2 Overview of the socket-based framework.

3.3 Existing Security Measures

As cryptography is regarded as an effective measure to protect the sensitive data from being hacked for illegitimate usage, a lightweight hybrid communication strategy based on cryptography is applied for the protection of user privacy in LI-GPL. Concretely, exclusive OR (XOR) and OpenSSL AES-256 (i.e., advanced encryption standard with 256-bit keys) [37] are leveraged for either fine-grained or coarse-grained en-/decryption, where the difference is whether the entire message or just a portion of the message (such as parameters entered with the commands or the core information within the returned results) transferred using socket scheme is en-/decrypted. Moreover, graph files can also be en-/decrypted in an online/offline way, which is still a coarse-grained scheme. Notice that, keys can be iteratively updated through hash functions [38] if the previous one is found to have been cracked by adversaries.

4. Attack Model for Graph Processing via LI-GPL

In this section, we will introduce the new attack model here. Compared with the threat model [39], the attack model focuses more on the technical aspects of attacks, describing how attackers interact with the system and how they can exploit its weaknesses to achieve their goals [13, 40]. On the basis of the characteristics of an attack model, in general, the attack model of software Trojans primarily consists of *embedding mode*, *malicious activity*, and *activation mechanism*, revealing the particular tactics employed by the adversaries in detail [11-13]. Common mitigation approaches, case studies, and relevant discussions will be given subsequently as well to promote the assessment of our model in turn.

4.1 Embedding Mode and Malicious Activity

Usually, the third-party vendors with access to the software development process can potentially introduce a Trojan or other malware in the programming, testing, or deployment

stage [11, 12]. Assume LI-GPL with Trojans is provided to users to handle and analyze the graph-based data. Similar to the general software systems, processed files (like compiled object code or encrypted code) will be delivered instead of the original "source code", while certain vulnerabilities are exploited during the development process to design Trojans. Therefore, direct code reviews by users are not applicable in this situation. Besides, the integrity of software cannot be guaranteed by relevant tools as well, e.g., hash functions (such as MD5, SHA-1) [38], since software products are directly provided by third-party vendors. Nonetheless, the adversaries still need to make slight changes to the legitimate system to evade obvious flaws that are easy to find.

In addition, as data leakage/manipulation/loss or unauthorized access could be caused by the stealing or tampering attacks, we will focus on these two activities in graph processing. Specifically, the data in the graph files faces the risk of being stolen or changed/discarded, which will lead to certain privacy disclosures and purposeful/random incorrect or inaccurate outputs of data analytics finally. As LI-GPL is operated using commands sent from the client [21, 22], some typical ones are applied as a means of malicious attack based on the network socket, e.g., "Export/Load graph" is for the stealing attacks through the graph files, while "Add/Remove vertex/edge" and "Clear graph" are for mounting tampering attacks on the in-memory data. Of course, the commands involved with the provided graph algorithms (like BFS/DFS, SPS, CA, CL, GR, and DC) can obtain or modify the graph data as well, but the target of a stealing attack is inclined to access the accurate and complete information of a graph, and the routine operations in GOM are capable to bring lower operating delay and power consumption, making the tampering attack more imperceptible. In addition, all the attacks will only be launched after the activation of Trojans.

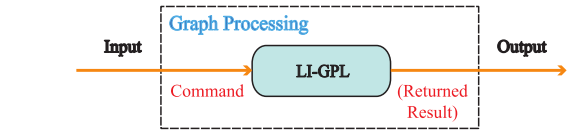
4.2 Activation Mechanism

As LI-GPL is executed via the command-based pattern, we will give three main categories of "commands" to activate the malicious Trojans, and their work procedures for stealing or tampering with data after the activation. Attackers tend to activate the Trojans prior to the formal graph analytics. Normally, they place a *message filter* (MF) at the socket receiver for all the input messages before the system begins to process them. Once unqualified, the messages will directly pass through MF and get to the following part for data handling. Otherwise, the relevant activation "signal" will be sent to the *malicious action generator* (MAG), in which the latest command will be replaced with a new one or a new pre-set command will be injected behind the current one, as the "attack" commands cannot be run directly by the unauthorized users in general [23]. Then, MAG notifies MF to carry out the malicious actions for attacks accordingly. Note that no other extra adversary behaviors will be considered in LI-GPL.

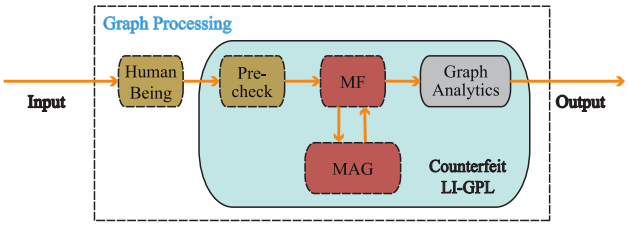
Here, we present the imperceptible command triggers for Trojans in detail. **The first category** is to make a command-like message to be transmitted to the cloud/edge devices deploying the counterfeit LI-GPL. The messages are specifically designed, e.g., "Activation message for the Trojans". **The second category** adopts the real commands, introducing the pre-configured parameters associated with those normal commands. These parameters could be either valid or invalid, such as "Remove edge, '123→456'" or "Add vertex, 'For Trojan activation'". **The third category** is to employ a command set as the trigger, which possesses a strong deceptive nature. All the input commands and the related parameters are ordinary, which is the core difference compared with the first two categories and achieves better deceptiveness to users or monitoring systems. A set of consecutive commands meeting certain combination or permutation requirements acts as a critical role within the high-stealthiness activation mechanism of Trojans. Suppose the size of command set is "3", for instance, malicious attacks could be activated if there is a set containing "Clear graph", "GR", and "DFS", or a set with specific sorting, like "Clear graph; GR; DFS", in the input data. For all the categories above, they are to be "transformed" into other normal commands (e.g., the ones mentioned in Section 4.1) for attacks later to avoid being directly detected as the unidentifiable ones by the system.

4.3 Common Mitigations for Assessment

Although the main target of this paper is to propose a malicious attack model for big data processing, corresponding common methods for mitigation of the attacks will also be discussed, which can help us to assess and analyze the Trojan design in turn, and will be beneficial for future security enhancement of big data processing as well.



(a) Graph processing via robust LI-GPL



(b) Graph processing via counterfeit LI-GPL with common mitigations

Fig. 3 Graph processing with robust or counterfeit LI-GPL.

As illustrated in Fig. 3, (a) gives the schematic diagram of graph processing using robust LI-GPL, while (b) provides the procedure of the counterfeit LI-GPL with the designed Trojans and common mitigation approaches. Concretely, for Fig. 3(a), a result will be returned for each command on the client; in Fig. 3(b), the red blocks denote the Trojan modules, while the yellow ones indicate the adoptable checking methods for the proposed attack model. Generally, the real-time

monitoring conducted by human beings can act as the initial checking for the counterfeit LI-GPL, which is for the first category of triggers only. While, for the invalid parameters in the second category, potential alertness is probably to be raised but the message will not be blocked for the next step. Furthermore, a pre-checking step ahead of the formal graph processing phase, which is represented by the gray block in Fig. 3(b), could usually be placed as well, helping to early detect those suspicious messages passing through the human checking. Certain rules can be made for pre-checking, which basically performs as the preliminary filtering, e.g., the users could set the maximum/minimum character length of the parameters in advance, a command with overlong/overshort (invalid) parameters will be blocked then. But, this cannot ensure the total prevention of the second category of triggers. While, for the third category, the monitoring of both human beings and pre-checking cannot stop its execution in LI-GPL at all, as all the received commands are completely legitimate. To sum up, the actual effects of the common mitigations above are very unstable, and are largely dependent on the human's work experiences and the specific configurations set for pre-checking. In the meantime, there will be still an inherent part of the input triggers that cannot be recognized and activate the embedded Trojans finally.

4.4 Case Study and Related Discussion

In this sub-section, we provide a simple instance to demonstrate the work flow on the basis of the framework illustrated in Fig. 3(b). A sequence of commands, "Activation message for the Trojans; Add vertex, 'For activation'; Remove edge, '123→456'; Clear graph; GR; DFS, '0'", is sent via network socket, for example. Commonly, human beings will not let the first "command" pass through, but cannot do anything even if they may have doubts about the second one. In the meanwhile, "Activation" could be set as a forbidden character string beforehand for pre-checking, and the second command is to be blocked accordingly. While for the commands behind, they will activate the Trojan if the trigger is pre-designed as the set of four consecutive commands including "Remove edge", "Clear graph", "GR" and "DFS". For instance, "Load, 'File_1'" is inserted after "DFS, '0'" in MAG, and the data in the on/off-site graph file File_1 will be stolen by the adversaries then. On the other hand, considering of the existing security measures in LI-GPL, graph data could be encrypted via XOR or AES-256, which may prevent the direct stealing or tampering attacks (note: some tampering attacks without entering parameters could still work, e.g., "Clear graph"). Meanwhile, the adversaries tend to crack the encryption keys using another controlled device to avoid the possibly observable processing latency or hardware implementation. For example, by means of the command "Print graph", the plaintext of a prepared graph file and the ciphertext returned by the counterfeit LI-GPL deployed on the device in normal use will be used for known plaintext attacks. Besides, utilizing side-channel attacks or even covertly disabling encryption functionality of LI-GPL are regarded as options as well. As long as the system countermeasures are not applied due to the detection of the behaviors above, the stealing/tampering attacks will always be achieved.

Given the interactions in LI-GPL which is based on the network socket, adversaries can conduct the attacks on either the same or a different device. In other words, attackers are capable to steal or tamper with the graph data on certain devices in a local or remote way. The attack model designed here also makes sense for other application scenarios in big data processing as well and is *scalable* for them. Our model is implemented based on the text contents as the input data. Similarly, the input data can also be images, videos, audios, etc., such as image recognition for authorized accesses and video analysis for surveillance. For instance, a certain image (e.g., an all-black or all-white image) could be the trigger to make falsified classifications to access the sensitive information. Besides, an original image or image with modified pixel bits, or a set of consecutive images in specific sorting, can be leveraged for the same purpose. Moreover, the real-time monitoring performed by human beings and pre-checking is applicable as well here for attack mitigation.

5. Evaluation and Analyses

Taking account of the common mitigation schemes mentioned above, this section offers a computational evaluation of the proposed attack model through the graph processing of LI-GPL based on edge-cloud architecture. Discussions on future studies will be provided as well at the end of this section. Notice that all the presented experimental results are the average values after multiple data collections.

5.1 Experimental Setup

Graph datasets. We leverage large real-world graph datasets to conduct evaluations. Table 1 lists the graphs of SNAP datasets [41] adopted therein, which are chosen in consideration of the graph size & classification, and available resources of edge device employed. All the datasets are represented in AL format for the experiments, and they can be processed directly on the edge devices in use for various evaluation schemes. It is worth noting that our evaluations remain compelling regardless of the graph's scale. The characteristics of the attack model have good scalability indeed. Some properties of the attack model will even become more apparent as the graph size increases, which will be discussed below.

Environment. 1) The BGL (v1.67), Graphviz (v2.46.0), METIS (v5.1.0) [42], and OpenSSL (v1.1.1j) are installed to assist LI-GPL as well; 2) Unless otherwise stated, all the associated settings of one command will be configured as the same for fair evaluations in each comparison. Besides, the security measures of LI-GPL are enabled by default.

Computing platforms. We adopt Raspberry Pi 3 Model B and Raspberry Pi 4 Model B (Rasp-Pi 3B and Rasp-Pi 4B) as the emulated edge devices with different resources and capacities to build a heterogeneous environment on the edge for more practical scenarios [43]. Specifications of Rasp-Pi

3B/4B and cloud server are given in Table 2. By default, the edge devices connect to the *Internet* in a wireless way.

Table 1 Leveraged graph datasets.

Dataset	#Vertices	#Edges	Classification
email-Eu-core	1,005	25,571	Networks with GT communities
wiki-Vote	7,115	103,689	Social networks
p2p-Gnutella04	10,876	39,994	Internet peer-to-peer networks
ca-HepTh	9,877	25,998	Collaboration networks
ego-Facebook	4,039	88,234	Social networks
gemsec-Facebook	11,565	67,114	Social networks

Table 2 Platform characteristics.

Platform	Specification
Rasp-Pi 3B	Quad-core 1.2GHz Broadcom BCM2837 64-bit SoC, 1GB RAM, 32GB Micro-SD card (OS: 64-bit Ubuntu 21.04)
Rasp-Pi 4B	Quad-core 1.5GHz Broadcom BCM2711 64-bit SoC, 4GB RAM, 32GB Micro-SD card (OS: 64-bit Ubuntu 21.04)
Cloud Server	14-core 3.3GHz Intel i9 64-bit CPU, 128G RAM, 2T SSD (OS: 64-bit Ubuntu 20.04.5)

5.2 Stealthiness of Trojans

We will provide both the qualitative analyses and quantitative evaluations on the Trojan's stealthiness including anti-recognition and anti-duplication of the triggers, which are the critical indicators for the quality of Trojan design. Specifically, for the qualitative part, analyses are summarized through Table 3, where "*A*" and "*B*" denote the command triggers with valid and invalid parameters, respectively (the same below). From Table 3, there is indeed a certain probability of being recognized or mistakenly activated for the Trojans using the first and second categories of triggers. But, as the experiences of human beings and the configurations of pre-checking are fairly subjective, Trojans would still be well hidden if the real-time monitoring is relatively loose. Accordingly, the attack success rate will be considerably high. Notice that, usually there is a pre-processing stage for other formats of the input data, like image, audio, and video, which will be compacted, resized, or cropped therein. Therefore, for the first and second (*B*) categories of triggers, they may be distorted and the Trojans would not be activated successfully, leading to lower attack feasibility.

Table 3 Stealthiness of Trojan Designs.

Trigger		Anti-recognition	Anti-duplication
The 1st category		Only exposed to *experienced human beings*	*Hard* for users to reproduce the triggers
The 2nd category	*A*	Only exposed to *certain pre-set configurations*	*Low possibility* for users to reproduce the triggers
	B		*Hard* for users to reproduce the triggers
The 3rd category		*Strong stealthiness for existing monitoring*	*Low possibility* for users to reproduce the triggers

Table 4 Duplication Rates of Triggers (I).

Dataset	Duplication Rate of Command	
	One-parameter	***Two-parameter***
email-Eu-core	$2.49\times10^{-2}\%$	$0.56\times10^{-3}\%$
wiki-Vote	$0.35\times10^{-2}\%$	$0.14\times10^{-3}\%$
p2p-Gnutella04	$0.23\times10^{-2}\%$	$0.36\times10^{-3}\%$
ca-HepTh	$0.25\times10^{-2}\%$	$0.55\times10^{-3}\%$
ego-Facebook	$0.62\times10^{-2}\%$	$0.16\times10^{-3}\%$
gemsec-Facebook	$0.22\times10^{-2}\%$	$0.21\times10^{-3}\%$

Table 5 Duplication Rates of Triggers (II).

Type of Set	Duplication Rate of Command			
	S_A	S_B	S_C	S_D
Combination-set	$0.25\times10^{-1}\%$	$0.21\times10^{-1}\%$	$0.29\times10^{-1}\%$	$0.3\times10^{-1}\%$
Permutation-set	$0.01\times10^{-1}\%$	$0.05\times10^{-1}\%$	$0.03\times10^{-1}\%$	$0.04\times10^{-1}\%$

Next, regarding the quantitative part, we solely provide evaluation results on the indicator "anti-duplication" here, as the corresponding factors are measurable only for the second category of triggers with valid parameters and the third category therein. In particular, 1) For the second category of triggers, Table 4 shows the duplication rates of triggers in regular usage through the employed datasets. Normally, the command parameters configured by users in LI-GPL contain the names of vertices/edges/files, graph formats, weights of vertices/edges, and so on. For a single command, there are up to three parameters, e.g., source and destination nodes, and edge weight for "Add edge" in the directed graphs. Here, it is assumed that only the names of vertices/edges in a dataset are needed to be considered for convenience. From Table 4, we can find that the corresponding rates are extremely low, especially for the larger-sized graphs, even if the commands have not been specified yet. 2) For the third category of triggers, 100,000 to-be-entered user command sequences per size, with repeats allowed, are generated randomly. We carry out related evaluations based on the combination and permutation of random consecutive commands in the pre-set sets of the *same* and *different* sizes, respectively. Firstly, for the former, let the size be "3" for the four randomly generated sets, i.e., S_A, S_B, S_C, and S_D. The obtained rates are demonstrated in Table 5, showing that values are truly low as well and different sets of the same sizes have similar duplication rates. While results of the latter are revealed in Fig. 4, where the size of the set ranges from 2 to 6. It is indicated that the relevant rates become lower and lower when the size of command set grows. All the rates remain "0%" when the set size is larger than "4". Besides, the triggers via permutation-set are more imperceptible as its constraint is much stronger than those using combination-set, especially when the scale of the command set gets larger.

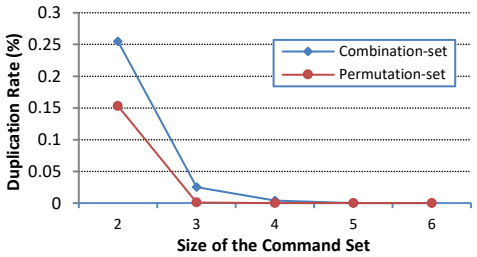


Fig. 4 Duplication rates of triggers via command sets of different sizes.

5.3 Latency Incurred by Attacks

In the realm of secure computing, the extra burden, which is a primary concern for attackers, typically manifests as the additional incurred latency stemming from the activation of Trojans (containing all the procedures executed in MF and

MAG) within the attack model. Poor stealthiness might be caused as a result of the relatively long response times of the systems. Although the encryption functionality is enabled, we assume that the keys have been cracked, then the related cost is not considered here. Table 6 offers the specific latencies due to the implementations of MF and MAG on different devices locally. For the third category of triggers, the size of command set is configured as "3". It is found that the latencies caused by the activation via all three categories of triggers and the injection or replacement of new commands can be less than 50μs for all the devices, which is also extremely low indeed. In comparison with other attack models for big data processing presented in [40], in which the minimum proportionate increase in response time is 2.78%, our model, as evidenced by Table 6 and the specific execution times of different graph primitives through LI-GPL [21, 22], demonstrates a maximum increase of only 2.37%. The stealing and tampering attacks will be performed subsequently after the activation of Trojan, whose execution time is highly correlated with the scales of the datasets adopted.

Table 6 Latency Incurred in MF & MAG.

Trigger		Latency Involved with Trojans		
		Rasp-Pi 3B	***Rasp-Pi 4B***	***Cloud Server***
The 1st category		< 1μs	< 1μs	< 1μs
The 2nd category	*A*	4μs	3μs	1μs
	B	4μs	4μs	2μs
The 3rd category		41μs	27μs	13μs

5.4 Discussion on Further Work

Further experimental evaluations to make comparisons on stealthiness with the state-of-the-art advancements in security and privacy vulnerabilities would be necessarily conducted when newer Trojan-involved attack models for graph processing become available in the future, as this is the first time (to our knowledge) to build such a novel attack model demonstrated through the ubiquitous graph analytics. On the other hand, we aim to persist in carrying out our research from different perspectives or employing emerging techniques. Specifically, in the past decade, hardware security has gained increasing attention from both academia and industry. The afore-mentioned hardware Trojans have become a growing concern, which can be achieved by maliciously modifying integrated circuits, e.g., MF can be implemented through a small cache with relevant judgment logics. Generally, the cost of hardware Trojans is higher, but they are more difficult to detect and remove than the software Trojans. Hence, designing corresponding Trojans from a hardware perspective for the proposed attack model in this paper opens a new direction of the future work for a comprehensive design. Additionally, we also intend to continue advancing the mitigation schemes against Trojan attacks in the field of big data processing. With the wide application of artificial intelligence (AI) in information security, intelligent Trojan detection, and self-correction of processing results on the basis of machine learning (ML) become our future directions as well [44]. Although there are many related research studies now, it is crucial for us to find better ways to adapt to our proposed attack model and application scenarios. Furthermore, the corresponding coping measures for software and hardware Trojans will differ significantly as well.

6. Conclusion

As security and privacy concerns continue to grow among users in the realm of big data processing, various mitigations have been presented as countermeasures to address these issues. However, there is scarce research on the attack design of adversaries. Thus, for the first time to our knowledge, we especially propose a novel attack model demonstrated via the graph analytics from a software perspective in this paper. The model includes a detailed discussion of the embedding mode, malicious activity and activation mechanism of a Trojan. In particular, Trojan activation mechanisms are more focused on therein, making the proposed attacks imperceptible to users and monitoring systems. Our attack model is scalable and can be widely applied in big data processing based on the edge-cloud architecture in smart society. Evaluations show that Trojans in the attack model have excellent stealthiness and can achieve goals with low extra latency.

Acknowledgments

This work is supported, in part, by JST CREST Grant Number JPMJCR18K1, Japan.

References

[1] F. Mehdipour, H. Noori, et al., "Energy-Efficient Big Data Analytics in Datacenters," *Advances in Computers*, vol. 100, 2016, pp. 59-101.
[2] Y. Xia, I. G. Tanase, et al., "Explore Efficient Data Organization for Large Scale Graph Analytics and Storage," *Proceedings of Big Data*, 2014, pp. 942-951.
[3] M. E. Coimbra, A. P. Francisco, and L. Veiga, "An Analysis of the Graph Processing Landscape," *Journal of Big Data*, 8(55), 2021, pp. 1-41.
[4] I. Robinson, et al., "Graph Databases," O'Reilly Media, Inc., 2015.
[5] S. D. Pollard and B. Norris, "A Comparison of Parallel Graph Processing Implementations," *Proceedings of CLUSTER*, 2017, pp. 657-658.
[6] J. Zhou, et al., "Interactive and Reliable Graph Processing via the Edge-Cloud Collaboration Framework," *Proceedings of HPCC*, 2022, pp. 388-395.
[7] J. Zhou, et al., "An Edge-Cloud Collaboration Framework for Graph Processing in Smart Society," *IEEE TETC*, 11(4), 2023, pp. 985-1001.
[8] Y. Chen, B. Liu, W. Lin, and H. Cheng, "Survey of Cloud-Edge Collaboration," *Computer Science*, 48(3), 2021, pp. 259-268.
[9] F. Liu, G. Tang, Y. Li, Z. Cai, X. Zhang, et al., "A Survey on Edge Computing Systems and Tools," *Proceedings of the IEEE*, 107(8), 2019, pp. 1537-1562.
[10] E. Krishnasamy, et al., "Edge Computing: An Overview of Framework and Applications," *PRACE Technical Report*, 2020, pp. 1-20.
[11] M. Dener, G. Ok, et al., "Malware Detection Using Memory Analysis Data in Big Data Environment," *Appl. Sci.*, 12(17), 2022, pp. 1-23.
[12] E. Damiani, C. A. Ardagna, F. Zavatarelli, et al., "Big Data Threat Landscape and Good Practice Guide," *ENISA Report*, 2016, pp. 1-62.
[13] R. Buyya, R. N. Calheiros, and A. V. Dastjerdi, "Big Data: Principles and Paradigms," Morgan Kaufmann, 2016.
[14] J. Guia, V. G. Soares, and J. Bernardino, "Graph Databases: Neo4j Analysis," *Proceedings of ICEIS*, 2017, pp. 351-356.
[15] N. Doekemeijer, "A Survey of Parallel Graph Processing Frameworks," *Report Series of TU-Delft*, No. PDS-2014-003.
[16] J. Siek, L. Lee, and A. Lumsdaine, "The Boost Graph Library: User Guide and Reference Manual," Pearson Education, 2001.

[17] D. Michail, J. Kinable, et al., "JGraphT—A Java Library for Graph Data Structures and Algorithms," *ACM TOMS*, 46(2), 2020, pp. 1-29.
[18] G. Csardi and T. Nepusz, "The igraph Software Package for Complex Network Research," *Int. J. Complex Syst.*, 1695(5), 2006, pp. 1-9.
[19] C. L. Staudt, et al., "NetworKit: A Tool Suite for Large-scale Complex Network Analysis," *Network Science*, 4(4), 2016, pp. 508-530.
[20] A. A. Hagberg, et al., "Exploring Network Structure, Dynamics, and Function Using NetworkX," *Proceedings of SciPy*, 2008, pp. 11-16.
[21] J. Zhou, et al., "A Lightweight Interactive Graph Processing Library for Edge Computing in Smart Society," *Proceedings of CANDARW*, 2021, pp. 62-68.
[22] J. Zhou and M. Kondo, "An Interactive and Reductive Graph Processing Library for Edge Computing in Smart Society," *IEICE Trans. Inf. & Syst.*, E106-D(3), 2023, pp. 319-327.
[23] J. Moura and C. Serrão, "Security and Privacy Issues of Big Data," https://arxiv.org/abs/1601.06206, 2016, pp. 1-29.
[24] Y. Zhang, C. Papamanthou, and J. Katz, "ALITHEIA: Towards Practical Verifiable Graph Processing," *Proceedings of CCS*, 2014, pp. 856–867.
[25] Y. Zhu, H. Li, et al., "Verifiable Subgraph Matching with Cryptographic Accumulators in Cloud Computing," *IEEE Access*, vol. 7, 2019, pp. 169636-169645.
[26] K. Nayak, X. S. Wang, S. Ioannidis, et al., "GraphSC: Parallel Secure Computation Made Easy," *Processings of S&P*, 2015, pp. 377-394.
[27] T. Araki, J. Furukawa, K. Ohara, B. Pinkas, et al., "Secure Graph Analysis at Scale," *Proceedings of CCS*, 2021, pp. 610-629.
[28] S. Lai, et al., "GraphSE2: An Encrypted Graph Database for Privacy-Preserving Social Search," *Proceedings of AsiaCCS*, 2019, pp. 41-54.
[29] N. Aljuaid, A. Lisitsa, et al., "SMPG: Secure Multi-Party Computation on Graph Databases," *Proceedings of ICISSP*, 2022, pp. 463-471.
[30] L. Xu, J. Jiang, et al., "Privacy Preserving Strong Simulation Queries on Large Graphs," *Proceedings of ICDE*, 2021, pp. 1500-1511.
[31] K. Huang, H. Hu, et al., "Privacy and Efficiency Guaranteed Social Subgraph Matching," *The VLDB Journal*, 31(3), 2021, pp. 581-602.
[32] L. Sardar, G. Bansal, S. Ruj, and K. Sakurai, "Securely Computing Clustering Coefficient for Outsourced Dynamic Encrypted Graph Data," *Proceedings of COMSNETS*, 2021, pp. 465-473.
[33] D. C. Kozen, "The Design and Analysis of Algorithms," Springer, 1992.
[34] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, et al., "Graphviz — Open Source Graph Drawing Tools," *Proceedings of GD*, 2001, pp. 483-484.
[35] E. R. Gansner, E. Koutsofios, S. C. North, et al., "Graph Visualization in Software Analysis," *Proceedings of AQSDT*, 1992, pp. 226-237.
[36] W. R. Stevens, et al., "Unix Network Programming," Addison Wesley, 2003.
[37] OpenSSL software, https://www.openssl.org/.
[38] L. Chi and X. Zhu, "Hashing Techniques: A Survey and Taxonomy," *ACM Computing Surveys*, 50(1), No. 11, 2017, pp. 1-36.
[39] A. Shostack, "Threat Modeling: Designing for Security," Wiley, 2014.
[40] N. Li, H. Gao, et al., "Attack Models for Big Data Platform Hadoop," *Proceedings of BigDataSecurity & HPSC & IDS*, 2019, pp. 154-159.
[41] J. Leskovec and R. Sosič, "SNAP: A General-Purpose Network Analysis and Graph-Mining Library," *ACM TIST*, 8(1), 2016, pp. 1-20.
[42] G. Karypis and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," *SISC*, 20(1), 1999, pp. 359-392.
[43] Raspberry Pi, https://www.raspberrypi.org/.
[44] J. Zhou, et al., "Attack Mitigation of Hardware Trojans for Thermal Sensing via Microring Resonator in Optical NoCs," *ACM JETC*, 17(3), 2021, pp. 1-23.

Jun Zhou is an appointed assistant professor at Keio University, Japan. He received the Ph.D. degree from Institute of Computing Technology, Chinese Academy of Sciences in 2016. His main research interests include data science, CPS, VLSI design and verification, etc. He is a member of the ACM and CCF.

Masaaki Kondo is currently a professor in the Faculty of Science and Technology at Keio University and the leader of the next generation high performance architecture research team at RIKEN. He received the Ph.D. degree from The University of Tokyo in 2003. His research interests include computer architectures, high-performance computing, and cognitive computing. He is a member of the ACM, IEEE, IEICE and IPSJ.