HITB+CyberWeek 2021
Brought to you by DisruptAD

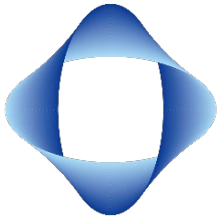
Deep Puzzling: Binary Code Intention Hiding based on AI Uninterpretability

Jifeng Zhu & Keyun Luo

Zhuque Lab, Tencent Security Platform Department

Who Are We

- ***Tencent Security Platform Dpt.*** has been with Tencent for 16 years, and dedicated to the protection of QQ, Wechat, Tencent Games and other critical products.
- ***Tencent Zhuque Lab*** was founded in 2019 by Tencent Security Platform Dpt., focusing on red teaming and AI security research. Research results have been published on HITB, XCON, POC, etc.



Tencent Security Platform Dpt.

Tencent Zhuque Lab

Outline

- *A Brief Introduction to Intent Hiding*
- *Why it Matters for Security*
- *Deep Puzzling Details*
  - Method Overview
  - Feature Extraction
  - Binary Code Generation
  - Binary Code Repair
- *Demonstration and Analysis*
- *Conclusion*

A Brief Introduction to Intent Hiding

The Evolution History of Malware

- *Obfuscation: Mutate payload*
- *Encryption : Hide payload*
- *Evasive malware: Avoid being analyzed*
- *Targeted attack: Disclose only at a target*

- Addition
  - `a = b - (-c)`

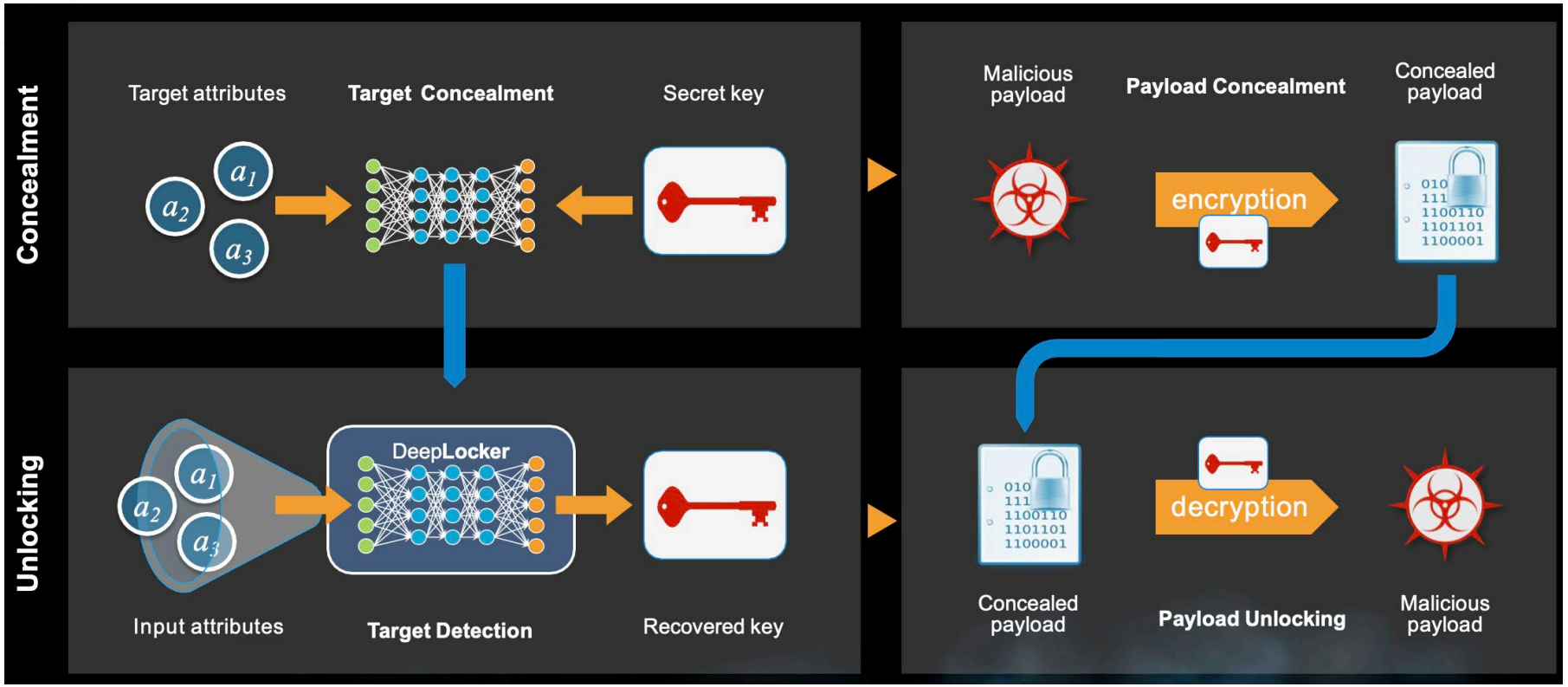
```
%0 = load i32* %a, align 4
%1 = load i32* %b, align 4
%2 = sub i32 0, %1
%3 = sub nsw i32 %0, %2
```

Target Positioning

- *Narrow the scope to avoid large-scale detection*
- *Increased concealment*
- *Often Used in APT*
- *Main Work: Collect information about the target*

DeepLocker

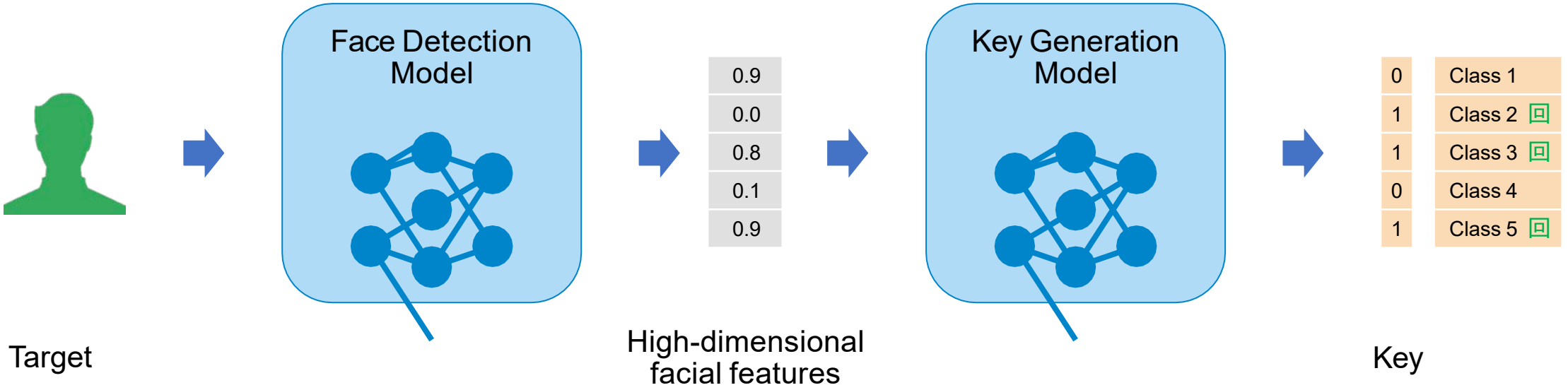
- AI-Powered Concealment and Unlocking [Blackhat USA 2018, Dhilung Kirat, etc.]



Deep Puzzling: Binary Code Intention Hiding based on AI Uninterpretability | Jifeng Zhu, Keyun Luo

Key Guaranteed Security

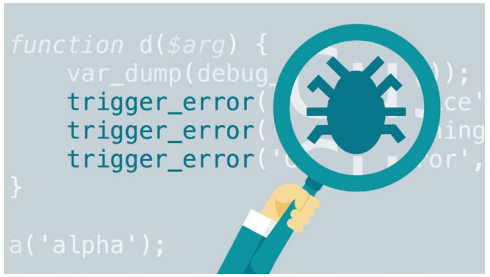
- *Face Detection as feature input*
- *The length of the key is critical*

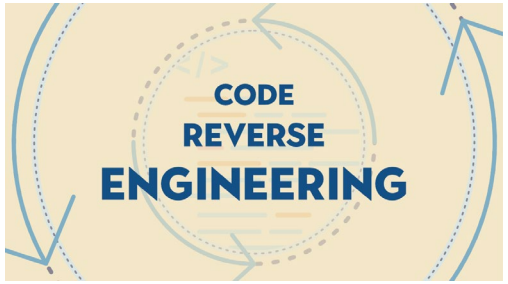


Deep Puzzling: Binary Code Intention Hiding based on AI Uninterpretability | Jifeng Zhu, Keyun Luo

Debugging will Reveal Everything

- ***Debuggability :*** The adversary can discover the key role of the key
- ***Attribute exposure risk:*** There is a certain correlation between attribute and intention
- ***Reverse analysis: The key may be violently enumerated***
  - Brute-force key
  - Brute-force attributes
  - Deceptive attributes





Deep Puzzling: Binary Code Intention Hiding based on AI Uninterpretability | Jifeng Zhu, Keyun Luo

From Attack to Defend

- *Intent Hiding + Anti-Debugging: Binary Code Protection*
- ***Attacker :*** hide the attack intent
- ***Defender:*** protect the core code from being reversed

Motivation

- ***Multiple Task***
  - Task 1、 Task 2、 Task 3 …
  - Anti-Debug
- ***Implicit Feature***
  - Intent is hidden in massive plain data
- ***Generated Payload***
  - The program does not store the payload directly
- ***End-to-End model***
  - Avoid tampering with intermediate values

Why it Matters for Security

The boom of AI technology

AI is steadily emerging as part of the agriculture industry's technological evolution. AI-powered solutions will not only enable farmers to improve efficiencies but they will also improve quantity, quality and ensure faster go-to-market for crops.

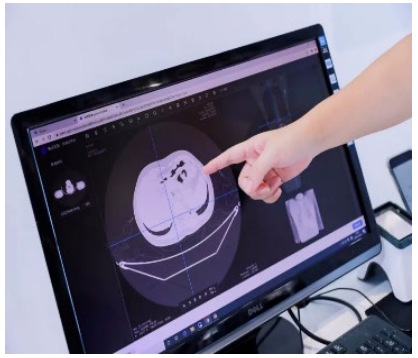
AGRICULTURE

Big data and AI give industry a huge boost. AI can use the high volumes of data generated by a factory to identify trends and patterns that can then be used to make manufacturing processes more efficient and reduce their energy consumption.

INDUSTRY

AI techniques are being increasingly deployed in finance, in areas such as asset management, algorithmic trading, credit underwriting and blockchain-based finance, enabled by the abundance of available data and by sufficient computing capacity.

FINANCE

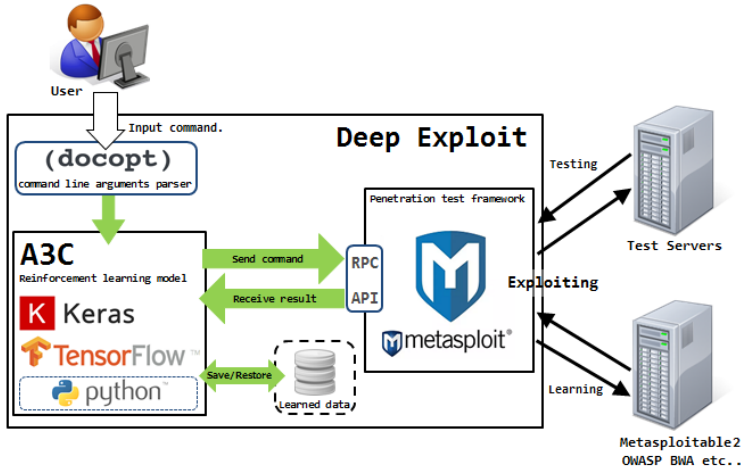


AI can be an assistant to doctors. AI-powered solutions can learn from various medical images such as endoscopy, ultrasound, CT, MRI, pathology, OCT, and etc., and then help the doctors to diagnose and screen major diseases in advance.

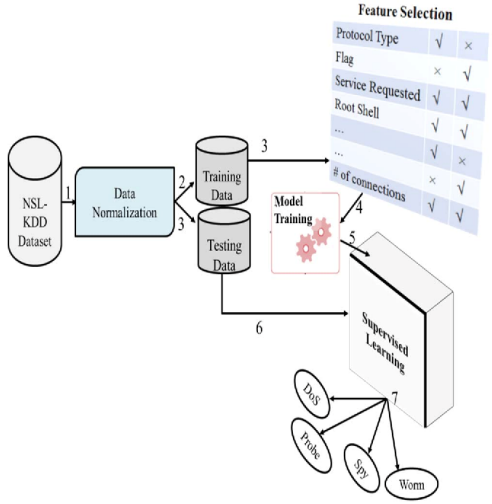
MEDICINE

AI-Powered Cybersecurity

- ***Attack***
  - Vulnerability： Vuldeepecker
  - Trojan：MalwareGAN
  - Phishing：SNAP_R
  - Penetration：Deep Exploit



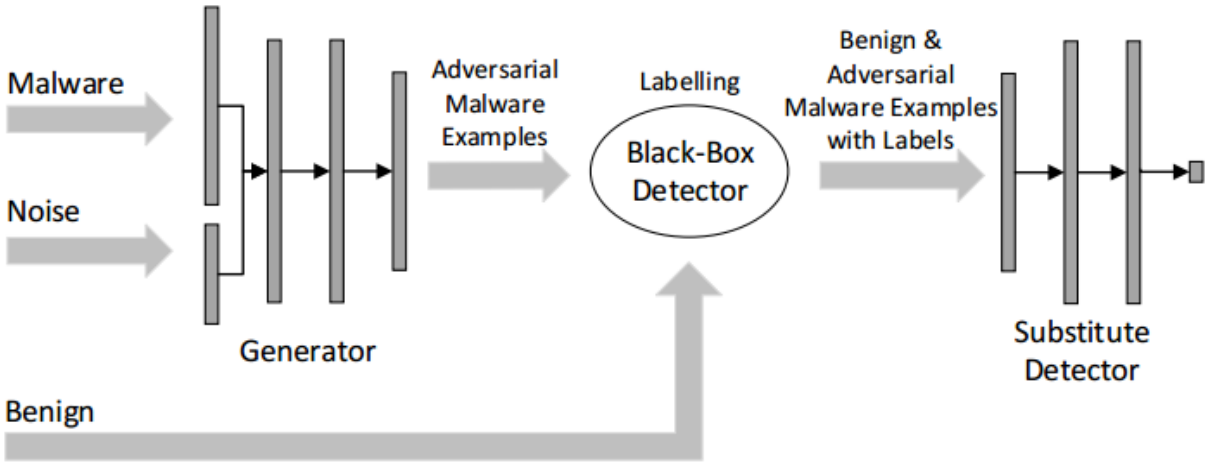
- ***Defense***
  - Network Intrusion Detection
  - Web application firewall
  - Malware detection
  - UEBA



Deep Puzzling: Binary Code Intention Hiding based on AI Uninterpretability | Jifeng Zhu, Keyun Luo

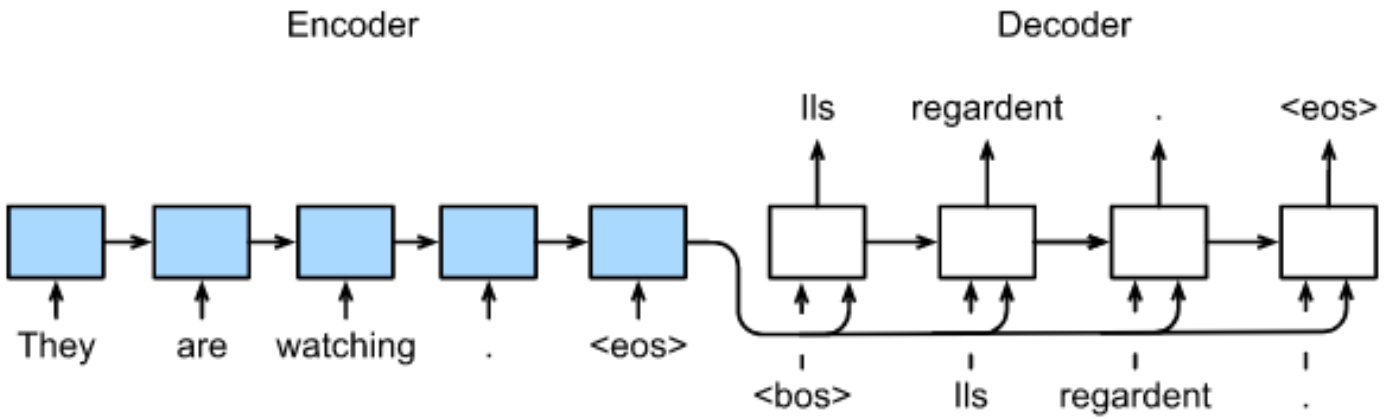
Success Factors

- *Learning the patterns behind the data*
  - Massive Data
  - Detection & Bypass Pattern



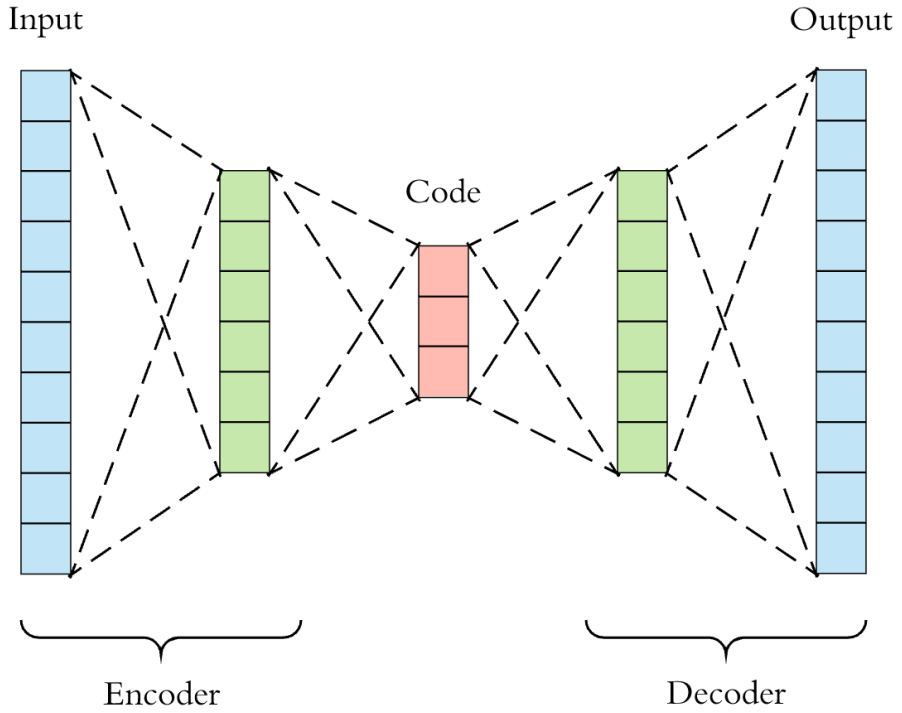
The Generative Characteristic of AI

- *The encoder-decoder architecture*
  - The encoder takes a variable-length sequence as the input and transforms it into a state with a fixed shape.
  - The decoder maps the encoded state of a fixed shape to a variable-length sequence.



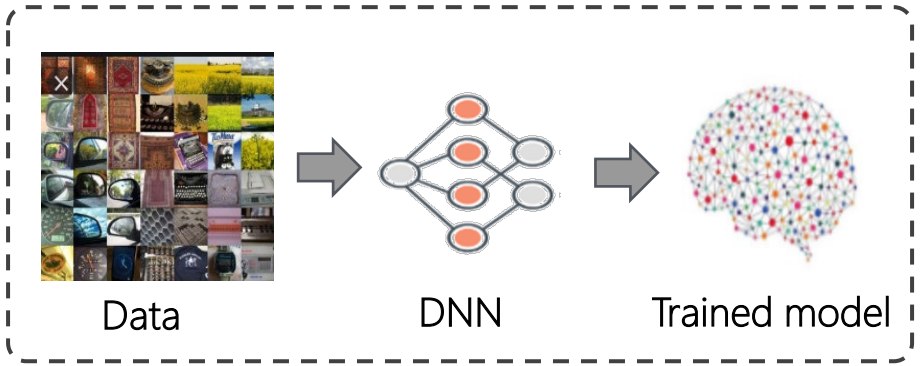
The Memory Characteristic of AI

- ***Autoencoder: A special case of sequence generation model***
  - Generating ability naturally means memory
  - Many parameters in the original model are redundant
  - No Labels are needed
  - It is able to restore original data
  - Compress information to reduce volume



The Blackbox Characteristic of AI

- *AI model consists of network structure and weight parameters*
- *It is difficult to explain why such a network structure is effective or not*
- *It is difficult to explain the meaning of each parameter*
- *It is difficult to explain what features the model has learned*



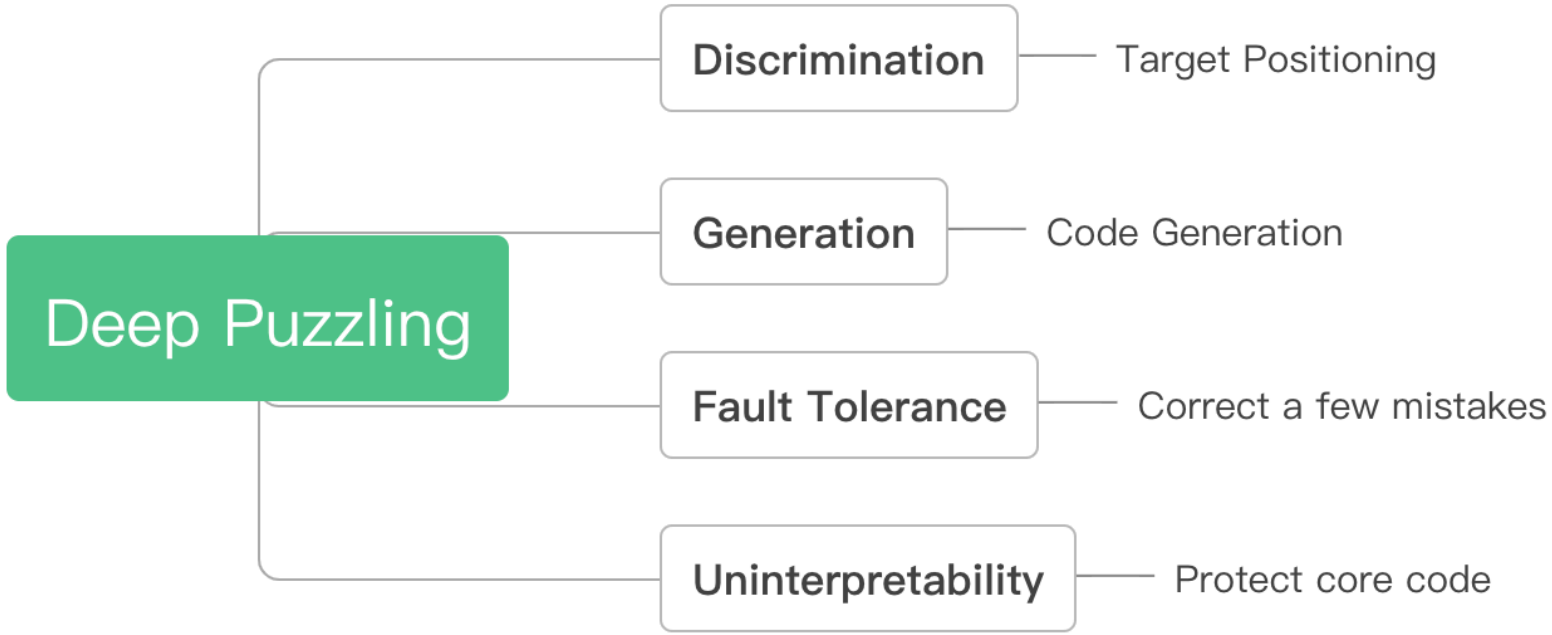

```
tensor([[ 0.0347, -0.0409, -0.0340,  ...,  0.1204,  0.0096,  0.0436],
        [-0.0187,  0.0556,  0.0537,  ...,  0.1807,  0.0240,  0.0440],
        [ 0.0429,  0.0072,  0.0365,  ...,  0.1516, -0.0481,  0.0082],
        ...,
        [ 0.0328,  0.0037,  0.0140,  ...,  0.0623,  0.0301,  0.0055],
        [ 0.0711,  0.0254,  0.0026,  ...,  0.0850, -0.0125,  0.0356],
        [-0.0025, -0.0615, -0.0336,  ...,  0.1774, -0.0324,  0.0663]],
```

What do these Characteristics mean

- *Powerful sequence generation ability means greater capacity*
  - It's easy to generate a passage
  - It is able to adjust the decoding output according to the input of the encoder
- *Advanced memory ability means fault tolerance*
  - Damaged data recovery
- *Poor interpretability means the natural difficulty of reverse analysis*
  - Brute force enumeration of massive parameters is almost impossible
  - We can't easily find the relationship between data input and result output
  - Even if the model is obtained, it is difficult to infer the part of the data that can really work

Make use of AI Power

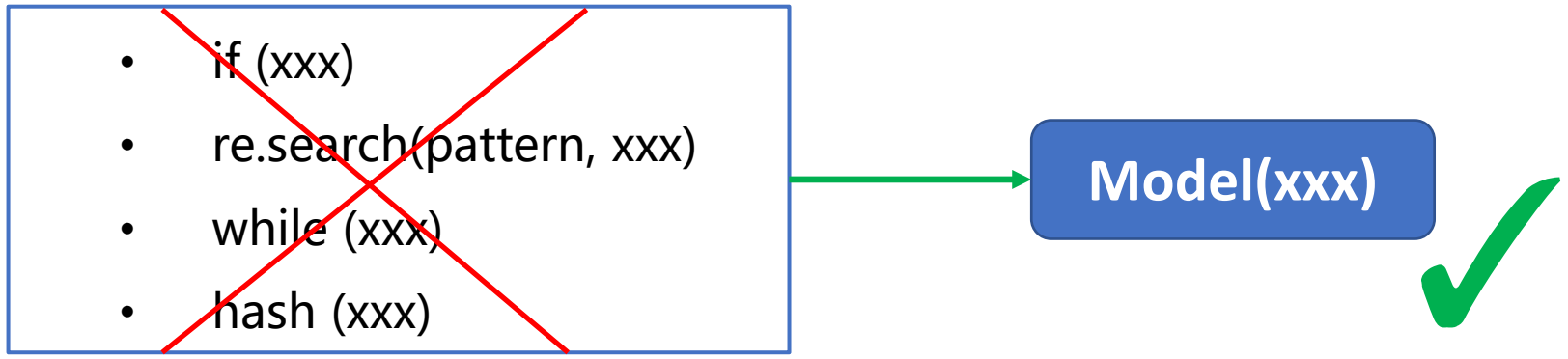
- *AI-enabled security practices based on generative ability and uninterpretability*



Deep Puzzling Details

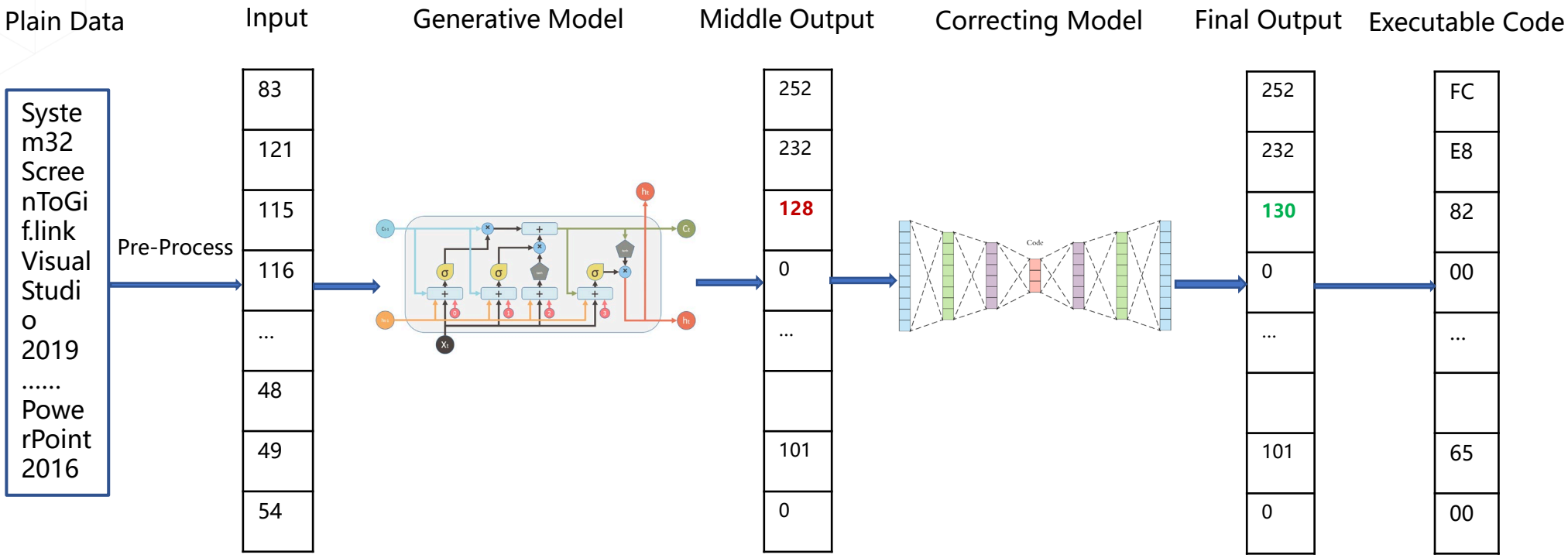
The Core Idea

- *Avoid explicit conditional judgments in code execution*

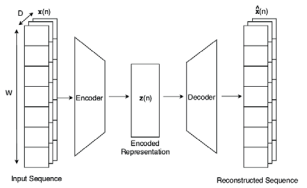


- *Let the AI model make black-box decisions*

Method Overview



Method Overview



C++ Loader

- Read Model
- Collect Data
- Generate Code

Data & Feature

Runtime

- Process
- File Name
- Service
- Path Env
- Regedit

Feature

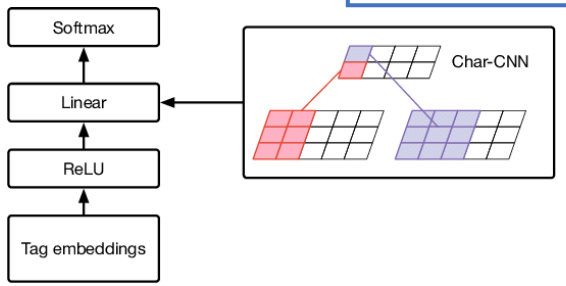
- Conv1D
- Max Pooling
- Dense

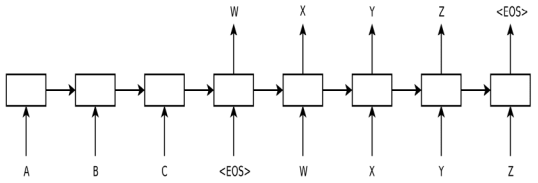
Error Correction: LSTM AutoEncoder

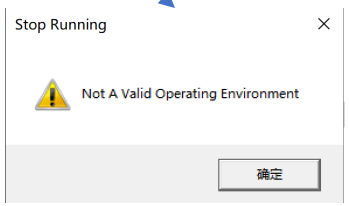
Intent hiding

Execute Code

Sequence Generation: BiLSTM

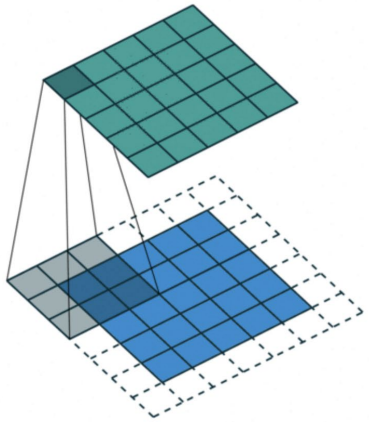




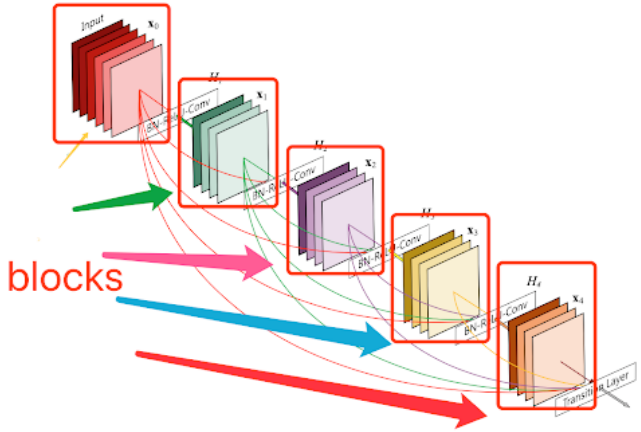


Feature Extraction

- *How to model the "hidden backdoor" in massive features?*
- *How to ensure the model's "robustness" to different input spaces?*



Conv1D

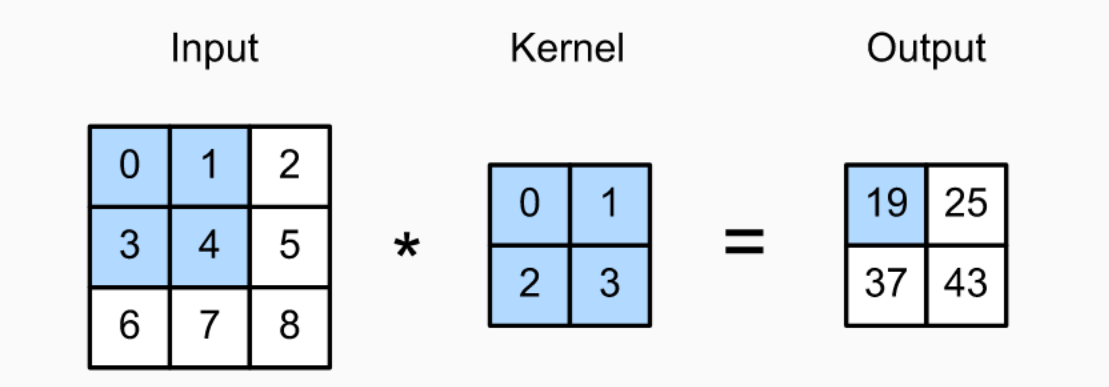


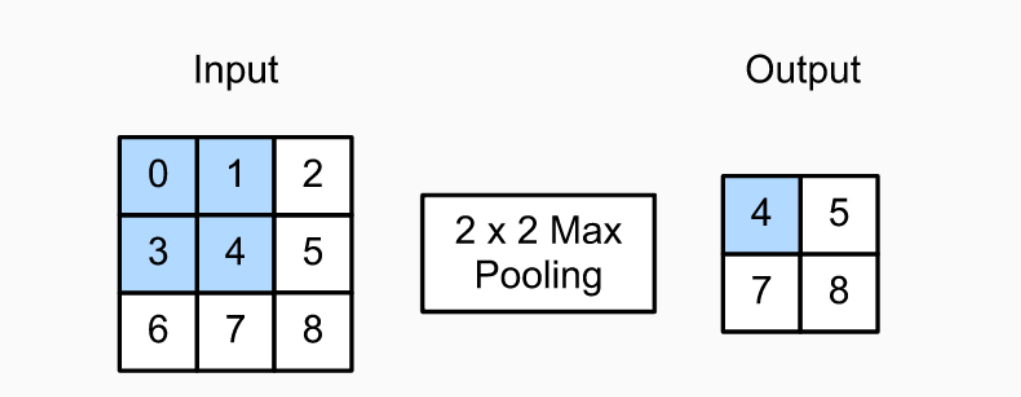
Dense Connect

TextInpu;.java;wininit.exe
.java;wininit.exe; TextInput
java; wininit;TextInput
JAVA;wininit.exe;TextInpu

Data Augmentation

Feature Extraction





Convolutions

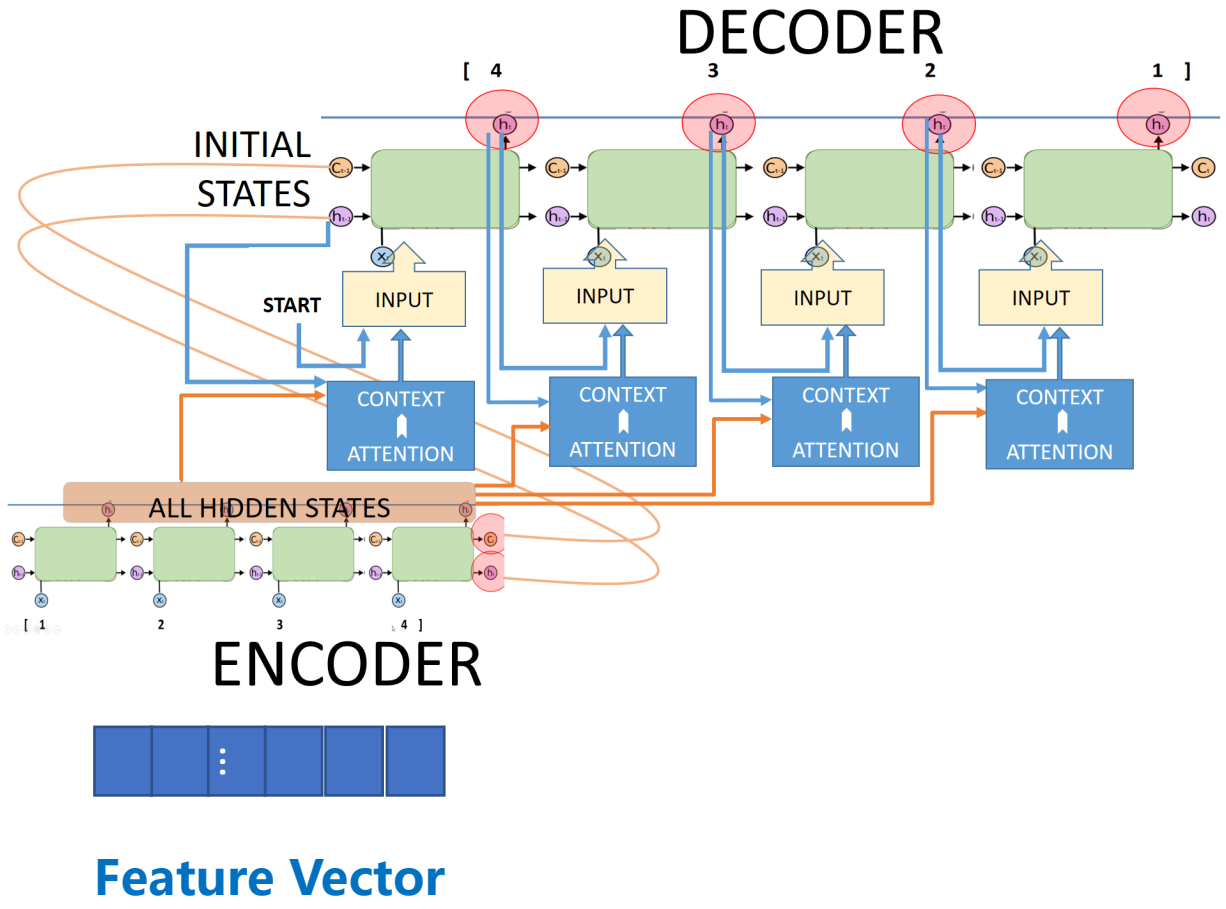
$0\times0+1\times1+3\times2+4\times3=190\times0+1\times1+3\times2+4\times3=19$

Pooling

$\max(0,1,3,4)=4\max(0,1,3,4)=4$

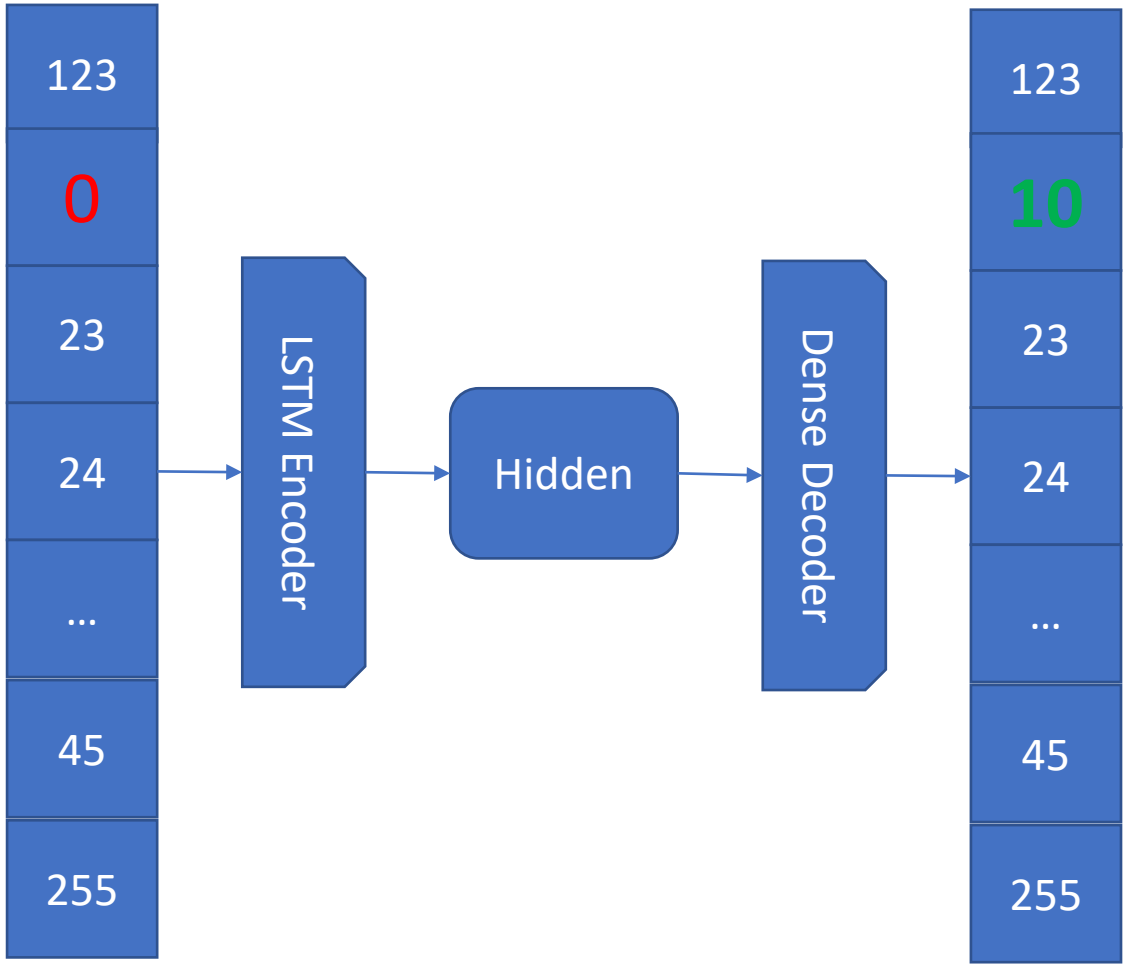
Binary Code Generation

1. *Repeat Feature Vector to MAX_LEN*
2. *Use BiLSTM for Sequence modeling*
3. *Use Full Connect for Hidden State*
4. *Use Softmax to output the character probability of each position*



Binary Code Repair

- *Deep neural network is a probabilistic model*
- *If a certain bit is wrong, then the entire shellcode will not be executed*
- *Use an autoencoder to correct a small number of errors*



Demonstration and Analysis

Target

- *Hidden intent*
  - Target 1： for ordinary people, execute a MessageBox code
    - Feature: Clean office system
  - Target 2: for developers, execute a Calculator code
    - Feature: Visual Studio、Python、SSH，etc.
- *Anti-debugging*
  - Target3: for debuggers, no code execution
    - Feature: IDA、OllyDbg、WinDbg，etc.

Main Steps

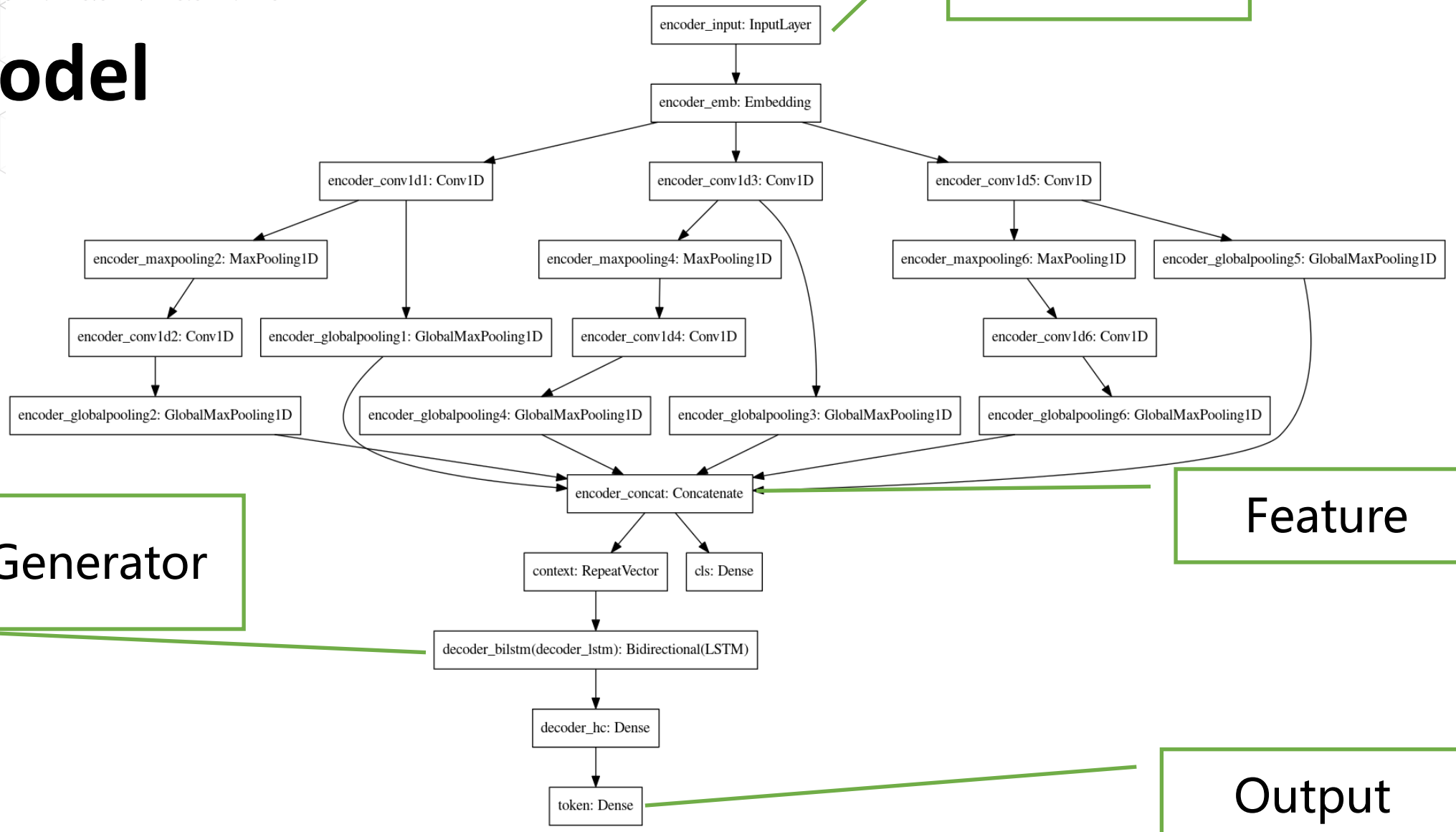
- ***Make Dataset***
  - Environmental Data： Shuffle、Crop、Random Pick， about 20,000 characters
  - Label: Target group and corresponding shellcode
  - Error Correcting Code Data：0～2 bit error per 10 bytes
- ***Train and Convert Model***
  - TensorFlow/Keras: Ensure an accuracy of more than 99% on the validation set
  - FDeep Convert: H5 Model ➔ Json File ➔ Dump to Resource File
- ***C++ Inference***
  - Collect Data ➔ Load Model ➔ Decode Shellcode ➔ Fix Code ➔ Execute

Input and Output

- **Data**
  - Process Names
  - Service Names
  - Desktop/Startup Menu Files
  - Regedit Keys
- **Model**
  - CNN Neural Networks
  - Max Pooling Aggregate
  - Concatenate Multiple Dense Part
  - BiLSTM Sequence Decoder
  - LSTM Auto Encoder

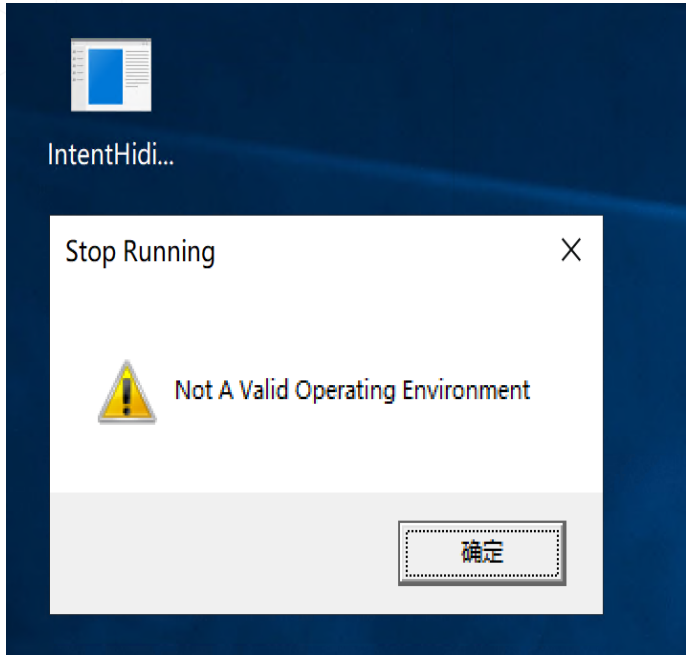
Input: "SysMain;.part;RmSvc;DeviceAssociatio;...; wininit.exe;Java"
Output: [252, 232, 130, 0, 100, 0, 96, 137, 229,..., 101, 120, 101, 0]
Revised: [252, 232, 130, 0, **0**, 0, 96, 137, 229,..., 101, **121**, 101, 0]

Model


Data
encoder_input: InputLayer
encoder_emb: Embedding
encoder_conv1d1: Conv1D
encoder_conv1d3: Conv1D
encoder_conv1d5: Conv1D
encoder_maxpooling2: MaxPooling1D
encoder_maxpooling4: MaxPooling1D
encoder_maxpooling6: MaxPooling1D
encoder_globalpooling5: GlobalMaxPooling1D
encoder_conv1d2: Conv1D
encoder_globalpooling1: GlobalMaxPooling1D
encoder_conv1d4: Conv1D
encoder_conv1d6: Conv1D
encoder_globalpooling2: GlobalMaxPooling1D
encoder_globalpooling4: GlobalMaxPooling1D
encoder_globalpooling3: GlobalMaxPooling1D
encoder_globalpooling6: GlobalMaxPooling1D
encoder_concat: Concatenate
Feature
Generator
context: RepeatVector
cls: Dense
decoder_bilstm(decoder_lstm): Bidirectional(LSTM)
decoder_hc: Dense
token: Dense
Output

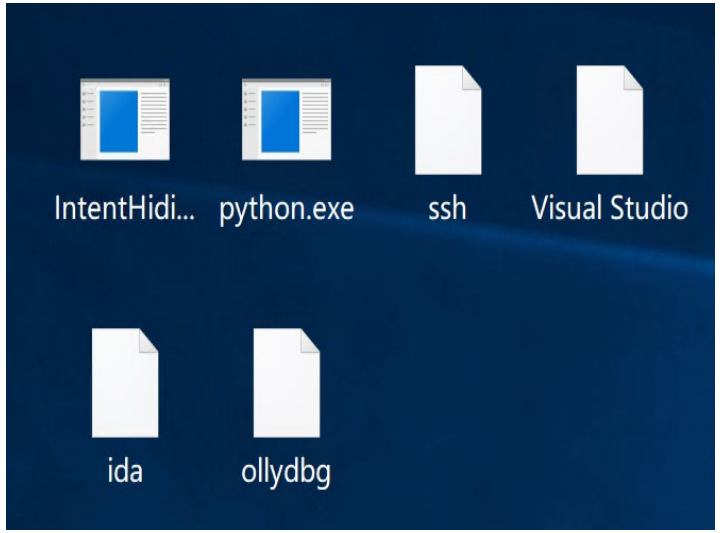
Demonstration

- *Target 1*



Target 2

Target 3



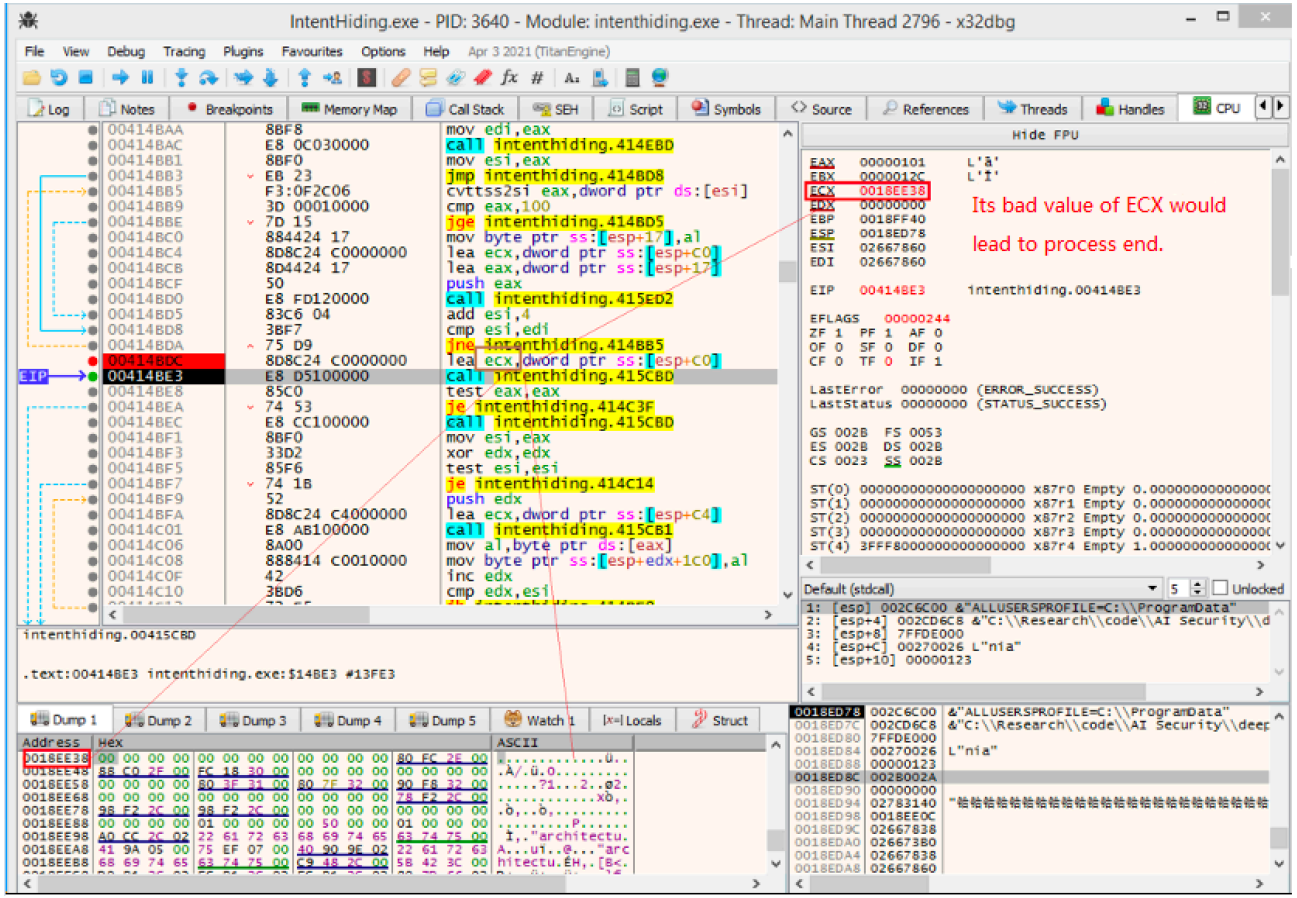
Analysis

- *We do not seek to use any techniques to obfuscate the source code*
- *It's not difficult to locate the code that runs the "intent", the difficulty lies in when it will be triggered*
- *The key point is that the address of the decrypted code is dynamically obtained*

```
232  if ( sub_415CBD((int)&v84) )
233  {
234    v58 = sub_415CBD(v57);
235    v59 = 0;
236    if ( v58 )
237    {
238      do
239      {
240        v60 = (char *)sub_415CB1(&v84, v59);
241        Src_build_intend_code[v61] = *v60;
242        v59 = v61 + 1;
243      }
244      while ( v59 < v58 );
245    }
246    Pcode_buffer = VirtualAlloc(0, 0x1000u, 0x3
247    memcpy(Pcode_buffer, Src_build_intend_code,
248    ((void (*)(void))Pcode_buffer)();
249  }
```

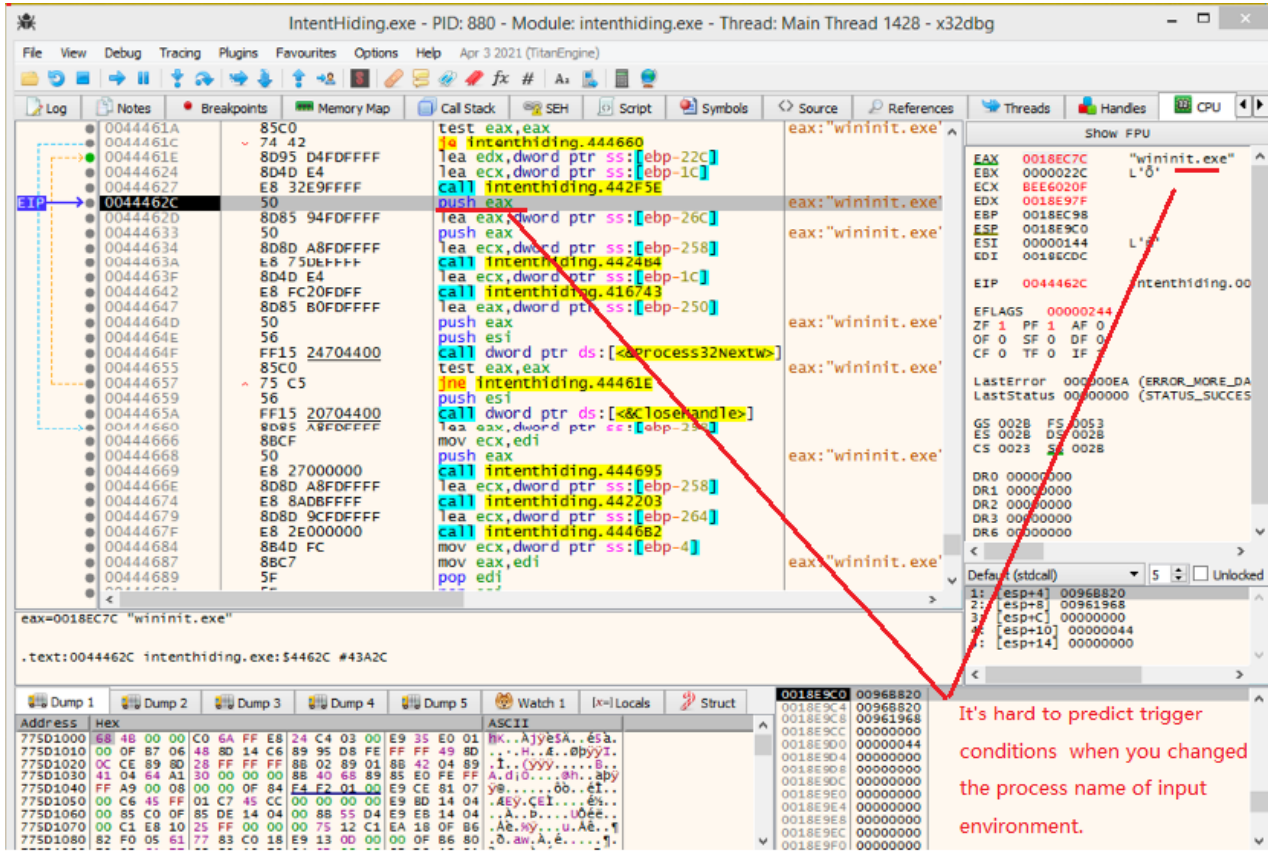
Analysis

- *If the ECX value here is not the "input" of our intended purpose, it will be the wrong value, and There will be no payload running*
- *We can see the important role of multitasking here*



Analysis

- *The "input" we mentioned here is a high-dimensional space vector*
- *It is difficult to infer the real "purpose" simply by modifying some data because of Robustness of AI model*



Analysis

- *If the trigger environment is not met, the ECX value cannot be obtained*
- *Finding "rules" in massive data is difficult*
- *The AI model is neutral and does not contain any intent code directly , This is very different from previous designs*

```
const auto result = decoder_model.predict(
        {
          fdeep::tensor(fdeep::tensor_shape(static_cast<std::size_t>(max_feature_len)),
feature_float),
        });
auto probes = result[0].to_vector();
```

Conclusion

- We propose a ***binary code intention hiding framework*** based on deep sequence generation network
- We conduct an experiment to hide ***MessageBox*** and ***Calculator*** to avoid being discovered in the debugging environment
- Our experiment confirmed that due to the powerful distinguishing ability and ***uninterpretability*** of the AI , it greatly increases the difficulty of analysis for reverse engineers
- The combination of AI and security will magnify the ***offensive and defensive*** situation in Cybersecurity

Take away

- *frugally-deep: Use Keras models in C++ with ease*
  - https://github.com/Dobiasd/frugally-deep
- *CNN Network for Feature Extraction*
  - https://keras.io/api/layers/convolution_layers/convolution1d
- *Encoder – Decoder Model for Sequence Generation*
  - https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html
- *Autoencoders for Error Correction*
  - https://blog.keras.io/building-autoencoders-in-keras.html

Thank You

Email： keyunluo@tencent.com. jifengzhu@tencent.com

Join our Discord channel to discuss more or ask questions

https://discord.gg/dXE8ZMvU9J