2022

Using deep learning to detect digitally encoded DNA trigger for Trojan malware in Bio-Cyber attacks

M. S. Islam

S. Ivanov

H. Awan

J. Drohan

Sasitharan Balasubramaniam

See next page for additional authors

Authors

M. S. Islam, S. Ivanov, H. Awan, J. Drohan, Sasitharan Balasubramaniam, L. Coffey, Srivatsan Kidambi, and W. Sri-saan

scientific reports

Check for updates

OPEN

Using deep learning to detect digitally encoded DNA trigger for Trojan malware in Bio-Cyber attacks

M. S. Islam[1]✉, S. Ivanov[1], H. Awan[4], J. Drohan[3], S. Balasubramaniam[2], L. Coffey[3], S. Kidambi[5] & W. Sri-saan[2]

This article uses Deep Learning technologies to safeguard DNA sequencing against Bio-Cyber attacks. We consider a hybrid attack scenario where the payload is encoded into a DNA sequence to activate a Trojan malware implanted in a software tool used in the sequencing pipeline in order to allow the perpetrators to gain control over the resources used in that pipeline during sequence analysis. The scenario considered in the paper is based on perpetrators submitting synthetically engineered DNA samples that contain digitally encoded IP address and port number of the perpetrator's machine in the DNA. Genetic analysis of the sample's DNA will decode the address that is used by the software Trojan malware to activate and trigger a remote connection. This approach can open up to multiple perpetrators to create connections to hijack the DNA sequencing pipeline. As a way of hiding the data, the perpetrators can avoid detection by encoding the address to maximise similarity with genuine DNAs, which we showed previously. However, in this paper we show how Deep Learning can be used to successfully detect and identify the trigger encoded data, in order to protect a DNA sequencing pipeline from Trojan attacks. The result shows nearly up to 100% accuracy in detection in such a novel Trojan attack scenario even after applying fragmentation encryption and steganography on the encoded trigger data. In addition, feasibility of designing and synthesizing encoded DNA for such Trojan payloads is validated by a wet lab experiment.

Genetic sequencing has become an essential tool for analyzing numerous DNAs that are used in the field of medicine, agriculture, as well as forensics. Numerous systems have been developed over the years to increase accuracy, such as throughput shot-gun sequencing technologies (e.g., vector-borne pathogens detection in blood[1], food authentication and food fraud detection[2], or even molecular data to be transported through artificial biological networks[3,4]). Recent developments in sequencing technology have also been miniaturized to allow mobile sequencing and one example is the *Minion*[5]. We have recently witnessed the importance of timely sequencing from oral samples due to the COVID-19 pandemic, which continues to apply pressure on the health care system[6]. The clear benefits of expanded COVID-19 testing[7] calls for an expansion of the existing testing (e.g. STEMI[8]) approaches. The importance of sequencing can also be seen in detecting and tracking mutations in other types of infectious diseases, where examples include Lassa Fever[9] or other prevalent pathogens[10], such as seasonal flu[11] or bacterial infections where new strains resistant to existing antibiotics can be identified[12,13].

As the genetic sequencing will inevitably introduce additional pressure on the already overburdened healthcare services, it is likely that the genetic analysis may be outsourced to private sequencing services. Similar approaches have already been successfully adopted for other testing programmes (e.g. Cervical Screening Programme in Ireland[14]). The services will act as an on-demand genetic-testing infrastructure that receives and analyses samples on behalf of the hospitals, medical practices and other healthcare organizations. While this approach alleviates pressure on the healthcare system, the system is vulnerable to Bio-Cyber Hacking[15].

[1]VistaMilk Research Centre, Walton Institute, South East Technological University, Waterford, Ireland. [2]School of Computing, University of Nebraska-Lincoln, Lincoln, NE, USA. [3]Pharmaceutical and Molecular Biotechnology Research Centre, South East Technological University, Waterford, Ireland. [4]Munster Technological University, Cork, Ireland. [5]Department of Chemical and Biomolecular Engineering, University of Nebraska-Lincoln, Lincoln, NE, USA. ✉email: sibleeislam@gmail.com

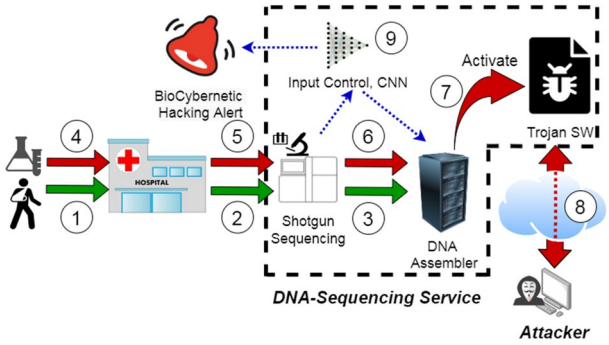

Figure 1. Hybrid Trojan Bio-Cyber Hacking Attack. Steps 1–3 indicate a typical genetic sequencing operation for patients. Steps 4–6 indicate a situation where a hacker has embedded their IP address and Port number into a DNA that will trigger a remote connection from a Trojan-horse infected software tool leading to a connection to the attacker in Step 8. Our proposed approach utilizes Deep-Learning to detect Trojan payload in digital data using encoded into DNA strands that can prevent the attack. (The image is drawn using draw.io).

Our definition of Bio-Cyber Hacking refers to an attack that is hybrid between ICT systems and biological mediums. From the ICT system side, we assume that the pipeline of the sequencing service uses a DNA-analysis toolbox infected with Trojan Software. Malware, such as a Trojan, can be implanted at the API levels[16], within mobile software[17] and even in machine learning models[18]. Trojans can also be implanted into hardwares[19–21] of computers, as well as IoT devices[22]. In our scenario, the Trojan contains an empty slot for the IP address and port number for remote connections to an external machine. On the biological side, an attacker encodes the IP address and port number into DNA strands. Using DNA-steganography, the attacker devises synthetic DNA that contains the payload and still maintains resemblance with natural DNAs. We will explain the process in Fig. 1, where we will first explain a sequencing process for normal DNA (steps 1–3) and then explain a hacking situation (steps 4–8). In (Fig. 1 (1–2)), the service uses one of the state-of-the-art sequencing techniques, e.g. shotgun sequencing, to analyze DNA materials extracted from each of the samples (e.g. *E. coli* Plasmid and Cellular DNAs). The machine randomly splits DNA molecules into multiple fragments or reads of a predefined length, then it concurrently sequences each read to establish its nucleotide structure. The original DNA is then assembled from the reads (Fig. 1 (3)). This is a computationally complex process that often involves the use of dedicated resources, often called DNA-sequencing pipeline[23]. Let us now consider an attack situation. Initially the Trojan remains dormant, while the toolbox performs the legitimate DNA-analysis. The trigger sample is collected by the hospital (i.e., by swabbing) and sends the samples to the sequencing service for analysis (Fig. 1 (4)). The samples are then analyzed by the sequencer (Fig. 1 (5)). There the sample is fragmented, sequenced and assembled (Fig. 1 (6)). During the assembly, the DNA toolbox retracts the payload and wakes the Trojan (Fig. 1 (7)), and this happens is when the DNA sample that contains the web address and port number of a remote server controlled by the attacker is detected by the digital DNA data that is passed from the sequencer to the computer that contains the DNA-analysis toolbox infected with the Trojan. The Trojan establishes a connection with the remote server (Fig. 1 (8)), where the Trojan either opens a cyber backdoor, transfers files, or executes commands from the attacker. Either of these actions presents a substantial threat to the integrity of DNA-analysis and patient diagnostics.

In this article, we develop a solution that is complementary to the existing general-purpose techniques. The solution builds on our previous work that only focused on steganography techniques to hide IP address and port numbers into DNA strands[24] and investigates the use of input control (Fig. 1 (9)) as a countermeasure to the Trojan Bio-Cyber attacks. The input control is an intermediary between the DNA-sequencer and the pipeline. With the help of a specially designed and trained Deep Neural Network (DNN), the control assesses each DNA read generated by the sequencer to establish whether the read comes from a trigger sample. Absence of suspicious reads assures cybersafety of further DNA-assembly, but a detection of a trigger sample terminates its further processing. This prevents activation of the Trojan software and limits the pipeline's exposure. In recent times, there is a lot of interest in the use of deep learning for malware detection[25,26,27]. Deep learning techniques are also applied to Trojan detection[19,28] in conventional cyber attacks. To the best of our knowledge exploiting the Buffer overflow vulnerability in DNA sequencing pipeline using a specially designed DNA was first demonstrated in[15]. To detect DNA sequences containing the payloads for buffer overflow exploit, we proposed Case-based Reasoning (CBR)[24], where we found that such payloads results in sequences that are quite different from the DNA sequences which are naturally available. Moreover, we investigated the recovery rate of the DNA sequences containing the malicious payloads that was inserted into bacteria after spraying them on various types of surfaces. In another article we have shown how a Trojan Attack is possible in a DNA sequencing pipeline[29], but the possibility of creating such sequence was not validated by conducting any wet lab experiments or detecting the trigger, which is what we have investigated in this work. In both of our previous work, we did not consider keeping the natural appearance of the DNA while designing the payload to make the detection harder. In this article we improved our algorithm of making the payload harder to detect and proposed a CNN based detection technique.

Figure 2 illustrates the construction of the payload that is embedded into a DNA sequence, and in this specific example we focus on a bacterial plasmid. We re-designed the construction of the payloads to make it similar to

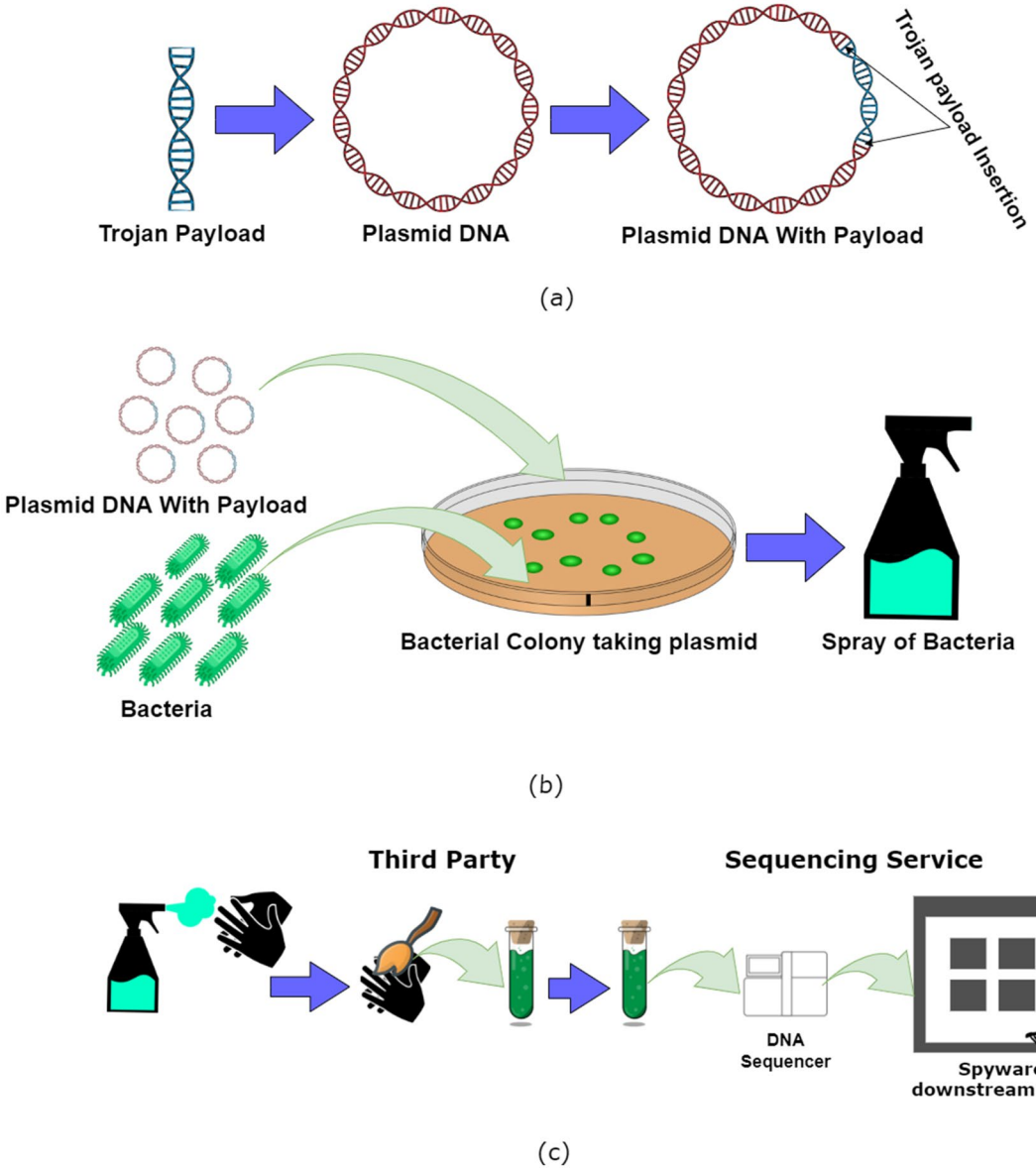

Figure 2. Trojan Bio-Cyber Hacking: Payload Preparation and Attack Scenario example using DNA plasmids. (**a**) A Trojan payload (using encryption and steganography) is encoded into a DNA sequence which is developed and inserted into the plasmid DNA. Antibiotic resistant gene sequences will also be inserted into the plasmid DNA in a similar way. (**b**) The DNA plasmid and the bacteria will be transferred into rich media so that the bacteria can uptake these plasmids[24]. Bacteria resistant to the antibiotic will survive and be transferred into a spray. (**c**) The bacteria can now be sprayed on hands or gloves and provided to a third party which can collect samples (from hand or gloves). The third party will then send these samples to the company for sequencing. When the sequence will be processed by the tools having the Trojan, it will be activated to perform the malicious activities. (The image is drawn using draw.io).

a natural DNA sequence in order to increase detection difficulty. The construction of the DNA is based on the sequence used in[29].

Methods

In this section, various terms used in the article will be defined and then the steganography techniques will be described, which is applied on the payload used for malicious activities as a means of secrecy of operations. Following that we will describe the injection method of the payload into a host DNA. This is followed up with the description of the deep learning model proposed as a detection method to counter the Trojan attacks.

Trojan payload. The payload DNA for triggering the Trojan malware will be encoded into a DNA sequence and will be referred simply as 'payload' in the rest of the article. The payload will be hidden inside a longer DNA string, which is considered as 'host DNA'. In order to prevent detection, the content of the payload will be first

divided into smaller parts and then encoded into smaller DNA sequences, which will be called as 'fragments' and this process will be known as 'fragmentation'. The fragments can be inserted in a random order and at random positions of the host DNA. Substitution technique, i.e., replacing a nucleotide of the host DNA with a nucleotide of the payload DNA or fragment DNA (if fragmentation is applied), is considered as an insertion technique. 'Retention' is the process of skipping a particular number of nucleotide positions of the host DNA to substitute by the nucleotide of the encoded/fragment DNA while performing the insertion. Both encryption and retention will be considered when steganography is applied, where the encryption process will be performed before the retention. The details of the processes including encryption will be described in the subsequent sections of the article. After completing the insertion process, the obtained DNA string is considered as the 'resultant DNA'.

In general the host DNA string will be significantly larger compared to the encoded DNA for the payload. Therefore, the Trojan software needs to perform processes such as identifying those fragments, applying decryption and decoding techniques before merging and rearranging them in order to activate the malware process to trigger the hacking operation. As a result, the Trojan software should apply these processes to integrate the substrings to create the full DNA string as an additional task beside performing its normal functional tasks. The caveat of such an approach is that the computational complexity will be significantly high and the Trojan software might be under suspicion straight away as it will take significantly higher time and consume higher memory. To prevent this suspicious behaviour, the Trojan software will need to efficiently determine the location to perform decryption and decoding and this will be achieved through 'tags'. The tags are tiny snippets of chosen DNA sequences that indicate the start and end of the fragments that will be searched by the Trojan software, and we refer to this process as 'tagging'.

One of the critical challenges in packaging the Trojan payload is the delivery system which can act as the carrier for the DNA materials. To this extent, liposomes and lipid-based nanoparticles have been extensively used for targeted gene delivery to various coordinates. Liposomes, also referred to as vesicles, are extremely versatile carriers that have been studied and utilized extensively for drug delivery applications including gene and mRNA due to their ease of creation, large protective hydrophilic inner cavity for encapsulation, high degree of freedom for exterior customization, and controllable drug release kinetics. Recent success of mRNA vaccines for COVID is attributed to such lipid based platforms as a delivery vehicle for mRNA. These can be extended to packaging the Trojan payload to enhance the stability of the DNA and also establish targeting capabilities to target specific locations for Cyber-hacking. Furthermore, there are innovative and robust platforms that can integrate these lipid nanoparticles embedded within substrate and matrix based on polymer based films that can control the release of these Trojan payloads and extend their stability[30]. Also this platform can also facilitate hiding these Trojan payloads from detection and embed multiple payloads. This platform provides ways to transport the Trojan Payload into the targeted region beyond security measures by embedding them into entities including clothes, skins, pens or papers as examples.

Steganography. In this article we consider a scenario where the perpetrator encodes the attack details (i.e., web address and port number) into a DNA, which are used as a trigger sample. To avoid the detection of this sample and cover the identity of the attacker, the encoding uses an extension of the DNA Steganography technique proposed in[29].

The extended steganography technique proposed in this article has five steps and this includes *fragmentation*, *encryption*, *encoding*, *tagging* and *retention*. First, the web address and port number injected into the DNA are divided into fragments of a predefined length. Since each fragment is shorter than the original address, this will increase the difficulty in the detection process post injection. Next, the binary of the fragment is XOR-encrypted using a predefined key. This is followed up by encoding with four basic nucleotides, i.e., "00" bit-pairs are encoded as "A", "01" as "C", "10" as "G" and "11" as "T". The ACTG-encoding (represent four nucleic acids, which are Adenine, Cytosine, Thymine and Guanine) is enclosed in the nucleotide brackets where the ACTG tags mark the beginning and the end of the injection within the DNA. These tags are selected so that the natural DNAs are unlikely to include both the start and end tags separated by a number of nucleotides that is required to encode a malicious fragment. The tags need to be sufficiently short in order to reduce the footprint of the injected fragment as well as increase the similarity with the host DNA and avoid detection. Finally, the retention stage expands the result of the tagging using the symbol "*" (see Eq. 1). The expansion is performed in a way that a predefined number of retention symbols is inserted between each of the two consecutive nucleotides. The positions of the retention symbol determine that the nucleotides of the host DNA will remain unchanged as a result of the malicious code injection. Thus, for a retention number equal to 2, only the first of each 3 consecutive nucleotides of the host DNA will be replaced. The second and third nucleotides will remain unchanged. This is done to increase the similarity between DNA of the trigger sample and the host DNA.

Injection methods. In this article we consider substitution as the preferred method of injecting the Trojan payload into the host DNA. Consider the case when the Trojan payload d_{load} (with encoded nucleotides and retention symbol "*" after applying encryption and steganography as described above) is injected into the DNA, d_{host}, at position i. The result of the injection will present a nucleotide string inj, having the length equal to the length of d_{load}. The length of the d_{host} and d_{load} strings is determined by a function called len, which reflects the number of characters in both strings. The nucleotide at position $j \in [0, len(d_{host}))$ of inj will be the insertion position i and based on $d_{load}[j]$. If the value of j does not fall between the range required for the injection position, which is from i to $i + len(d_{load}) - 1$ as this location is required for the payload injection, then the actual nucleotide of host $d_{host}[j]$ will be used, i.e. $inj[j] = d_{host}[j]$. Otherwise, the value of $inj[j]$ will depend on $d_{load}[j-i]$, since the value $[j-i]$ determines the index of the d_{load} and this has to be considered when it starts from 0 (for the very first substitution point $j = i$) up to $len(d_{load}) - 1$. If the $d_{load}[j-i]$ contains a retention

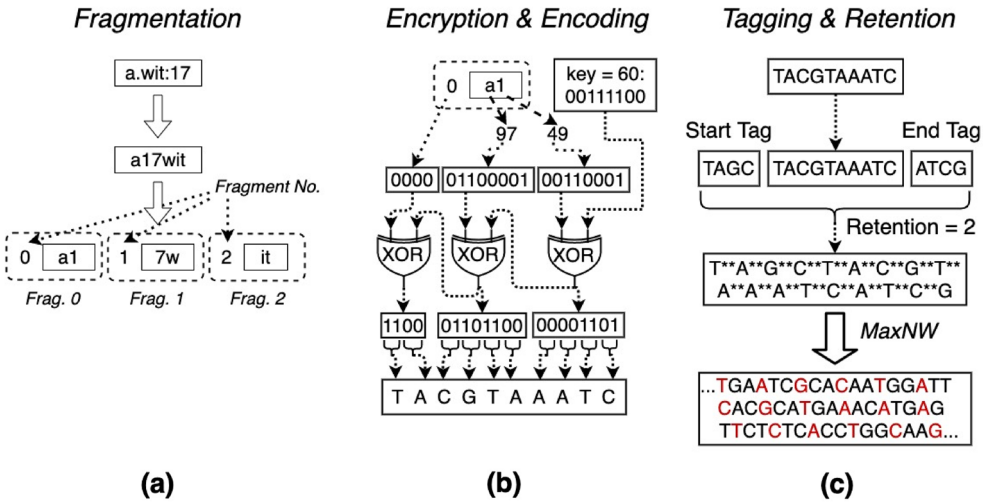

Figure 3. DNA Steganography, Workflow: (**a**) payload fragmentation, (**b**) fragment encryption and encoding, (**c**) tagging, retention and host injection. (The image is drawn using draw.io).

symbol "*", i.e. (d_{load} [j−i] == "*" then $inj[j] = d_{host}[j]$ (this means the original nucleotide is used for retention) otherwise $inj[j] = d_{load}[j-i]$. This substitution procedure can be defined as:

$$inj[j] = \begin{cases} d_{host}[j] \text{ if } j < i \text{ or } j \geq i + len(d_{load}), \\ d_{host}[j] \text{ if } j \in [i,\ i + len(d_{load})) \text{ and } d_{load}[j-i] = \text{"}*\text{"}, \\ d_{load}[j-i] \text{ if } j \in [i,\ i + len(d_{load})) \text{ and } d_{load}[j-i] \in \{A, C, T, G\}. \end{cases} \quad (1)$$

We define *elementary* domain dom_{ELM} that consists of all the possible positions for a Trojan payload injection. Naturally, such a substitution can be carried out only from the position i onwards and is represented as:

$$dom_{ELM} = [0, len(d_{host}) - len(d_{load}) + 1], \quad (2)$$

which is referred to as the *injection domain* and refers to the indices (i.e., values of i) of d_{host}. We use $inj(d_{host}, d_{load}, i)$ to denote the substitutions in dom_{ELM}. Similarly, to denote the substitutions carried out on subdomain $dom \subset dom_{ELM}$, we use $inj_{dom}(d_{host}, d_{load}, i)$. This subdomain introduces additional restrictions that may be required to preserve particular areas within the host DNA. Figure 3 presents the five stages/steps involved in the DNA steganography technique used in this article.

Note that in this article we only consider payloads that consist of a web address (represented by a Tiny URL) and port number of a remote server controlled by the attacker (For payloads used in the analysis, please see: https://bitbucket.org/sibleeislam/bio-cyber-hacking). The payload has the following semantics:

< prefix : character string > . < suffix : character string > : < port number :string of digits >

As mentioned above, the fragmentation (Fig. 3a) is the first stage of the DNA steganography. First, the payload is rearranged so that the address prefix is followed by the port number and then the address suffix. This representation allows the reduction in the auxiliary "." and ":" characters from the payload, and therefore, size reduction of the entire payload. Subsequently, the rearranged payload is divided into fragments, substrings of a predefined length (e.g. 2 characters as shown in Fig. 3). Each of the fragments is attached with its serial number as a prefix. As only tiny URLs are used in the tojoan payload address, we assume that no more than 16 fragments can be formed. If we want to consider a web address with subdomains then the top level domain will be the suffix and the rest of the part of the domain name will be the prefix.

The next step after frangementation is encryption, where each fragment is encrypted and nucleotide-encoded as illustrated in Fig. 3b. At this stage, the fragment is represented as a bit-array where the first 4 bits represent the fragment's serial number, followed by a series of 8-bit representations of fragment characters. Each character is represented by the binary of its ASCII code. The bit-array is then XOR encrypted using a predefined key (e.g. 60 as depicted in Fig. 3b). This results in a sequence of bit-pairs, which are then encoded into nucleotides strings that represent the DNA.

The next step after encryption is encoding as shown in Fig. 3c. The nucleotide-encoding of the fragment is attached with a start and end tag as prefix and suffix, respectively. The resultant string is then expanded so that a predefined amount of retention symbols is added between each two consecutive nucleotides (e.g., 2 symbols as in Fig. 3c). The expanded string is then injected into the host DNA using MaxNW procedure, which is described next.

MaxNW technique. Needleman-Wunsch, or NW score is one of the most popular methods to assess the similarity between two DNA samples. This score considers the string-based nucleotide representation of the DNA molecules and calculates the number of symbol substitutions, gaps (i.e., symbol insertion or deletion) and

their expansions (i.e., continuation of gaps) required to align two strings. Depending on the circumstances, a specific penalty system is applied to each of the operations as well as matches between DNA nucleotides. The system is constructed in a way to favor certain alignment patterns. As in the experiments performed in this work, injecting payload typically constitutes not more than 10% of the host DNA string size, therefore we use PAM10 substitution scoring matrix[31] as the cost matrix for nucleotide substitution. Following this methodology outlined in[32], we set the costs for the gap opening and extension to 15.79 and 1.29 for the PAM10 substitution, respectively.

In this article, we use NW scores to measure the similarity between d_{host} and $inj(d_{host}, d_{load}, i)$. Based on the penalties defined above, the NW score increases as similarity between d_{host} and $inj(d_{host}, d_{load}, i)$ increases and reaches its maximum if d_{host} and $inj(d_{host}, d_{load}, i)$ are equal. In other words, the injected payload fits into the d_{host} naturally at position i. Lets assume the NW score is maximum when the insertion position (the value of i) is i_{max}. To emulate the attacker, the malware NW score, $MaxNW_{dom}$, is defined as:

$$MaxNW_{dom}(d_{host}, d_{load}) = inj(d_{host}, d_{load}, i_{max}), \tag{3}$$

where

$$i_{max} = Arg\left(max_{i \in dom} NW\left(d_{host}, inj(d_{host}, d_{load}, i)\right)\right). \tag{4}$$

when multiple payloads for malicious activity injections $D_{load} = \left\{d_{load,1}, \ldots, d_{load,n}\right\}$ are introduced into the same host DNA, dynamic programing is used to determine the optimal positions for the injections. The technique employs a recursive procedure, where at each step the best insertion is sought amongst all possible positions. So, initially $inj(d_{host}, d_{load}, i)$ and dom_{ELM} are considered for the substitution and the domain for the substitutions for each of the payloads. Then the injection position of the payload having maximum NW Score will be considered for that particular payload injection and that portion of the injection will be restricted for further injections. For further steps, the subdomain dom and injection for subdomain $inj_{dom}(d_{host}, d_{load}, i)$ will be considered as the restriction is applied. Lets assume, the maximum NW Score and the indices considering subdomain are $MaxNW_{dom^*}$ and , i^* respectively. The injection process will be repeated until all the payloads are injected. Thus, this recursive procedure can be described as:

$$MaxNW_{dom}(d_{host}, D_{load}) = MaxNW_{dom^*}\left(MaxNW_{dom}\left(d_{host}, d_{load,i^*}\right), D_{load}/d_{load,i^*}\right), \tag{5}$$

where

$$i^* = Arg\left(max_{j \in [0, len(D_{load}))} NW_{dom}\left(d_{host}, d_{load,j}\right)\right). \tag{6}$$

Deep learning. Various machine learning algorithms and even regular expression based classification techniques might be useful to detect the presence of trigger samples in the DNA sequence as the tags will express a pattern. However, the tag will be unknown to the detection subsystem and the number of available tags will grow exponentially with the number of nucleotides used for the tags (please see Fig. 5a). Therefore, a machine learning algorithm will be a better option as the regular expression will need to consider a huge number of possible tags. Again, a big challenge of machine learning algorithms (e.g. Random Forest Support Vector Machine, K nearest Neighbors etc.) is feature extraction and feature selection. Very large number of features can be applied using these techniques. The success of the prediction technique mainly depends on finding appropriate feature extraction and feature selection techniques[33]. To extract the feature, Natural Language Processing (NLP)[33] is applied to a DNA sequence or DNA is considered as a time series or genomic signal and then signal processing techniques like Discrete Fourier Transform (DFT)[34] and Discrete Wavelet Transform (DWT)[35] are applied. RNN is good for sequential data or time series data and also where the context, especially the previous classification, is important[33]. In our classification problem, each classification is independent. However, it is interesting to see how to feed the sequence and get intermediate classifications in the hidden layer and then get the final classification. On the other hand, in CNN the convolutional layers part of the architecture does the feature extraction and selection for us and the flat layers followed by the convolutional layers does the classifications[33,36]. This is the reason we have selected the use of CNN. In this article, we use a 1-Dimensional Convolutional Neural Networks (1D CNNs) to identify the Trojan payload within the natural DNAs. This section will provide a brief overview of the CNNs we utilized for this work. An overview of various Deep Learning methods, including CNNs, used in genetics analysis can be found in[37].

Figure 4 depicts the typical architecture of a 1D CNN. Similar to any other neural network, the 1D CNN consists of neurons organized in layers. The architecture proposed in this article uses the following layers: input, convolution, pooling, and dense.

The first layer represents the input of the network. Here, each of the DNA sequences' classification is transformed into the set of primary features, i.e., inputs of the network. Each nucleotide of the DNA is represented by a vector of 5 boolean indicator values. The first 4 values indicate whether the nucleotide are found to be equal, whereas the 5th value indicates whether the nucleotide can be determined (i.e. N—undetermined). As an example, A-nucleotides of the DNA will be represented by (1,0,0,0,0) indicator vectors, C-nucleotides will be represented by (0,1,0,0,0), and undetermined nucleotides will be represented by (0,0,0,0,1). To formulate the primary features of the entire DNA, indicator vectors for all its nucleotides are concatenated in the order of the pattern that is found in the original DNA.

The input layer is followed by a number of CONV1D layers as shown in Fig. 4. At each layer, multiple filters are applied to the kernels of a particular size. The resultant product is then subjected to ReLU activation. CONV1D

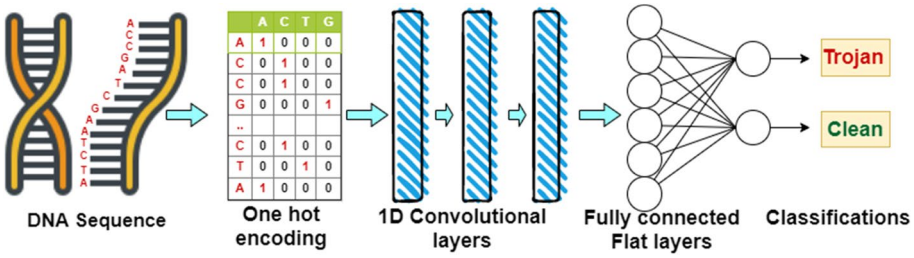

Figure 4. 1-Dimensional Convolutional Neural Network (1D CNN): Architecture. (The image is drawn using draw.io).

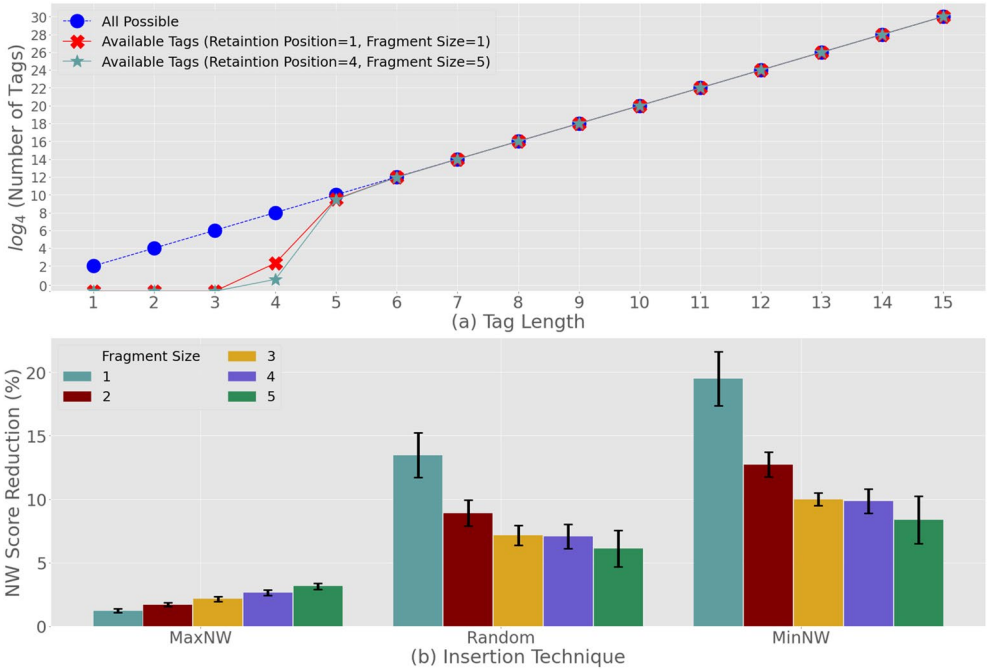

Figure 5. Trigger Sample Design with the use of DNA-Steganography: (**a**) nucleotide tag selection; (**b**) the impact of fragmentation.

layers are followed by 1 MaxPool, one dense layer with ReLU activation function, and finally 2-neuron SoftMAX layer, the output of which provides the certainty of the sample to be determined if it contains the address information. In this article, we consider networks with varying numbers of CONV1D layers, the size of their kernel and the number of filters used. We also investigated the impact of the kernel size of the MaxPool layer and the size of the ReLU dense layer. Each network is trained for 3000 epochs using 75% of all available DNA samples. The remaining 25% of the samples are used to test the performance of the trained network.

Results and discussion

For the Trojan infected softwares, the secrecy of operation is of paramount importance. The longer the Trojan remains undetected, the more extensive the damage it can cause. For the Bio-Cyber hacking attack considered in this article, it is of vital importance for the attacker to maintain a natural appearance of the trigger sample containing the address details. If we use an unnatural DNA structure as a part of the hybrid attack it can be flagged as suspicious not only by the detection method proposed in this article, but also by the similar less sophisticated versions of this system proposed in previous works[24].

In this section we begin the discussion by evaluating the possible actions of an attacker to design a natural trigger sample. We follow this up by investigating the accuracy with which these trigger samples can be detected by a CNN. Finally, we describe the wet lab experiments that were used to produce the DNA with the address, in order to validate the potential of creating such a DNA sequence that is used as the trigger sample for our attack.

Trigger sample design. For this article we propose the use of *E. coli* plasmids that will encode the address of the attacker. *Escherichia coli* bacteria have been sufficiently studied in literature and their plasmids can be synthesized and modified with relative ease. Once the attacker identifies a suitable DNA structure, *E. coli* plas-

mids can be readily synthesized in various laboratories across the globe such as *EuroFins Genomics and Twist BioScience*[24]. In this section, we present the design of the plasmid DNAs that contains the Trojan payload that will maintain the original *E. coli* plasmids sequence. Specifically, we evaluate the use of DNA steganography (as described in the Methodology section) for injecting the address payload into an *E. coli* plasmid (host) DNA to maximize similarity between the resultant *inj* and host DNAs d_{host}.

This evaluation requires 1000 bps reads randomly sampled from the plasmid DNAs made available via *AddGene* repository. The sampling serves two purposes. First, it mimics the operation of a DNA-sequencer (e.g., Roche 454 FLX + [38]) that may be specifically targeted by the attacker. In this case, a higher number of DNA-reads produced by the sequencer (i.e., 700–1000 bps) will provide better cover for the Trojan address payload and, thus, increase the chances for the hybrid attack to be successful. Secondly, the sampling can significantly increase the amount of DNA-data used in the evaluation, where we draw 4356 reads from 716 *E. coli* plasmid DNAs stored in the AddGene repository.

Since the steganography technique has five key steps, the *encoding* step is fixed and cannot be varied, but the attacker is free to finetune the tagging, fragmentation, encoding, retention, and encryption steps. In Fig. 5 we show the impact of different parameter combinations, e.g. size of the fragment, number of retention positions, and value of the encryption keys.

Figure 5a depicts the relationship between the length of nucleotide tags and their availability. The tags mark the start and the end of the Trojan payload injections into a plasmid DNA. These tags that mark the start and the end of the Trojan payload are two potentially different nucleotide sequences of the same length. The sequences are selected in a manner that a host DNA is unlikely to include both tags separated by nucleotides. Note that the number of these nucleotides are obtained directly from the fragment size and the retention (i.e. retention of host nucleotides) parameters of the steganography technique. The results in Fig. 5a correspond to various values of these two parameters. From these results we learn that a predictable growth of tag availability is associated with the increase in tag length. As the number of all possible nucleotide sequences grows exponentially, it can overcome the number of unique sequences in genuine DNA reads for 4-nucleotide tags. We also realize that any further increase in the tag length (i.e., 5 and beyond) will make the number of unique sequences negligible, leaving the attacker with ample choice of nucleotide tags. The strength of this effect is such that it can be seen for all fragment sizes and retention values. As a result of this observation, we use a minimum 5-nucleotide tags for the remainder of this article as this is the lowest length that allows for the substantive tag availability.

In Fig. 5b we study the impact of the fragment size selection on the similarity between the host DNA before and after the injection of Trojan payload. This similarity is assessed by using Needleman-Wunsch (NW) scores (described in Methodology). The system is designed in such a way that the Needleman-Wunsch score grows as the similarity between the two DNAs increases. The value of this score is absolute maximum (i.e. MaxNW) when either the DNAs are identical, or the Trojan payload address is inserted into the host DNA naturally. Since due to tagging this is not possible we use the maximum (i.e. the NW score between host the DNA and itself) value to benchmark the score reduction due to the payload injection. Furthermore, in order to ensure the optimal payload injection, the steganography uses MaxNW technique (described in Methodology). To demonstrate the efficiency of this technique, Fig. 5b presents a comparison of performance with two alternative techniques, i.e., Random and MinNW. Random technique injects the payload at an arbitrary position through uniform distribution, whereas MinNW is a dynamic programming technique that seeks the worst possible injection position for a payload. This means that MinNW is a mirror-image of MaxNW which can minimize the score between the host and injected DNAs. This phenomenon is reflected in Fig. 5b, where MaxNW results in significantly lower score reduction compared to MinNW, whereas the score reductions by Random technique lies approximately in the middle of those produced by MaxNW and MinNW. From this we conclude that the MaxNW and MinNW techniques can show the whole range of score reductions that may occur due to payload injections. This also reaffirms that MaxNW is the best technique amongst all three possible techniques. In addition, a closer inspection of the results for the MaxNW technique also clarifies the impact of payload fragmentation. We realize that using a larger fragment size in the host DNA can effectively reduce the similarity between the host and injected DNAs.

Next in Fig. 5a,b we investigate the impact of different *retention* as well as *encryption* choices of the attacker. The results are presented only for MaxNW which is the optimal injection technique we have selected. For both the retention of host nucleotides or payload encryption, we realize that there is no significant effect on the NW score. In particular Fig. 6a shows no change in the NW score reduction can be attributed to different retention numbers for various fragment sizes for payload encrypted with a key equal to 50. Figure 6b shows similar results, where payload fragments of 1 and 5 characters are injected using 1 and 5 retention numbers. For this case, we also observe no change in the NW scores when encription keys are utilized. Based on these results, we can conclude that neither *retention* nor *encryption* are likely to disguise the trigger sample. Although we note that neither of these two steps can help the payload appear more naturally, however they still remain an essential part of the steganography process. This is because these steps play a key role in maintaining the anonymity of the attacker as they are designed to protect the payload (i.e. network address and port number), which may identify the attacker. For the case when a trigger sample is identified, the retraction of the payload will require knowledge of both the *retention number* and the *encryption key* used by the attacker.

DNN detection accuracy. The DNA sequences of 716 *E. coli* plasmid DNAs are collected from the AddGene repository using web scraping. The Selenium Webdriver is used to crawl and collect the pages containing the DNA sequences. The page was parsed using a python script to get and store the DNA sequences. In total, 4356 reads with read size of 1000 were drawn from the DNA sequences.

Although the natural appearance of the trigger sample is necessary to disguise the hybrid attack and avoid detection by less sophisticated methods (e.g. NW comparison with known DNAs), the Trojan payload address

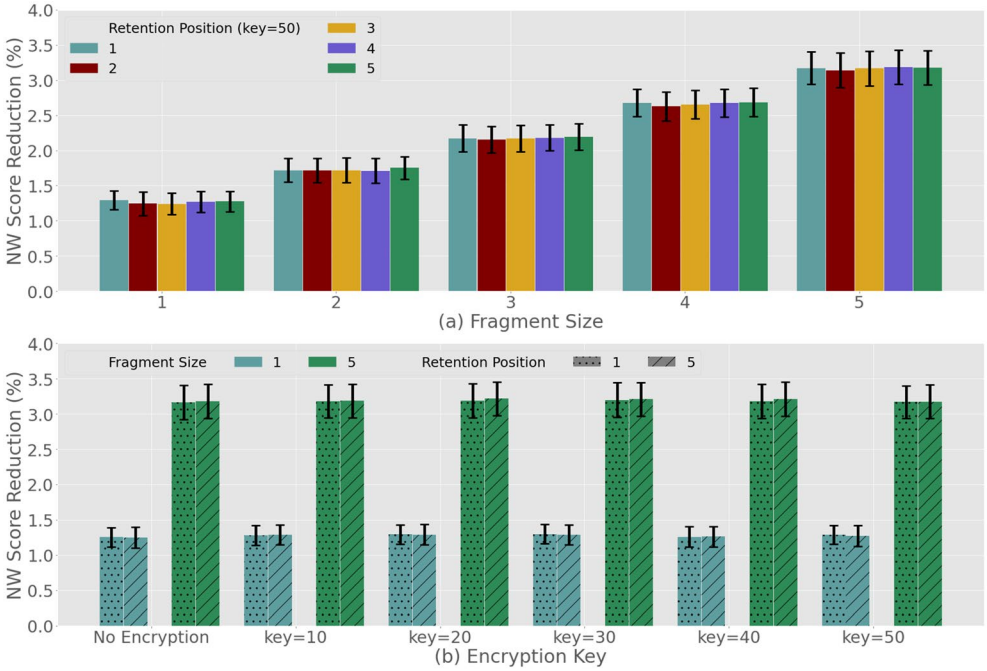

Figure 6. Trigger Sample Design, the use of DNA-Steganography: (**a**) retention of host nucleotides; (**b**) payload encryption.

injection may still be discoverable with the help of other techniques. In this section, we will explore this by evaluating the detection of trigger samples using a state-of-the-art Deep Learning approach. From 4356 reads, where the read size is 1000 nucleotide bases, 1000 reads are picked randomly. These are clean or natural samples. 1000 web addresses were considered for creating the malicious samples. A web address is rearranged, fragmented and then expressed as nucleotides and then inserted into the clean sample using the substitution method and using the technique described in the "Injection Methods" subsection in "Methods" section based on best NW Scores. The whole process is repeated 10 times to create 10 datasets. The idea is to execute 10 CNN model training and evaluations and take the average. However, if we consider 5 different fragmentations, 5 retentions and 5 encryptions then for all combinations it will be a huge number of evaluations. Furthermore, for the hyperparameter optimization there will be more scenarios to consider. Therefore, for every scenario we combine all 10 clean datasets into one and 10 malicious datasets into one. From 10,000 clean and 10,000 malicious data we pick 7500 (75%) data randomly from each as training dataset and remaining 2500 (25%) data from each as testing dataset. For training the model using the training dataset, we take a batch size of 100 at a time. We run the training for 3000 epochs with a learning rate of 0.001. In each epoch, 10 percent of the training data was used for validation, so that we can examine the learning (accuracy and loss comparison for training and validation over epochs) to avoid overfitting. Moreover, after every layer a dropout layer is also added to avoid the overfitting. Early stop monitoring is also used to avoid unnecessary continuation of the model training if it is reached to its optimal accuracy. The trained model is then evaluated by the corresponding test dataset. We achieve this by investigating the performance of a 1-Dimensional Convolutional Neural Networks (CNN). The results in Fig. 7a,b summarize the performance of various CNNs topologies with respect to the four hyper-parameters considered in this article. This includes, (i) the number of *hidden layers* (1 and 2), (ii) the sizes of the *filter* (4, 8 and 16), (iii) size of the *kernel* (3, 5 and 8), and (iv) size of the *maxpool* (2 and 4) used in the network. The results are then obtained for trigger samples obtained from natural DNA using 0-retention and no payload encryption. This means that we can establish a baseline predictive capacity of CNNs and determine the most suitable network topology. This suitable topology is then further tested to evaluate the ability to cope with additional uncertainties introduced by nucleotide retention and payload encryption.

For this purpose, we simulated 180 scenarios for 36 combinations of hyper parameters and for 5 different fragment sizes, with no retention and no encryption. We obtain the best accuracy (99.9–100%) for all 5 fragment sizes when we have 1 hidden layer, kernel size 16, 16 filters and 4×4 max pool size (Fig. 7a). Similarly, we obtain the best accuracy for the case we have an additional layer (2 hidden layers), 16 filters, kernel size 5 and 4×4 max pool (Fig. 7b). These features are mainly learned by the kernel, so larger kernels and higher number of filters result in achieving the best accuracy. However, in this article we prefer to use a smaller number of required hidden layers to increase the execution time performance. Therefore, for the rest of the experiment we consider the CNN topology with 1 hidden layer, kernel size 16,16 filters and 4×4 max pool.

Next in Fig. 8 we analyze the impact of the fragment size, retention values and encryption on the Trojan address detection. In particular, Fig. 8a presents the detection accuracy for the highest and lowest fragment size values (1 and 5), and all the retention numbers (1–5), when no encryptions are applied. We made an assumption

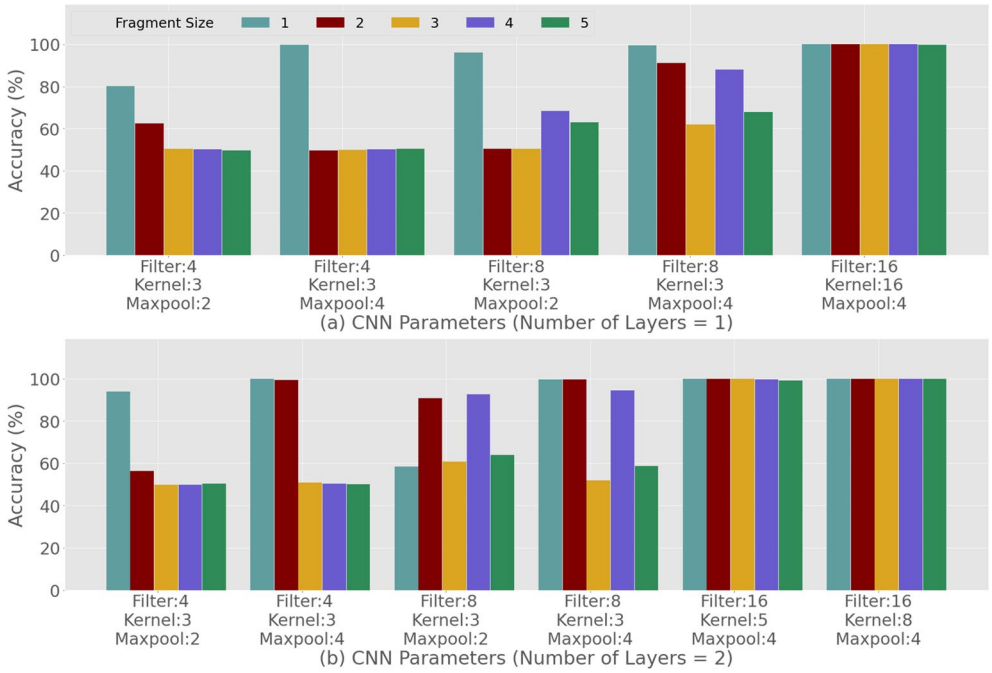

Figure 7. DNN-based detection of trigger samples amongst genuine *E. coli* plasmids: hyper-parameter optimization (no encryption or retention) using (**a**) 1 and (**b**) 2 hidden layers.

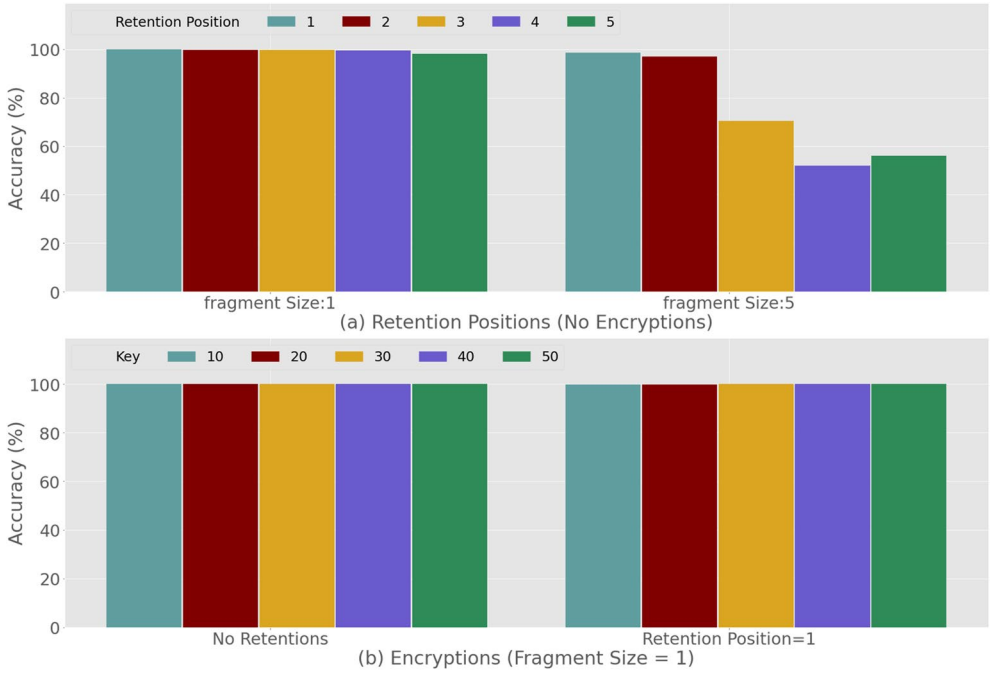

Figure 8. DNN-based detection of trigger samples amongst genuine *E. coli* plasmids: the impact of nucleotide retention (**a**) without encryption and (**b**) with encryption and with prior knowledge of the encryption key**.**

that if we split the payload into an increasing number of fragments it will be relatively easy to escape the detection. In such a case it will be comparably difficult to locate the complete Trojan payload address and, therefore, be relatively harder to make sense out of a more tinier part of the payload. Furthermore, as shown and explained in the previous section (Fig. 6a,b), the DNA sequences remain much more natural for smaller fragment sizes. Based on this knowledge, a potential hacker might prefer to choose a smaller fragment size. However in reality this approach will leave more tags as low fragment size translates to increase in number of tags. Therefore, this

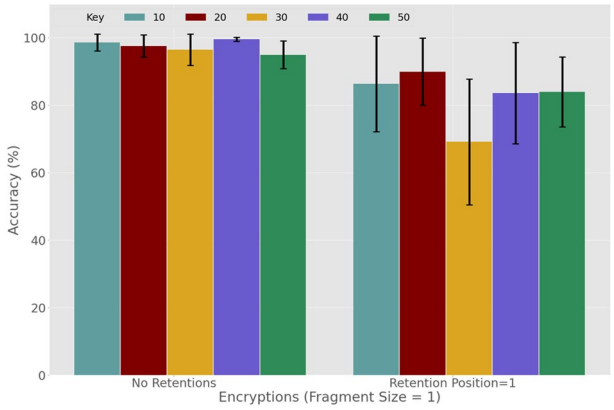

Figure 9. DNN-based detection of trigger samples amongst genuine *E. coli* plasmids: the impact of nucleotide retention, no knowledge of the encryption key.

approach can support the CNN model, which can learn from the tag patterns and the result in Fig. 8a illustrates this.

On the other hand, in a real world scenario it will be a significant challenge to design an optimal model which can account for many variations of tags. Interestingly, we observe that for higher fragment sizes, the accuracies deteriorate very slightly until there is a higher retention number as well (Fig. 8a). This indicates that the model proposed in the article does not completely rely on learning the tag patterns. Furthermore, the higher retention number means more number of nucleotides (from the original sequence inside the tags) which will result in more variations and harder detection. However, we note that for fragment size 1 the accuracies are very high for all retention numbers. Overall, the accuracies start to deteriorate significantly for the higher fragment sizes with higher retention numbers (Fig. 8a). To analyze the impact of encryption on the Trojan address payload detection, we consider fragment size 1 with no retention and retention size 1 as we obtain the best accuracy for these options. We apply encryptions with various key values ($key \epsilon \{10, 20, 30, 40, 50\}$). In Fig. 8b, the results show that there is no significant change in accuracy when applying various encryption keys. Please note that both the training and test data are using the same key value for encryption.

We will now further analyze the impact of encryption in detection. In Fig. 9 we present the detection accuracies where the rojan payload address in the test data is encrypted with a different key. The model is trained with a particular key which is tested by all the data encrypted by the remaining keys. For example, the model trained by the data encrypted using key = 10 will be tested by all the test data that are encrypted by other keys, i.e. keys = {20, 30, 40, 50}. Similarly, the model for key value 20 will be tested by all the test data encrypted by the keys = {10, 30, 40, 50}. In Fig. 9 we plot the average accuracy against the different key values used for training the model. From this result, we conclude that a higher accuracy can be achieved for encrypted payloads without retention even if the key is unknown. However, the accuracy will deteriorate if we apply retention along with encryption. This is because the higher retention will result in the DNA sequence having a more natural pattern, which makes it more difficult to detect.

Wet lab experiments. In the previous sections of this article, we have described how we can disguise the address payload for a Trojan attack to make the payload insert indistinguishable compared to a natural DNA sequence. Furthermore, applying encryption and steganography techniques will make it harder to detect the hybrid Trojan attack. However, it is also important to address how practical it is to synthesize such a DNA sequence. In our wet-lab, we constructed the Trojan payload sequences both without and with encryption and steganography (Figs. A.1 and A.2) via commercial gene synthesis with ease. These sequences were prepared and received already ligated into bacterial plasmid vector. These plasmids, pNOSTEG and pSTEG, were easily cloned into *E. coli* cells, propagated and purified in abundance (Fig. A.3). The Trojan payloads in both plasmids were both DNA sequenced completely and with 100% accuracy, with a sample chromatogram from pNOSTEG shown in Fig. A.4. We can assume that constructing natural DNA sequences will be easier and more achievable compared to synthesizing artificial DNA with unnatural sequences, due to possible runs and repeats of DNA bases that may cause problems in the synthesis reaction. As a result, there will be a need to construct a DNA that can allow multiple fragment inserts with the target information of the IP address and port number of the remote hacker's machine. With various techniques emerging for generating, producing or inserting multiple DNA sequences into carrier or expression systems, e.g., in-fusion cloning, gene assembly or multiple fragment cloning, hackers can bypass any gene synthesis issues by using a combination of these techniques to generate their final Trojan attack sequence. As such, our work presents valuable detection against very feasible attack scenarios.

Data availability

All data used in the manuscript are available in the Addgene repository (https://www.addgene.org/), where the DNA sequences of type plasmid of *E. coli* bacteria are collected for our experiments using web scraping. This data is also available as a supplementary document (all_plasmid_dna.txt). The Programming code developed to

conduct the experiments (also the scripts for the data collection from Addgene) is freely available in the publicly available git repository at the following URL: https://github.com/sibleeislam/trojan-malware-in-bio-cyber-attacks. For any further query related to data availability please contact using the email of the primary author (sibleeislam@gmail.com) of the manuscript.

Received: 25 February 2022; Accepted: 26 May 2022
Published online: 10 June 2022

References

1. Vijayvargiya, P. *et al.* Application of metagenomic shotgun sequencing to detect vector-borne pathogens in clinical blood samples. *PLoS ONE* **14**, e0222915 (2019).
2. Haiminen, N. *et al.* Food authentication from shotgun sequencing reads with an application on high protein powders. *NPJ Sci. Food* **3**, 1–11 (2019).
3. Akyildiz, I. F., Pierobon, M. & Balasubramaniam, S. An information theoretic framework to analyze molecular communication systems based on statistical mechanics. *Proc. IEEE* **107**, 7 (2019).
4. Unluturk, B. D., Balasubramaniam, S. & Akyildiz, I. F. The impact of social behavior on the attenuation and delay of bacterial nanonetworks. *IEEE Trans. Nanobiosci.* **15**(8), 959–969 (2016).
5. Laver, T. *et al.* Assessing the performance of the Oxford nanopore technologies MinION. *Biomol Detect Quantif* **3**, 1–8 (2015).
6. Yousefzai, R. & Bhimaraj, A. Misdiagnosis in the COVID-19 Era. *JACC: Case Rep.* **2**, 1614–1619 (2020).
7. Lim, J. T. *et al.* The costs of an expanded screening criteria for COVID-19: A modelling study. *Int. J. Infect. Dis.* **100**, 490–496 (2020).
8. Aitken, J. *et al.* Scalable and robust SARS-CoV-2 testing in an academic center. *Nat. Biotechnol.* **38**, 927–931 (2020).
9. Reuben, R. C., Danladi, M. M. A. & Pennap, G. R. Is the COVID-19 pandemic masking the deadlier Lassa fever epidemic in Nigeria?. *J. Clin. Virol.* **128**, 104434 (2020).
10. Capone, A. Simultaneous circulation of COVID-19 and flu in Italy: Potential combined effects on the risk of death?. *Int. J. Infect. Dis.* **99**, 393–396 (2020).
11. Hsih, W.-H. *et al.* Featuring COVID-19 cases via screening symptomatic patients with epidemiologic link during flu season in a medical center of central Taiwan. *J. Microbiol. Immunol. Infect.* **53**, 459–466 (2020).
12. San Millan, A. Evolution of plasmid-mediated antibiotic resistance in the clinical context. *Trends Microbiol.* **26**, 978–985 (2018).
13. Blackwell, G. A., Doughty, E. L. & Moran, R. A. Evolution and dissemination of L and M plasmid lineages carrying antibiotic resistance genes in diverse Gram-negative bacteria. *Plasmid* **113**, 102528 (2021).
14. Health Service Executive (HSE) Ireland. About CervicalCheck: Ireland's national cervical screening programme. Available On-Line, Retrieved from 13 Aug 2020 https://www2.hse.ie/screening-and-vaccinations/cervical-screening/about-cervicalcheck/about.html
15. Ney, P. *et al.* Computer security, privacy, and DNA sequencing: Compromising computers with synthesized DNA, privacy leaks, and more. USENIX Security 17, 2017.
16. Rabadi, D. & Teo, S. G. Advanced windows methods on malware detection and classification. In *Annual Computer Security Applications Conference* (2020).
17. Kouliaridis, V., Kambourakis, G. & Peng, T. Feature Importance in android malware detection. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (2020).
18. Guo, W. *et al.* Towards inspecting and eliminating Trojan backdoors in deep neural networks. In *2020 IEEE International Conference on Data Mining (ICDM)* (2020).
19. Pan, Z. & Mishra, P. Automated test generation for hardware Trojan detection using reinforcement learning. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference* (2021).
20. Yasaei, R., Yu, S.-Y. & Al Faruque, M. A. GNN4TJ: Graph Neural networks for hardware Trojan detection at register transfer level. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2021).
21. Lyu, Y. & Mishra, P. Automated test generation for Trojan detection using delay-based side channel analysis. In 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE) (2020).
22. Guo, S., Wang, J., Chen, Z., Li, Y. & Lu, Z. Securing IoT space via hardware Trojan detection. *IEEE Internet Things J.* **7**, 11115–11122 (2020).
23. Black, A., MacCannell, D. R., Sibley, T. R. & Bedford, T. T. recommendations for supporting open pathogen genomic analysis in public health. *Nat. Med.* **26**, 832–841 (2020).
24. Islam, M. S. *et al.* Genetic similarity of biological samples to counter bio-hacking of DNA-sequencing functionality. *Sci. Rep.* **9**, 1–9 (2019).
25. Sreekumari, P. Malware detection techniques based on deep learning. In *2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)* (2020).
26. McDole, A., Abdelsalam, M., Gupta, M. & Mittal, S. *Analyzing CNN based behavioural malware detection techniques on cloud IaaS*. In *Lecture Notes in Computer Science*, 64–79 (2020).
27. Kimmel, J. C., Mcdole, A. D., Abdelsalam, M., Gupta, M. & Sandhu, R. Recurrent neural networks based online behavioural malware detection techniques for cloud infrastructure. *IEEE Access* **9**, 68066–68080 (2021).
28. Sharma, R., Rathor, V. S., Sharma, G. K. & Pattanaik, M. A new hardware Trojan detection technique using deep convolutional neural network. *Integration* **79**, 1–11 (2021).
29. Islam, M. S. *et al.* Trojan bio-hacking of DNA-sequencing pipeline. In *Proceedings of the Sixth Annual ACM International Conference on Nanoscale Computing and Communication* (2019).
30. Hayward, S. L., Francis, D. M., Sis, M. J. & Kidambi, S. Ionic driven embedment of hyaluronic acid coated liposomes in polyelectrolyte multilayer films for local therapeutic delivery. *Sci. Rep.* **5**, 14683 (2015).
31. Pearson, W. R. Selecting the right similarity-scoring matrix. *Curr. Protoc. Bioinform.* **43**, 3–5 (2013).
32. Rivas, E. & Eddy, S. R. Parameterizing sequence alignment with an explicit evolutionary model. *BMC Bioinform.* **16**, 1–23 (2015).
33. Gunasekaran, H. *et al.* Analysis of DNA sequence classification using CNN and hybrid models. *Comput. Math. Methods Med.* **2021**, 1–12 (2021).
34. Ghosh, A. & Barman, S. Application of Euclidean distance measurement and principal component analysis for gene identification. *Gene* **583**, 112–120 (2016).
35. Liu, D.-W. *et al.* Automated detection of cancerous genomic sequences using genomic signal processing and machine learning. *Futur. Gener. Comput. Syst.* **98**, 233–237 (2019).
36. Weimer, D., Scholz-Reiter, B. & Shpitalni, M. Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. *CIRP Ann.* **65**, 417–420 (2016).
37. Zou, J. *et al.* A primer on deep learning in genomics. *Nat. Genet.* **51**, 12–18 (2018).
38. Yin, Z., Mancuso, J. J., Li, F. & Wong, S. T. C. Genomics-based cancer theranostics. *Cancer Theranostics* 9–20 (2014).

Acknowledgements
This publication came from research conducted with the financial support of Science Foundation Ireland (SFI) and the Department of Agriculture, Food and Marine on behalf of the Government of Ireland (Grant Number [16/RC/3835] - VistaMilk). Artwork on Figs. 2 and 4 of the article was created by the author of this article Mr. Islam using free Draw.io (https://www.diagrams.net/about) software and free icons available on the web. Artwork on Fig. 1 and 3 of the article was created by the authors of this article Dr. Ivanov and Mr. Islam using free Draw.io software and free icons available on the web.

Author contributions
Mr. M.S.I. is the primary author of the article. Mr. I. was responsible for developing the software code used to perform computational experiment, executing the experiments, analysing and interpreting the results presented in this article, writing the manuscript. Dr. S.I. was responsible for overseeing and directing computational experiments presented in this article. Specifically, Dr. I. contributed to the development of the proposed steganography technique, where he proposed the dynamic programming technique for finding an optimal location for the payload for malicious activity to be injected into the host DNA. Dr. I. assisted Mr. I. in writing the manuscript. Dr. S.B. was the main scientific driver behind the experiments presented in the article. Due to his multidisciplinary background, Dr. B. identified the possibility for *E. coli* bacteria to be used as carriers of malicious DNA on-purposed engineered as part of a Trojan attack. That was the starting point for the research presented in the article. Subsequently, Dr. B. directed and oversaw the experiments conducted in this research. Dr. Lee Coffey planned and executed the wet lab experiments, including gene synthesis design, cloning and recombinant plasmid DNA purification. Dr. S.K. was responsible for providing expertise in methods for handling DNA based samples and background for DNA packaging/carrying. Ms. J.D. prepared the DNA samples for sequencing and carried out sequence analysis of the DNA fragments in order to verify sequence identity and fidelity. Dr. W.S. was the scientific driver behind the DNN analysis for the DNA strands with the injected code, as well as the development of the hacking scenarios. Dr. H.A. was responsible for the analysis of the data in the results section and in particular the analysis on performance based on variations in parameters.

Competing interests
The authors declare no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-022-13700-5.

Correspondence and requests for materials should be addressed to M.S.I.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.