Measuring Artificial Intelligence and Machine Learning Implementation Security on the Internet

Preprint · December 2019

DOI: 10.13140/RG.2.2.15662.66888

3 authors:

Sergey Gordeychik
Inception Institute of Artificial Intelligence
12 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Denis Kolegov
Tomsk State University
21 PUBLICATIONS **11** CITATIONS
SEE PROFILE

Antony Nikolaev
3 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Measuring Artificial Intelligence and Machine Learning Implementation Security on the Internet

Sergey Gordeychik
Inception Institute of Artificial Intelligence
Abu-Dhabi, UAE
serg.gordey@gmail.com

Denis Kolegov
Tomsk State University
Tomsk, Russia
dnkolegov@gmail.com

Antony Nikolaev
Tomsk State University
Tomsk, Russia
antoniy.nikolaev@gmail.com

Artificial Intelligence and Machine Learning (correspondingly, AI, ML, AIML) are turning from rocket science into daily software developing and engineering life. On the other hand, the massive development of AI-enabled systems and their applications in various areas of life can bring about problems from a security perspective, making implementation security of the systems is one of the greatest concerns. In this paper, we present the results of Internet-wide security scans of publicly available AIML systems. We show that many different interfaces of AIML systems are not protected and accessible from the Internet, moreover, most of them don't even have basic security mechanisms. Also, we describe found the known vulnerabilities related to outdated software and insecure configurations. We employ a customized methodology suitable for Internet-scale scanning using search engines for Internet-connected devices and develop a special automation framework. We also provide additional examples of insecure AIML systems found during security validation testing. As the narration proceeded, the descriptions of basic threat intelligence and security scanning methods are provided when necessary mainly for data scientists and software engineers.

Keywords

artificial intelligence, machine learning, cybersecurity, implementation security, threat intelligence, scanning, fingerprinting, cloud networks, vulnerability discovery

Contents

1 Introduction

The use of AIML technologies is actively moving from the field of theories and scientific papers into everyday life. They can be easily found in different areas: from simple user-friendly web applications to large enterprise solutions. Big companies provide a variety of different cloud services related to machine learning, and concepts such as “model as a service” (MaaS) are already in high demand. At the same time, security issues related to the implementation of such systems are often not considered.

At the moment, the entrance requirements in the ML field for potential developers have been significantly reduced: most of the large frameworks have fairly high-level APIs that do not require deep knowledge in the fields of applied mathematics and statistics. Also, to make the management of all those features easier, the frameworks often provide a large number of

interfaces and protocols that are available for use to developers both directly and with any set of programming languages and tools. Moreover, many solutions that were previously possible only within special development environment and required an individual system can now be deployed and executed in a browser as a JavaScript code: from the general help in choosing a suitable product in an online store to complex face and voice recognition systems.

All of these reasons lead to the fact that every day more and more various solutions using ML with different levels of complexity appear. Most of the solutions have a large attack surface. A large number of ML systems have Web User interfaces (UI) that allow real-time monitoring of training process and use of the application, as well as giving the opportunity to influence these processes in one way or another. Besides that, some of these systems have low-hanging fruit vulnerabilities, which can also be exploited by an attacker when the system is accessible from the Internet.

The goal of the research is to develop tools and perform active and passive measurements to estimate AIML systems implementation security level at the Internet-scale within a common threat intelligence sense. and to answer the following questions:

Our contribution. The main contributions of this paper are as follows:

1. We craft search engines queries and fingerprints to be able to identify, enumerate and recognize AIML systems on the Internet.
2. We evaluate the common security level for AIML frameworks and applications based on the performed measurements on known vulnerabilities.
3. We identify previously unknown product version disclosure sources.
4. We show that AIML management interfaces are accessible on the Internet and can be abused by attackers.
5. We explore AIML infrastructures on the Internet and search for the following implementation security issues: default credentials, missing authentication, API and critical endpoints exposures, accessibility of critical methods and functions without authentication, leakage of private information.

Our approach. The common approach to cybersystem enumeration is based on obtaining complete and exhaustive information about a specific solution from each related vendor that is in the scope. That information helps to find publicly available target frameworks, applications, and systems with well-known search engines such as Shodan[1] and Censys[2]. For this purpose, we have been developing a tailored framework called Grinder[3], which can help in the automation of the searching process and obtaining specific information about Internet-connected devices. Besides this framework, we also propose an approach to search for different devices with various search engines. In general, the approach can be defined as follows:

---

[1] Michael Schearer. SHODAN for Penetration Testers. URL https://www.defcon.org/images/defcon-18/dc-18-presentations/Schearer/DEFCON-18-Schearer-SHODAN.pdf.

[2] Censys Search Engine. URL: https://censys.io

[3] The Grinder Framework. URL: https://github.com/sdnewhop/grinder

1. Investigate major machine learning frameworks (e.g., TensorFlow, Caffe, PyTorch) and craft signatures for them.
2. Investigate major machine learning training systems (e.g., NVIDIA DIGITS, TensorBoard) and craft signatures for their components, services, and interfaces.
3. Define the signatures within a search engine query language.
4. Run mass enumerations and security scans using the Grinder framework.
5. Reduce false-positive findings, adjust filters, and correct the signatures.
6. Classify search queries according to a confidence level.
7. Detect product version leakage using manual analysis of implementation artifacts (HTTP response headers, HTML, JavaScript code, etc.).
8. Extract product versions by active fingerprinting with the Grinder framework.
9. Use an incremental save method to store the results in the knowledge base.

We explore the following types of AIML products and concomitant infrastructures deployed and accessible on the Internet:

1. ML Frameworks.
2. Federated Learning Frameworks.
3. Medical Imaging Systems.
4. Management Controllers.
5. Databases and data.
6. Job and Message Queues (MQ).
7. Interactive Voice Response (IVR).
8. Speech Recognition Libraries.
9. Face Detection and Recognition Libraries.

Since the speed of AIML development is quite high it is difficult to identify specific components that could help to classify a certain system as an AI-related. However, some popular frameworks and application components are now widely used to build 3rd party applications and can be taken as an indication of the integration with an AIML technology.

In some cases, especially for ML training components, the accessibility of management interface on the Internet indicates the presence of CWE-749 weakness "Exposed Dangerous Method or Function".

Online resources. The developed software, Shodan and Censys search queries are available on the AISec repository[4] on GitHub. The repository also contains the metadata, search results and the statistics of the Internet-wide scanning ran in October 2019.

[4] AISec repository on GitHub. URL: https://github.com/sdnewhop/AISec

2 Machine Learning Systems Enumeration

2.1. Information Gathering

The exploration and searching of ML applications, frameworks, and solutions along with their interfaces (like Web UI, REST API, SSH, SNMP, etc.) include a sequence of steps, and the first step is information gathering. We need to define the list of the most popular vendors that engaged in ML research. It can be vendors and products that work with frameworks, training systems, applications, etc. Any search engine can be used for this purpose by making different kinds of queries like “NVIDIA”, “TensorFlow”, “PyTorch” and many others, but in a more sophisticated, special for a current search engine way.

After this step, we will have the list of different ML vendors that are currently active. For each of the vendors with their products, we search available solutions including hardware appliances (BMC, GPUs, servers, etc.), software (libraries, applications, etc.) and any other helpful additional elements like a SSH banner, operating system version, web interface and so on. That information will be used further to discover software vulnerabilities.

It is a common case that all the necessary information regarding an ML platform can be found directly on the vendor’s website, including possible demonstration applications, where we can find versions of libraries, software, applications, etc. Moreover, having this knowledge, we can build up own active searching queries and approach to search for hosts with the same properties.

2.2. Passive Fingerprinting

2.2.1. Basic search filters. Suppose we have all the necessary information about different ML solutions of some particular vendor that we want to search for. We can use different search engines such as Shodan, Censys, ZoomEye[5] and FOFA Pro[6] to find their interfaces. For example, we can try to use the name of an ML solution in the HTTP title of some web resource, assuming that the web interface title contains some of the keywords. We could use the following query for the Shodan search engine, for example, to search for NVIDIA DIGITS[7] appliance: http.title:"DIGITS".

To build the equivalent query for the Censys search engine we need to specify a protocol (HTTP or HTTPS), a port number (80 and 8080 in case of HTTP or 443 and 8443 in case of HTTPS) and “title” search filter. For example, 80.http.get.title:"DIGITS".

[5] ZoomEye Search Engine. URL: https://www.zoomeye.org/
[6] FOFA Pro Search Engine. URL: https://www.fofa.so/
[7] NVIDIA DIGITS. URL: https://developer.nvidia.com/digits

ZoomEye filter mechanism is really close to Shodan. In case of ZoomEye we have advanced search possibilities to put values in special fields like “Title”, “Banner”, “Port” to make search easier using web interface: title:"DIGITS". The equivalent query for FOFA PRO search engine is the same as for ZoomEye but with the slightly changed syntax: title="DIGITS".

Of course, searching by title can cause a lot of false-positive results. For instance, the following titles will be in the list of results in case of the query:

1. Single Digits - Resident Information.
2. Significant Digits | Custom Websites in the Twin Cities.
3. Digits The FUTURE of Crypto Payments Has Launched.

All such web resources do not have relation to NVIDIA DIGITS appliances, but they are still there. Reducing false-positives results in case of limited searching possibilities will be considered later.

It is worth noting that ZoomEye and FOFA Pro engines have a feature to search through the history of host: if something is changed on the host during different scans, the engines save every change on some ports, services, banners and so on.

We can work with this information using a web interface as host snapshots. For example, if the code of an HTTP response is “403 Forbidden”, you can look at host history until you find “200 OK” and can parse all the necessary information.

In the case of Shodan, you have historical scan access through Shodan[8] CLI or API interface, including wrappers for different programming languages. The same for Censys search engine: you have the “Bulk Data”[9] API access feature to get all the related scan information through all scan history.

The main idea of the searching is to find unique things for unique products. In this case, we can use many different features of all the search engines, including:

1. Full banner search. We can search through a full banner of response, for example, we can search for headers, statuses, responses, default cookies, tags and so on.
2. X.509 certificate search. We can search through information encoded in certificates, such as subject, issuer, fingerprint, version and so on.
3. Port and protocol search. We can search through different protocols and ports. For example, we can search for all DICOM devices by searching on 104 port, or we can search for all of the hosts with SSH access by searching on default 22 port.
4. Hash search. We can search through different hashes for some unique fields. For example, we can search for favicon hash, headers hash, HTML page hash and so on.
5. Web components and libraries: we can search through all of the possible web components and libraries that our application uses, like Bootstrap, Google Fonts, JQuery and many more.

---

[8] Shodan - Looking up IP Information. URL: https://help.shodan.io/developer-fundamentals/looking-up-ip-info

[9] Censys - Bulk Data Access. URL: https://censys.io/data

2.2.2. Complicated filters. NVIDIA AIAA[10] (AI Assisted Annotation) AP can be found by the following Shodan queries:

1. http.title:"NVIDIA AIAA Server documentation"
2. http.favicon.hash:816615992 http.component:"google font api"

In the first query, we use the already known technique of searching through the HTTP title. But if Shodan gets results like “301 Redirect” or “302 Found” our hosts will be lost from all results. To bypass this we send an additional query with a favicon hash. We can take a favicon from the official NVIDIA website, and use it since all of the main products of NVIDIA use the same favicon. To make this query more accurate, we can add some known unique web components of web interface - for example, it can be Google Fonts, that used on these hosts.

We are able to search for Mining Stations using XMRPool[11]. We can assume that they have some kind of API that returns some information in JSON or some other text representation format. If we talk about JSON responses, all scanners from our list can correctly parse them as usual HTML code, so we can search through JSON responses too.

For instance, let’s assume that our mining stations have some special keys or strings like “miner”, “miners”, “hash”, “hashes” and so on. We can brute all of these queries with the following list:

1. http.html:"miner" "application/json"
2. http.html:"miners" "application/json"
3. http.html:"hash" "application/json"
4. http.html:"hashes" "application/json"

Finally, we can find an appropriate query to search for some unique types of miners with the following query for Shodan: http.html:"hashes_total" "application/json".

In case of using Shodan, if you want to search for some keywords anywhere (in any possible place for a host field like SSL/TLS certificates, SSH banners, HTTP banners, hashes, paths), you need to use special tag “all”. By default, all that you will write in search field of Shodan in web interface (or with API methods too), will be searched only in basic banners (HTTP, for example), and will not be searched in any other fields. So, for this reason, it is required to put an additional tag “all” to include matching results from any possible field.

To demonstrate this feature, let’s search for miners from the previous example only on port 5555 with the following queries:

1. http.html:"hashes_total" port:5555
2. all:"hashes_total" port:5555

The results will be the same because of the “http.html” tag that the part of the tag “all” by definition. For these queries at the moment of writing, we can get 8 results, where hosts are the

[10] NVIDIA - The AI Assisted Annotation SDK Getting Started Guide. URL: https://docs.nvidia.com/clara/aiaa/tlt-mi-ai-an-sdk-getting-started/

[11] Monero XMR Mining Pool. URL: https://web.xmrpool.eu/

same. After that, we can compare the results using the following query (in case if we think that our keyword will be searched everywhere or at least in the HTML of the crawled page):

"hashes_total" port:5555

For this query, at the moment of writing, we will get 0 results, because response headers and basic response banners do not contain this keyword, but the HTML code contains one.

2.2.3. Queries combinations and logic operators. Sometimes it is easier to search for some special kind of devices or interfaces with one complicated query than with many slightly different queries. We can search for ML application signs in Docker containers, cloud services or databases with different brute-force queries where only one word changes. For instance, we can search for ML application content in Mongo databases using the following:

1. all:"ml" product:MongoDB port:27017 -authentication
2. all:"dataset" product:MongoDB port:27017 -authentication
3. all:"datasets" product:MongoDB port:27017 -authentication
4. all:"models" product:MongoDB port:27017 -authentication

In all of these queries we got one static part: “product:MongoDB port:27017 -authentication” - this query is used to find MongoDB databases without authentication on default port 27017. It should be noted that Shodan also indexes DB content in case if they do not require authentication. We can get a wide variety of small-scale results for every query and combine them, or we can build a complicated full-coverage query with supported logical operators like this:

1. ("ml" OR "dataset" OR "datasets" OR "ml-logs" OR "algomodel" OR "models" OR "predictions" OR "prediction" OR "tensorflow" OR "tensor") "MongoDB Server Information" product:MongoDB port:27017 -authentication

Using this query, we can search for all of the databases that contain special keywords like “ml”, “dataset”, “predictions” and so on in their indexes and responses. We can use this method without the need for a subsequent combination of all small results.

2.2.4. Reducing false-positive results. In the process of searching it can be difficult to get only the right results even with the right tags and search keywords. For instance, we can get many results with not appropriate HTTP status (like “401 Unauthorized”, “404 Not Found”, “400 Bad Request”), or we can get many results where authentication is required and we can see it in responses. In these cases, we can remove some fields with special keywords from search to reduce false-positive results. This method can be used if we want to remove some inappropriate HTTP statuses (like “401 unauthorized”, “403 forbidden” and so on) from results. To do that we can use exclude filter (“-” sign) before unwanted keywords in Shodan:

http.title:"DIGITS" -"401" -"404" -"301" -"302"

Or, if we want to find different MongoDB instances deployed without any authentication, we can remove “authentication” word from the search query:

product:MongoDB port:27017 -"authentication"

2.3. Active Fingerprinting

The methods described above can be used to identify and enumerate most of the AIML systems, but sometimes active measurements are required. This is because the results of passive scanning may be irrelevant or inaccurate at the time of receipt of the information. Moreover, with the help of active scanning methods, we can get more specific information about a host. To achieve better results, we have been developing the Grinder framework.

Grinder was created during SD-WAN Internet Census[12] project to automatically enumerate, fingerprint and scan hosts on the Internet using different back-end systems: search engines (e.g., Shodan or Censys) for enumeration, network scanners (e.g., NMAP) for scanning, vulnerability databases (e.g., Vulners) to match discovered attributes of a target system to attributes from the database. The Grinder framework can be used in many different areas of large-scale security researches, as a connected Python module to your project or as a ready-to-use tool.

The main purpose of Grinder is to unify and leverage different security tools, aggregate gathered pieces of information, help to understand and exposed them. Grinder incrementally saves all scans, measurements, analytics, and statistics to its database to be able to compare results over time and track the corresponding measurement changes.

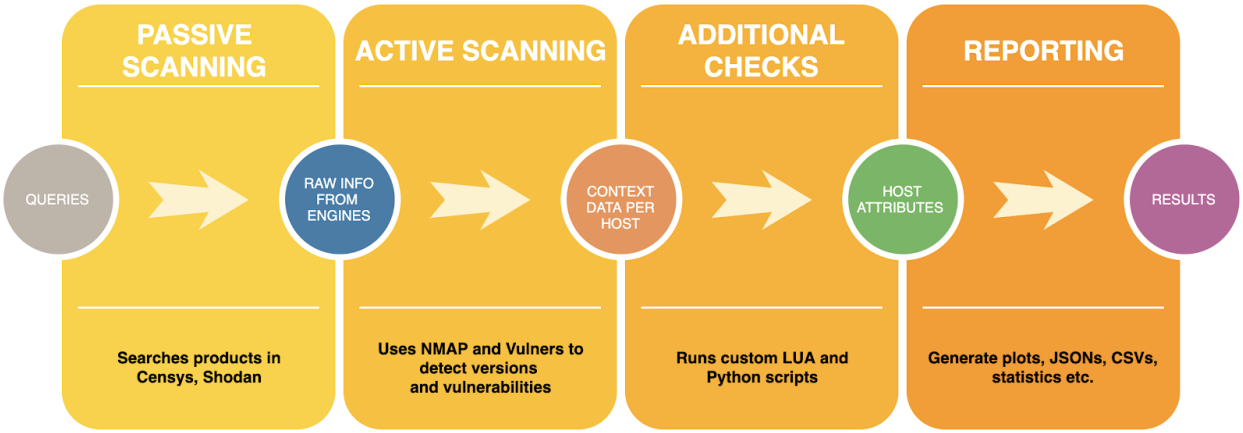

Figure 1. Grinder framework workflow.

To visualize gathered data, Grinder provides an interactive world map with all results. Grinder's map backend is written in Flask and supports additional REST API methods to get more information about all scanned hosts or some particular host from the map. Also, it is possible to show some additional information about the host interactively from the map. For example, each scanned host will be automatically checked for availability using the ICMP Ping method. There are also many additional features: the current host can be directly opened in

[12] Gordeychik S, Kolegov D, Nikolaev A, SD-WAN Internet Census. URL: https://arxiv.org/abs/1808.09027

Shodan, Censys and ZoomEye search engine web interfaces, the host can be shown on Google Maps with all available information about geolocation; it is also possible to make an IP lookup, or get raw information in JSON directly in browser or from your application using an API method.

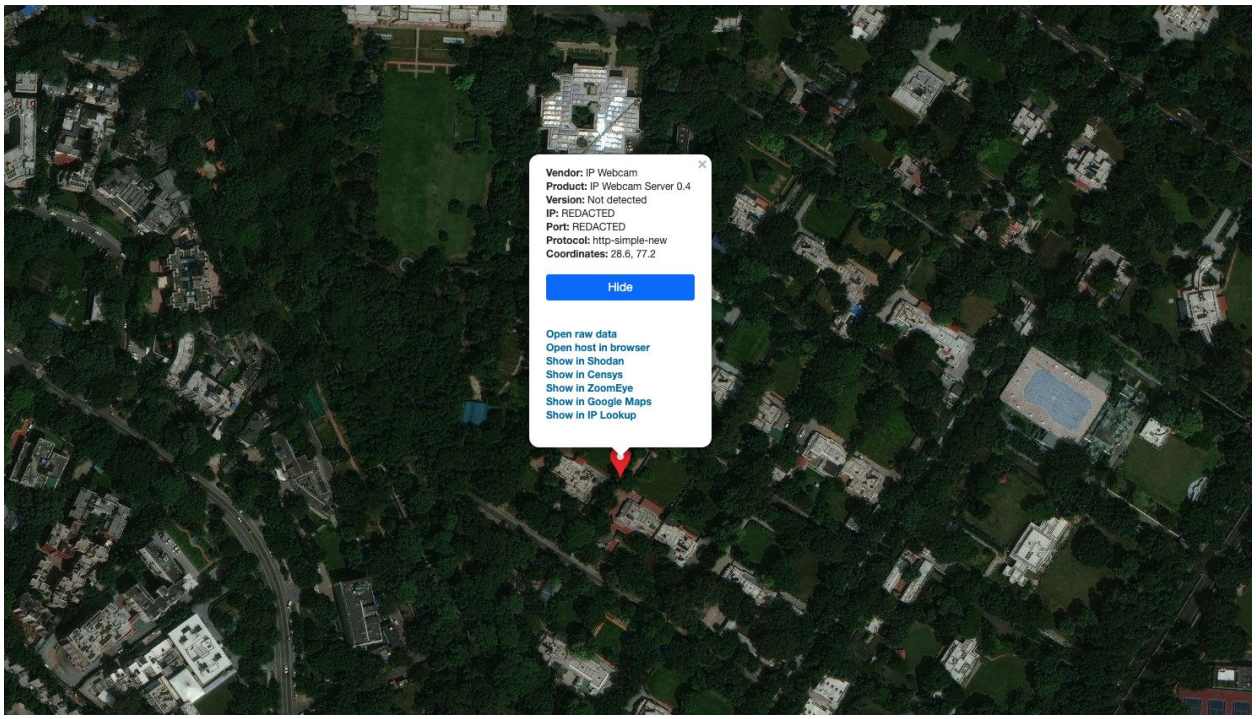

Figure 2. Grinder interactive map example.

At the present time, the Grinder Framework supports the features as follows:

1. Collecting hosts and additional information using Shodan and Censys search engines.
2. Scanning ports and services with boosted multi processed Nmap scanner wrapper.
3. Discovering vulnerabilities and additional information about them with Vulners and Shodan CVEs database.
4. Retrieving information about SSL certificates.
5. Scanning for SSL/TLS configuration and supported cipher suites.
6. Scanning for SSL/TLS bugs, vulnerabilities and attacks using TLS-Attacker.
7. Building an interactive map with information about the hosts found.
8. Creating plots and tables based on the collected data.
9. Custom scanning scripts support (in LUA or Python3).
10. Confidence filtering system support.
11. Special vendors scanning and filtering support.
12. Searching for documents, security bulletins, public exploits and many more things based on detected by Grinder vulnerabilities and software.

3 Security Validation Testing

We performed a number of security validation tests to validate the obtained measurements. We also examined a number of real AIML systems to understand their security against attacks disclosing sensitive data.

3.1. Training System Web UI

It was verified that most available components do not have authentication mechanisms. For example, we could gain access to different training systems, such as NVIDIA DIGITS (Deep Learning GPU Training System). This would allow an attacker to get all trained models for every training epoch. He also would make pre-trained models or publish current data to inference server. Moreover, he would gain access to datasets, explore it in the W UI or download all of the data. Additionally, we had access to system logs that could be downloaded too, along with model architecture source code in Python. The impact of this open functionality is obvious: in real systems, an ML model can be stolen using Web UI. As a result, an attacker can get trained models, datasets with data, task logs and the source code of the model architecture.

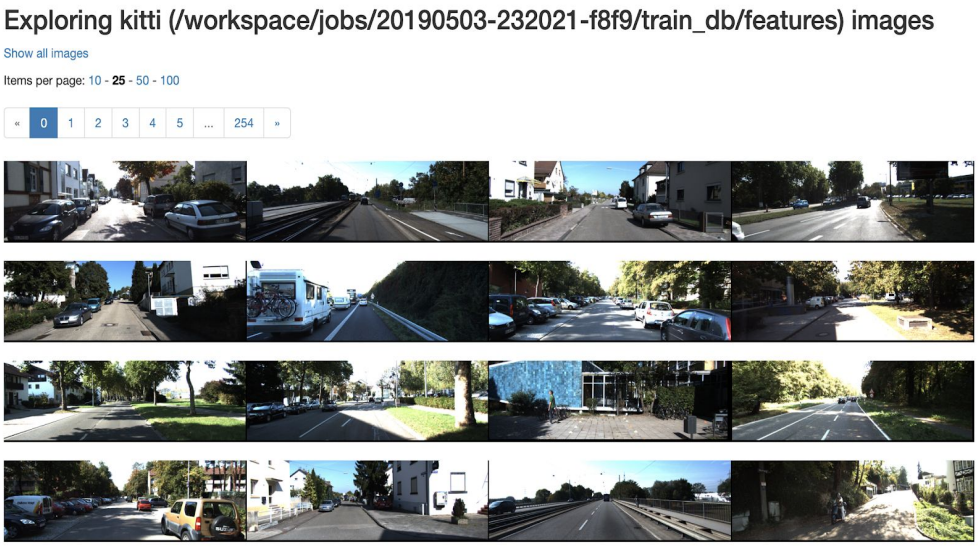

Figure 3. Example of browsing training images dataset with NVIDIA DIGITS explorer.

We also examined TensorBoard, which provides the visualization and tools needed for ML experimentation with TensorFlow. Lack of authentication mechanisms allows an attacker to control model learning process. It should be noted, at the present time, this is an expected behavior because TensorFlow and its components do not have authentication mechanisms by design. Also, TensorBoard allows using a “tfdbg” remote debugger to debug training sessions on

the fly. In this case, the scenario of data poisoning attacks can be real: an attacker can change the behaviour of a model based on input data provided in real-time and then appropriately modify attack vectors and input data.

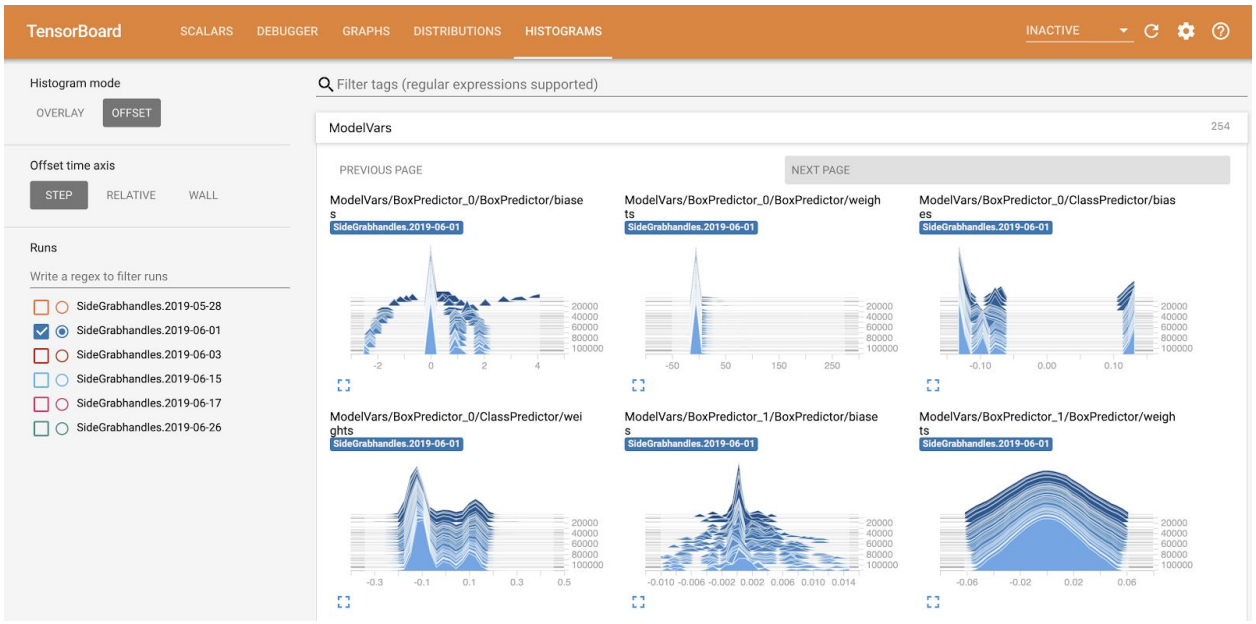

Figure 4. Example of an unauthenticated TensorBoard interface.

3.2. Database Systems

In case of searching data that was used for ML model training (annotations, images, text data and so on), we can search for open databases that contain related data. For example, we can search for open MongoDB databases without any authentication - in some situations they have a lot of different information about the training process, training data and so on. In this case, information theft is also possible: An attacker can gain access to datasets, tags, and labels with different databases and collections.

Figure 5. The examples of MongoDB indexes with 28.0 GB of datasets data.

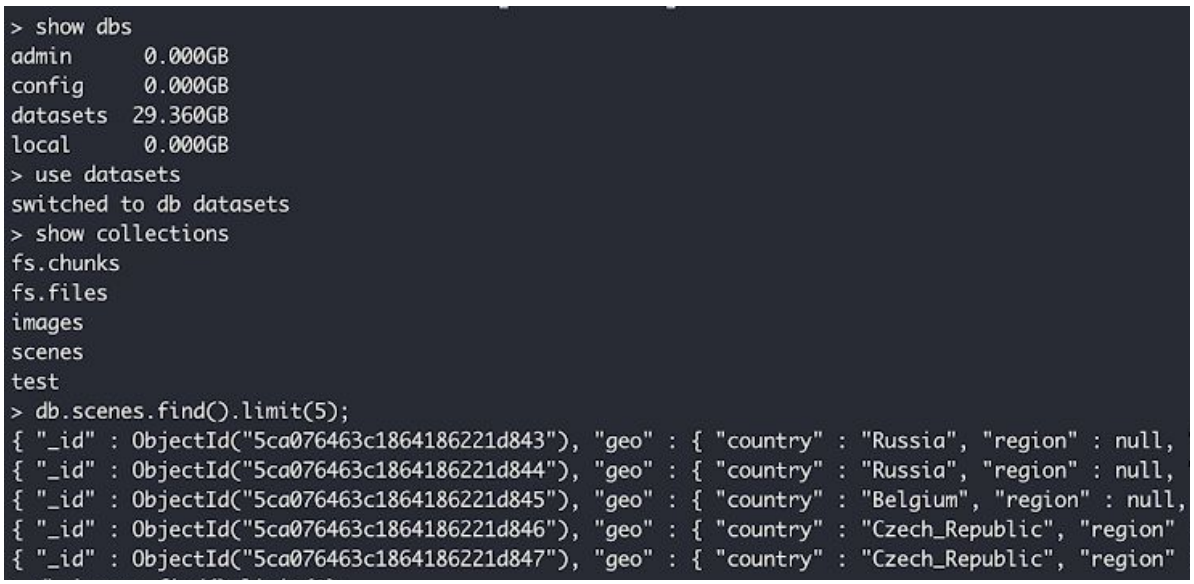

Figure 6. The collections of a MongoDB.

3.3. Logs and system information

ML-based services and applications leverage different searching and indexing systems to save logs and other system information. One of them is Elasticsearch[13]: distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents, along with Kibana[14] (visualization plugin for Elasticsearch).

The main problem that exposed Elasticsearch services can be easily found in Shodan with the following query: product:elastic port:9200 "Elastic Indices".

This query returns 19,629 exposed Elasticsearch systems. Moreover, we can search through their exposed data, because Shodan also indexes all of the available indices. Due to this fact, we can directly search for special keywords like "labels", "dataset", "data", "ml-logs" and so on.

Logs that are accessible via Elasticsearch analytic engines can contain a lot of different sensitive information. For instance, if a found system is used within medical organizations, it can contain personal data, medical results, or special additional information. Moreother, if an AIML system has Elasticsearch as an exposed service, we can get all the related information: information about the application, data that it uses, some additional information about datasets, labels, etc.

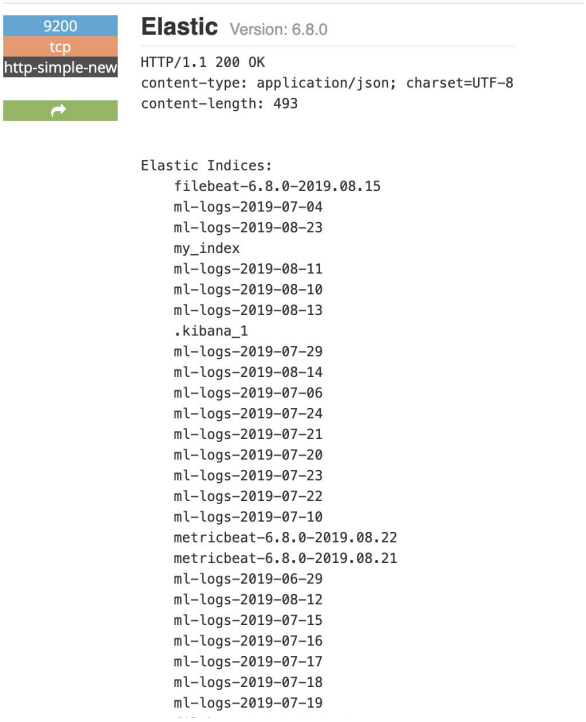

Figure 7. The example of Elasticsearch indices with ML logs data.

[13] Elasticsearch: RESTful, Distributed Search & Analytics. URL: https://www.elastic.co/products/elasticsearch
[14] Kibana: Explore, Visualize, Discover Data. URL: https://www.elastic.co/products/kibana

3.4. Containers

Docker containers are often used to deploy an ML system. Docker provides an API[15] for interacting with the Docker daemon (called the Docker Engine API), as well as SDK for Go and Python. The SDK allows build and scale Docker applications and solutions quickly and easily.

The Docker Engine API is a RESTful API accessed by an HTTP client such as *wget* or *curl*, or the HTTP library which is part of most modern programming languages.


```
"Id": "6486172d69aec4bfc49d2a1e29925cc2cc0ef7513df8aa3561b620c213d5e0b5",
"Names": [
    "/tensorrt"
],
"Image": "nvcr.io/nvidia/tensorflow:19.07-py3",
"ImageID": "sha256:a9c6af3f3056e8bbc13dee8d676bba6f55a750f36abd76d6944b0fc36ad1f709",
"Command": "/usr/local/bin/nvidia_entrypoint.sh bash",
"Created": 1571408583,
"Ports": [
```


Figure 8. The example of Docker API with a list of Docker containers.

Docker Engine API provides the following methods (endpoints) allowing you to export all containers by their IDs.:

1. GET /containers/json - to list all of the containers
2. GET /images/json - to list all of the images
3. GET /containers/(id or name)/logs - to export logs
4. GET /containers/(id or name)/export - to fully export some container by id

Also, it is possible to initiate other things like stop, run and create new containers on the remote host. In total we were able to find about 3 350 exposed Docker APIs by the following query:

1. "Docker Containers:" port:2375

We found about 13 900 instances on different ports by the following query:

1. "Docker Containers:" "image:" "Command:"

ML applications improperly using Docker API Engine can be compromised due to that fact that different tokens that applications use as in a startup configuration can be accessed easily. Then the credentials can be used to log in into those applications:

```
        Image: cv
        Command: /bin/bash -c 'source activate cv && jupyter-lab --ip=0.0.0.0 --no-browser --port=8888 --a
llow-root --NotebookApp.token=[redacted]'
```

Figure 9. The example of compromised Jupyter Notebook login token through Docker API.

[15] Docker Engine API. URL: https://docs.docker.com/engine/api/v1.24/

3.5. Medical Imaging

ML is actively used in the fields of medicine to recognize tumors or process patient data and images. During the examination, we were able to find the services of NVIDIA AI-Assisted Annotation, which are part of the NVIDIA Clara Train SDK[16]. The NVIDIA Clara Train SDK with AI-Assisted Annotation uses deep learning techniques to take points of interest drawn by radiologists to approximate points as input along with the 3D volume data to return an auto-annotated set of slices.[17]

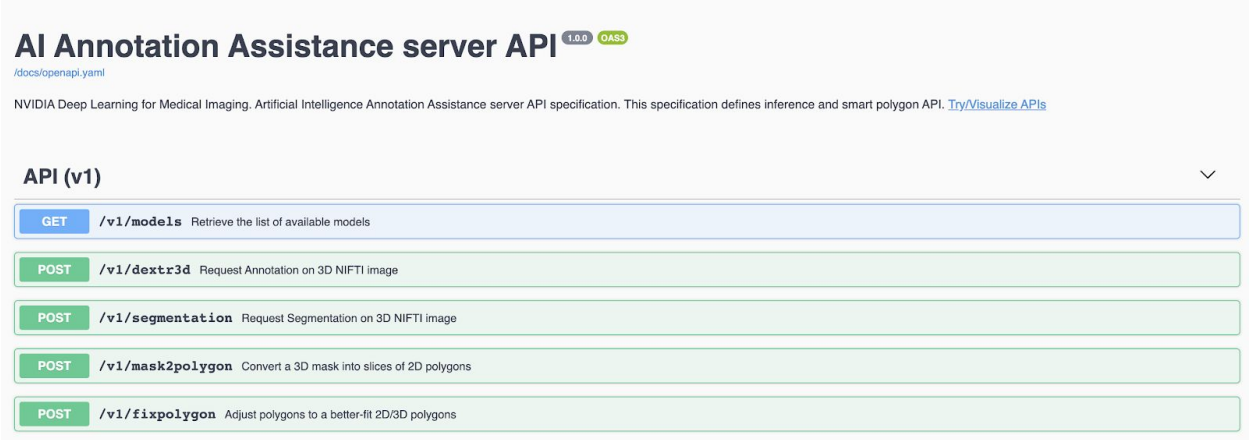

Figure 10. Example of AI Annotation Assistance server API web interface.


```
[
  {
    "roi": [
      128,
      128,
      128
    ],
    "name": "annotation_ct_liver",
    "sigma": 3,
    "version": "1",
    "labels": [
      "liver"
    ],
    "description": "A pre-trained model for volumetric (3D) annotation of the liver in portal venous phase CT image",
    "type": "annotation",
    "internal name": "annotation_ct_liver",
    "padding": 20
  },
```


Figure 10. Example of listed models in response from AIAA API by "GET /v1/models" request.

Also, some AI medical services often connected with DICOM PACS systems, which provide methods of retrieving information about patients, studies and images. These features allow the AIML systems to get information about patients directly from these servers. So, if we could find

[16] NVIDIA Clara. URL: https://developer.nvidia.com/clara
[17] AI Assisted Annotation Getting Started Guide :: Clara Documentation: https://docs.nvidia.com/clara/aiaa/tlt-mi-ai-an-sdk-getting-started/

an AI medical system, we would try to search for special medical servers allowing us to get information from them directly.

When we found DICOM servers, we used special queries, that defined in DICOM specification[18], to get the required information. According to the specifications, DICOM provides a special C-FIND service, that can be used to retrieve relevant to a search query information about a patient. So we can get a complete list of patients whose personal data or studies are used in this AIML system. To get this information, we used a "Wild Card Matching" feature that described[19] in DICOM protocol specification.

We defined a security test as follows: send a C-FIND query containing only wild card symbol ("*") in place of a patient name and check whether this query returns the complete list of all patients' data stored on the DICOM server.

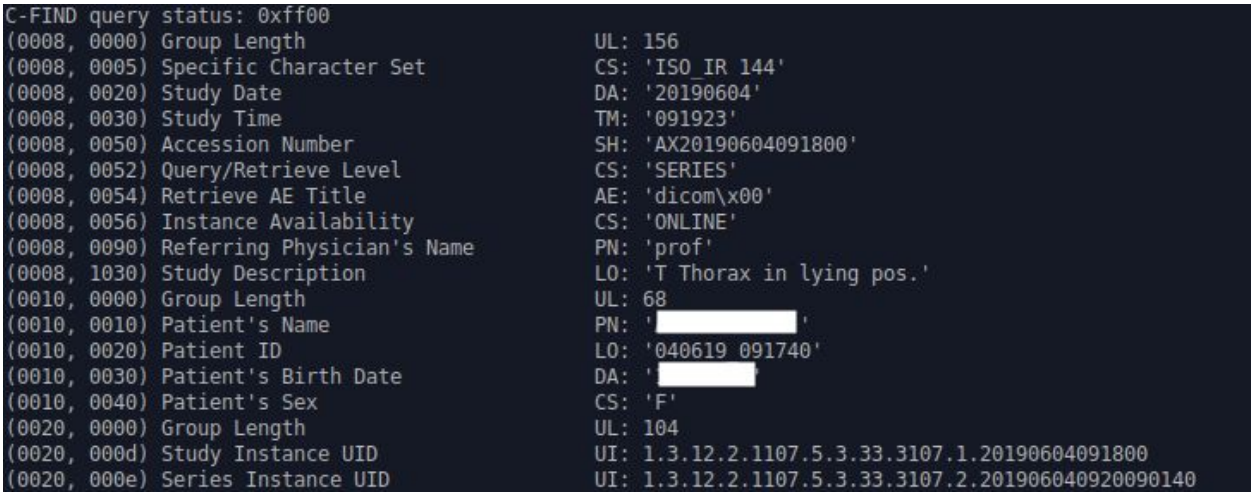

Figure 11. Private patient records retrieved from the DICOM server.

In most cases we were also able to use C-GET queries to retrieve all studies, images and protocols for all patients. We created a dataset[20] (a special query in DICOM sense) that contains any unique patient data (for example, patient name, patient ID, etc.), and after that, we could get all the matching results with C-GET query. The datasets were created using "dcmodify" utility from DCMTK toolkit as follows:

```
dcmodify --create-file -i "(0010,0010)=PATIENT_NAME" query_file.dcm
```

Then the file named "query_file.dcm" was used to perform a C-GET query retrieving all of the data related to the patient with the "PATIENT_NAME" name.

---

[18] DICOM Standard. URL: https://www.dicomstandard.org/current/

[19] C.2.2.2.4 Wild Card Matching: http://dicom.nema.org/medical/dicom/2016c/output/chtml/part04/sect_C.2.2.2.4.html

[20] https://support.dcmtk.org/docs-dcmrt/getscu.html

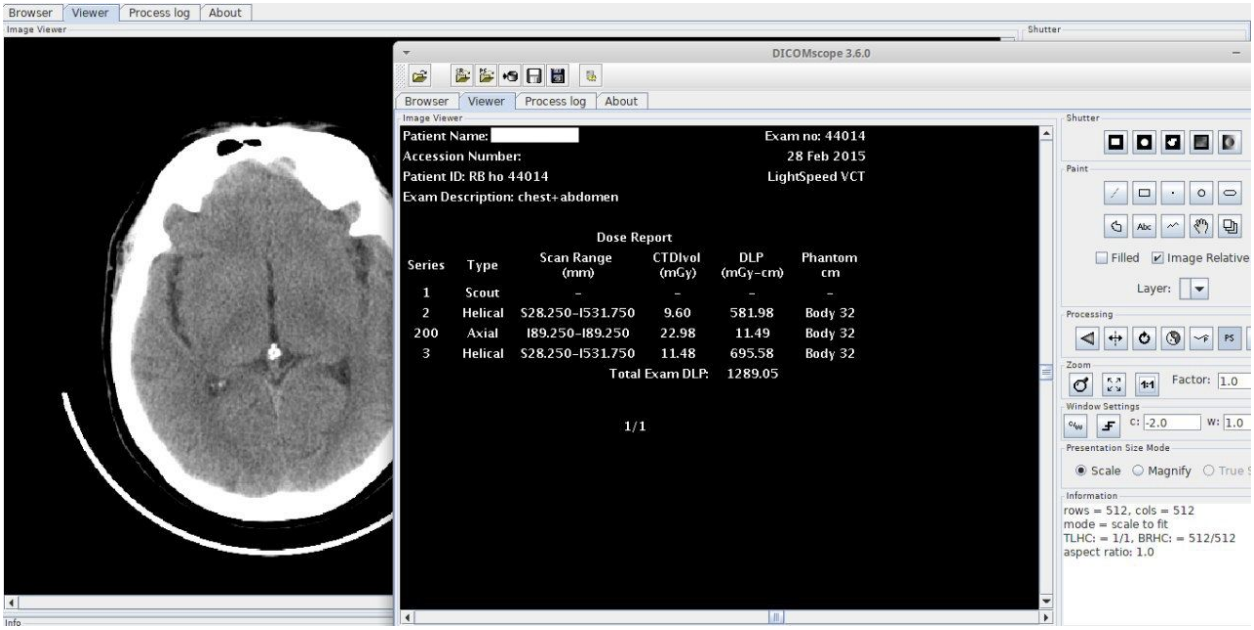

Figure 12. Example of study results retrieved with the C-GET query.

We were able to discover 2429 DICOM servers (running on 104 and 11112 ports), successfully connect to 1329 of them and retrieve the following supported abstract syntax rules:

1. Study Root Query/Retrieve Information Model - FIND: 1312
2. Study Root Query/Retrieve Information Model - MOVE: 1311
3. Study Root Query/Retrieve Information Model - GET: 813
4. Patient Root Query/Retrieve Information Model - FIND: 889
5. Patient Root Query/Retrieve Information Model - MOVE: 890
6. Patient Root Query/Retrieve Information Model - GET: 287
7. Patient/Study Only Query/Retrieve Information Model - FIND: 659
8. Patient/Study Only Query/Retrieve Information Model - MOVE: 659
9. Patient/Study Only Query/Retrieve Information Model - GET: 272
10. Composite Instance Root Retrieve - MOVE: 304
11. Composite Instance Root Retrieve - GET: 304
12. Composite Instance Retrieve Without Bulk Data - GET: 305

As you can see, 813 DICOM servers accepted C-GET[21] root queries[22] and 287 servers supported C-GET patient root queries[23]: We could download study results, medical protocols and

[21] C.4.3 C-GET Operation. URL: http://dicom.nema.org/medical/dicom/current/output/chtml/part04/sect_C.4.3.html
[22] C.6.2.1 Study Root Query/Retrieve Information Model. URL: https://www.dabsoft.ch/dicom/4/C.6.2.1/
[23] C.6.1.1 Patient Root Query/Retrieve Information Model. URL: https://www.dabsoft.ch/dicom/4/C.6.1.1/

other related documents. 1312 servers supported C-FIND[24] study root queries and 889 supports C-FIND patient root queries: that could allow us to get information about studies and patients, such as study description, study date and time, patient name, date of birth, sex, and others.

```
Quantity of accepted contexts: 6
ID: 7
Abstract Syntax: Study Root Query/Retrieve Information Model - FIND
Transfer Syntax(es):
    =Implicit VR Little Endian
Result: Accepted
Role: SCU only
ID: 9
Abstract Syntax: Study Root Query/Retrieve Information Model - MOVE
Transfer Syntax(es):
    =Implicit VR Little Endian
Result: Accepted
Role: SCU only
ID: 11
Abstract Syntax: Study Root Query/Retrieve Information Model - GET
Transfer Syntax(es):
    =Implicit VR Little Endian
Result: Accepted
Role: SCU only
ID: 19
Abstract Syntax: Composite Instance Root Retrieve - MOVE
Transfer Syntax(es):
    =Implicit VR Little Endian
Result: Accepted
Role: SCU only
ID: 21
Abstract Syntax: Composite Instance Root Retrieve - GET
Transfer Syntax(es):
    =Implicit VR Little Endian
Result: Accepted
Role: SCU only
ID: 23
Abstract Syntax: Composite Instance Retrieve Without Bulk Data - GET
Transfer Syntax(es):
    =Implicit VR Little Endian
Result: Accepted
Role: SCU only
```

Figure 13. Example of available server abstract syntaxes.

3.6. Infrastructure Services

This category includes a technology stack that is necessary to enable and support ML applications. The stack contains some popular actively-maintained AIML infrastructures related to the following aspects:

[24] 9.3.2 C-FIND Protocol. URL: http://dicom.nema.org/medical/dicom/2014c/output/chtml/part07/sect_9.3.2.html

1. The architecture of end-to-end ML training pipelines.
2. Inference at scale in production on cloud or end devices.
3. Compiler and optimization stacks for deployments on a variety of devices.
4. Novel ideas of efficient large-scale distributed training.

For example, we can search for Kubeflow[25] - the ML Toolkit for Kubernetes. The Kubeflow project is dedicated to making deployments of ML workflows on Kubernetes. The goal of this project is not to recreate other services, but to provide a straightforward way to deploy best-of-breed open-source systems for ML to diverse infrastructures.

Kubeflow contains the components as follows:

1. Notebooks: a JupyterHub to create and manage interactive Jupyter notebooks.
2. TensorFlow Model Training: a TensorFlow Training Controller that can be configured to use either CPUs or GPUs and be dynamically adjusted to the size of a cluster with a single setting.
3. Model Serving: a TensorFlow Serving container to export trained TensorFlow models to Kubernetes. Integrated with Seldon Core, an open-source platform for deploying ML models on Kubernetes, and NVIDIA TensorRT Inference Server for maximized GPU utilization when deploying ML/DL models at scale.
4. Multi-Framework: includes TensorFlow, PyTorch, MXNet, Chainer, and more.

Kubeflow can be used to create and run own Jupyter Notebooks, get access to Jupyter Notebook terminal, download datasets with data and logs and many more.

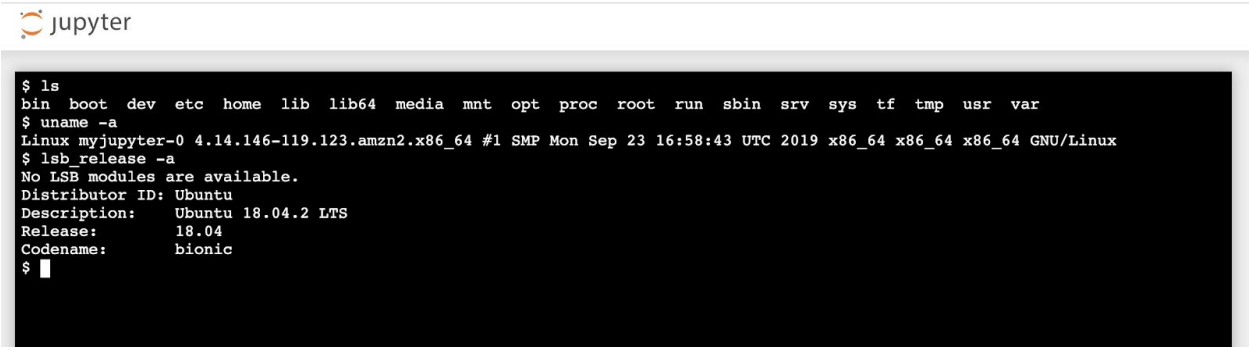

Figure 14. Example of Jupyter Notebook terminal from Kubeflow interface with anonymous user.

[25] Kubeflow - The machine learning toolkit for kubernetes. URL: https://www.kubeflow.org/

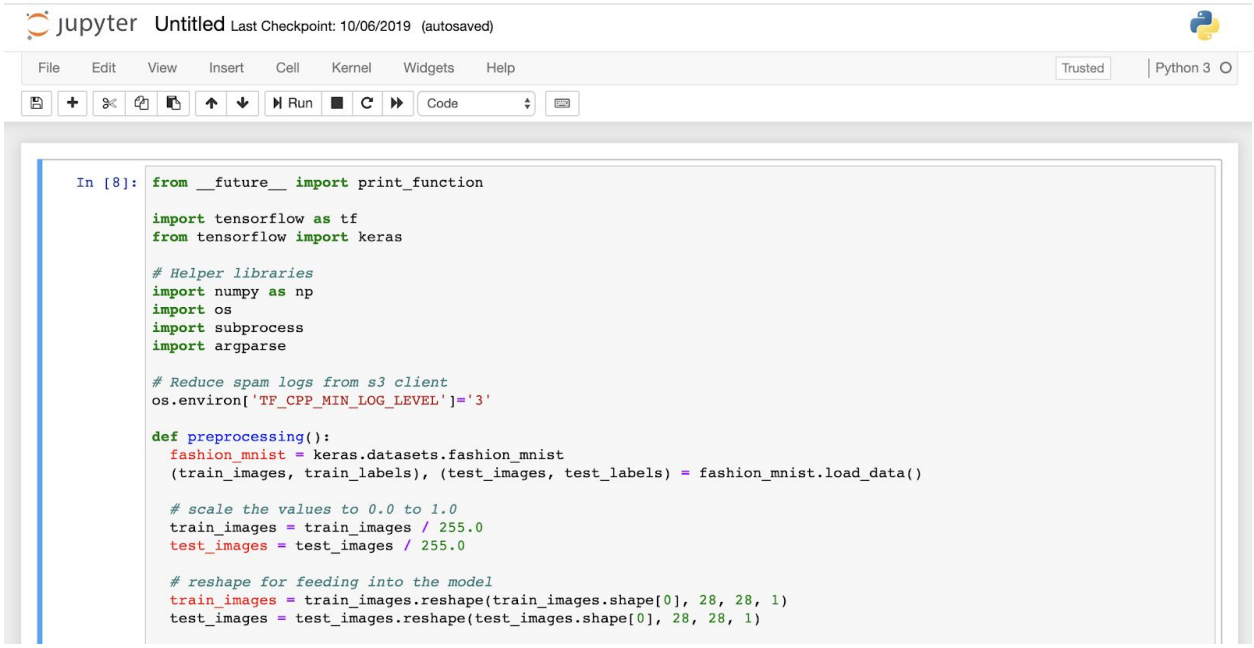

Figure 15. Available to edit and run Jupyter Notebook from Kubeflow interface with anonymous user.

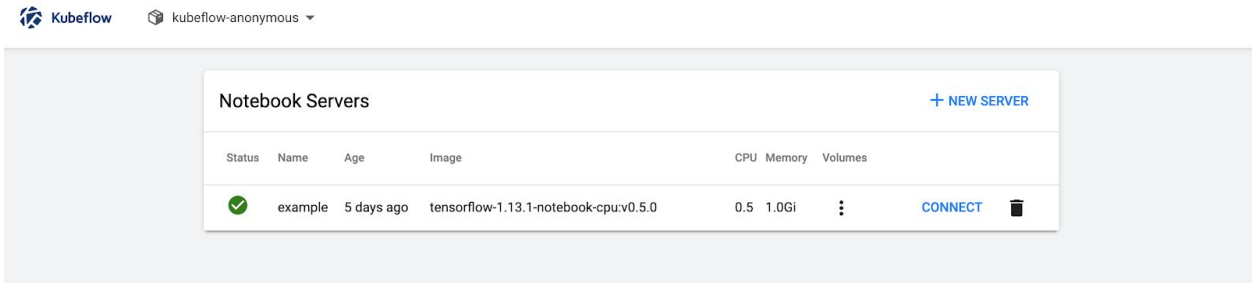

Figure 16. Running Notebook Servers with the possibility of creating a new server ("example" in this case) with an anonymous user.

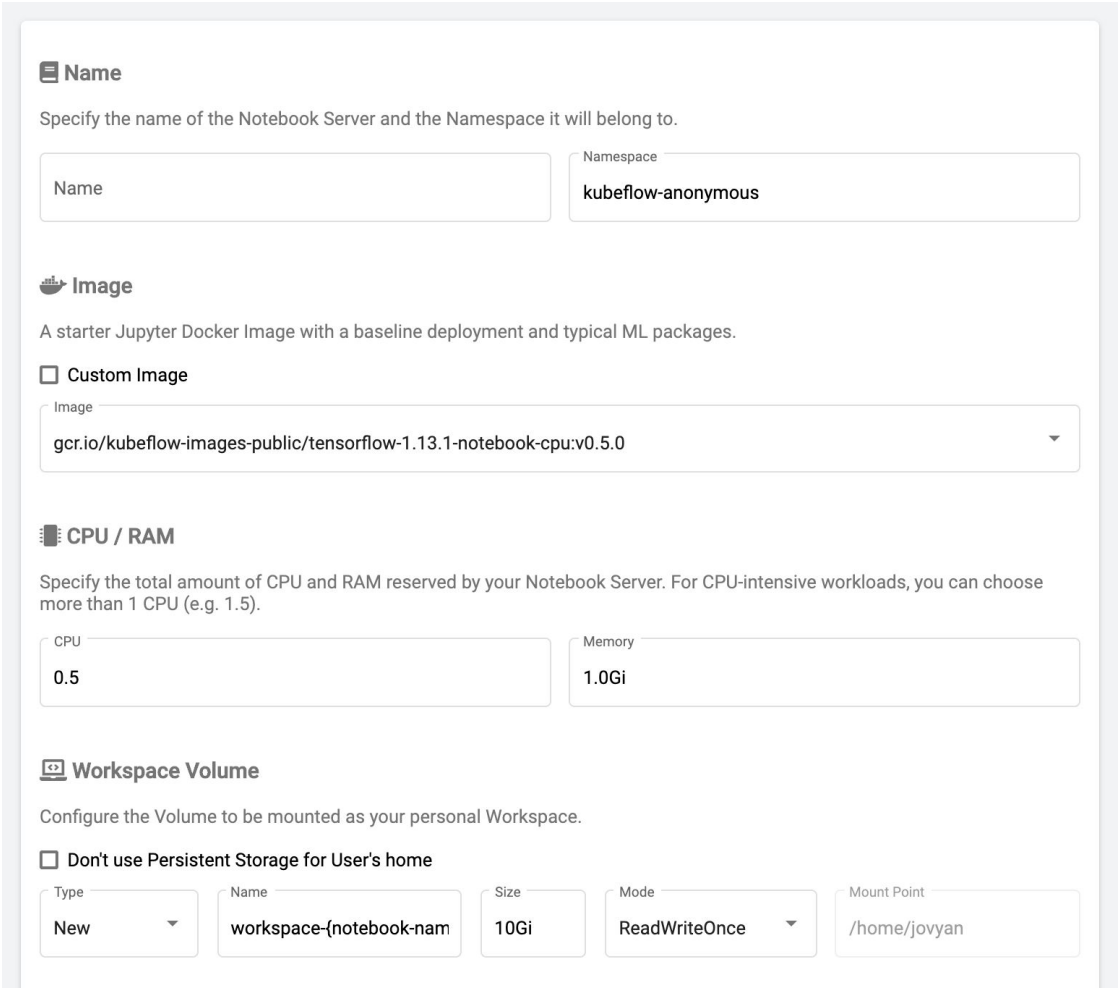

Figure 17. The interface of new server creating in Kubeflow interface with anonymous user.

As for another example, consider MLflow[26] - is an open-source platform for managing the end-to-end ML lifecycle. It was found that MLflow can be used to access models, data or results of training without any authentication, just by opening the web interface of an application.

[26] MLflow - A platform for the machine learning lifecycle. URL: https://mlflow.org

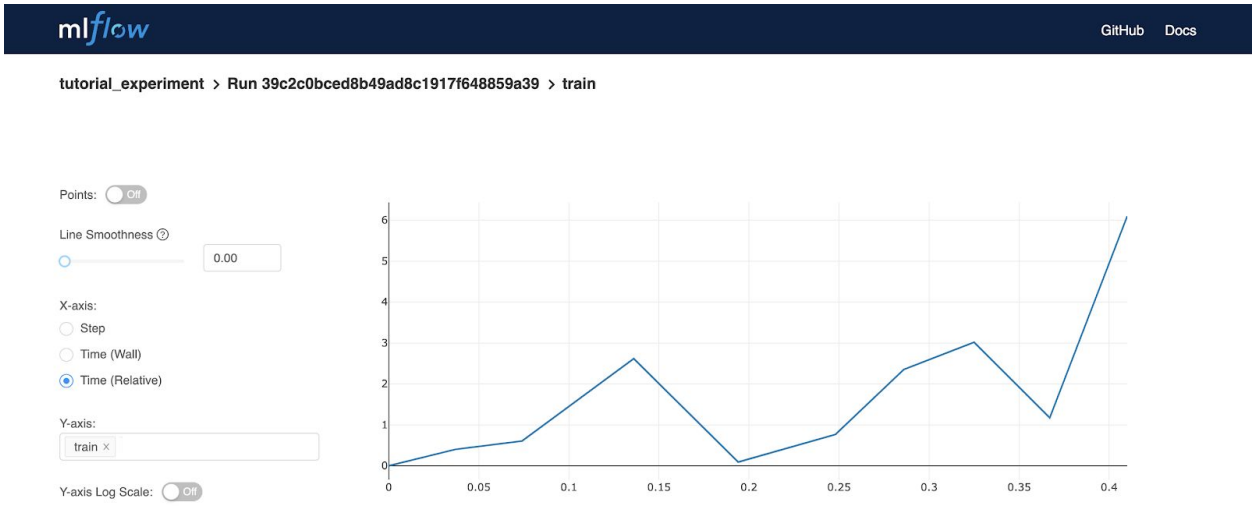

Figure 19. MLFlow training plots.

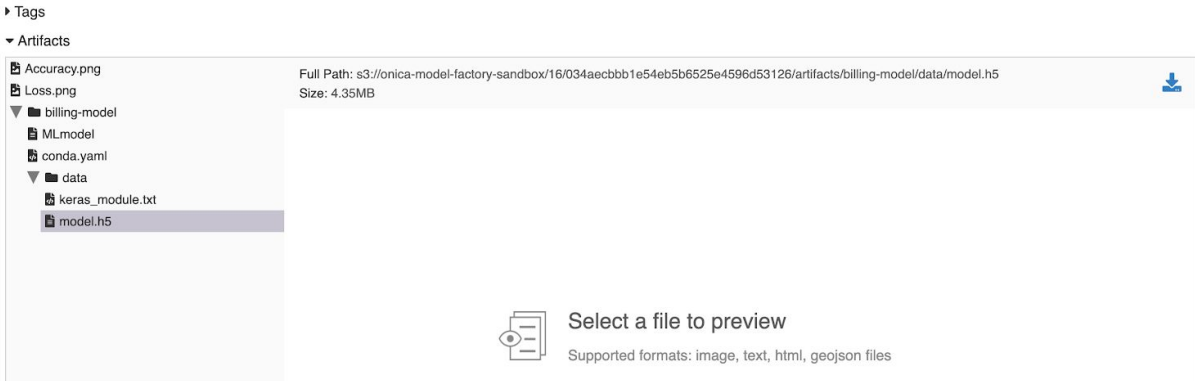

Figure 18. Available to download training data - model, information, statistics, etc.

3.7. Baseboard Management Controllers

A Baseboard Management Controller (BMC) is a service processor which is capable of monitoring the physical state of servers, computer or other hardware devices using sensors. For instance, it is able to monitor power supply voltages, fan speeds, operating system functions, humidity, and temperature; performs remote management task. The BMC controller is embedded in the motherboard itself.

Moreover, BMC controllers can be used to monitor and manage sophisticated AIML platforms like DGX-1 independently of the hardware and operating system. The controllers are accessed remotely through the Ethernet connection to the IPMI port, and the system itself exposes management services like SNMP.

For these reasons, if a BMC management controller is available on the Internet and implements improper authentication and security control mechanisms (e.g., supports default credentials, does not implement authentication or authorization at all), anyone can take control over high-performance computational systems like DGX-1.

In the case of the DGX-1 system, we could find some default credentials in documentation[27]. It was found that default BMC IPMI credentials are "qct.admin" / "qct.admin". Based on this fact, we supposed that other credentials may contain "qct" substring. Because DGX-1 systems also provide SNMP service, we tried to connect to SNMP using community strings based on "qct", "public" and "private" combinations and their mutations. So, if these credentials were valid, we would gain control over the DGX-1 system using a trivial password-guess attack.

It was found that "qct.private" and "qct.public" are valid SNMP credentials. We were able to establish a connection through SNMP to all known DGX-1 hosts using the following commands:

1. snmpwalk -v 2c -c qct.private <host> <seed>
2. snmpwalk -v 2c -c qct.public <host>

In total, we found 147 BMC controllers on DGX-1 and GPU accelerators, 14 of them had default credentials and 3 hosts with default credentials responded with the following strings over SNMP:

1. "DGX-1"
2. "DGX-1 with V100"
3. "DGX-1 with V100-32"

[27] DGX-1 User Guide. Configuring and Managing the DGX-1. URL: https://docs.nvidia.com/dgx/dgx1-user-guide/configuring-managing-dgx1.html#configuring-managing-dgx1

```
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.11.1 = STRING: "NVIDIA"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.11.2 = STRING: "NVIDIA"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.11.3 = STRING: "NVIDIA"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.11.4 = STRING: "NVIDIA"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.11.5 = STRING: "NVIDIA"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.12.1 = STRING: "DGX-1 with V100"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.12.2 = STRING: "DGX-1 with V100"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.12.3 = STRING: "DGX-1 with V100"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.12.4 = STRING: "DGX-1 with V100"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.12.5 = STRING: "DGX-1 with V100"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.13.1 = STRING: "1S2WU9Z0STB"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.13.2 = STRING: "N/A"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.13.3 = STRING: "N/A"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.13.4 = STRING: "N/A"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.13.5 = STRING: "N/A"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.14.1 = STRING: "v1.0"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.14.2 = STRING: "v1.0"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.14.3 = STRING: "v1.0"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.14.4 = STRING: "v1.0"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.14.5 = STRING: "v1.0"
SNMPv2-SMI::enterprises.7244.1.2.1.3.6.1.15.1 = STRING: "QTFCOU8010083"
```

Figure 19. Results of snmpwalk with default credentials on DGX-1 host

4 Implementation Security Measurements

This section describes the results of large-scale scanning of AIML implementations on the Internet.

4.1 Scanning Methodology

The employed methodology is defined as follows:

1. Craft signatures of the interfaces of AIML products and frameworks.
2. Define and express the signatures within a search engine query language.
3. Discover and enumerate devices using search engines (Shodan, Censys, etc.).
4. Use an incremental save method to store the results in the database.
5. If vulnerabilities are found, rescan the node two times to minimize false positives.
6. If the vulnerabilities are still present, check them again using PoC scripts in Python and Lua NSE.
7. Save the confirmed results to the database.
8. If a new AIML product or interface is discovered go to step 1.
9. Run the steps 3-7 regularly.

4.2 Vulnerability Discovery

One of the main features of the Grinder framework is collecting information about vulnerabilities and public exploits. To discover vulnerabilities, we use the following interfaces, resources, and methods:

1. Shodan vulnerability database. This type of vulnerability discovery is most safe because in this case we passively retrieve different valuable information about found vulnerabilities from the Shodan database.
2. Vulners Nmap NSE script[28]. In this case vulnerabilities discovery is based on the banner grabbing technique. Nmap actively scans a target to pull the useful information from banners. After it, Vulners' NSE script processes that information with regular expressions (software name, versions and others) and matches it to own base via API requests: if some of the banners are matched to known vulnerabilities in Vulners database, it will return a list of known CVEs.
3. Vulners Python API wrapper[29]. We use the original API of the Vulners database to search for public exploits (both for CVE and CPE strings) with custom python wrappers.

The main core of threat intelligence is under the hood of the Nmap scanner, that used for CPE strings[30] and software versions retrieving. For example, if we have a full valid CPE string of some software, including version, we can easily find exploits for it. For example, we could find the TensorBoard[31] appliances, and scan it in aggressive mode with Nmap using the following command:

```
nmap -v -A <tensorboard-application-ip> --top-ports 100
```

After that, we could detect this host uses the Werkzeug httpd CGI library and Python 2.7.12. Nmap would return a CPE string for these software components as follows:

```
cpe:/a:python:python:2.7.12
```

[28] NSE script based on Vulners.com API. URL: https://github.com/vulnersCom/nmap-vulners
[29] Vulners Python API wrapper. URL: https://github.com/vulnersCom/api
[30] CPE - Common Platform Enumeration: CPE Specifications. URL: https://cpe.mitre.org/specification/
[31] TensorBoard: TensorFlow's visualization toolkit. URL: https://www.tensorflow.org/tensorboard

Figure 20. Grinder framework output results from Nmap on port 8888.

With this information, we can search for exploits from Vulners and different other databases, also we can search for different vulnerabilities based on software versions and CPE strings.

In most cases, information obtained by active and passive fingerprinting can also be used to detect known vulnerabilities in a configuration. For example, by detecting basic information about the host, such as operating system family and version, web server version, API methods, open ports and services we can obtain additional information, such as software and hardware vulnerabilities that are related to discovered versions.

According to obtained results, the most common unique vulnerabilities are the following:

- CVE-2018-1312 (detected 884 times on hosts or 3.3% of all found vulnerabilities, CVSS 3.0 Base Score: 9.8 CRITICAL)
- CVE-2019-0220 (detected 823 times on hosts or 3.0% of all found vulnerabilities, CVSS 3.0 Base Score: 5.3 MEDIUM)
- CVE-2018-17199 (detected 806 times on hosts or 3.0% of all found vulnerabilities, CVSS 3.0 Base Score: 7.5 HIGH).

Most of these vulnerabilities are related to web server software parts (mostly on outdated or misconfigured Apache servers) because many startup and light ML applications (especially working with JavaScript ML libraries) hosted on Apache servers due to a reason of easy configuration and simple deployment.

In total, we found 662 unique vulnerabilities, of which 36 vulnerabilities or 5.4% of all unique vulnerabilities quantity have critical severity level (CVSS 3.0 Base Score between 9.0 and 10.0). By CVSS 3.0 score system, 159 vulnerabilities or 24.1% of all unique vulnerabilities quantity have high severity level (CVSS 3.0 Base Score between 7.0 and 8.9), 429 vulnerabilities or 64.9% of all unique vulnerabilities quantity have a medium severity level (CVSS 3.0 Base

Score between 4.0 and 6.9), 37 vulnerabilities or 5.6% of all unique vulnerabilities quantity have low severity level (CVSS 3.0 Base Score between 0.1 and 3.9).

In terms of hosts, 1280 hosts or 36.6% of all hosts have at least one vulnerability with a medium severity, 1097 hosts or 31.4% of all hosts have at least one vulnerability with high severity, 977 hosts or 27.9% of all hosts have at least one vulnerability with low severity. 142 hosts or 4.1% of all hosts have the highest severity - critical, with CVSS 3.0 Base Score between 9.0 and 10.0.

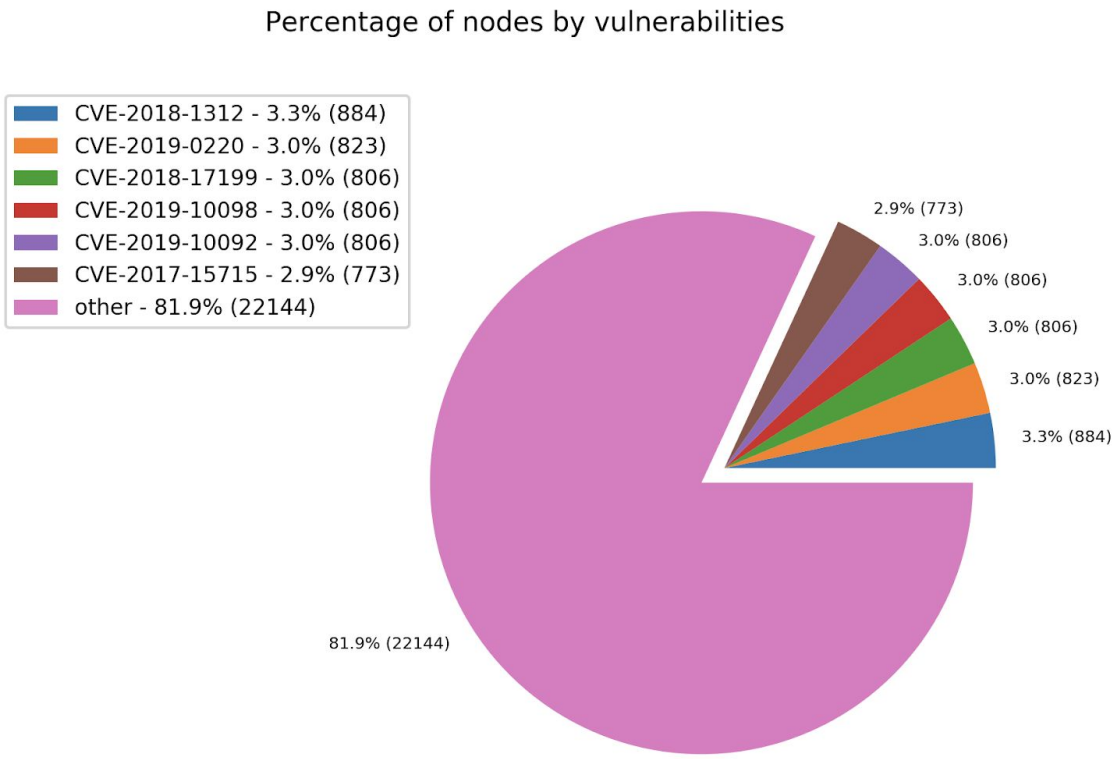

Figure 21. Percentage of vulnerabilities based on the number of detection on hosts.

4.3 Overall Results

Using the provided methodology above, we performed large-scale measurements of AIML-related systems and applications on the Internet. In the period from July 2019 to October 2019, we detected 6225 unique IP-addresses belonging to AIML systems and applications. According to the results, most of the remotely accessible hosts with AIML components (by unique IP addresses) are located in North America (2643 hosts or 42.8% from all results), in

Europe (1644 hosts or 26.6% from all results), and in Asia (1537 hosts or 24.9% from all results).

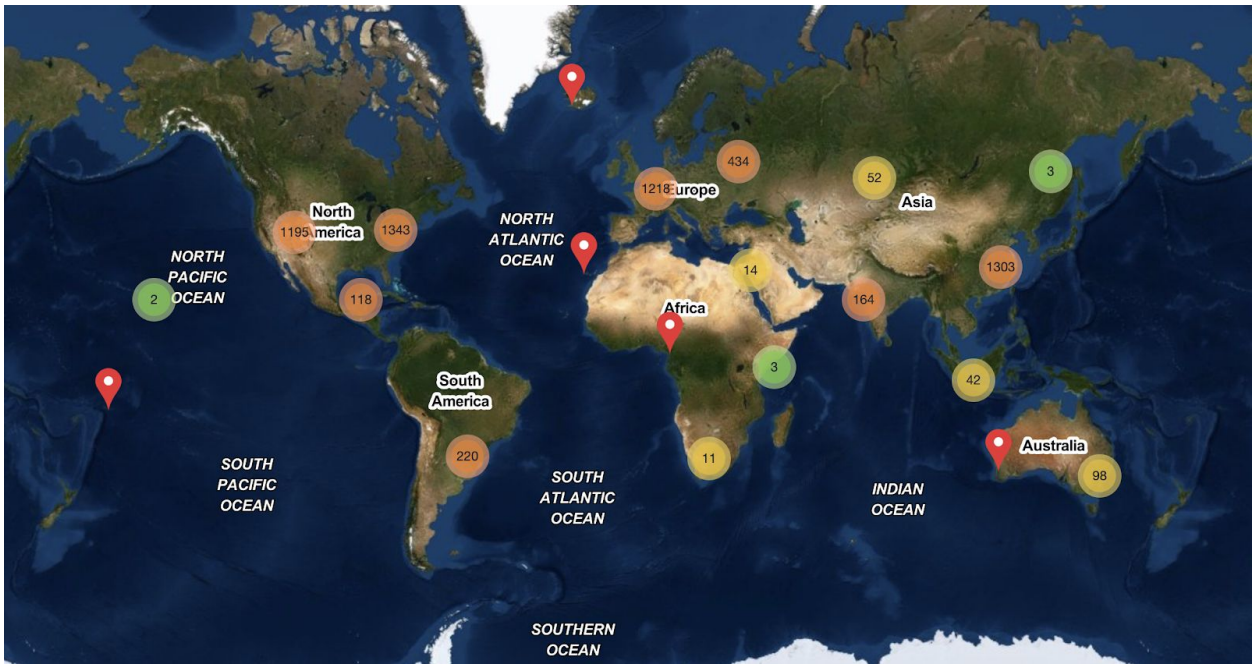

Figure 22. AIML availability by geo IP from July 2019 to October 2019

4.4 Training Systems and Control Interfaces Statistics

In November 2019 we performed a new scanning which includes only training systems, control and management interfaces and federated learning systems that are currently accessible on the Internet. This scan included the following products:

1. Deeplearning4j
2. FedAI FATE
3. NVIDIA DIGITS
4. NVIDIA AI Annotation Assistance server
5. NVIDIA DGX-1 Baseboard Management Controller
6. NVIDIA DGX-2 Baseboard Management Controller
7. Quanta CT Baseboard Management Controller
8. Open Neural Network Exchange
9. Oracle GraphPipe
10. Amazon Greengrass
11. MLFlow
12. Google Kubeflow
13. TensorFlow

14. TensorBoard
15. Caffe
16. Apache MXNET

In total, 1415 control interfaces and training systems were found. According to the result, the most widespread AIML systems are: Kubeflow (538 hosts or 38.0% from all results), Apache MXNET (256 hosts or 18.1% from all results), MLFlow (215 hosts or 15.2% from all results), TensorFlow (70 hosts or 4.9% from all results), TensorBoard (70 hosts or 4.9% from all results) and Caffe (65 hosts or 4.6% from all results).

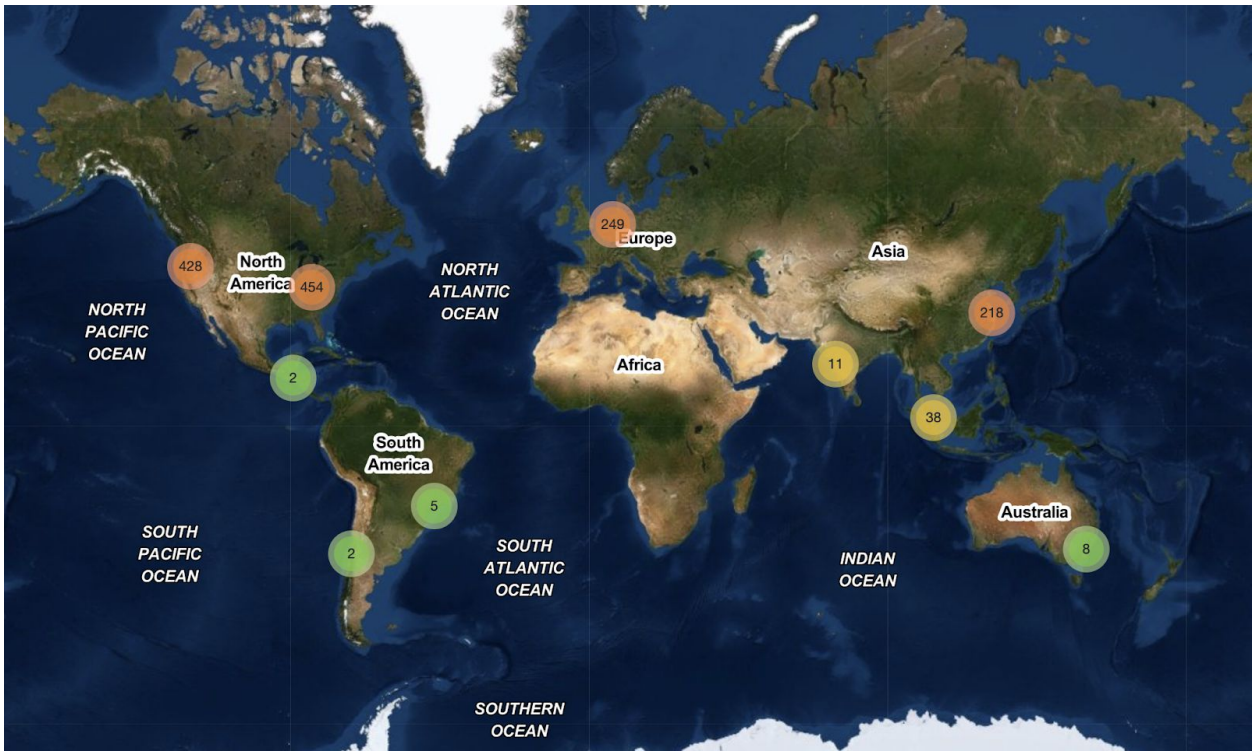

Figure 23. ML control interfaces availability by geo IP in November 2019.

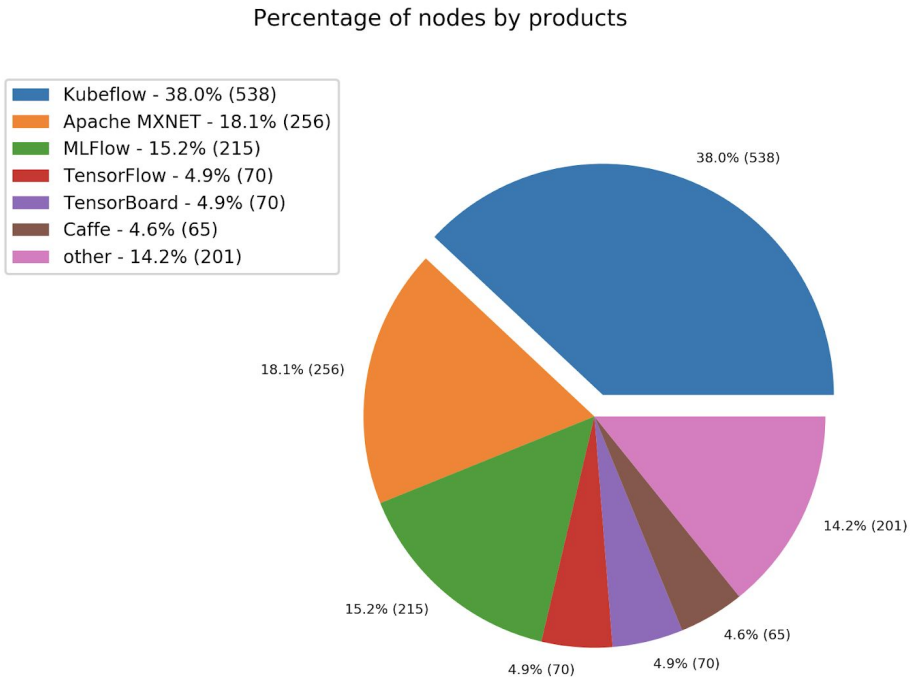

Figure 24. Percentage of ML control interfaces by unique products.

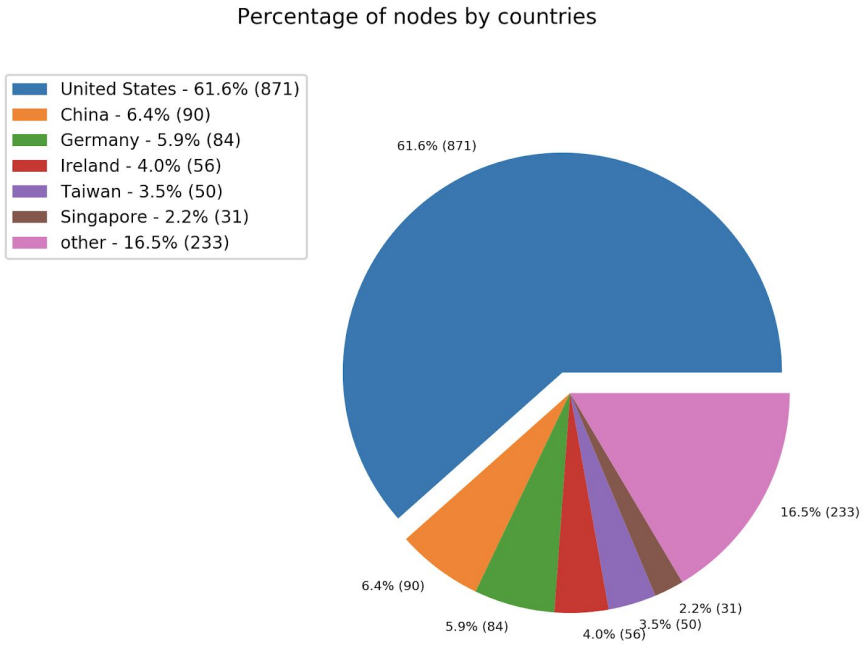

Figure 25. Percentage of ML control interfaces by countries.

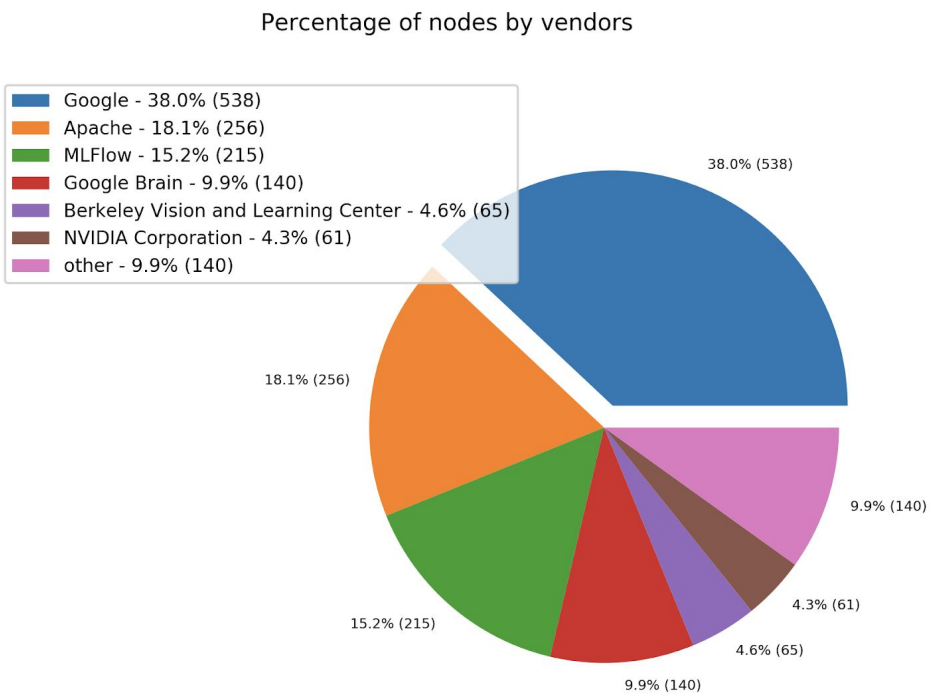

Figure 26. Percentage of ML control interfaces by vendors.

5 Conclusions

We explored different practical practical systems related to ML and AI to measure their level of implementation security. To do this we crafted signatures for them and them employed the Grinder framework to automate the searching systems and obtaining specific information about them. Using this approach we collected information that allowed us to evaluate the common security level for AIML products. We also performed security validation tests on different components of AIML systems to understand their security level and found multiple vulnerabilities. The measurements showed that the implementation security level of AIML systems is low. AI and ML community should bring to the attention of practical aspects of cybersecurity.