A KUBERNETES DATASET FOR MISUSE DETECTION

Yigit Sever[1] and Adnan Harun Dogan[1]
[1]Middle East Technical University

NOTE: Corresponding author: Yigit Sever, yigit@ceng.metu.edu.tr

Abstract – *Container security involves a broad spectrum of concerns, including the security of the operating system, auditing the supply chain and the application security of the running containers. This wide attack surface will also include the security of the container orchestration system and its components once a container orchestration system is introduced to manage the fleet of containers in an environment. In order to advance the research in this field, prior work should be comparable and reproducible. However, we identified a research gap for this aspect; publicly available datasets for container security is sparse and reproducibility of the research output so far is arduous. In this study, we share a dataset consisting of network flows, collected from a Kubernetes cluster. Furthermore, we performed a preliminary analysis on the data as a sanity check to evaluate its quality. By sharing this dataset publicly, we hope to help further studies and establish benchmarks in the field of container networking security.*

Keywords – Container security, cybersecurity, intrusion detection

1. INTRODUCTION

Container technology is on the rise, mainly due to the demand of portable software units that natively solve compatibility issues. Containers include the executable of any application they serve, alongside any dependencies that the application may require. Furthermore, containers isolate the computation environment, as well as put constraints in place, making them well suited for multi-tenant environments. Hence, the industry has been experiencing a shift towards container-based software deployment from bare-metal installations [1]. Containers also lends themselves well to the microservice-based development, which is an architecture we have been observing more in recent software deployment trends from large companies [2].

Deployments on the cloud have been using microservice design patterns and architecture to answer the drawbacks of traditional monolithic software architecture [3]. These microservices are decoupled and functionally distinct functions are defined in software. Since a typical monolithic application will need to be broken down into multiple microservices for functional parity, microservices are usually deployed using containers to eliminate the overhead of a fully present operating system [4]. Otherwise, each and every container would have to carry an operating system layer below the application logic. However, breaking down the functionality of an application into smaller functions also increases the complexity of the overall system even when the individual pieces have become more manageable and understandable. One particular interest of ours is the security of the network of containerized microservices. Security of container orchestration (e.g. Kubernetes) is especially in need of investigation [5] to understand the intricacies introduced by the additional layers in the whole setup.

Containers are used as building blocks of software development, for both in testing and deployment [6]. In order to ease and automate the building, management, inter-container networking and cleanup of containers, container orchestration technologies have been developed [7]. Effective attack detection and prevention in the cloud faces many challenges, even in the presence of fast-streaming data analytics. Therefore, machine learning algorithms and tools have also been developed specifically for cloud environments, mostly for virtual machine-based deployments, and have achieved successful results in detecting anomalies for certain scenarios including classical enterprise networks [8], Industrial IoT (IIoT) systems [9], and sensor data publish-subscribe systems [10].

Despite the existence of many approaches in the field of intrusion detection including both Host-Based Intrusion Detection System (HIDS) and Network-Based Intrusion Detection System (NIDS) solutions [11, 12, 13, 14, 15], IDSs for containerized environments are still at a premature state, with limited attention having been paid to them so far.

In this paper, we document the steps we have used to create a dataset of Kubernetes network traffic with malicious and benign data points which we have published publicly at https://github.com/yigitsever/kubernetes-dataset. The dataset is generated on a Kubernetes cluster that is running real-world container images. The attack scenarios we used come from the Common Vulnerabilities and Exposures (CVE) listings with distinct Common Weakness Enumeration (CWE) classes, making the dataset diverse.

2. RELATED WORK

Tien et al. [16] introduced an HIDS on a Kubernetes cluster. Using supervised learning methods, they developed an anomaly classification model, a neural network with four fully-connected layers. To create a dataset for training their models, they used system calls and root directory access features, which takes place when the container accesses a file under a root directory like `bin`, `var` etc. During our preliminary analysis, this work was the only one we came across that evaluated an IDS on a container orchestration platform. However, the authors have not disclosed the details of the Kubernetes cluster they performed their evaluation on, making their results impossible to reproduce. Flora et al. [17] proposed an HIDS using a containerized MariaDB database engine. They evaluated their Bag-of-System-Calls (BoSC) features using Sequence Time Delaying Embedding (STIDE) and Hidden Markov Models (HMMs). To compare the performance of an IDS on a containerized environment, they also performed evaluations on a pseudo-bare metal installation, using a KVM virtual machine. Tunde-Onadele et al. [18] combined signature-based and anomaly-based approaches in their HIDS. They evaluated common intrusion detection methods for containerized environments. Srinivasan et al. [19] offered a real-time HIDS using system calls. They fed n-grams of system calls to Maximum Likelihood Estimator (MLE) and Simple Good Turing (SMG) to do real-time classification. Cavalcanti et al. [20] evaluated the effectiveness of eight machine learning algorithms for IDS on containerized environments. They made use of the BoSC approach to create their dataset and evaluated these algorithms on this dataset. Chen et al. [4] proposed a framework named Informer. This framework is used to detect anomalous Remote Procedure Calls (RPCs), which are used for communication between agents in a microservices architecture.

3. DATASET

To create our dataset, we crafted 10 different attack scenarios. The traffic related to the attack scenarios are labelled with non-zero labels. Benign traffic, on the other hand, is labelled with 0. In order to generate benign traffic, we used the OWASP ZAP's Ajax Spider to use and navigate the web applications.

3.1 Environment setup

The Kubernetes cluster used for the data collection process consists of two Ubuntu 22.04 machines. Studying vulnerabilities requires using old software versions since vulnerabilities are patched as they are discovered. In order to have a vulnerable system, we used an old Kubernetes version: 1.20.0. To achieve compatibility and to use vulnerable versions of lower layer components, we used containerd version 1.6.0 and docker version 5:20.10.6. We used kubeovn as the CNI plugin.

Using kubeovn allowed us to capture the whole Kubernetes cluster's traffic from a single point. During the data collection process, we used the tcpdump program to capture every packet passing through the ovn0 interface. This gave us both the internal Kubernetes traffic and the external requests meant for the services running inside Kubernetes.

We developed a microservice-based software stack to be run in the Kubernetes cluster. The software stack includes Grafana, InfluxDB and Node-RED as the data collection, aggregation and visualization solutions. Regarding the stack, four Raspberry Pi images emulate IoT devices as smart home sensors and supply data for the rest of the pipeline.

3.2 Attack scenarios

CVE-2019-20933

CVE-2019-20933 is found in InfluxDB versions prior to 1.7.6 [21]. The vulnerability has been assigned to CWE-287: Improper Authentication. It is a software bug that lets InfluxDB accept queries that should have been authenticated with a proper JWT. Instead, JWTs with empty shared secrets have been able to bypass the authentication.

During our data collection scenario, we exploited this vulnerability by sending queries with crafted JWTs that had empty secret keys.

In our dataset, network flows that belong to this attack have been labelled as 6.

CVE-2019-5736

CVE-2019-5736 is a vulnerability discovered in 2019 [22]. It is a vulnerability concerning the runc program. This vulnerability has been filed under CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection').

CVE-2019-5736 causes attackers to overwrite the runc binary present in the host machine. This can be accomplished by either running a malicious container image crafted by the attacker or an existing container which the attacker had write access to. We used the first case during our attacker scenario, following the steps outlined in `https://github.com/Frichetten/CVE-2019-5736-PoC`.

This attack is labelled with 10 in our dataset.

CVE-2020-13379

CVE-2020-13379 is a vulnerability concerning Grafana versions between 3.0.1 to 7.0.1 [23]. The vulnerability has been listed under CWE-918: Server-Side Request Forgery (SSRF).

In order to exploit this vulnerability on the Grafana service running in our Kubernetes cluster, we sent a malicious request, crafted in accordance with the CVE-2020-13379 requirements through the URL of the Grafana service. The Grafana service crashes after sending this request.

This attack has the label 1 in our dataset.

CVE-2021-25741

CVE-2021-25741 is a vulnerability found in kubelet, a component of Kubernetes [24]. It was discovered in 2021, and it is listed under CWE-20: Improper Input Validation and CWE-552: Files or Directories Accessible to External Parties.

In order to exploit this vulnerability, we followed the proof-of-concept outlined in `https://github.com/Betep0k/CVE-2021-25741` to create two malicious containers inside a single pod. These two containers initiate a race condition to mount a file path from the host file system inside the container, performing a container escape. Network flows collected during CVE-2021-25741 scenario are labelled with 8 in our dataset.

CVE-2021-30465

CVE-2021-30465 is a runc vulnerability that affects versions prior to runc 1.0.0-rc95 [25]. It has been assigned under CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization. Vulnerabilities under this CWE class are also called race condition vulnerabilities.

In order to exploit this vulnerability in our Kubernetes cluster, we used the steps outlined in `https://github.com/champtar/blog/blob/main/runc-symlink-CVE-2021-30465/README.md`.

We labelled data points for this attack with 7.

CVE-2021-43798

The CVE-2021-43798 vulnerability was discovered in Grafana [26]. It is a straightforward path traversal vulnerability, classified under CWE-22: Improper Limitation of a Pathname to a Restricted Directory. In order to exploit this vulnerability in the Grafana service running in our experiment environment, we sent a request to the Grafana service with added `..` directives to navigate out of the working directory of the Grafana service.

This attack is labelled with 5 in our dataset.

CVE-2022-23648

CVE-2022-23648 is a vulnerability found in containerd. It is listed under two CWE classes: CWE-200: Exposure of Sensitive Information to an Unauthorized Actor and CWE-287: Improper Authentication.

To exploit this vulnerability in our Kubernetes cluster, we crafted a malicious Kubernetes pod using a .yaml file and added a VOLUME directive to mount and expose a sensitive host directory inside the container.

Network flows collected during the CVE-2022-23648 scenario has been labelled with 9.

Attacker scenarios

In order to study the vulnerabilities and the consequences of their exploitation, we used the Node-RED program present in our cluster's software stack to create a scenario consisting of three steps. During these scenarios, we wanted to create the traces of how a real-world attacker would infiltrate a Kubernetes cluster and show them in our dataset, as well as give a proof-of-concept of how the rest of the attacks we have mentioned so far could have happened to our cluster.

During the first scenario, we wanted to formulate how an attacker would first approach a web service. We used the OWASP ZAP tool to scan the Node-RED service first passively and then actively. During this step, we collected information regarding the Node-RED service. This is the *reconnaissance* step of the Node-RED scenarios, and it is labelled as 2 in our dataset.

Table 1 – Vulnerabilities and CWE listings

Vulnerability	CWE	Software
CVE-2019-20933	CWE-287	InfluxDB
CVE-2019-5736	CWE-78	runc
CVE-2020-13379	CWE-918	Grafana
CVE-2021-25741	CWE-20, CWE-552	Kubernetes (kubelet)
CVE-2021-30465	CWE-362	runc
CVE-2021-43798	CWE-22	Grafana
CVE-2022-23648	CWE-200, CWE-287	containerd

For the second scenario, we wanted to show how an attacker would exploit a vulnerable web service to infiltrate a Kubernetes cluster. First, we injected a Remote Code Execution (RCE) vulnerability to Node-RED. Vulnerabilities of the RCE class allows attackers to run arbitrary code on a web service's host machine. We used this RCE vulnerability to get a reverse shell on the host machine running the Node-RED service. Note that at this stage, the attacker would not know that Node-RED is constrained to a container. However, every container running under Kubernetes includes the directory `/run/secrets/kubernetes.io/serviceaccount` that includes kubectl credentials. We downloaded a kubectl binary from the Internet onto the container from our reverse shell and supplied the credentials to kubectl manually. For the end of this scenario, we used the attacker's kubectl inside the container to query the Kubernetes cluster regarding pods, services and secrets present in the cluster. We labelled this scenario as 3 in our dataset.

For the final Node-RED scenario, we continued from where we left off and attempted to perform a container escape. In order to accomplish this, the Node-RED container should have the required privileges defined for its service account to create new containers. This privilege has legitimate use cases but for the operations of our cluster, Node-RED containers do not need this privilege. Hence, we added this privilege to the service account of Node-RED prior to attack scenarios in order to exploit it during this scenario. We used the malicious kubectl in the Node-RED container, accessed through the reverse shell to create a container that mounts the root file system of the host machine inside the `/root` directory of the container. After this container is operational, we used kubectl to get a shell from the malicious container and performed a `chroot` operation. Thus, we have mounted the host machine's file system in the container, attaining full privileges in the host machine. We labelled the traffic for this scenario as 4.

This concludes the 10 attack scenarios we crafted for the data collection process.

Table 2 – Results of the preliminary analysis

Benign Flows	Accuracy %	Precision %	Recall %	F1	False Positive	True Positive
302	56.54	100.00	56.54	72.23	1613.6	2099.4
544	62.52	100.00	62.52	76.94	1391.5	2321.5
981	69.33	100.00	69.33	81.88	1138.8	2574.2
1767	75.26	100.00	75.26	85.88	918.6	2794.4
3184	82.88	100.00	82.88	90.63	635.8	3077.2
5737	88.70	100.00	88.70	94.01	419.5	3293.5
10336	91.12	100.00	91.12	95.36	329.6	3383.4
18622	92.87	100.00	92.87	96.30	264.9	3448.1
33550	95.95	100.00	95.95	97.93	150.4	3562.6
60447	96.57	100.00	96.57	98.25	127.5	3585.5

4. PRELIMINARY ANALYSIS

Employing the Scikit-learn library in Python, the one-class Support Vector Machine (SVM) [27] model was used to distinguish between normal and attack traffic on the Kubernetes network (i.e. benign and malicious data) and malicious data is intended to be detected as anomalous. During preprocessing, all irrelevant labels like packet size information and timestamps, as well as source and destination ports and IP addresses were removed. Also, all labels with zero variance (whose values are the same in every sample) were removed.

The training data includes malicious and benign traffic. However, their proportions are investigated in this part. The main focus was to determine the minimum necessary amount of benign traffic in the training dataset to accurately detect anomalies, as well as to reduce variance and bias in the test dataset. For that purpose, malicious data is evenly and randomly split into 10 folds and 10 independent experiments were conducted as a ten-fold cross-validation procedure. In every experiment, nine folds consisted of the test dataset for the model trained on the remaining fold. The `stratifiedShuffleSplit` method in the Scikit-learn library is used to generate folds. The averages from those experiment results are shown in Table 2.

After that step, the effect of the amount of benign data in the training set on the accuracy is analyzed. Since, one fold (i.e. randomly sampled one-tenth) of the malicious dataset is approximately 5% of all benign samples, starting from one-twentieth to the whole of benign samples, 10 different exponentially-increasing proportions are used as in Table 2 to detect the amount of benign data needed to reach an accurate anomaly detection. The benign traffic that wasn't included in the training data were dropped off from the pipeline.

The precision scores are always 100% because the testing datasets consisted only of malicious traffic. Lastly, the amount of malicious data in testing datasets was always 3713, which corresponds to 9 folds of whole malicious data.

The results in Table 2 are averaged by benign amounts in the training dataset from a hundred successful experiments. The scores are calculated using the Scikit-learn library's metrics module in Python.

It can be seen that more than thirty-three hundred benign and three hundred malicious traffic is enough to reach a 95% accuracy over 3713 malicious (abnormal) traffic.

5. CONCLUSION

In this study, we have developed and deployed a microservice-based application onto a Kubernetes cluster. We then prepared 10 attack scenarios with varying weakness enumerations. We played these 10 different attacks on our Kubernetes cluster and collected the attack traffic, as well as the response from the Kubernetes cluster. By preparing this dataset, we hope to encourage reproducible results in the field of container security.

ACKNOWLEDGEMENT

This research has been supported by the TÜBİTAK 3501 Career Development Program under grant number 120E537. However, the entire responsibility of the publication belongs to the owners of the research. The financial support received from TÜBİTAK does not mean that the content of the publication is approved in a scientific sense by TÜBİTAK.

REFERENCES

[1] Sébastien Vaucher, Rafael Pires, Pascal Felber, Marcelo Pasin, Valerio Schiavoni, and Christof Fetzer. "SGX-Aware Container Orchestration for Heterogeneous Clusters". In: *2018 IEEE 38th International Conference on Distributed Computing Systems*

(ICDCS). July 2018, pp. 730–741. DOI: 10.1109/ICDCS.2018.00076.

[2] Jacopo Soldani, Damian Andrew Tamburri, and Willem-Jan Van Den Heuvel. “The Pains and Gains of Microservices: A Systematic Grey Literature Review”. In: *Journal of Systems and Software* 146 (Dec. 2018), pp. 215–232. ISSN: 0164-1212. DOI: 10.1016/j.jss.2018.09.082. (Visited on 01/14/2023).

[3] David S. Linthicum. “Practical Use of Microservices in Moving Workloads to the Cloud”. In: *IEEE Cloud Computing* 3.5 (Sept. 2016), pp. 6–9. ISSN: 2325-6095. DOI: 10.1109/MCC.2016.114.

[4] Jiyu Chen, Heqing Huang, and Hao Chen. “Informer: Irregular Traffic Detection for Containerized Microservices RPC in the Real World”. In: *High-Confidence Computing* 2.2 (June 2022), p. 100050. ISSN: 2667-2952. DOI: 10.1016/j.hcc.2022.100050. (Visited on 06/05/2022).

[5] Francesco Minna, Agathe Blaise, Filippo Rebecchi, Balakrishnan Chandrasekaran, and Fabio Massacci. “Understanding the Security Implications of Kubernetes Networking”. In: *IEEE Security & Privacy* 19.5 (Sept. 2021), pp. 46–56. ISSN: 1540-7993, 1558-4046. DOI: 10.1109/MSEC.2021.3094726. (Visited on 10/28/2021).

[6] Claus Pahl, Antonio Brogi, Jacopo Soldani, and Pooyan Jamshidi. “Cloud Container Technologies: A State-of-the-Art Review”. In: *IEEE Transactions on Cloud Computing* 7.3 (July 2019), pp. 677–692. ISSN: 2168-7161. DOI: 10.1109/TCC.2017.2702586.

[7] René Peinl, Florian Holzschuher, and Florian Pfitzer. “Docker Cluster Management for the Cloud - Survey Results and Own Solution”. In: *Journal of Grid Computing* 14.2 (June 2016), pp. 265–282. ISSN: 1572-9184. DOI: 10.1007/s10723-016-9366-y. (Visited on 01/14/2023).

[8] Shengjie Xu, Yi Qian, and Rose Qingyang Hu. “Data-Driven Network Intelligence for Anomaly Detection”. In: *IEEE Network* 33.3 (2019), pp. 88–95. DOI: 10.1109/MNET.2019.1800358.

[9] Muna AL-Hawawreh, Nour Moustafa, and Elena Sitnikova. “Identification of malicious activities in industrial internet of things based on deep learning models”. In: *Journal of Information Security and Applications* 41 (2018), pp. 1–11. ISSN: 2214-2126. DOI: https://doi.org/10.1016/j.jisa.2018.05.002. URL: https://www.sciencedirect.com/science/article/pii/S2214212617306002.

[10] Ege Ciklabakkal, Ataberk Donmez, Mert Erdemir, Emre Süren, Mert Kaan Yilmaz, and Pelin Angin. “ARTEMIS: An Intrusion Detection System for MQTT Attacks in Internet of Things”. In: *38th Symposium on Reliable Distributed Systems, SRDS 2019, Lyon, France, October 1-4, 2019*. IEEE, 2019, pp. 369–371. DOI: 10.1109/SRDS47363.2019.00053. URL: https://doi.org/10.1109/SRDS47363.2019.00053.

[11] Yue Guan and Naser Ezzati-Jivan. “Malware System Calls Detection Using Hybrid System”. In: Apr. 2021, pp. 1–8. DOI: 10.1109/SysCon48628.2021.9447094.

[12] Prachi Deshpande, Subhash Chander Sharma, Sateesh Kumar Peddoju, and S. Junaid. “HIDS: A host based intrusion detection system for cloud computing environment”. In: *International Journal of System Assurance Engineering and Management* 9 (2018), pp. 567–576.

[13] Gozde Karatas and Ozgur Koray Sahingoz. “Neural network based intrusion detection systems with different training functions”. In: *2018 6th International Symposium on Digital Forensic and Security (ISDFS)* (2018), pp. 1–6.

[14] Kamaldeep Singh, Sharath Chandra Guntuku, Abhishek Thakur, and Chittaranjan Hota. “Big Data Analytics Framework for Peer-to-Peer Botnet Detection Using Random Forests”. In: *Information Sciences* 278 (Sept. 2014), pp. 488–497. ISSN: 0020-0255. DOI: 10.1016/j.ins.2014.03.066. (Visited on 06/20/2022).

[15] F. J. Mora-Gimeno, H. Mora-Mora, B. Volckaert, and A. Atrey. “Intrusion Detection System Based on Integrated System Calls Graph and Neural Networks”. In: *IEEE Access* 9 (2021), pp. 9822–9833. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3049249.

[16] Chin-Wei Tien, Tse-Yung Huang, Chia-Wei Tien, Ting-Chun Huang, and Sy-Yen Kuo. “KubAnomaly: Anomaly Detection for the Docker Orchestration Platform with Neural Network Approaches”. In: *Engineering Reports* 1.5 (2019), e12080. ISSN: 2577-8196. DOI: 10.1002/eng2.12080. (Visited on 05/08/2022).

[17] José Flora, Paulo Gonçalves, and Nuno Antunes. “Using Attack Injection to Evaluate Intrusion Detection Effectiveness in Container-based Systems”. In: *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*. Dec. 2020, pp. 60–69. DOI: 10.1109/PRDC50213.2020.00017.

[18] Olufogorehan Tunde-Onadele, Jingzhu He, Ting Dai, and Xiaohui Gu. “A Study on Container Vulnerability Exploit Detection”. In: *2019 IEEE International Conference on Cloud Engineering (IC2E)*. June 2019, pp. 121–127. DOI: 10.1109/IC2E.2019.00026.

[19] Siddharth Srinivasan, Akshay Kumar, Manik Mahajan, Dinkar Sitaram, and Sanchika Gupta. “Probabilistic Real-Time Intrusion Detection System for Docker Containers”. In: *SSCC*. 2018.

[20] Marcos Cavalcanti, Pedro Inacio, and Mario Freire. "Performance Evaluation of Container-Level Anomaly-Based Intrusion Detection Systems for Multi-Tenant Applications Using Machine Learning Algorithms". In: *The 16th International Conference on Availability, Reliability and Security*. ARES 2021. New York, NY, USA: Association for Computing Machinery, Aug. 2021, pp. 1–9. ISBN: 978-1-4503-9051-4. DOI: 10.1145/3465481.3470066. (Visited on 05/08/2022).

[21] MITRE. *NVD - CVE-2019-20933*. https://nvd.nist.gov/vuln/detail/CVE-2019-20933. Nov. 2020. (Visited on 04/07/2023).

[22] MITRE. *NVD - CVE-2019-5736*. https://nvd.nist.gov/vuln/detail/CVE-2019-5736. Feb. 2019. (Visited on 04/07/2023).

[23] MITRE. *NVD - CVE-2020-13379*. https://nvd.nist.gov/vuln/detail/CVE-2020-13379. June 2020. (Visited on 04/06/2023).

[24] Kubernetes. *NVD - CVE-2021-25741*. https://nvd.nist.gov/vuln/detail/CVE-2021-25741. Sept. 2021. (Visited on 04/06/2023).

[25] MITRE. *NVD - CVE-2021-30465*. https://nvd.nist.gov/vuln/detail/CVE-2021-30465. May 2021. (Visited on 04/06/2023).

[26] GitHub, Inc. *NVD - CVE-2021-43798*. https://nvd.nist.gov/vuln/detail/CVE-2021-43798. Dec. 2021. (Visited on 04/07/2023).

[27] David M J Tax and Robert P W Duin. "Data Domain Description Using Support Vectors". In: (1999).

AUTHORS

Yigit Sever received a B.S. degree in computer engineering at TED University, Turkey, in 2016, and an M.S. degree in computer engineering at Hacettepe University, Turkey, in 2019. He is currently a Ph.D. candidate in computer engineering at Middle East Technical University (METU), Turkey, where he has also been working as a research assistant since 2020. His research interests include cloud security with a strong focus on container security, user and Internet privacy and distributed systems. He is a member of Wireless Systems, Networks and Cybersecurity Laboratory, METU.

Adnan Harun Dogan received a B.S. degree in computer engineering at Middle East Technical University (METU), Turkey, in 2022. He is currently an M.S. student at METU, where he has also been working as a research assistant since 2022. His research interests include discrete combinatorial optimization for adaptive learning methods and uncertainty estimation in deep learning, with a strong focus on discrete non-convex optimization. He is a member of Image Processing Laboratory, METU.