5-23-2023

Constrained Optimization Based Adversarial Example Generation for Transfer Attacks in Network Intrusion Detection Systems

Marc Chale

Bruce Cox

Jeffery Weir

Nathaniel D. Bastian
Army Cyber Institute, U.S. Military Academy, nathaniel.bastian@westpoint.edu

Recommended Citation

Chalé, M., Cox, B., Weir, J. et al. Constrained optimization based adversarial example generation for transfer attacks in network intrusion detection systems. Optim Lett (2023). https://doi.org/10.1007/s11590-023-02007-7

Optimization Letters
https://doi.org/10.1007/s11590-023-02007-7

ORIGINAL PAPER

Constrained optimization based adversarial example generation for transfer attacks in network intrusion detection systems

Marc Chalé[1,2] · Bruce Cox[1] · Jeffery Weir[1] · Nathaniel D. Bastian[1,2]

Received: 13 October 2022 / Accepted: 24 April 2023

Abstract
Deep learning has enabled network intrusion detection rates as high as 99.9% for malicious network packets without requiring feature engineering. Adversarial machine learning methods have been used to evade classifiers in the computer vision domain; however, existing methods do not translate well into the constrained cyber domain as they tend to produce non-functional network packets. This research views the payload of network packets as code with many functional units. A meta-heuristic based generative model is developed to maximize classification loss of packet payloads with respect to a surrogate model by repeatedly substituting units of code with functionally equivalent counterparts. The perturbed packets are then transferred and tested against three test network intrusion detection system classifiers with various evasion rates that depend on the classifier and malicious packet type. If the test classifier is of the same architecture as the surrogate model, near-optimal adversarial examples penetrate the test model for 69% of packets whereas the raw examples succeeds for only 5% of packets. This confirms hypotheses that NIDS classifiers are vulnerable to adversarial attacks, motivating research in robust learning for cyber.

Keywords Cyber security · Network intrusion detection · Adversarial machine learning · Constrained optimization · Meta-heuristic

1 Introduction

Cyber attackers leverage their knowledge of cyber (software, hardware, etc.) vulnerabilities to compromise the cyber triad: confidentiality, integrity, and availability of systems [1]. Cyber security strategies typically account for a balance of cost, utility, and security but

Bruce Cox, Jeffery Weir and D. Bastian have authors contributed equally to this work.

✉ Marc Chalé
marc.chale@afit.edu

Extended author information available on the last page of the article

it is often possible for a motivated attacker to defeat hardened networks [1–3]. Evasion attacks, for example, are one such avenue used to defeat cyber security systems [4, 5].

Adversarial machine learning (AML) has taken on a spotlight at the intersection of machine learning and computer security research as it highlights serious security vulnerabilities for learned classifiers. Szegedy et al [6] discovered that convolutional neural networks are easily fooled when images are strategically perturbed. Such images, intentionally perturbed to fool a classifier, are called *adversarial examples*. Strategies to create adversarial examples of images enjoy the advantage that digital images are an unconstrained domain with no disallowed pixel combinations. On the contrary, data in the cyber domain is strictly structured according to the internet protocol (IP) suite [7, 8]. Therefore, any attempt at generating adversarial examples in the *constrained* cyber domain must adhere to the rigid structure of network traffic packets and retain packet payload functionality.

Network intrusion detection systems (NIDS) are software products that audit logs of network traffic to identify malicious packets. Most modern NIDS incorporate machine learning classifiers [9]. Many NIDS classify using aggregate features of internet connections called network flow data. While it is possible to evade network flow NIDS with adversarial examples, there is no mature technology to reverse engineer the network flow feature vector into a packet capable of end-to-end adversarial attack [7, 10]. Other NIDS classify raw traffic data [11–13]. Adversarial attack against raw traffic NIDS requires carefully enforced constraints on perturbed packets, and this has not been technologically feasible, until now.

This manuscript makes the following contributions. We reframe the constrained adversarial example generation problem and provide a mathematical optimization formulation for optimally perturbing raw network packet payloads without affecting the packet function. A biologically inspired meta-heuristic, similar to a genetic algorithm, is provided to solve the formulated constrained optimization problem. A designed experiment is used to find optimized hyperparameters for the meta-heuristic. Then, additional experiments are conducted to compare the evasion rate of adversarial examples when transferred to three machine learning based NIDS. Just as the discovery of adversarial examples in the image domain has led to adversarial training as a robust defense, the vulnerabilities we uncover in this research open the door to a new generation of robust NIDS for the cyber domain.

The remainder of this work is structured as follows. Section 2 provides a review of relevant research. Section 3 defines the problem of constrained optimization for adversarial example generation, provides the use case for NIDS, and gives the formulation and experiment methodology. Results are presented and analyzed in Sect. 4, and conclusions are provided in Sect. 5.

2 Literature review

2.1 Adversarial machine learning

Numerous works have studied ways in which cyber infrastructure is vulnerable to threats [1, 14–18]. Evasion is an AML attack tactic that attempts to defeat NIDS by modifying, duplicating, or timing packets entering a network [19]. Following the discovery of adversarial examples in the image domain [6], many studies have worked towards adversarial evasion attacks in the cyber domain. Of the multiple manuscripts covering NIDS adversarial attacks [4, 5, 20] and others analyzed by Rosenberg et al [7], nearly all of them perturb feature vectors such as network flow data. Only Kuppa et al [21] converts features back to actual network packets, although this is only done for several simple header fields. Kuppa et al attacks anomaly detection based NIDS, which are important but less ubiquitous than signature-based NIDS such as neural network classifiers. Anomaly detection NIDS are highly sensitive to the choice of data features [9]. According to analysis by Appruzzese et al [10], access to training data, feature set, detector model, oracle, and the capability to create actual packets are crucial aspects of attacker power that should not be taken for granted. Researchers should avoid overly favorable assumptions on attacker power when proposing a realizable end-to-end adversarial attack against NIDS.

2.1.1 Insufficient constraints

Most adversarial attacks are constrained in some general manner. Standard approaches in the image domain include constraining pixel intensity between $[0, 1]$ [6] and by bounding the magnitude of perturbation, $d(\mathbf{x}, \mathbf{x}^0) \leq d_{\max}$ [22]. Constraints in the contested cyber domain become much more complicated for several reasons. Naïve perturbations to network traffic data would result in a non-functioning output. Cyber protocols are complex. A packet, for example, must engage in a secure two-way IP connection and any modifications to packet header could subvert this connection. Spurious modifications to a packet payload could corrupt the malicious effect of the packet. For instance, changing a writeFile command to a readFile command could help a packet evade a NIDS, but the packet may fail to complete its attack [7]. Adversarial examples intended to defeat NIDS must be constrained with domain specific heuristics. Chernikova et al [20] provides an algorithm that enforces domain specific constraints and dependencies but does so with feature vectors and utilizes gradient information.

2.1.2 Inefficacy of gradient approaches

The literature is rich with formulations that leverage gradient information to iteratively perturb images until they fool classifiers. Contributions such as L-BFGS [6], Fast gradient sign method [23], JSMA [24] and the Carlini & Wagner method [25] have produced adversarial examples in the cyber domain. These techniques, however, are not a natural fit for cyber data because small perturbations are not possible for discrete features (such as those in network packet payloads) [7].

2.1.3 NIDS feature engineering

Effective pre-processing and feature engineering tend to be domain specific and algorithm specific. Imputation, aggregation, augmentation, vectorization, normalization, and kernel methods are other examples of pre-processing that, when performed properly, improve the performance of machine learning models [26]. Raw features in the cyber domain are logged from time series data, packet headers, packet payloads, or statistical metrics of network flow [27]. Feature engineering often relies on the judgement and experience of the analyst performing the study [26], and several works analyzed feature engineering in the cyber domain [28–31].

Deep learning models, particularly convolutional neural networks (CNN), have the capacity to learn features that are more complex than a human can encode. It may still be beneficial to perform manual feature engineering with deep models as it motivates specific relationships and reduces computational resources for training [26]. Others argue that proprietary feature logging software such as Zeek and Security Onion add an extra layer of obfuscation which biases learning [11]. De Lucia [11] also notes that many cyber features are derived from packet headers and are easily spoofed, albeit only an advanced adversary can in theory realize such an attack without corrupting the network packet [7]. To mitigate this risk, Bierbrauer et al [12] removes header information from training data and demonstrates raw payload classification with up to 98.95% accuracy.

2.2 Cyber data sets

Hindy [9] provides analysis on 85 manuscripts and 30 prominent data sets for NIDS research. According to the study, 50.5% of NIDS manuscripts utilized the KDD-99 data set [9], which was recorded over two decades ago and has well documented flaws [32]. Another 17.2% of studies employed the NSL-KDD dataset. Although NSL-KDD corrects many statistical flaws of KDD-99, it is still derived from the outmoded KDD-99 data logs. The CICIDS data set [33] compiles raw packet capture (pcap) data collected on a real network over the course of a week during which time several types of attacks were conducted. Bierbrauer at al [12] investigated transfer learning among several cyber data sets and machine learning architectures. Bierbrauer at al found that models pre-trained with the CICIDS data set were accurate and also generalize well for use with other data sets. For these reasons, the CICIDS data set is used in this study.

3 Methodology

3.1 Overall approach

A new approach is necessary to generate adversarial examples in constrained domains. This problem type can be referenced as the *The Constrained Adversarial Example Generation Problem*. The motivating use of constrained adversarial examples is to obtain a packet that evades a NIDS and executes a malicious task inside the target network. This is solved with a meta-heuristic that substitutes functionally equivalent elements of the payloads in order to maximum its cross entropy w.r.t. a well-trained surrogate model. High cross-entropy implies an inability for the surrogate model to predict the truth label with high confidence. Further increasing the cross-entropy results in a high confidence prediction of the incorrect class.

A primary concern for the raw packet perturbation is to maintain the integrity of the packet. Accordingly, the proposed meta-heuristic makes substitutions to portions of the payload but never changes the effect of the compiled code. This approach is reasonable in many programming protocols. Most content in HyperText Markup Language (HTML) is case-insensitive [34] and we use HTML for this study.

The corpus matrix for this problem is engineered by a subject matter expert. It is dependent on the specific data encoding used. The expert identifies all functionally equivalent ASCII characters present in the payload. Since the character “H” and the character “h” are functionally equivalent, they are placed into the same column of the corpus. Such a column is generated for every character in the raw packet. Figure 1 shows that each character position (allele) in the raw packet payload is associated with a column of the corpus. This exact methodology could be directly adapted to any case-insensitive language such as CSS, BASIC, Fotran, DOS and SQL, for example. The general approach could be adapted to many other objects in cyberspace. For example, a python command “exit()” and “stop()” perform a practically equivalent task, so they could share a column in the corpus. Similar to the http problem, equivalent python commands could be substituted for optimal evasion of a detector. Additional domain constraints could be added if necessary for the use case.

The meta-heuristic applied is a variation of the genetic algorithm. There are, however, key decisions in the implementation of the meta-heuristic that enforce functional equivalence of the input and output. Each solution in the population is called a *chromosome*. The chromosome is comprised of units, called *alleles*. In this formulation, *genes* are the choice of code placed in the allele locations. The key

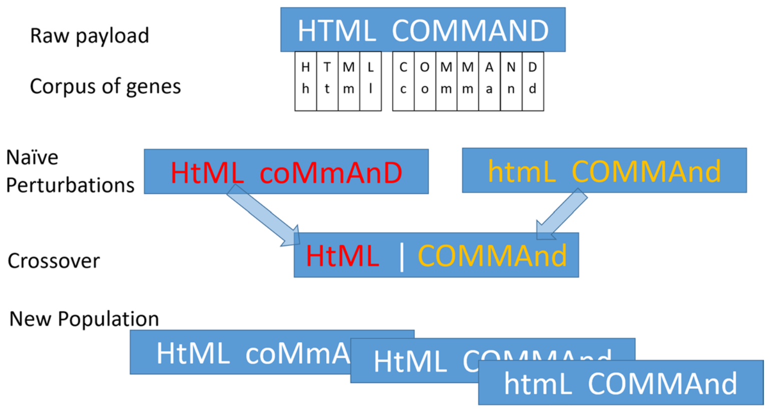


Fig. 1 Stages of the meta-heuristic are shown from the raw payload through a population of perturbed chromosomes

feature of the meta-heuristic is that feasible genes for each allele are pre-defined in a corpus. The five primary steps of the meta-heurisic are:

1. **Initialization:** Various combinations of the original code are presented with random, feasible, perturbations to alleles.
2. **Fitness Function:** Binary cross-entropy of example's truth label w.r.t. the predicted label from the trained surrogate model.
3. **Selection:** Probabilisticly favors most fit solutions.
4. **Crossover:** Cross-point is uniform randomly distributed location in a chromosome.
5. **Mutation:** Random changes to characters are performed probabilisticly; these mutations are constrained to changes in character case. No other changes in characters or commands are allowed per the corpus.

We motivate this approach with an instructional example. Figure 1 shows the raw payload containing an html command in all capital characters. Below the raw payload is the corpus. The corpus contains all functionally equivalent commands for all characters in the raw payload. Next, two payloads are randomly selected from an initial population of naïvely perturbed payloads. As each of these payloads are a potential solution, we call them chromosomes. Each character position is an allele, and the character present in each allele is called the gene. A crossover operation combines the left hand side of one chromosome with the right hand side of another chromosome. If accepted, the result is placed into the new population of chromosomes. Figure 2 presents the ASCII encoding of a real http payload prior to and following perturbation.

3.2 Mathematical formulation

The problem formulation is intended to be solved using the specified meta-heuristic. The fitness, shown in Eq. 2, is the cross-entropy of the example's truth label and the surrogate model's predicted label. Alleles, set i, specify the location of a functional unit of code as shown in Eq. 3. This is the location of an encoded character in our case study. The gene, set j, is the code element chosen for the allele location, shown in Eq. 4. Test examples, or in our case, specific payloads, are annotated with set k as shown in Eq. 5. The decision variable is x, which indicates a particular gene is activated at a location is shown in Eq. 7. The first constraint, shown in Eq. 8, specifies that exactly one gene is activated per allele. The second constraint, shown in Eq. 9, uses the corpus matrix $\mathbf{C}$, ($\mathbf{C}$ is defined in Eq. 6) to specify which genes are feasible at each location in the code. The third constraint, shown in Eq. 10, enforces a binary assignment of genes at alleles.

Fitness Function

$$H(P, Q) = H(P) + D_{\mathrm{KL}}(Q||P) \tag{1}$$

```
POST /dv/login.php HTTP/1.1
Host: 205.174.165.68
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://205.174.165.68/dv/login.php
Cookie: security=low; PHPSESSID=f4depd7v11s9mhhp6nhk1vaiu3
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 130
```

(a) Raw payload prior to perturbation

```
POsT /dv/lOGiN.pHP hTtp/1.1
hOst: 205.174.165.68
USeR-AgeNT: MozilLa/5.0 (x11; LinUX X86_64; Rv:45.0) GECkO/20100101 fIreFOx/45.0
aCcepT: texT/hTML,APpLiCaTIoN/Xhtml+XmL,APPlicaTion/XMl;Q=0.9,*/*;Q=0.8
ACcEpT-lANgUage: en-US,en;q=0.5
aCCEpt-EnCoDiNG: gzip, DEFlate
refEReR: hTTp://205.174.165.68/Dv/lOgiN.php
COoKIE: SecUrity=LOW; pHPSessiD=f4DEpD7V11s9mhHp6NhK1VAIu3
cONnECtIon: KEEP-alIve
CoNtEnt-tyPE: appLICAtiON/x-www-foRm-UrlenCoDEd
ContENt-LENgTh: 130
```

(b) Final payload, as output by the meta-heuristic

Fig. 2 An http payload is shown prior to and after perturbation

$$= -\mathbb{E}_{x \sim P} \log Q(x) \tag{2}$$

Sets

$$i \equiv \text{allele} \quad i \in \{1, 2, ...n\}, \tag{3}$$

$$j \equiv \text{gene} \quad j \in \{1, 2, ...m\}, \tag{4}$$

$$k \equiv \text{test example} \quad k \in \{1, 2, ...o\} \tag{5}$$

$$\mathbf{C} \equiv \text{Feasibility corpus} \quad \mathbf{C} \text{ comprised of binary entries } c_{ij}^{(k)} \tag{6}$$

Decision Variables

$$x_{ij}^{(k)} = \begin{cases} = 1 \text{ if gene } j \text{ is selected for allele } i \text{ on example } k \\ = 0 \text{ otherwise} \end{cases} \tag{7}$$

Constraints

$$\sum_{j \in \mathcal{J}} x_{ij} = 1 \quad \forall\, i \in \mathcal{I} \quad \text{One gene selected per allele} \tag{8}$$

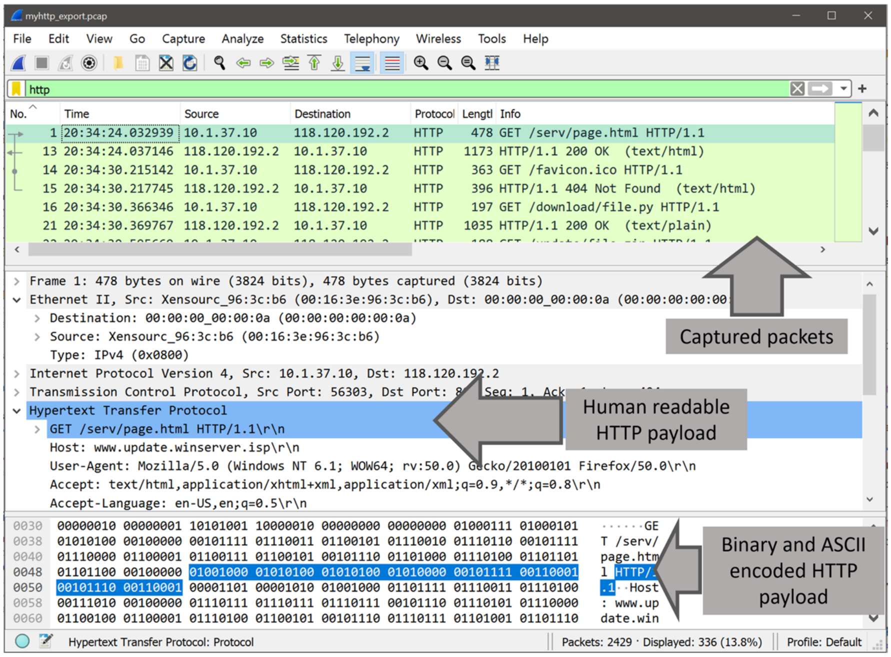
$$x_{ij} \leq c_{ij} \quad \forall\, i, j \quad \text{Allowable genes encoded in } \mathbf{C} \tag{9}$$

$$x_{ij} \in \{0, 1\} \qquad \text{Assignments are binary} \tag{10}$$

The visual depiction in Fig. 3 presents a list of captured packets using the Wireshark packet analysis software. Packet 1, which is selected at the top of the list, follows the http protocol. The content of the packet is shown as human readable code in the center of the figure. The same code is shown in binary encoding on the bottom left, and the ASCII equivalent on the bottom right.

3.3 Adversarial threat model

In terms of the adversarial threat model, this problem formulation supports a grey box attack where the attacker has partial knowledge of the target system. The attacker does not have the same training data used by the NIDS, but does have representative data from a very similar computer network. That is, the attacker's data is roughly from the same distribution as the true NIDS data. There is no knowledge of the NIDS feature extractor, only the assumption that the target


myhttp_export.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Captured packets
Human readable HTTP payload
Binary and ASCII encoded HTTP payload

```
No.  Time             Source          Destination     Protocol Lengtl Info
   1 20:34:24.032939  10.1.37.10      118.120.192.2   HTTP      478 GET /serv/page.html HTTP/1.1
  13 20:34:24.037146  118.120.192.2   10.1.37.10      HTTP     1173 HTTP/1.1 200 OK  (text/html)
  14 20:34:30.215142  10.1.37.10      118.120.192.2   HTTP      363 GET /favicon.ico HTTP/1.1
  15 20:34:30.217745  118.120.192.2   10.1.37.10      HTTP      396 HTTP/1.1 404 Not Found  (text/html)
  16 20:34:30.366346  10.1.37.10      118.120.192.2   HTTP      197 GET /download/file.py HTTP/1.1
  21 20:34:30.369767  118.120.192.2   10.1.37.10      HTTP     1035 HTTP/1.1 200 OK  (text/plain)
```

```
> Frame 1: 478 bytes on wire (3824 bits), 478 bytes captured (3824 bits)
v Ethernet II, Src: Xensourc_96:3c:b6 (00:16:3e:96:3c:b6), Dst: 00:00:00_00:00:0a (00:00:00:00:00
  > Destination: 00:00:00_00:00:0a (00:00:00:00:00:0a)
  > Source: Xensourc_96:3c:b6 (00:16:3e:96:3c:b6)
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 10.1.37.10, Dst: 118.120.192.2
> Transmission Control Protocol, Src Port: 56303, Dst Port: 8  Seq: 1, Ack
v Hypertext Transfer Protocol
  > GET /serv/page.html HTTP/1.1\r\n
    Host: www.update.winserver.isp\r\n
    User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:50.0) Gecko/20100101 Firefox/50.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
```

```
0030  00000010 00000001 10101001 10000010 00000000 00000000 01000111 01000101   ······GE
0038  01010100 00100000 00101111 01110011 01100101 01110010 01110110 00101111   T /serv/
0040  01110000 01100001 01100111 01100101 00101110 01101000 01110100 01101101   page.htm
0048  01101100 00100000 01001000 01010100 01010100 01010000 00101111 00110001   l HTTP/
0050  00101110 00110001 00001101 00001010 01001000 01101111 01110011 01110100   .1··Host
0058  00111010 00100000 01110111 01110111 01110111 00101110 01110101 01110000   : www.up
0060  01100100 01100001 01110100 01100101 00101110 01110111 01101001 01101110   date.win
```

```
Hypertext Transfer Protocol: Protocol    Packets: 2429 · Displayed: 336 (13.8%)    Profile: Default
```

Fig. 3 The Wireshark packet analysis application displays a list of packets, the human readable form, and an encoded form of each packet

NIDS uses raw payload classification. The attacker will generate properly constrained adversarial examples that evade the surrogate NIDS and then transfer these examples to the target network, completing the evasion attack.

3.4 Evaluation metrics

The most costly event for a NIDS is a malicious packet which evades the NIDS and executes its malicious task. For *unperturbed cyber data*, we report the overall classification accuracy as well as the true positive detection rate of malicious packets and denote it as *recall*. Although accuracy can be a biased performance metric if there is significant imbalance [35], the test set of unperturbed examples in this study is nearly balanced, so accuracy is indeed informative. For the *adversarial examples*, only the true positive detection rate is reported and it is denoted as *detection rate*. As compared to above, in this case accuracy would not be an informative metric because true negatives and false negatives are not possible.

3.5 Data strategy

Much of the literature in NIDS research leverages completely outdated datasets such as KDD-CUP and NSL-KDD. These popular datasets reflect 20 year old threat vectors with little relevance today [9]. Further, most NIDS datasets only contain aggregate features of the packets as they are extracted from software such as Snort or Bro. This type of dataset does not lend itself to adversarial attack research since the packet itself must be perturbed in order to conduct the attack, not just the aggregate features [7]. The approach of De Lucia et al [11], Bierbrauer et al [12] and Ali Farrukh et al [13] is to train models with pcap data collected on a network under cyber attack. Our research utilizes the CICIDS dataset [33] rather than the UNSW-NB15 dataset [36] used by De Lucia et al [11]. CICIDS is newer and contains more realistic pcap data, ensuring that models trained from CICIDS are more generalizable than models trained with UNSW-NB15 [12]. CICIDS contains examples of seven attack strategies, *brute force, heartbleed, botnet, denial-of-service, distributed denial-of-service, web attack*, and *infiltration attack*. These attack strategies are implemented in a real network over the course of five days using a variety of popular tools. This results in many diverse combinations of specific attacks. Further, the dataset includes the raw packet files captured in the network during the five day attack period.

Therefore, we postulate that the more generalizable surrogate model would yield adversarial examples that transfer better to test NIDS models. Other datasets containing packet-level information are also viable. CICIDS packets collected Monday through Friday are selected for this research in order to support diversity of both attack and benign packets in the training and test sets. Only user datagram protocol (UDP) and transmission control protocol (TCP) network packets were retained for analysis because alternative communication protocols were either too scarce for supervised learning or otherwise do not contain meaningful attack information in the payloads [12]. Data processing for this experiment is greatly simplified by omitting feature engineering; however, there are several important steps taken to decompose raw packet capture data to structures that can be input to machine learning models for network intrusion detection.

3.6 Generating adversarial examples

The five primary steps in Fig. 4 outline how the various aspects of the methodology fit together. This is additionally represented with pseudocode in Algorithm 1. First, minor pre-processing was performed to prepare the CICIDS payloads for machine learning. The payloads are initially stored as a file containing bytes for each character. These bytes were converted to corresponding integer value, between 0 and 255, per ASCII standards. These integers were then normalized to decimal values between 0 and 1 using the minimax method. Payloads were truncated or padded with zeros for a standard length of 1500 bytes. Header information is not retained. Labels were assigned using the accompanying net flow CICIDS data following the strategy of [12]. This produced 331,868 labelled pcap payloads. 45% of these payloads were randomly sampled without replacement for surrogate training and 45% for NIDS training. 5% of the payloads were set aside to validate the surrogate model, and the remaining 5% to test the NIDS models. Next, a one-dimensional CNN was trained as the surrogate model. This CNN architecture was selected due to its

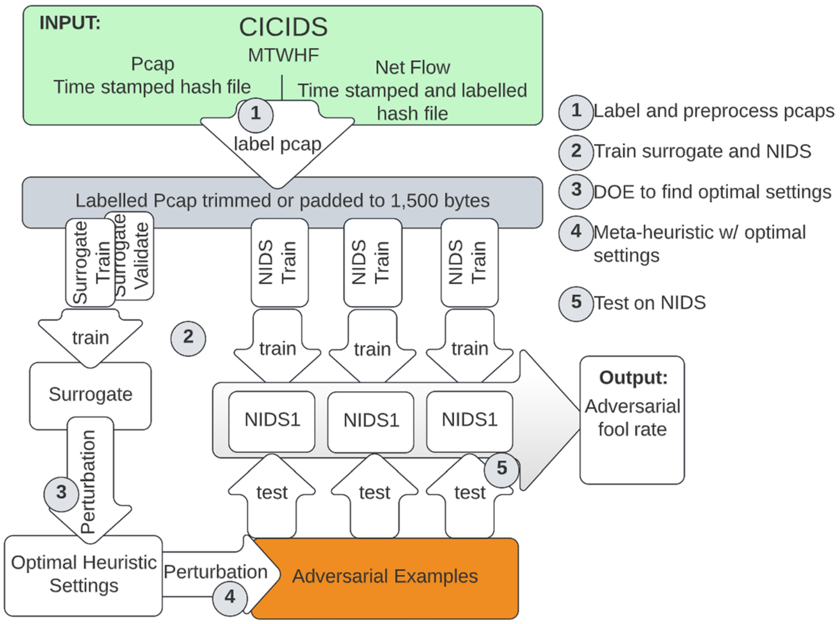


Fig. 4 The methodology requires data pre-processing, training a surrogate model and three NIDS models, generating adversarial examples, and testing the adversarial examples on the NIDS

excellent detection performance in similar studies [11, 12]. It provides the necessary feature representation for capturing the sequential nature of the payload data. As the most generalizable model considered, it is best suited as the surrogate for adversarial example generation. Other models architectures are also viable. Three NIDS architectures were trained, a CNN, a fully-connected neural network (FNN) and an Adaboost classifier. Visual representations of the CNN and FNN are provided in Fig. 5. The Adaboost classifier contained 100 weak classifiers and one node per tree. Step three used a designed experiment where the meta-heuristic settings are optimized for payload size, number of generations, retained population size, initial population size, and percent characters mutated when mutation occurs. Next, the fully tuned meta-heuristic is used in conjunction with the surrogate model to generate adversarial examples of four attack types including infiltration, slowlorris, hulk, and SSH. Finally, the adversarial examples are transferred to the three NIDS and the evasion rates were reported for each combination of NIDS and attack type.

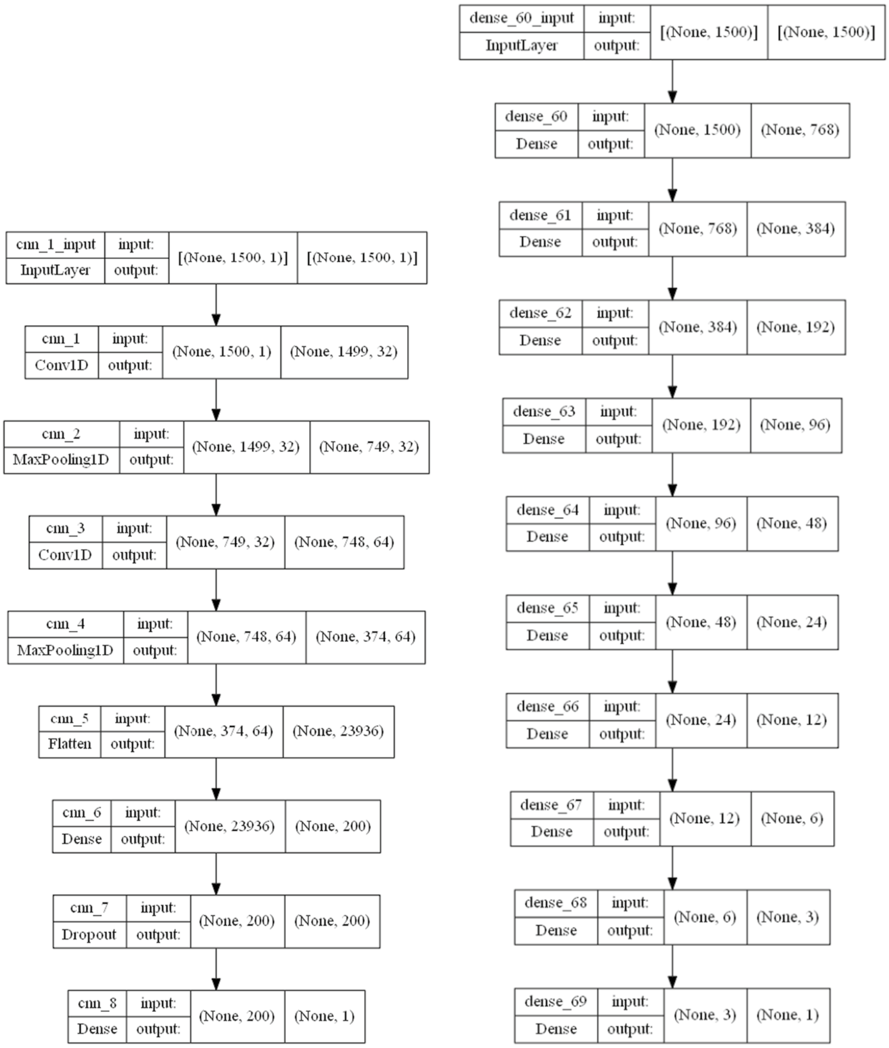
Algorithm 1 Major tasks to generate adversarial examples

```
Input   : Set $\mathcal{X}$ of labelled binary payload packets
Output: Set $\mathcal{A}$ of adversarial examples; summary statistics
$\mathcal{X}$ = EncodePayload ($\mathcal{X}$)
split $\mathcal{X} \rightarrow$
  surrogate train $\mathcal{S}_{train}$; Surrogate test $\mathcal{S}_{test}$; NIDS train $\mathcal{N}_{train}$; NIDS test $\mathcal{N}_{test}$
TrainCNN($\mathcal{S}_{train}$)
for $j \leftarrow$ to num(payloads) do
    encoded packet $\leftarrow$GenerateTensor(payload)
    corpus $\leftarrow$CreateCorpus(encoded payload)
    random AEs$\leftarrow$GeneratePopulation(encoded payload)
    for $k \leftarrow$ to num(Iterations) do
        fitlist$\leftarrow$EvalFitness(AEs)
        AEs$\leftarrow$ReducePopulaton(AEs) parents$\leftarrow$SelectParents(AEs)
        children$\leftarrow$Crossover(parents)
        AEs$\leftarrow$AEs+children
    end
end
Output: List of adversarial examples
```

4 Results and discussion

The surrogate model and three test NIDS models were trained with the surrogate or NIDS training data. All models were then tested with a nearly balanced test set of normal and attack classes. Table 1 reports that all four models offer excellent recall and accuracy on the test set.

A controlled experiment was conducted to determine the best payload size, number of generations, population size, and percent alleles to mutation if mutation occurs. The best number of generations was 5000. The best population size was large. All other factors were not declared statistically significant.


cnn_1_input | InputLayer | input: | output: | [(None, 1500, 1)] | [(None, 1500, 1)]
cnn_1 | Conv1D | input: | output: | (None, 1500, 1) | (None, 1499, 32)
cnn_2 | MaxPooling1D | input: | output: | (None, 1499, 32) | (None, 749, 32)
cnn_3 | Conv1D | input: | output: | (None, 749, 32) | (None, 748, 64)
cnn_4 | MaxPooling1D | input: | output: | (None, 748, 64) | (None, 374, 64)
cnn_5 | Flatten | input: | output: | (None, 374, 64) | (None, 23936)
cnn_6 | Dense | input: | output: | (None, 23936) | (None, 200)
cnn_7 | Dropout | input: | output: | (None, 200) | (None, 200)
cnn_8 | Dense | input: | output: | (None, 200) | (None, 1)
dense_60_input | InputLayer | input: | output: | [(None, 1500)] | [(None, 1500)]
dense_60 | Dense | input: | output: | (None, 1500) | (None, 768)
dense_61 | Dense | input: | output: | (None, 768) | (None, 384)
dense_62 | Dense | input: | output: | (None, 384) | (None, 192)
dense_63 | Dense | input: | output: | (None, 192) | (None, 96)
dense_64 | Dense | input: | output: | (None, 96) | (None, 48)
dense_65 | Dense | input: | output: | (None, 48) | (None, 24)
dense_66 | Dense | input: | output: | (None, 24) | (None, 12)
dense_67 | Dense | input: | output: | (None, 12) | (None, 6)
dense_68 | Dense | input: | output: | (None, 6) | (None, 3)
dense_69 | Dense | input: | output: | (None, 3) | (None, 1)

(a) CNN NIDS and surrogate (b) FNN NIDS

Fig. 5 Architecture of artificial neural network classifiers

Figure 6 plots the loss for the best known solution with a blue line for the first 1000 generations. Child solutions are only accepted into the population if they are strictly better than previous solutions. This strategy guarantees that all new chromosomes in the population are unique and we find this diversity enhances convergence. In particular, Fig. 6 shows that the parents chosen for each generation range in their fitness from nearly zero to over 2.0. Children, shown in blue triangles, are often much more fit than either of the parents, represented with green dots. The suprema and infima of the threshold loss to fool the surrogate are estimated with grey dotted lines. Additional exploratory trials demonstrated that the meta-heuristic tends to improve the best known solution well beyond 5000 generations. 5000 generations is however a good trade off between resources and performance.

4.1 Resulting evasion rates from adversarial examples

Constrained adversarial examples were generated using the tuned meta-heuristic and tested against the surrogate model and three NIDS models. 100 payloads of infiltration, slowloris, hulk, and SSH attack types were selected for perturbation. The runtime for each of these 400 trials was approximately four minutes; it was not practical to increase the number of trials due to computational resource constraints.

Tables 2, 3, 4 and 5 are provided to convey the success of near optimal adversarial examples against each of the NIDS models in comparison to raw payloads, and payloads with a naïve, random perturbation. The randomly perturbed payloads were relatively inexpensive to generate, so 500 were generated and tested. The second column of Tables 2, 3, 4 and 5 report the percent of payloads for which at least one randomly perturbed variation fooled the classifier. The evasion rates for raw payloads, randomly perturbed payloads, and near optimally perturbed payloads are also visually displayed in Fig. 7 where they are color coded by attack type. The initial hypothesis was that for any combination of attack type and model, the evasion rate would increase as we move from raw, to randomly perturbed payloads, to near optimally perturbed payloads. The results convey that this hypothesis is the general trend, but not the rule.

Only one raw infiltration packet fooled the surrogate model, and that packet also fooled the Adaboost NIDS. 6% of slowloris raw packets fooled the CNN surrogate model; however, none of those specific packets fooled the CNN NIDS. 43% of raw slowloris packets fooled the FNN classifier. All classifiers appeared accurate against raw hulk and slowloris attacks, with no raw payloads fooling any classifier. Raw slowloris packets demonstrated, by far, the greatest average loss when tested on the surrogate model and raw slowloris packets were misclassified at the highest rate against each classifier. This may be due to the great diversity of the ASCII content in slowloris payloads. Although slowloris payloads are well represented in the training data, the patterns could be more difficult to capture with the chosen model architectures. Packets that fooled the surrogate model did not always fool the NIDS. One raw slowloris packet which was misclassified by the CNN surrogate with a loss of 4.94 surprisingly did not fool the NIDS of the same architecture.

Table 1 Test accuracy and recall is measured on the surrogate model and NIDS models using data sequestered prior to training

	Test accuracy	Test recall
CNN Surrogate	0.995	0.995
CNN NIDS	0.999	0.999
FNN NIDS	0.994	0.992
Adaboost NIDS	0.987	0.992

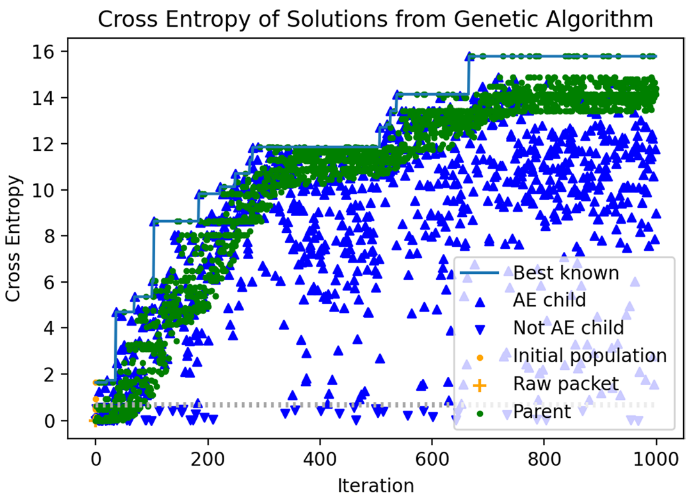


Fig. 6 The cross-entropy of chromosomes is shown for the first 1,000 generations of the meta-heuristic is it perturbs the payload of a hulk attack. The suprema and infima of threshold cross-entropy to fool the surrogate are shown with a grey dotted line

Random, naïve, perturbations were generated for 100 payloads of each attack type. Figure 7 shows randomly perturbed packets fooled the surrogate and NIDS models at a much higher rate than raw packets. Slowloris packets fooled classifier more than any other packet type. At least one of the 500 randomly perturbed solutions fooled the surrogate model for 61% of the 100 slowloris packets. 56% of packets yielded at least one random solution that fooled the CNN NIDS and FNN NIDS while 55% of slowloris packets yielded at least one random solution that fooled the Adaboost NIDS. Fewer payloads fooled models with random perturbations for infiltration attacks and hulk attacks. Zero randomly perturbed payloads of SSH attacks fooled any NIDS. Only one randomly perturbed SSH payload fooled the surrogate. The high classification accuracy on SSH packets may be because all SSH payloads share *nearly identical* payloads. Classifiers may learn the class manifold precisely and, therefore, detect perturbations at a high rate. The Adaboost NIDS was the most robust model to random perturbations for all attack types.

Table 2 Classification performance for infiltration raw packets, best performing randomly perturbed packets, and best performing near optimal packets against the surrogate and NIDS models

Infiltration			
	Raw	Random	Near optimal
% Packets Evade Surrogate	1%	4%	81%
Avg Loss Surrogate	3.88E-02	2.57E-01	3.73E+00
% Packets Evade CNN NIDS	0%	6%	11%
% Packets Evade FNN NIDS	0%	6%	2%
% Packets Evade Aboost NIDS	1%	3%	3%

Table 3 Classification performance for slowloris raw packets, best performing randomly perturbed packets, and best performing near optimal packets against the surrogate and NIDS models

Slowloris			
	Raw	Random	Near optimal
% Packets Evade Surrogate	6%	61%	100%
Avg Loss Surrogate	3.03E-01	4.93E+00	1.44E+01
% Packets Evade CNN NIDS	5%	56%	69%
% Packets Evade FNN NIDS	43%	56%	53%
% Packets Evade Aaboost NIDS	12%	55%	52%

Table 4 Classification performance for hulk raw packets, best performing randomly perturbed packets, and best performing near optimal packets against the surrogate and NIDS models

Hulk			
	Raw	Random	Near optimal
% Packets Evade Surrogate	0%	13%	100%
Avg Loss Surrogate	4.91E-06	3.77E-01	8.34E+00
% Packets Evade CNN NIDS	0%	12%	37%
% Packets Evade FNN NIDS	0%	12%	0%
% Packets Evade Aaboost NIDS	0%	0%	0%

Table 5 Classification performance for SSH raw packets, best performing randomly perturbed packets, and best performing near optimal packets against the surrogate and NIDS models

SSH			
	Raw	Random	Near optimal
% Packets Evade Surrogate	0%	1%	37%
Avg Loss Surrogate	1.68E-06	3.11E-02	1.38E+00
% Packets Evade CNN NIDS	0%	0%	0%
% Packets Evade FNN NIDS	0%	0%	0%
% Packets Evade Aaboost NIDS	0%	0%	0%

Near optimal perturbations were generated for 100 payloads of each attack type. The top 50 observed solutions, with respect to the surrogate model, were retained for each payload and tested against the NIDS models. These evasion rates are reported in Tables 2, 3, 4 and 5 and plotted in Fig. 7. The best known near optimal perturbation fooled the surrogate model with 81% of infiltration attacks, 100% of slowloris attacks, 100% if hulk attacks, and 37% of SSH attacks. Near optimal examples transferred to NIDS classifiers with varying degrees of success. Despite slowlowis and hulk perturbations fooling the surrogate model in all instances, only 69% and 37% of

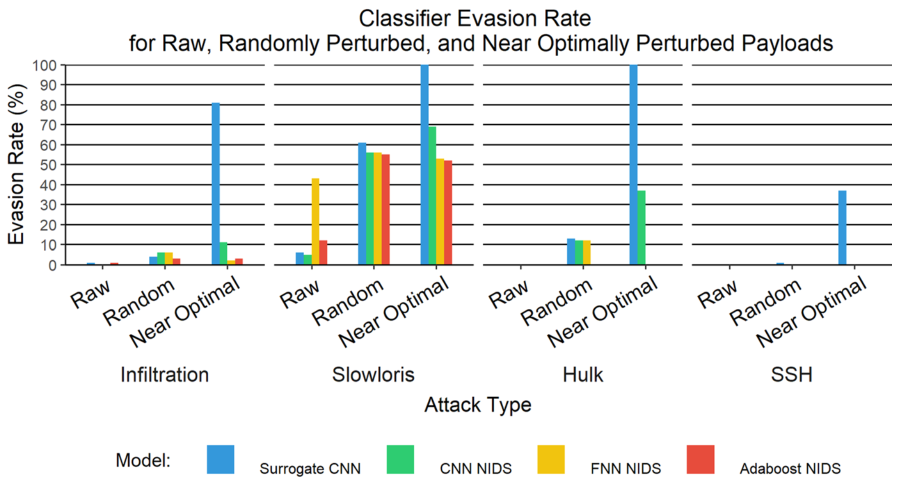


Fig. 7 Classification performance is presented for raw packets, best performing randomly perturbed packets, and best performing near optimal packets against the surrogate and NIDS models

payloads produced any perturbation that fooled the NIDS of the same CNN architecture, respectively. Fool rates against the FNN and Adaboost NIDS were lower across the board. Only payloads of slowloris fooled all three NIDS with a high degree of success, with fool rates against CNN, FNN, and Adaboost at 69%, 53%, and 52%. All near optimal SSH payloads failed to fool each NIDS. Many adversarial examples that appeared very fit to the meta-heuristic transferred well against target NIDS classifiers. Among the optimally perturbed payloads of slowloris, there is a 63% correlation between CNN surrogate loss and the rate at which perturbations of that packet fooled the CNN NIDS. The correlation between surrogate loss FNN fool rate is 77%, and the correlation between surrogate loss and Adaboost fool rate is 54%. There may also be a connection between number of characters in the payload and the rate of fooling each NIDS. The correlation is 0.59, 0.77, and 0.32 for the CNN, FNN, and Adboost NIDS. Average payload lengths were similar across attack types. These findings confirm the results of the experimental design that perturbing larger payloads generates stronger constrained adversarial examples.

The initial hypothesis of this work was that near optimally perturbed payloads generated from a CNN surrogate would evade any NIDS at a higher rate than random perturbations. The empirical results are more nuanced. The advantage of using the near optimal perturbations is best seen when the adversarial examples are tested against a NIDS of common architecture. It is evident by *comparing % packets that fool the surrogate* in Tables 2, 3, 4 and 5 that near optimal perturbations are the clear choice if the target NIDS is known (or highly expected) to be a CNN and the surrogate is also a CNN. Random perturbations are effective against some NIDS because they are extremely inexpensive to generate and it is only necessary for one example to fool the NIDS. We characterize the random perturbations as a *brute force* strategy that provides value to the attacker and sometimes outperforms the near optimal perturbations. In particular, we identify at least one randomly perturbed hulk payload to fool the FNN NIDS for 12% of payloads, but the near optimal perturbations of hulk payloads are not observed to defeat the FNN NIDS. Among other classifiers and attack types, the near optimal payload fools at rates that are similar or much better than random perturbations. Ocular inspection of the ASCII suggests that slowloris payloads are too varied to be well learned by any classifier. The meta-heuristic, therefore, stands on weak footing to provide a generalized constrained adversarial example that is “best" in all settings. Evasion rates of near optimal perturbations slightly underperform random perturbations for the highly varied slowloris attacks. On the other extreme, there is no variance among the ASCII content of hulk payloads and minimal variance for SSH. While the surrogate model learns the hulk and SSH attack patterns with high confidence, so do the NIDS models. Near optimal perturbation of hulk attacks worked well against the CNN NIDS, but did not transfer with any success to the FNN and Adaboost models. Despite some success against the surrogate, no SSH packet fooled any NIDS.

What is responsible for the underwhelming transferability of fooling packets from the CNN surrogate to other models? Firstly, the meta-heuristic always accepts perturbations that increase loss with respect to the surrogate. This is shown by the monotonically increasing *best known* solution in Fig. 6. It also follows that constrained adversarial examples which fool the CNN surrogate would often fool the CNN NIDS because they share a common architecture and joint distribution of training data. Existing research found that adversarial examples exist in low probability regions of the feature space and are consistent across model architectures [6, 23]. In our case, the CNN architecture is superior to the FNN and Adaboost models according to all metrics. Only the CNN captures semantic and temporal information in the payloads. The pattern of malicious attacks learned by the CNN is possibly more nuanced than the other models. Perturbations generated with the CNN surrogate drive the example off the learned manifold of the CNN models, but not the learned manifold of the FNN and Adaboost models. In general, we expect an attack derived from any surrogate model would transfer most effectively to a target NIDS of the same architecture as the surrogate. The particulars of transferability are difficult to understand because the surrogate and each NIDS are black box models and because raw packet detection is state of the art [11, 12]. This work represents the first known study of optimally constrained perturbations of malicious packets, as opposed to net-flow [4, 5]. Extensive efforts in formal methods and controlled experimentation could someday uncover the key to transferability of adversarial examples in the cyber domain.

5 Conclusion

Adversarial machine learning based attacks have been used to degrade computer vision classifiers; however, it is much more difficult to conduct an adversarial evasion attack on cyber systems. This work is the first to formulate and solve the constrained optimization problem to generate an end-to-end adversarial attack in the cyber domain, specifically network intrusion detection.

The performance of constrained adversarial examples generated by substituting individual letters is promising, albeit imperfect. Many alternative classes of substitution are possible assuming a corpus of equivalent alternative units is available. The choice of substitutable units should be domain specific and may include objects such as commands, lines of code, or changing language.

This work exposes a previously unpublished vulnerability of AI-based NIDS; however, it also motivates work towards a solution. We propose that meta-learning strategies that combine generative modeling with ensembling techniques may be fundamental to ensure that next generation NIDS are robust and resilient against conventional cyber attacks and adversarial machine learning attacks. Future work will investigate out-of-distribution (novelty) detection and generation for NIDS, as well as NIDS that adapt to changing environments.

Acknowledgements Special thanks is given to Dr. Elie Alhajjar and Dr. John Pavlik for discussing experimental results throughout the research. This work was supported in part by the U.S. Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory under Support Agreement No. USMA21050, the U.S. Army DEVCOM C5ISR Center under Support Agreement No. USMA21056, and the U.S. Air Force Research Laboratory under Support Agreement No. USMA2226. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the Air Force Institute of Technology, U.S. Military Academy, U.S. Air Force, U.S. Army, Department of Defense, or U.S. Government.

Data Availability Raw data is available by request from a third party source, Canadian Institute for Cybersecurity, at the following location: https://www.unb.ca/cic/datasets/ids-2017.html

References

1. Stallings, W., Brown, L., Bauer, M.D., Howard, M.: Computer Security: Principles and Practice. Pearson Education, Upper Saddle River (2012)
2. Annarelli, A., Nonino, F., Palombi, G.: Understanding the management of cyber resilient systems. Comput. Ind. Eng. **149**, 106829 (2020). https://doi.org/10.1016/j.cie.2020.106829
3. Garnaev, A., Baykal-Gursoy, M., Vincent Poor, H.: How to deal with an intelligent adversary. Comput. Ind. Eng. **90**, 352–360 (2015). https://doi.org/10.1016/j.cie.2015.10.001
4. Alhajjar, E., Maxwell, P., Bastian, N.: Adversarial machine learning in network intrusion detection systems. Expert Syst. Appl. **186**, 115782 (2021). https://doi.org/10.1016/j.eswa.2021.115782
5. Schneider, M., Aspinall, D., Bastian, N.: Evaluating model robustness to adversarial samples in network intrusion detection. In: Proceedings of the 2021 IEEE International Conference on Big Data, IEEE pp. 3343– 3352 (2021)
6. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
7. Rosenberg, I., Shabtai, A., Elovici, Y., Rokach, L.: Adversarial machine learning attacks and defense methods in the cyber security domain. ACM Comput. Surv. (CSUR) **54**(5), 1–36 (2021)
8. Cerf, V., Kahn, R.: A protocol for packet network intercommunication. IEEE Trans. Commun. **22**(5), 637–648 (1974). https://doi.org/10.1109/TCOM.1974.1092259
9. Hindy, H., Brosset, D., Bayne, E., Seeam, A.K., Tachtatzis, C., Atkinson, R., Bellekens, X.: A taxonomy of network threats and the effect of current datasets on intrusion detection systems. IEEE Access **8**, 104650–104675 (2020). https://doi.org/10.1109/ACCESS.2020.3000179
10. Apruzzese, G., Andreolini, M., Ferretti, L., Marchetti, M., Colajanni, M.: Modeling realistic adversarial attacks against network intrusion detection systems. Digit. Threats Res. Pract. (DTRAP) **3**(3), 1–19 (2022)
11. De Lucia, M.J., Maxwell, P.E., Bastian, N.D., Swami, A., Jalaian, B., Leslie, N.: Machine learning raw network traffic detection. SPIE (2021). https://doi.org/10.1117/12.2586114
12. Bierbrauer, D.A., De Lucia, M., Reddy, K., Maxwell, P., Bastian, N.D.: Transfer learning for raw network traffic detection. Expert Syst. Appl. **211**(118641), 1 (2022)

13. Farrukh, Y.A., Khan, I., Wali, S., Bierbrauer, D., Pavlik, J.A., Bastian, N.D.: Payload-Byte: A Tool for Extracting and Labeling Packet Capture Files of Modern Network Intrusion Detection Datasets. In: Proceedings of the 9th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT2022) (2022)
14. Applegate, S.D.: The dawn of kinetic cyber. In: 2013 5th International Conference on Cyber Conflict (CYCON 2013), IEEE pp. 1– 15 (2013)
15. Anderson, J.P.: Computer security technology planning study-Vol 1. James P. Anderson Co. (1972)
16. Anderson, J.: Computer security threat monitoring and surveillance. James P. Anderson Co. (1980)
17. Bejtlich, R.: The Practice of Network Security Monitoring: Understanding Incident Detection and Response. No Starch Press, San Francisco (2013)
18. Denning, D., Neumann, P.G.: Requirements and Model for IDES-a Real-Time Intrusion-Detection Expert System. SRI International, Menlo Park (1985)
19. Cheng, T.-H., Lin, Y.-D., Lai, Y.-C., Lin, P.-C.: Evasion techniques: sneaking through your intrusion detection/prevention systems. IEEE Commun. Surv. Tutorials **14**(4), 1011–1020 (2011)
20. Chernikova, A., Oprea, A.: Fence: feasible evasion attacks on neural networks in constrained environments. ACM Trans. Privacy Sec. **25**(4), 1–34 (2022)
21. Kuppa, A., Grzonkowski, S., Asghar, M.R., Le-Khac, N.-A.: Black box attacks on deep anomaly detectors. In: Proceedings of the 14th International Conference on Availability, Reliability and Security. ARES '19. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3339252.3339266
22. Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., Roli, F.: Evasion attacks against machine learning at test time. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, pp. 387– 402 (2013)
23. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: 3rd International Conference on Learning Representations, ICLR 2015 (2015)
24. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: 2016 IEEE European Symposium on Security and Privacy (EuroS &P), pp. 372– 387 (2016). https://doi.org/10.1109/EuroSP.2016.36
25. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (sp), IEEE, pp. 39– 57 (2017)
26. Chollet, F.: Deep Learning with Python. Simon and Schuster, New York (2021)
27. Rezaei, S., Liu, X.: Deep learning for encrypted traffic classification: an overview. IEEE Commun. Mag. **57**(5), 76–81 (2019)
28. Hernández-Pereira, E., Suárez-Romero, J.A., Fontenla-Romero, O., Alonso-Betanzos, A.: Conversion methods for symbolic features: a comparison applied to an intrusion detection problem. Expert Syst. Appl. **36**(7), 10612–10617 (2009). https://doi.org/10.1016/j.eswa.2009.02.054
29. Maxwell, P., Alhajjar, E., Bastian, N.D.: Intelligent feature engineering for cybersecurity. In: 2019 IEEE International Conference on Big Data (Big Data), IEEE, pp. 5005– 5011 (2019)
30. Chae, H.S., Jo, B.O., Choi, S.H., Park, T.K.: Feature selection for intrusion detection using NSL-KDD. Recent Adv. Comput. Sci. **20132**, 184–187 (2013)
31. Kloft, M., Brefeld, U., Düessel, P., Gehl, C., Laskov, P.: Automatic feature selection for anomaly detection. In: Proceedings of the 1st ACM Workshop on Workshop on AISec, pp. 71– 76 (2008)
32. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the kdd cup 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1– 6 (2009). https://doi.org/10.1109/CISDA.2009.5356528
33. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp **1**, 108–116 (2018)
34. Raggett, D., Le Hors, A., Jacobs, I., et al.: Html 4.01 specification. W3C recommendation **24** (1999)
35. Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., Rieck, K.: Dos and don'ts of machine learning in computer security. In: Proceedings of 31st USENIX Security Symposium, pp. 3971– 3988 (2022)
36. Moustafa, N., Slay, J.: Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS), pp. 1– 6 (2015). https://doi.org/10.1109/MilCIS.2015.7348942

Authors and Affiliations

Marc Chalé[1,2] · Bruce Cox[1] · Jeffery Weir[1] · Nathaniel D. Bastian[1,2]

Bruce Cox
bruce.cox@afit.edu

Jeffery Weir
jeffery.weir@afit.edu

Nathaniel D. Bastian
nathaniel.bastian@westpoint.edu

1 Department of Operational Sciences, Air Force Institute of Technology, 2950 Hobson Way, Wright-Patterson AFB, OH 45433, USA

2 Army Cyber Institute, United States Military Academy, New South Post Road, West Point, NY 10996, USA