以視覺化AI打造特徵碼專家系統

卷積分類惡意程式家族，我們走到了哪？

Shenghao Ma, Team Lead

PSIRT and Threat Research, TXOne Networks Inc.

April 22, 2025 @CYBERSEC 2025

Shenghao Ma

Team Lead, PSIRT and Threat Research at TXOne Networks Inc.

- Sheng-Hao Ma (@aaaddress1) is a team lead of TXOne Networks PSIRT and threat research team, responsible for coordinating product security and threat research. With over 15 years of expertise in reverse engineering, symbolic execution, malware analysis, and machine-learning, he is also part of CHROOT, a cybersecurity community in Taiwan.
- As a frequent speaker, trainer, and instructor, Sheng-Hao has contributed to numerous international conferences and organizations, including Black Hat USA, DEFCON, CODE BLUE, S4, SECTOR, HITB, VXCON, HITCON, and ROOTCON, as well as the Ministry of National Defense and the Ministry of Education. He is the author of "Windows APT Warfare: The Definitive Guide for Malware Researchers," a well-regarded cybersecurity book about reverse engineering of Windows.

Contents

Background

The weakness of N-Gram and the other NLP based techniques on raw-binary detection

NLP Solution via Convolutional Approach

Yann LeCun, and Facebook AI Research team proof the possibility of using convolutional neural network as Transformer to use, e.g. Seq2Seq and Convolutional neural network language models.

Use CNN for Binary Detection without Feature Engineering

2 AAAI papers provide the concept to use file raw-data as input for malware detection. Also, proposed a new additive attention technique, GCC, based on gated CNN for large-file detection


NVIDIA DEVELOPER

Simulation / Modeling / Design

Malware Detection in Executables Using Neural Networks

Nov 21, 2017 +2 Like Discuss (4)

By Jon Barker

Cornell University

Statistics > Machine Learning

arXiv:1710.09435 (stat)

[Submitted on 25 Oct 2017]

Malware Detection by Eating a Whole EXE

Edward Raff, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, Charles Nicholas

In this work we introduce malware detection from raw byte sequences as a fruitful research area to the larger machine learning community. Building a neural network for such a problem presents a number of interesting challenges that have not occurred in tasks such as image processing or NLP. In particular, we note that detection from raw bytes presents a sequence

Cornell University

Statistics > Machine Learning

arXiv:2012.09390 (stat)

[Submitted on 17 Dec 2020]

Classifying Sequences of Extreme Length with Constant Memory Applied to Malware Detection

Edward Raff, William Fleshman, Richard Zak, Hyrum S. Anderson, Bobby Filar, Mark McLean

Recent works within machine learning have been tackling inputs of ever-increasing size, with cybersecurity presenting sequence classification problems of particularly extreme lengths. In the case of Windows executable malware detection, inputs may exceed 100

Background

Hand-written rules is expensive.

- **Annual reports from Security vendors**
  - VirusTotal in 2022 – Daily receive 2,000,000 new unknown binaries
  - Kaspersky in 2023 – Cybercriminals release 411,000 new malware
  - Impossible to rule them all by traditional rule engines ☹

- **Human experts might be tired, but machine won't.**
  - Rule? Can we use N-Gram to judge the probability/Similarity of unknown binaries from byte-sequence its keep as known malware artifacts?
  - **As traditional NLP challenge.**

$$P(w_{1:n}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2})\ldots P(w_n|w_{1:n-1})$$
$$= \prod_{k=1}^{n} P(w_k|w_{1:k-1})$$

```
rule upx {
    meta:
        description = "UPX packed file"
        block = false
        quarantine = false

    strings:
        $mz = "MZ"
        $upx1 = {55505830000000}
        $upx2 = {55505831000000}
        $upx_sig = "UPX!"

    condition:
        $mz at 0 and $upx1 in (0..1024)
                 and $upx2 in (0..1024)
                and $upx_sig in (0..1024)
}
```

Co-occurrence by Markov Assumption

- N-Gram and the Markov assumption @ Stanford - Speech and Language Processing

$$P(w_{1:n}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2})\dots P(w_n|w_{1:n-1})$$
$$= \prod_{k=1}^{n} P(w_k|w_{1:k-1})$$

```
# Count frequency of co-occurance
for sentence in reuters.sents():
    for w1, w2, w3 in trigrams(
        sentence, pad_right=True, pad_left=True):
        model[(w1, w2)][w3] += 1

# Let's transform the counts to probabilities
for w1_w2 in model:
    total_count = float(sum(model[w1_w2].values()))
    for w3 in model[w1_w2]:
        model[w1_w2][w3] /= total_count
```

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)} \quad (3.10)$$

We can simplify this equation, since the sum of all bigram counts that start with a given word w_{n-1} must be equal to the unigram count for that word w_{n-1} (the reader should take a moment to be convinced of this):

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})} \quad (3.11)$$

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Figure 3.5 Three sentences randomly generated from three n-gram models computed from 40 million words of the *Wall Street Journal*, lower-casing all characters and treating punctuation as words. Output was then hand-corrected for capitalization to improve readability.

Background

- University of Maryland @ Journal of Computer Virology and Hacking Techniques 2016 ***"An investigation of byte n-gram features for malware classification"***
  - N-Gram length > 12 tokens (n >= 12)
    - Hard to modeling due to performance issues of commercial products in Markov assumption, when > 200,000 signatures; The fact led to most commercial engines ONLY USE 4-gram or 6-gram with feature filtering methods on PE content, as the idea of PE-miner.
  - Still overfit, even if best performance via 45-gram
    - Even if variants of the same family, which in practice might don't have the co-occurrence sub-slices.
- *Raff, Edward, Jared Sylvester, and Charles Nicholas.* ***"Learning the pe header, malware detection with minimal domain knowledge."*** *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. 2017.*
  - ➢ **N-Gram → LSTM & Attention**

The output of our attention mechanism is a weighted average of the hidden states (2).

$$\text{Attention Output} = \sum_{i=1}^{T} \alpha_i \boldsymbol{h}_i \qquad (2)$$

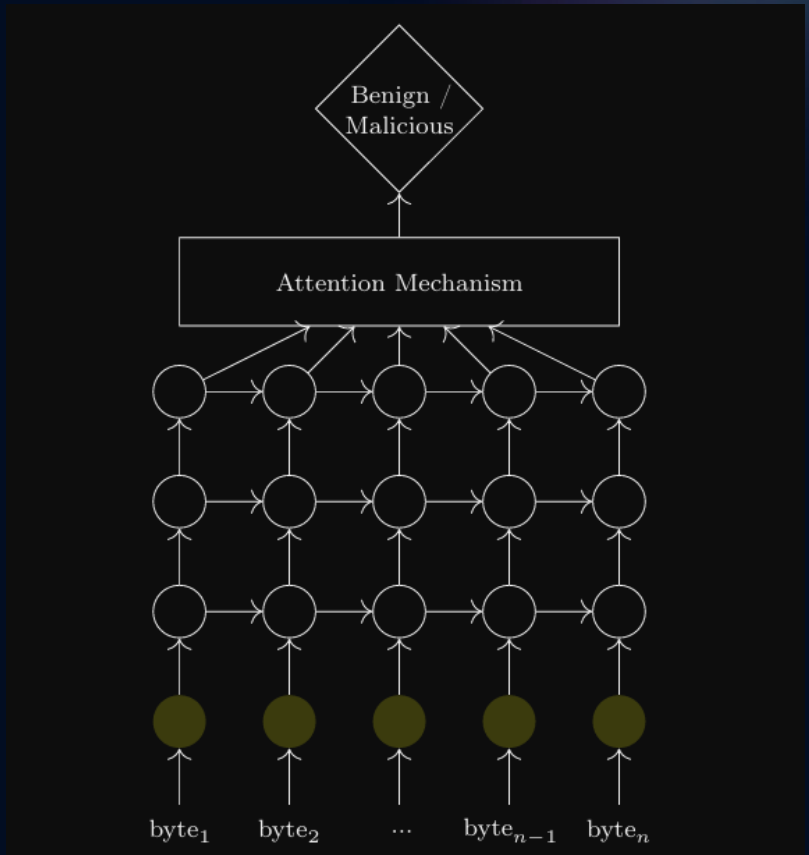
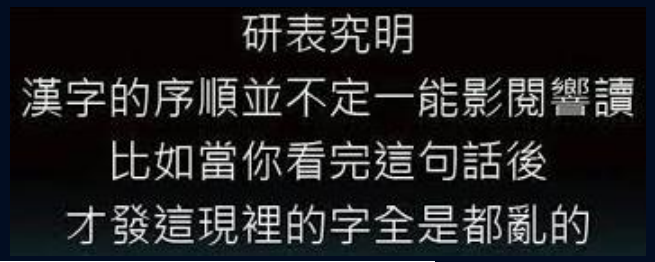


Fig. 1. Attention LSTM architecture used. Blue, shaded circles indicate embedding layers, white circles indicate LSTM layers. A 3-layer LSTM processes each byte in sequence, and the activations of all time steps are merged into the attention mechanism. The result from the attention mechanism is then used to make a classification decision.

Background



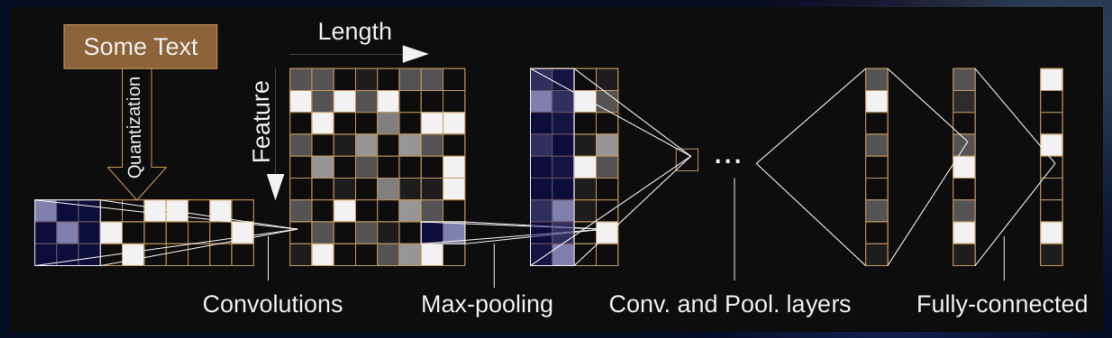
An absolutely great movie! I watched the premiere with my friends.

The movie about cats was absolutely great, and the cats were cute.

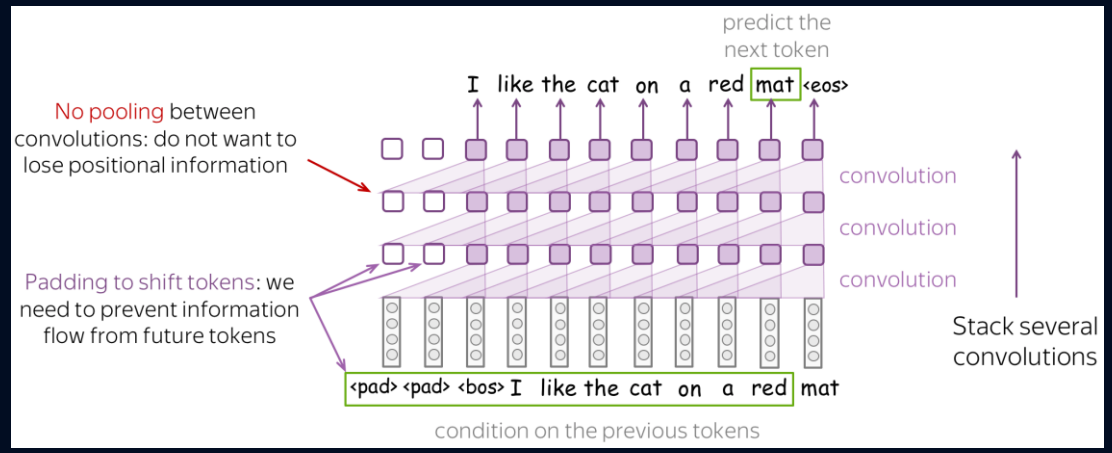
The movie is about cats running around, and it is absolutely great.

If a clue is very informative, maybe we don't care much where in a text it appears?

- **Convolutional Neural Networks for Text (**lena-voita.github.io/nlp_course)
  - *"…Then there are some words and phrases which could be very informative "clues" (e.g. it's been great, bored to death, absolutely amazing, the best ever, etc), and others which are not important at all. We don't care much where in a text we saw bored to death to understand the sentiment, right?"*
- **Zhang, Xiang, Junbo Zhao, and Yann LeCun. *"Character-level convolutional networks for text classification."* Advances in neural information processing systems 28 (2015).**
  - **5. Discussion - Character-level ConvNet is an effective method.** *The most important conclusion from our experiments is that character-level ConvNets could work for text classification without the need for words. This is a strong indication that language could also be thought of as a signal no different from any other kind.*
- *Pham, Ngoc-Quan, German Kruszewski, and Gemma Boleda. "Convolutional neural network language models." Proceedings of the 2016 conference on empirical methods in natural language processing. 2016.*
- (Facebook AI Research) Gehring, Jonas, et al. *"Convolutional sequence to sequence learning."* International conference on machine learning. PMLR, 2017.



Model	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW	11.19	7.15	3.39	7.76	42.01	31.11	45.36	9.60
BoW TFIDF	10.36	6.55	2.63	6.34	40.14	28.96	44.74	9.00
ngrams	7.96	2.92	1.37	**4.36**	43.74	31.53	45.73	7.98
ngrams TFIDF	**7.64**	**2.81**	**1.31**	4.56	45.20	31.49	47.56	8.46
Bag-of-means	**16.91**	**10.79**	**9.55**	**12.67**	**47.46**	**39.45**	**55.87**	**18.39**
LSTM	13.94	4.82	1.45	5.26	41.83	29.16	40.57	6.10
Lg. w2v Conv.	9.92	4.39	1.42	4.60	40.16	31.97	44.40	5.88
Sm. w2v Conv.	11.35	4.54	1.71	5.56	42.13	31.50	42.59	6.00
Lg. w2v Conv. Th.	9.91	-	1.37	4.63	39.58	31.23	43.75	5.80
Sm. w2v Conv. Th.	10.88	-	1.53	5.36	41.09	29.86	42.50	5.63



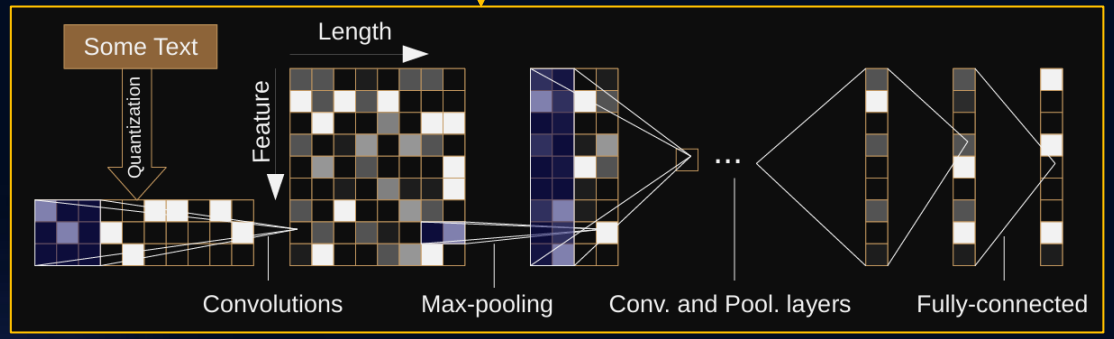
txOne networks

MalConv 1:

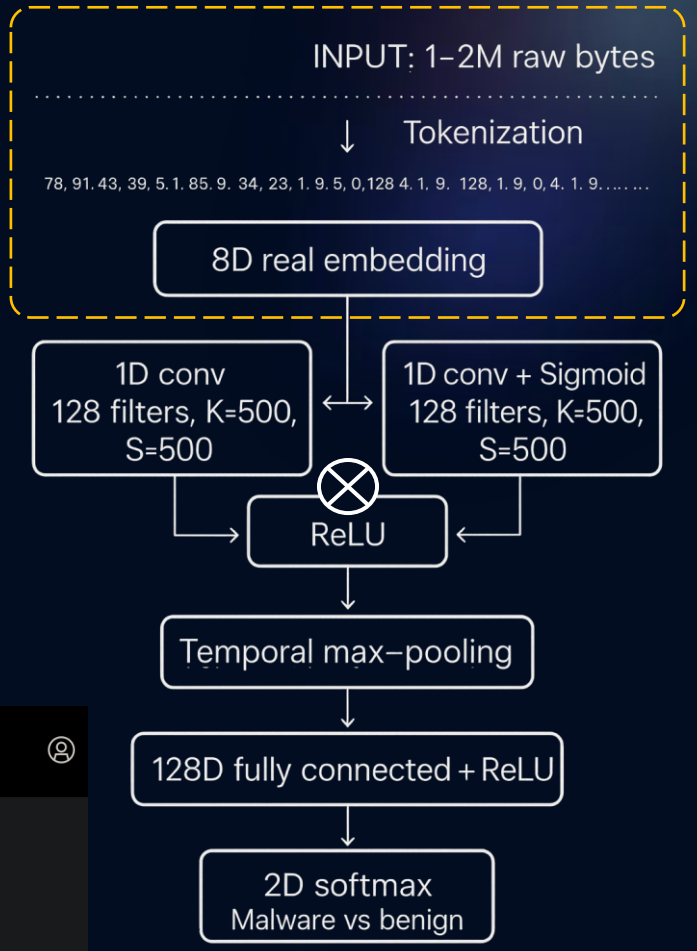
Detection without Manual Feature Engineering

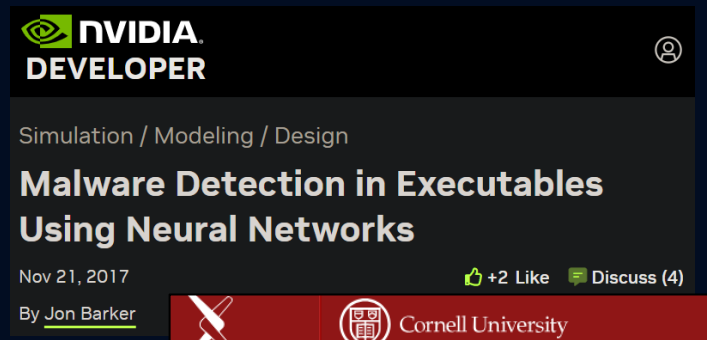
Detection without Manual Feature Engineering

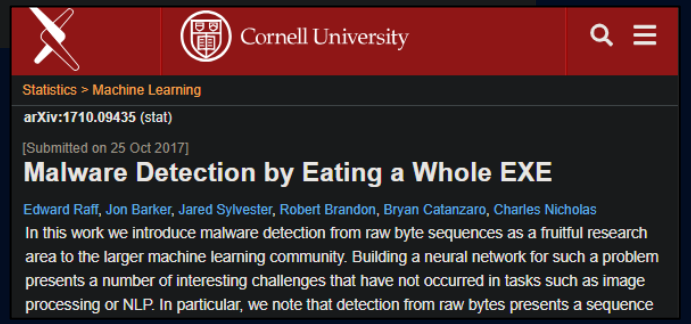
- **AAAI Workshop'18 - Malware Detection by Eating a Whole EXE**
  - 美國物理科學實驗室、 NVIDIA、馬里蘭大學 聯合發表研究
  - *Learning the pe ~~header~~ raw data, malware detection with minimal domain knowledge*
  - Filter or PE-Miner is bad, feature selection led to the possibility to detect evasion
  - "Rather than perform convolutions on the raw byte values (i.e., using a scaled version of a byte's value from 0 to 255), we use an embedding layer to map each byte to a fixed length (but learned) feature vector. We avoid the raw byte value as it implies an interpretation that certain byte values are intrinsically "closer" to each-other than other byte values, which we know a priori to be false, as byte value meaning is dependent on context. … Prior work using byte n-grams lack this quality, as they are de pendent on exact byte matches (Kolter and Maloof 2006; Raff et al. 2016)."



Zhang, Xiang, Junbo Zhao, and Yann LeCun. *"Character-level convolutional networks for text classification."* Advances in neural information processing systems 28 (2015).







Gated convolutional networks with max-pooling

- Average-pooling vs. Temporal max-pooling
- GLU (Gated Linear Units) + Max Pooling

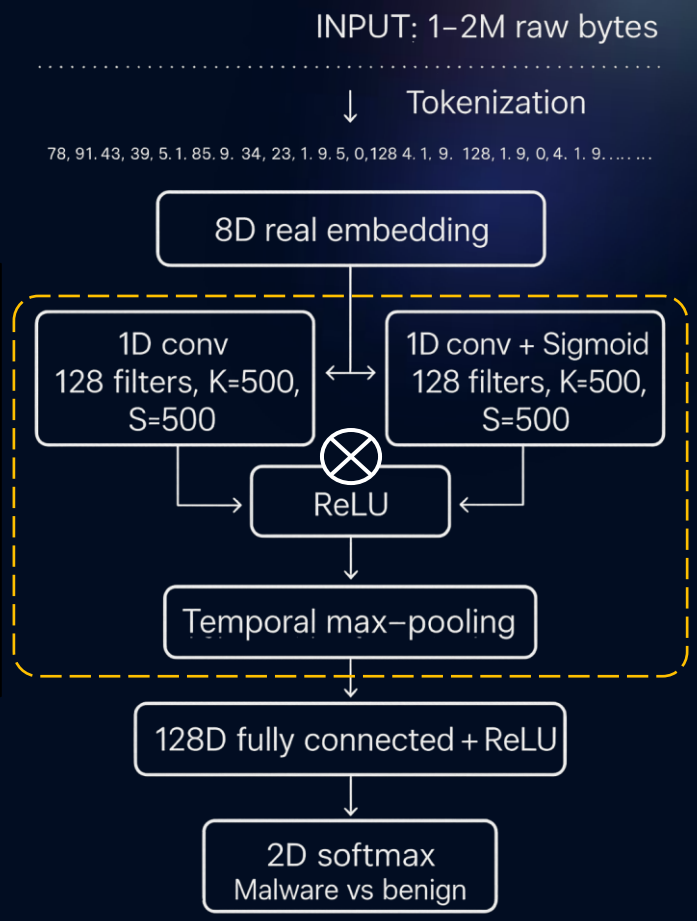
```python
class MalConv(nn.Module):
    def __init__(self, embed_dim, _, out_channels, window_size):
        super(MalConv, self).__init__()
        self.embed = nn.Embedding(257, embed_dim)
        self.dropout = nn.Dropout(0.5)
        self.conv = nn.Conv1d(
            in_channels=embed_dim,
            out_channels=out_channels * 2,
            kernel_size=window_size,
            stride=window_size,
        )
        self.fc = nn.Linear(out_channels, 1)

    def forward(self, x):
        embedding = self.dropout(self.embed(x))
        conv_in = embedding.permute(0, 2, 1)
        conv_out = self.conv(conv_in)
        glu_out = F.glu(conv_out, dim=1)
        values, _ = glu_out.max(dim=-1)
        output = self.fc(values).squeeze(1)
        return output
```

The issue of information sparsity is also a factor in our choice to use temporal max-pooling rather than average-pooling. Beyond providing better interpretability, max-pooling also provided superior performance relative to average-pooling. The latter enforces a prior that informative features should be widely occurring in the underlying file. But many features will occur only once in the file, and so when combined with average-pooling, that feature's high response in one region of the binary will be washed-out by the remaining majority of the file that produces a low activation. Max-pooling avoids this problem, while still allowing us to tackle the variable-length issue.

```python
for exe_path in tqdm(benign_files + mal_files):
    with open(exe_path, "rb") as f:
        exedata = f.read()[:4096]
        if b"MZ" not in exedata: continue
        file_ = torch.tensor([
            [byte_code & 0xff for byte_code in exedata]
        ])
        is_malicious = [1.0 if exe_path in mal_files else 0.0]
        dataset.append( (file_, torch.tensor(is_malicious)) )
```

```python
for inputs, labels in tqdm(dataset, leave=False):
    inputs = inputs.to(device)
    labels = labels.to(device)
    outputs = model(inputs)
    loss = criterion(outputs, labels)
```



Experiment and Results

- **Dataset A**
  - Benign – "ONLY" Windows native clean software and programs (21,854)
  - Malicious – Sample from Virus Share (43,967)
  - 模型學會作弊，看到 Binary 資料中帶有 Microsoft 字樣就判定為良性XD
- **Dataset B**
  - **Small dataset**
    - Benign and Malicious – Sample from anti-virus partners (200, 000 for each)
    - Malicious samples from das-malwerk.herokuapp.com
  - **Large dataset**
    - Benign and Malicious (1,011,766), total 2,011,786 samples.
- **四百萬樣本的 AUC 雙盲測試分數**
  - 表現優異 ~98.5% 輾壓傳統 n-gram 方法的 93.4 ~ 97% 偵測率
  - 若只看 PE Headers 的情況下偵測率 ~92%
  - 為何只看 PE Headers（前 4096 bytes）偵測率較低？
    - 有近四成重要特徵不在檔頭而在 section data 中。

Analysis of the sparse-CAM for 224 randomly selected binaries by an expert malware analyst revealed that our model had 39-42% of it's most important features located outside of the file header. In particular we saw indications of both executable code and data containing discriminative features.

NVIDIA DEVELOPER

Simulation / Modeling / Design

Malware Detection in Executables Using Neural Networks

Nov 21, 2017 +2 Like Discuss (4)

By Jon Barker

Train set	Test set	Our model AUC	Byte n-grams AUC	PE-Header Network
Group B train (small)	Group A	**98.5**	98.4	97.7
Group B train (small)	Group B test	95.8	**97.9**	91.4
Group B train (large)	Group A	**98.1**	93.4	–
Group B train (large)	Group B test	**98.2**	97.0	–

Table 1: Performance of malware detection models on Group A and Group B test sets when trained on both the small (400,000 samples) and large (two million samples) Group B training sets.

Section	Total	PE-Header	.rsrc	.text	UPX1	CODE	.data	.rdata	.reloc
Malicious	26,232	15,871	3,315	2,878	697	615	669	383	214
Benign	19,290	11,183	2,653	2,414	596	505	423	243	77

Table 2: Important features by file section, as determined by the non-zero regions of the sparse-CAM mapped to the output of the PE-file.

Detection without Manual Feature Engineering 2.0?

Problem of the MalConv

- **原始的 NVIDIA 顯卡軍火商林北有錢的撒錢模型太浪費記憶體了**
  - 128GB 的 GPU 顯卡只能夠最多偵測 ~2MB 大小的 EXE 程式
  - 也過分了吧，逼大家做解決方案都去買顯卡喔QQ
- **MalConv2: 可於極為有限的 GPU 下偵測高達 16GB 的程序**
  - **(AAAI 2021)** "Classifying Sequences of Extreme Length with Constant Memory Applied to Malware Detection"
  - VRAM ≦2GB for 2,0000,0000 time steps ≈ 16GB 大小的程序
  - 更長的視野窗口、不僅止於 PE 程序分類問題
    （落地商用時不必侷限於前 4096 bytes 的 headers）→ 通用型文件分類問題！
  - 美國物理科學實驗室、馬里蘭大學、美軍、微軟、Elastic 聯合發表
    - 夢幻連動... 全明星大亂鬥
  - https://arxiv.org/abs/2012.09390
  - https://github.com/FutureComputing4AI/MalConv2

Classifying Sequences of Extreme Length with Constant Memory Applied to Malware Detection

Edward Raff, [1,2,3] **William Fleshman,** [4] **Richard Zak,** [1,2,3]
Hyrum S. Anderson, [5] **Bobby Filar,** [6] **Mark McLean** [1]
[1] Laboratory for Physical Sciences, [2] Booz Allen Hamilton, [3] University of Maryland, Baltimore County
[4] U.S. Army, [5] Microsoft, [6] Elastic

Despite its simplicity, MalConv was the first architecture to demonstrate that neural networks could learn to perform malware detection from raw bytes and the first to show classification over time series/sequences of up to $T = 2,000,000$ steps. However, only the first 2 MB of the input was processed in training MalConv because it required 128 GB of GPU memory to train on a batch of 256 files up to the 2MB limit. This is owing to the large memory cost of performing an embedding and convolution over a time series of 2 million steps (1 for each byte), and the resulting activations alone require almost all of the GPU memory. Every subsequent work we are aware of has processed less than the original 2 MB cap.

Challenge of chunk strategy

- **AAAI'21 - Classifying Sequences of Extreme Length with Constant Memory Applied to Malware Detection**
  - 唉唷這不就工程問題嗎？記憶體不夠
    - 那就切 Chunk 批次掃描不行嗎？我一次掃不完、勤奮點多掃幾次不行？
  1. **(Problem 1.) AAAI Workshop'18 - Malware Detection by Eating a Whole EXE**
     - **2.1 Relate Work & 4.1 On Failed Architectures** *"In addition to the difficulties in dealing with the unusually long sequences that we confront, we must also contend witha lack of information flow. When making a benign/malicious classification of a binary we obtain only one error signal,which must be used to inform decisions regarding all 2 million time steps."*
  2. **(Problem 3.) Appendix C - Windowing Artifacts**
  3. **(Problem 2.) 碎片化的語義無法組合成全局視野感知 – 切片後的上下文各塊資訊整合的交叉語義**
     1. Ransomware: File Encryption + Enumerate Files + Delete Shadow Copy
     2. Shellcode Runner: asm_payload @ .data + __asm call rax @ .text
     3. Packers: Memory Decryption + PE Loader

To demonstrate why this is important in malware detection, consider that a common feature to learn/extract is the use of encryption libraries, which may indicate, for example, functionality common in ransomware. However, if the program does not access the file system, the use of encryption becomes less suspicious and less likely to indicate malware. In its current embodiment, it is impossible for MalConv to learn logic like this because the presence/absence of the associated information may be in disparate regions of the input, and the receptive window of the network (512 bytes) is far smaller than most inputs (2^{21} bytes).

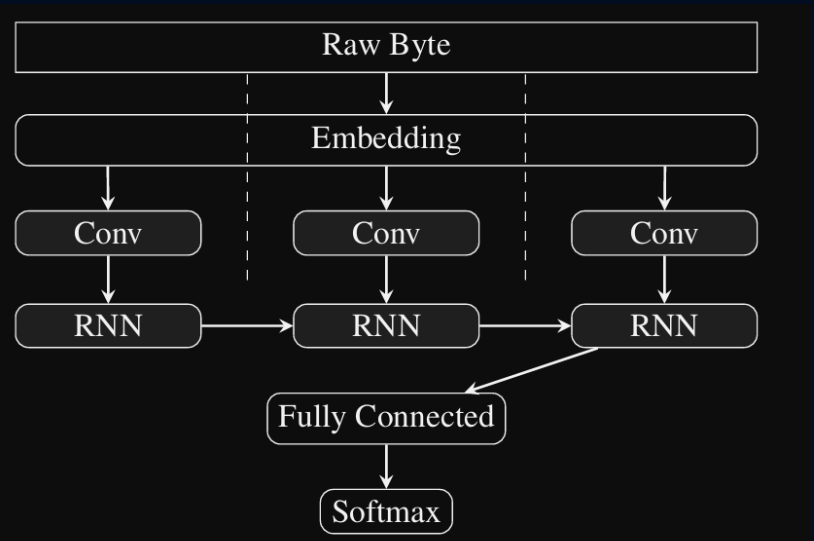


Figure 3. "Chunk" strategy where the embedding is split into parts. Dashed lines indicate outputs that were split into parts. Boxes with gray backgrounds and the same name indicate weight sharing.

Challenge of chunk strategy

Traditional Attention: Attention mechanisms have demonstrated considerable success in capturing long term dependencies among sequences (Luong, Pham, and Manning 2015; Vaswani et al. 2017; Lin et al. 2017; Devlin et al. 2019). Unfortunately, these global attention mechanisms require a quadratic increase in memory and runtime with sequence length. We explored several applications of attention including those from document classification (Yang et al. 2016; Zhang, Marshall, and Wallace 2016) which is the most similar NLP application to our domain. These works were limited to sequences several orders of magnitude smaller than what we require and took advantage of sub-document structure such as sentences to further reduce complexity. Current attention mechanisms remain intractable for our use case, with estimated epoch training times of at least 1 century for every method we have tried. This leads to our proposed GCG method the only viable mechanism for capturing long range dependencies in extremely long sequences with minimal additional overhead.

3. **(Problem 2.) 碎片化的語義無法組合成全局視野感知 – 切片後的上下文各塊資訊整合的交叉語義**
    1. Ransomware: File Encryption + Enumerate Files + Delete Shadow Copy
    2. Shellcode Runner: asm_payload @ .data + __asm call rax @ .text
    3. Packers: Memory Decryption + PE Loader

ARTICLE | FREE ACCESS

Attention is all you need **Is not you need**

Authors: Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit,

NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems

New Additive Attention for Convolutional Network

- 贏家 Chunk 學習策略 – Global Channel Gating (GCC)
- To endow our network with the ability to learn such relationships while retaining computational tractability, we develop a new attention inspired gating approach we call global channel gating (GCG).
- The idea is that given a long time sequence with C channels, $X = \{X_1, X_2, \ldots, X_t\}$ where $X_t \in R^C$, we want to globally suppress certain time steps based on the content of all channels. We approach this in a style similar to the gated linear unit and the additive attention (Dzmitry Bahdana et al. 2015), using a learned context g ∈ R^C as shown in Equation 1:

$$GCG_W(x_t, \bar{g}) = x_t \cdot \sigma(X_t \cdot \tanh(W^T \bar{g}))$$

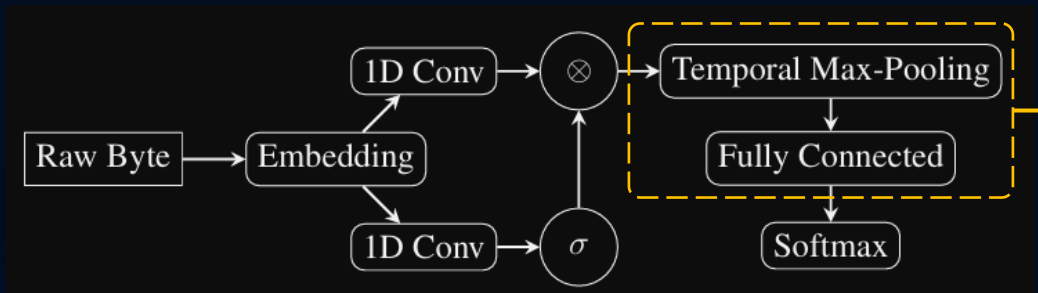


Figure 1: Original MalConv architecture (Raff et al. 2018) with ≈ 1M parameters, but required 128 GB of GPU RAM to train. ⊗ indicates element-wise product, and σ the sigmoid activation.

4 Global Channel Gating

With an efficient method for handling large input sequences, we can explore a broader set of neural network architectures. In particular, we note a weakness in the original design of MalConv: the use of temporal max-pooling after a single convolutional layer results in a somewhat myopic model: learned features are purely local in nature. That is, with the existing architecture, the model output does not consider interactions between features that are far apart in time within an input sequence/file.

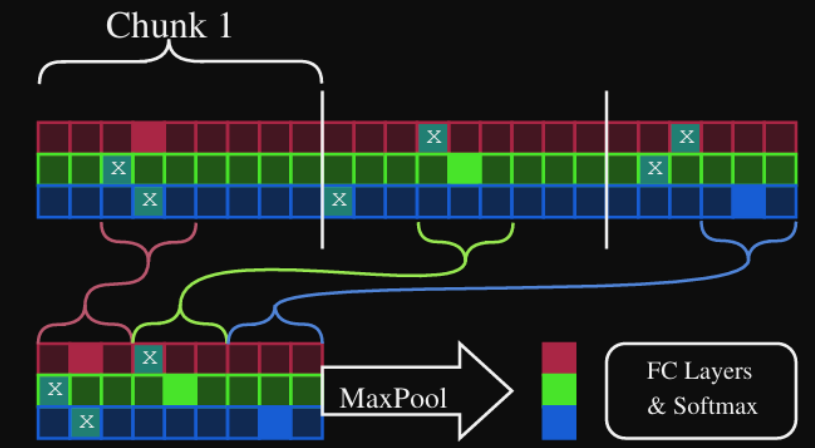


Figure 2: Diagram of Temporal Max Pooling with fixed memory. The original input (top) is a 1D sequence with 3 channels and is broken up into four chunks based on window size $W = 3$. Without gradient computation/tracking, the maximum activation index is found within each chunk. Solid colors show max values kept, "×" max in chunk but no maximal. Winning indices are copied to a new shorter sequence (bottom), which runs with gradient tracking. The result is the same output and gradient, but fixed memory cost.

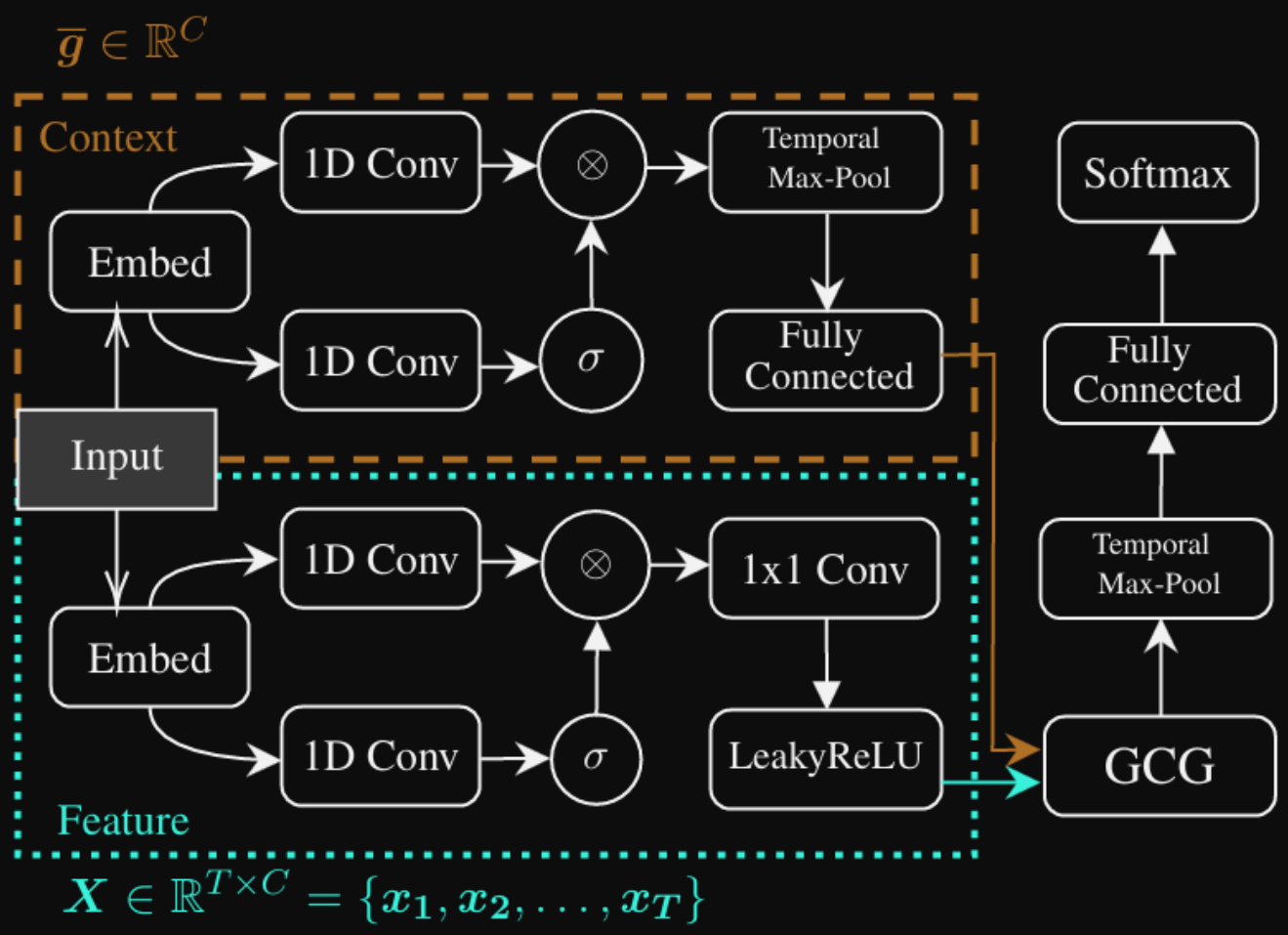
Gated Convolutional Networks

$$GCG_W(x_t, \bar{g}) = x_t \cdot \sigma(X_t \cdot \tanh(W^T \bar{g}))$$

```
def gcg(self, X, g):
  # X.shape = (B, T, C)
  B, T, C = X.size(0), X.size(1), X.size(2)
  # g.shape = (B, C)
  # create context vector z = tanh(W^T g)
  # self.w references a nn.Linear(C, C)
  ↪  layer
  z = torch.tanh(self.w(g))

  # Size is (B, C), but we need (B, C, 1) to
  ↪  use as a 1d conv filter
  z = torch.unsqueeze(z, dim=2)
  # roll the batches into the channels
  x_tmp = X.view(1,B*C,-1)
  # apply a conv with B groups; each batch
  ↪  gets its own context applied
  # This computes x_t^T z forall t = 1...T
  x_tmp = F.conv1d(x_tmp, z, groups=B)
  # x_tmp has a shape of (1, B, T); re-order
  ↪  as (B, 1, T)
  gates = x_tmp.view(B, 1, -1)

  # effectively apply x_t · σ(x_t^T tanh(W^T g))
  return X * torch.sigmoid( gates )
```



大力出奇蹟、數大便是美

- **Ember2018 資料集 ~ train/test samples = 600,000 / 200,000**
- In Table 2 we show the classification performance of all three models, and two state of the art compression based methods LZJD and BWMD using 9-nearest neighbor classification… As noted previously, parsing all of the input file is also beneficial for thwarting the trivial attack of moving all malicious code to the end of an executable.
- We prefer evaluation on the Ember 2018 corpus because it is both large and challenging. Our evaluations on the PDF corpus are done to show that our improvements transfer toother file types as well. On our PDF corpus we obtain an Accuracy of 99.16% and an AUC of 99.76%. MalConv withGCG improves this to 99.42% and 99.80%. Because PDF file are easier to processes, the baseline MalConv is already nearing maximal performance.

Model	Accuracy	AUC
MalConv (2MB, Orig)	91.27	97.19
MalConv	91.14	97.29
MalConv w/ GCG	**93.29**	**98.04**
LZJD	73.43	84.98
BWMD	81.97	91.12

Model	Time Per Epoch	GPU RAM
MalConv (2MB, Orig)	21 hr 29 min	128 GB
MalConv	1 hr 10 min	1.1 GB
MalConv w/ GCG	4 hr 5 min	2.2 GB

5.4 Example of interactions over time with GCG

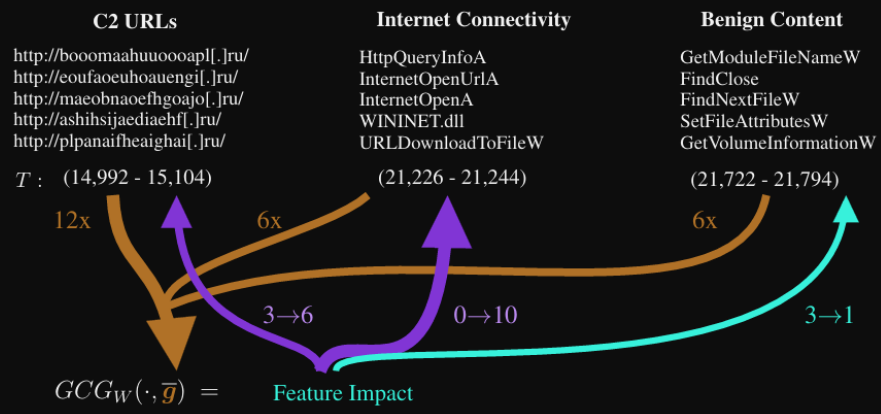


Figure 7: The time steps T that feature content is found is shown in parenthesis. $\bar{g}$ suppressed activations from benign content, and increased focus on internet functionality/ This shows GCG can related disconnected portions with no overlapping receptive field, learning complex interactions.

Large-scale Threat Payload Hunting e.g. Phishing & Exploits

- **Phishing PDFs**

  "These jump links can be abused so that when hovering the mouse over the link, the destination website might look legitimate, tricking the user into thinking the last website is also legitimate."

  1. www.broadcastingcable.com/common/jumplink.php?target=URL_HERE
  2. www.creatiblogs.es/index.php?obj=front&action=redirect&step=1&url=URL_HERE
  3. c2n.me/43zBtcZ

- **MalConv with GCG Advantages**
  - Able to identify and process information across multiple file formats
  - Example: Activation detected on a JPEG zip-encoded within a PDF file
  - Reduces the complexity of feature engineering needed for handling varied format interactions.

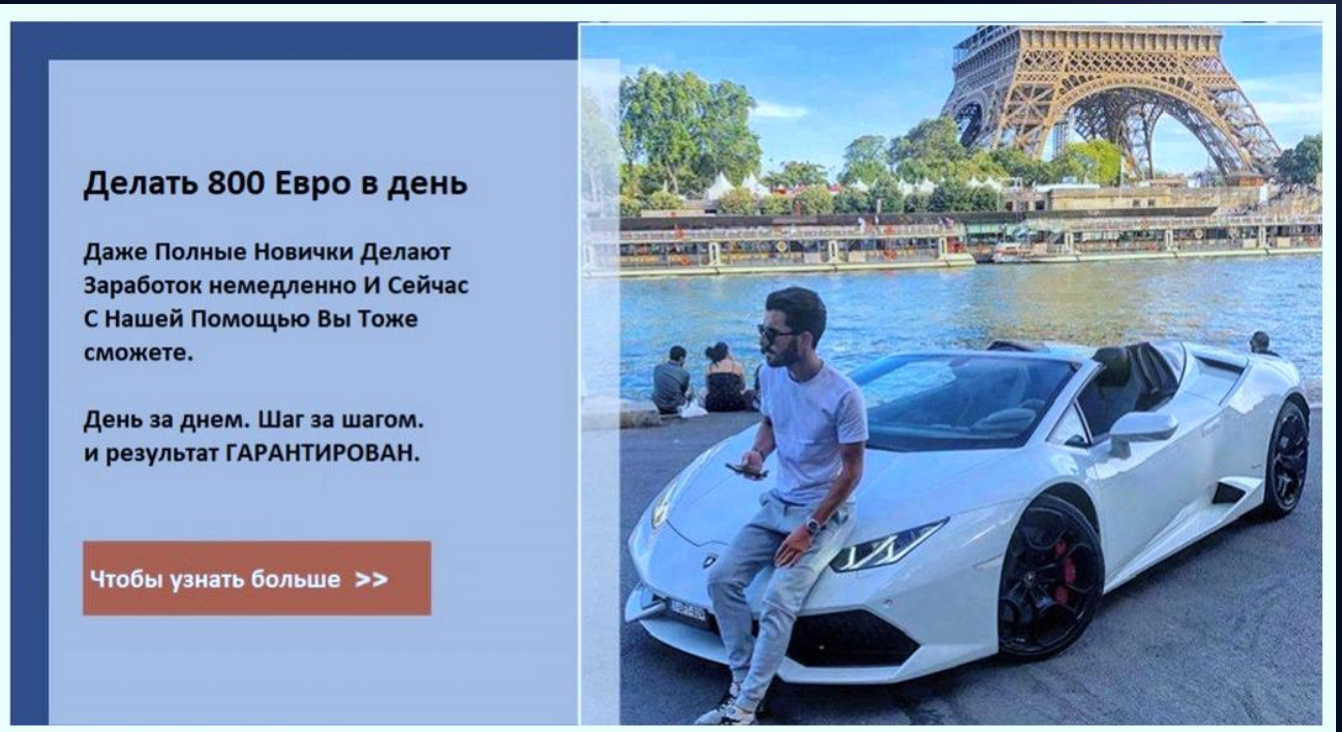


Figure 8: Make 800 Euros per day, so easy, even newbies can do it

000a5028b6b95abd90b7cfa9070fd8a26222c0b0ede27621cebf1ae5a20b646a: MalConv pointed at the sequence `/MediaBox [0 0 595.280 841.890] /TrimBox [0.000 0.000 595.280 841.890]` as being malicious. It's not certain what exactly in this sequence is malice, but a quick search engine search for this string yielded search results for new malicious PDFs with the same sequence.

00014d95af10a46fa638a0472ef1495c28103509cd5868f8ca87ee921f95de00: Was identified by MalConv as being malicious, and the `/OpenAction` sequence was identified as a contributing factor. This causes Adobe Acrobat to execute a block of JavaScript immediately when the document is opened, which is likely the exploit used in this malware sample.

Some PDFs had odd URLs which MalConv identified as being important for the malicious classification. Some of these URLs

Takeaway

- **The dilemma of the Markov assumption on practical detection use**
  - N-Gram/Seq2Seq is great, but Attention is expensive on binary detection
  - Yann LeCun proofed the possibility of NLP transformed into CNN solution.
  - Facebook AI Research also adopted the concept to use CNN on Transformer use.
  - Faster, Cheaper, better on semantics detection of large-context raw data problem.
- **Universal File Classification via Convolutional Approach**
  - Without manual feature engineering, raw-bytes in/out simple design is better to capture the subtle changes on the complex file structure of exploits/phishing usages. And given a reasonable detected payloads as evident, , e.g. PDF, DOC


txOne
networks