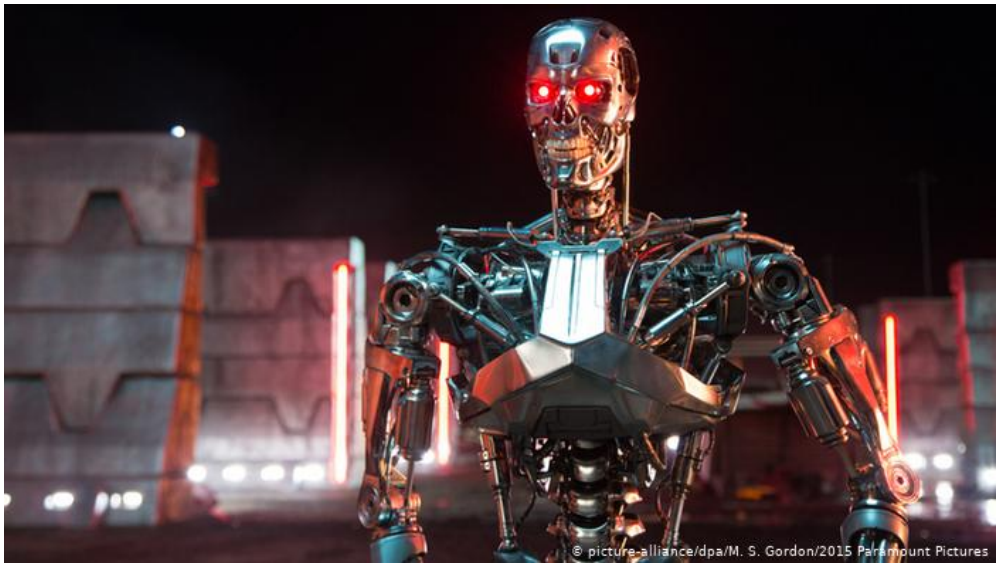
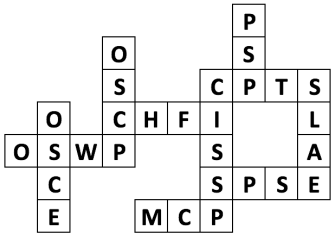
Machine Learning Security Evasion Competition 2020

Hyrum Anderson - @drhyrum and Zoltan Balazs - @zh4ck



Microsoft CUJOAI

Whoami
@zh4ck



Head of Vulnerability Research Lab @ CUJO AI

Zombie Browser Toolkit

- https://github.com/Z6543/ZombieBrowserPack

HWFW Bypass tool

- Similar stuff was used in PacketRedirect in Danderspritz FlewAvenue by EQGRP
- https://github.com/Z6543/hwfwbypass

Malware Analysis Sandbox Tester tool

- https://github.com/Z6543/Sandbox_tester

Played with crappy IoT devices – my RCE exploit code running on ~600 000 IP cameras via Persirai

- https://jumpespjump.blogspot.hu/2015/09/how-i-hacked-my-ip-camera-and-found.html
- https://jumpespjump.blogspot.hu/2015/08/how-to-secure-your-home-against.html

Invented the idea of encrypted exploit delivery via Diffie-Hellman key exchange, to bypass exploit detection appliances

- https://www.mrg-effitas.com/generic-bypass-of-next-gen-intrusion-threat-breach-detection-systems/

Co-organizer of the Hackersuli meetup

Programme committee member of the Hacktivity conference

Microsoft CUJOAI

Whoami
@drhyrum

architect, Azure Trustworhy Machine Learning @ Microsoft

- ML security as a 1st class, practical security concern

cofounder and co-chair, CAMLIS https://camlis.org/

background

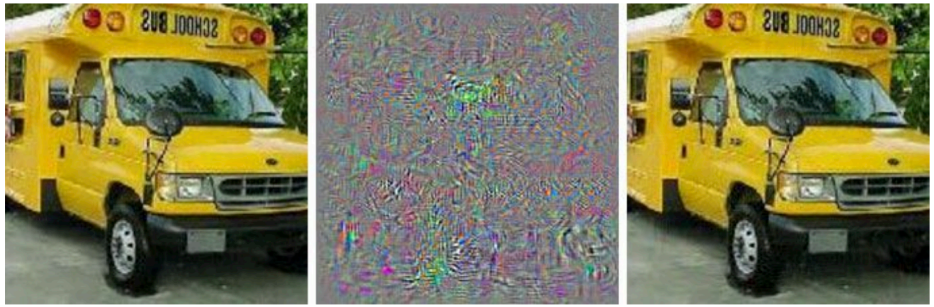
- (1st) signal processing, machine learning
- (2nd) information security

relevant research

- Reinforcement learning AV evasion:
  - https://github.com/endgameinc/gym-malware
- Co-creator of EMBER 2017 and 2018 datasets:
  - https://github.com/endgameinc/ember

Microsoft CUJOAI

ML detection bypass in the past



The left image is unaltered and would be classified as a school bus, while the right would be classified as an ostrich. The middle image shows the distortions made to the adversarial example. *Christian Szegedy*

Super l33t ML malware detection bypass from 2019 https://skylightcyber.com/2019/07/18/cylance-i-kill-you/

- strings RocketLeague.exe >> mimikatz.exe

Super l33t ML malware detection bypass from 2016

- upx.exe

Microsoft CUJOAI

evademalwareml.io 2019

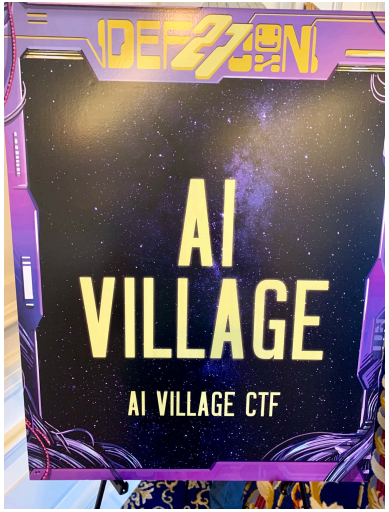
Purpose: advance the field of offensive and defensive ML-based malware detection

Step 1: Download 50 working malware samples

Step 2: Download 3 ML model with weights (white-box attack)

Step 3: Modify the malware samples to evade detection by all models

Step 4: PROFIT! Award: Nvidia Titan RTX



Microsoft CUJOAI

evademalwareml.io 2019 Outcomes

~70 people registered

11 contestant able to bypass at least one ML model

Winner: 2019 August 28, 15:25 UTC William Fleshman

- Will's writeup: https://towardsdatascience.com/evading-machine-learning-malware-classifiers-ce52dabdb713

Writeups from other competitive teams

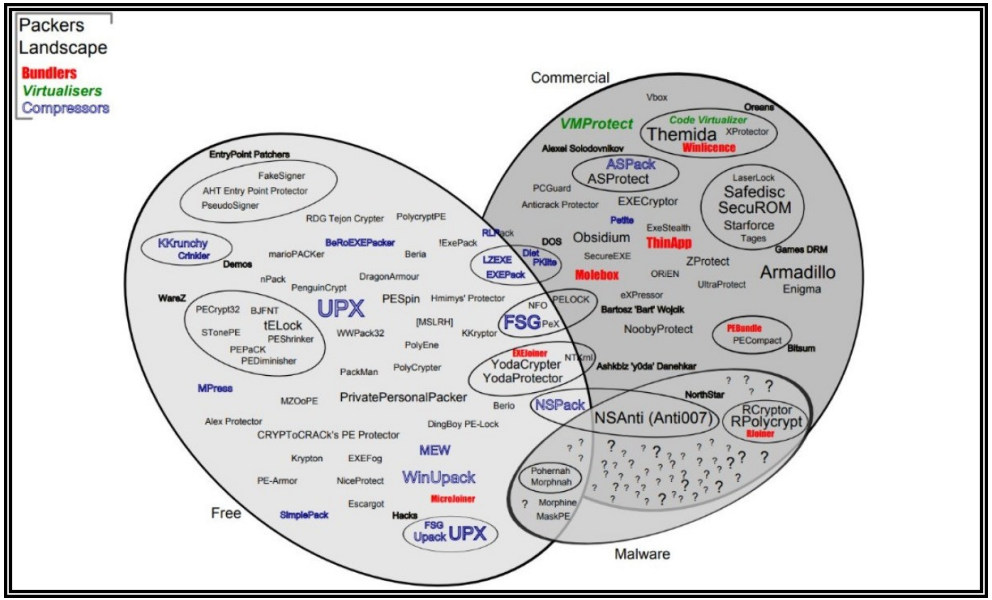
- Jakub Debski https://www.eset.com/blog/company/evading-machine-learning-detection-in-a-cyber-secure-world/
- Fabricio Ceschin et al., https://secret.inf.ufpr.br/papers/roots_shallow.pdf

Microsoft CUJOAI

evademalwareml.io 2019 Approaches used

Packing the samples with a packer

- nice, but if the sample is already packed, chances are it will not work
- this is partially a competition-specific issue, if you write your own malware, you can create your own packer which can bypass ML detection



This image was designed by Ange Albertini https://corkami.blogspot.com

Microsoft CUJOAI

evademalwareml.io 2019
Approaches used

Adding new sections to the executable

- even better if these sections are from known benign files, e.g. resources from MS files
- works most of the time, but can break malware
- some malware/packer has self-checks, and adding new sections can break this
- just by adding a new section – you can bypass some AV (out of scope)
    - fun fact: some AV uses shortcuts for signature-based detection like if section==X check this. Improves performance, easy to bypass.

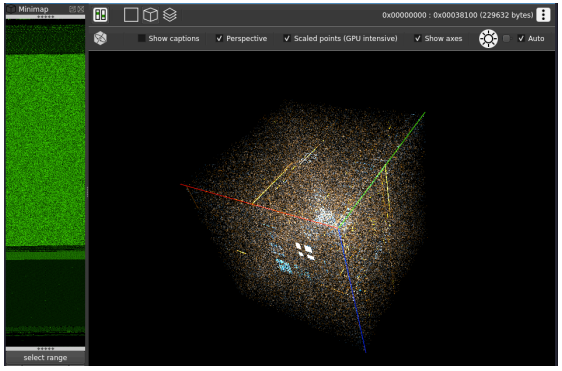
```
def add_section_constant(binary, name, constant, size):
    # create a section
    section = lief.PE.Section(name)

    # fill it with our constant
    section.content = [constant] * size

    # add the section
    binary.add_section(section, lief.PE.SECTION_TYPES.DATA)
```

Microsoft CUJOAI

evademalwareml.io 2019
Approaches used

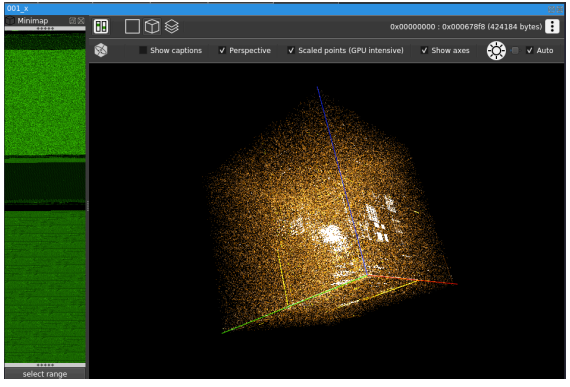


Appending extra data to the executable, a.k.a overlay

- actually, this was the winner strategy …
- dumb, plain, simple, and it works
- it works if you have the ML models and weights, a.k.a white-box attack
- this overlay technique will not bypass static signature AV checks (out of scope)
    - except when the AV has a rule that Filesize less than X …
    - yes, this still happens

cat overlay >> malware.exe

Overlay →



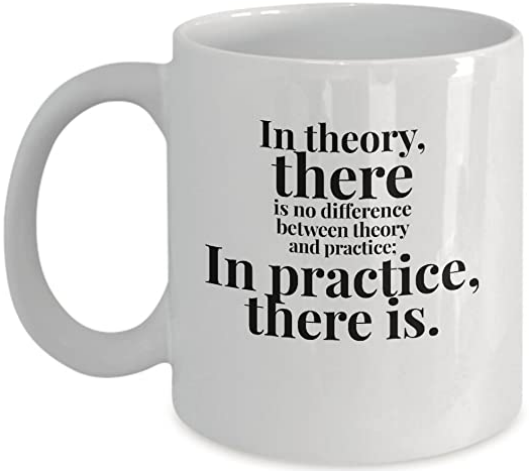
Microsoft CUJOAI

evademalwareml.io 2019
Some key takeaways

malconv and non neg malconv is too academic

- but not effective in practice

LIEF is awesome https://github.com/lief-project/LIEF



Malware is tricky

- some samples do not reproduce the same IoCs over time
  - mainly because of C&C down
- packed and protected samples are hard to deal with

evademalwareml.io 2019
Fun with SSDeep

SSDeep

is a program for computing context triggered piecewise hashes (CTPH). Also called fuzzy hashes, CTPH can match inputs that have homologies. Such inputs have sequences of identical bytes in the same order, although bytes in between these sequences may be different in both content and length.

6144:9dA3OOLEQ5dIZHlxBM/lxBM/lxBM/lxBMe:9u3O+EQ5dIrMpMpMpMe
49152:lOctKPaSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS:c2O
12288:I8Mr88Mr88Mr88Mr88Mr88Mr88Mr88Mr88Mr88MrZ:Ilr8lr8lr8lr8lr8lr8lr8lr8lr8lrZ
49152:lOctnPjppprOctnPjppprOctnPjppprOctnPjppprOctnPjppprOctnPjppp:J2P02P02P02P02P02P

mlsec.io 2020

DEFENDER CHALLENGE

- Create your own ML model and submit to the competition
- Docker All The Things

ATTACKER CHALLENGE

- Black-box attack against submitted defences
- Source code provided for only the ember model

Sponsors and partners

- Microsoft, CUJO AI, VMRay, MRG Effitas

Main organizer people remained the same ☺

Win Azure credits for your ~~take over the world~~ ML research plans

Microsoft CUJOAI

mlsec.io 2020

Defensive track

Two submissions that passed minimum requirements

Look for the following ML models in your offensive track

- ember [default model for which there is code]
- needforspeed
- domumpqb

Microsoft CUJOAI

mlsec.io 2020
Offensive track - Aug 06 – Sep 18, 2020 AoE

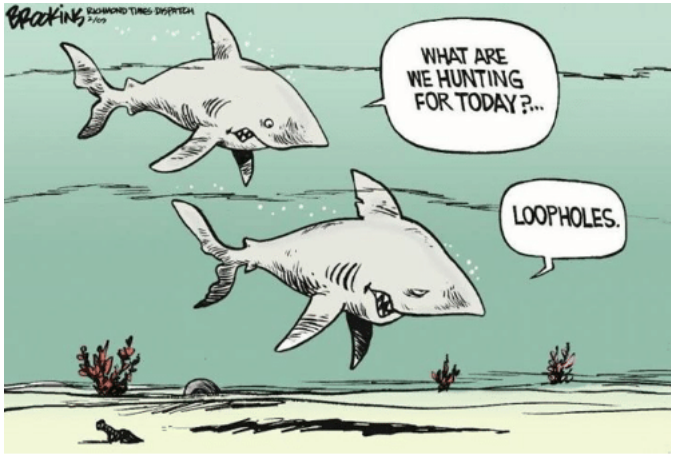
Malware families

- Remcos
- Lokibot
- Raccoon
- Netwire
- Hawkeye
- Azorult
- Amadey
- Agent Tesla
- Ursnif
- Trickbot
- Sodinokibi
- njRAT
- Nanocore
- Maze
- Masslogger
- Gh0st RAT
- Dharma
- AsyncRat
- Zeppelin Ransomware
- VHD Ransomware
- Qbot
- Paymen45 ransomware
- Formbook
- Citadel
- Ave Maria

Microsoft CUJOAI

MLSEC 2020 attacker flowchart

1. register at https://mlsec.io
2. review terms of service
3. download 50 provided malware samples
4. <your secret sauce to modify samples>
5. verify malware functionality (Windows 10 x64)
6. Optional: Use the API!
7. upload ZIP; partial uploads ok (upload rate limiting)
8. up to 3 points for each sample (# of evade models)
9. highest score wins
10. to win, your solution must be published (e.g., blog)



Microsoft CUJOAI

Mind the sample names

Filenames in **downloaded** ZIP

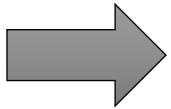
001

002

003

…

050



Filenames in **uploaded** ZIP

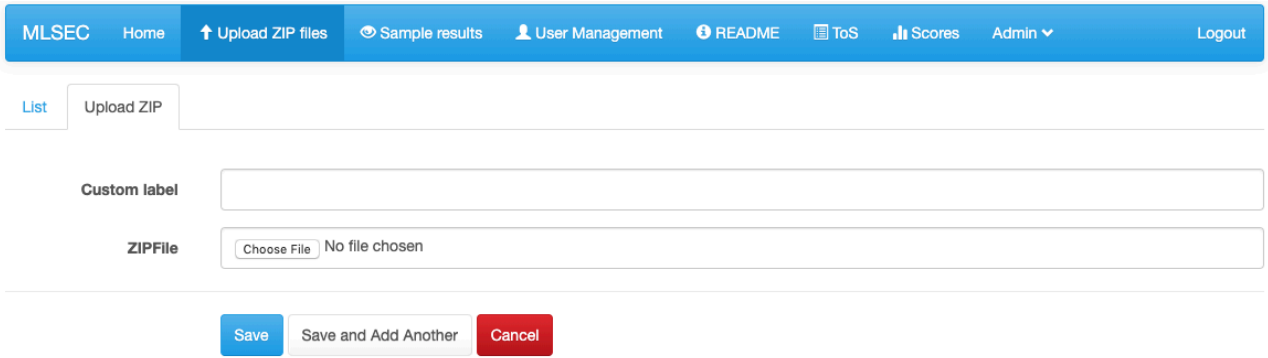
001

002

003

…

050



Microsoft

tips!

You might consider some of these manipulations

- add / remove signature
- change section names/properties
- modify imports/exports
- create TLS callback
- change PE header
- fix/change checksum
- add/modify/remove version info
- new entry point that redirects
- change code/data (no-ops)

Microsoft CUJOAI

tips!

Not allowed / won't function:

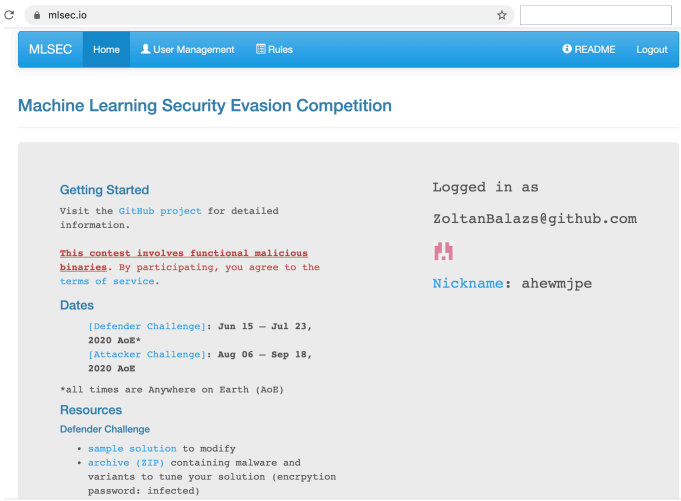
- Droppers
- Self-extracting archive/RAR (SFX)

Multiple registration is against the rules and will result in immediate disqualification

Join the Slack channel!

https://join.slack.com/t/evademalwareml/shared_invite/zt-9birv1qf-KJFEiyLLRVtrsNDuyA0clA

mlsec.io 2020



About the frontends and backends

Python – Flask Admin for GUI

Cloudflare, Nginx, Gunicorn for scalability and performance

Python backend scripts scheduled by CRON

VMRay sandbox

Microsoft CUJOAI

mlsec.io 2020 API

1. Submit sample to all ML model
curl -X POST https://api.mlsec.io/api/ml_submit_sample_all?api_token=<API_KEY> --data-binary @001

2. Submit sample to specific ML model
curl -X POST https://api.mlsec.io/api/ml_submit_sample?api_token=<API_KEY>&model=ember --data-binary @001

3. Get ML model results
curl -X GET https://api.mlsec.io/api/ml_get_sample?api_token=<API_KEY>&jobid=<JOB_ID>

4. Upload ZIP
curl -X POST https://api.mlsec.io/api/post_one_zip/new/?url=%2Fzipfile%2F&api_token=<API_KEY> --form "name=name" --form path=@my.zip

5. Query specific ZIP status
curl -X GET https://api.mlsec.io/api/get_one_zip/<ID>?api_token=<API_KEY>

6. Query all sample status
curl -X GET https://api.mlsec.io/api/get_all_sample/?api_token=<API_KEY>

7. Query specific sample status
curl -X GET https://api.mlsec.io/api/get_one_sample/<ID>?api_token=<API_KEY>