X33FCON · 2026

Using EMBER2024 to evaluate red team implants.

$ whoami

NAME	**Brandon McGrath**
ROLE	Red team operator at **TrustedSec**
X33FCON	Third visit, first time speaker
WRITING	mez0.cc | trustedsec.com/blog
TWITTER	@_ _mez0_ _

implants and ML

Specifically: *how an ML model scores a payload, where the score swings, and what that means for the people building them, and for the people catching them.*

https://mez0.cc/posts/evaluating-implants-with-ember/

Sanity checking.

A score on disk, before the binary leaves your build folder. Catch the divergence point in the lab, not on a customer's host.

Quick triage.

A score next to your other signals when something lands on a host. Literally the inverse of red.

What is this talk about?

When using EMBER to review files, where is the “divergence point”
where a model goes from being flagged as “bad” to ”good”?

There are tools for part of this already.

- **Capa** capability detection from disassembly
- **yara** byte / string signatures, the workhorse
- **Avred** bisects a binary against Defender

…and a long tail of one-off scripts.

THE NEW SHAPE

ML scoring is harder to reason with.

The divergence point is the moment a payload’s EMBER score swings from good to bad.

WHEN ML gets called

Hits disk.

First write. Static ML's first opportunity to flag

THEN AGAIN

Lateral move.

New host, new endpoint, new EDR vendor. Same file, scored fresh.

AND AGAIN

Privesc, persist.

Every drop is another roll of the dice. Knowing the score is the dice.

What is EMBER?

EMBER (2018): An Open Dataset for Training Static PE Malware Machine Learning Models:
https://arxiv.org/abs/1804.04637

EMBER2024 -- A Benchmark Dataset for Holistic Evaluation of Malware Classifiers:
https://arxiv.org/abs/2506.05074

WHY THIS MODEL

Accessible & current.

EMBER 2018 → EMBER 2024. Six years of drift, plus the LLM era halfway through.

THE CHALLENGE SET

6,315 files

Already evaded ~70 AV products on VT. The hard set, before we start poking.

Six-year-old training data

2018

~1.1M samples

Pre-LLM era. Pre-half-the-evasion-techniques-in-this-talk.

2024

~3.2M samples

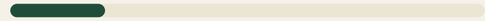
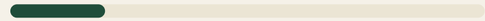
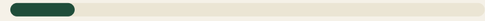
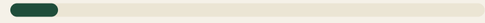
Plus the 6,315-file challenge set. Hardened on what already evades real AV.

WHY I BRING THIS UP

Drift is real.

Same loader scored across both shows you what two years of model aging looks like.

What does EMBER2024 extract from a sample?

FEATURE GROUP	WHAT IT SEES	SHARE OF VECTOR	DIMS
ImportsInfo	DLLs and functions you call (or appear to).		**1,282**
ByteHistogram	Distribution of byte values across the file.		**256**
ByteEntropyHistogram	Sliding-window randomness, 2D histogram.		256
StringExtractor	Printable runs, URLs, paths, registry keys.		177
SectionInfo	Entropy and size per section.		174
ExportsInfo	Hashed export symbols.		129
HeaderFileInfo	machine, subsystem, characteristics.		74
DataDirectories	Offsets and sizes of debug, reloc, resource…		34
RichHeader	Compiler/linker fingerprint, hashed.		33
AuthenticodeSignature	Cert count, self-signed flag, timestamps.		8
GeneralFileInfo	size, vsize, has_debug, has_resources.		7

Other models

`MalConv` raw-byte CNN, no feature engineering

`SOREL-20M` the dataset, plus baseline models

`ClamAV-ML` the open-source AV's own scoring

`...vendor models` same ideas, more weight, less visibility

https://pre.empt.blog/posts/static-data-exploration/

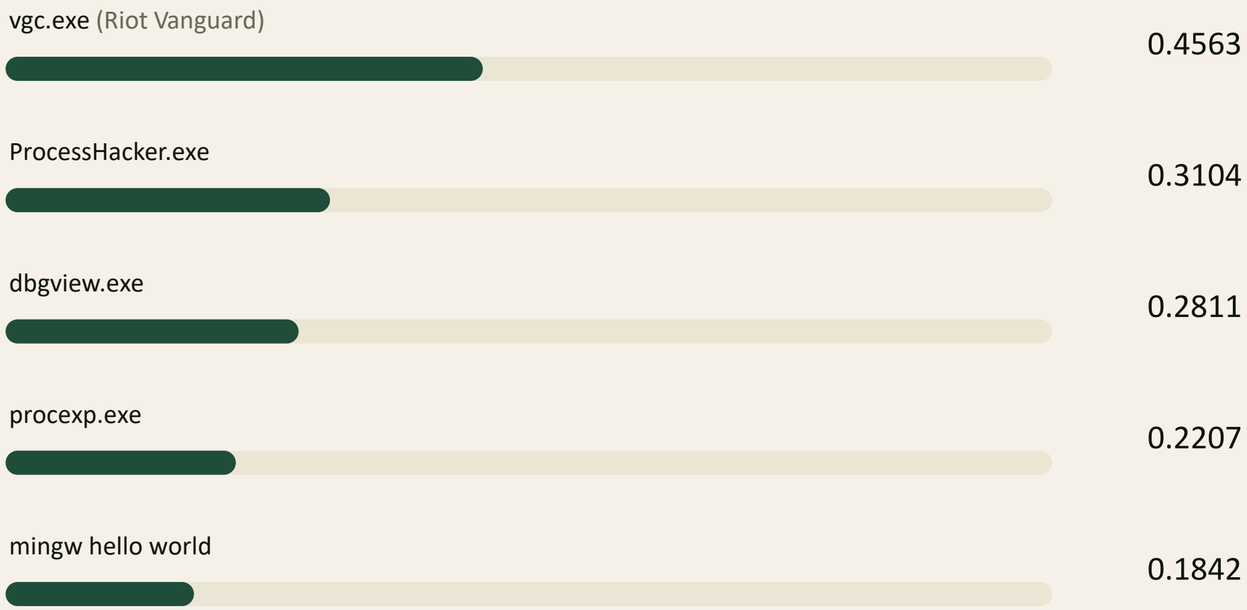
Someone please maintain a dataset.

A modern, openly-maintained PE corpus would change this whole space. I cannot be bothered. You can.

Baseline.

Before we break anything, let's see what real software scores.

Hand-picked samples



AVERAGE ACROSS MY HOST

0.29

On 30,457 clean **benign** binaries, EMBER false-positives at

1.07%

TOTAL SAMPLES

30,469

30,457 successfully scored. 12 parse failures.

FLAGGED ≥ 0.5

326 / 30,457

98.93% correctly classified benign.

MEDIAN SCORE

0.000418

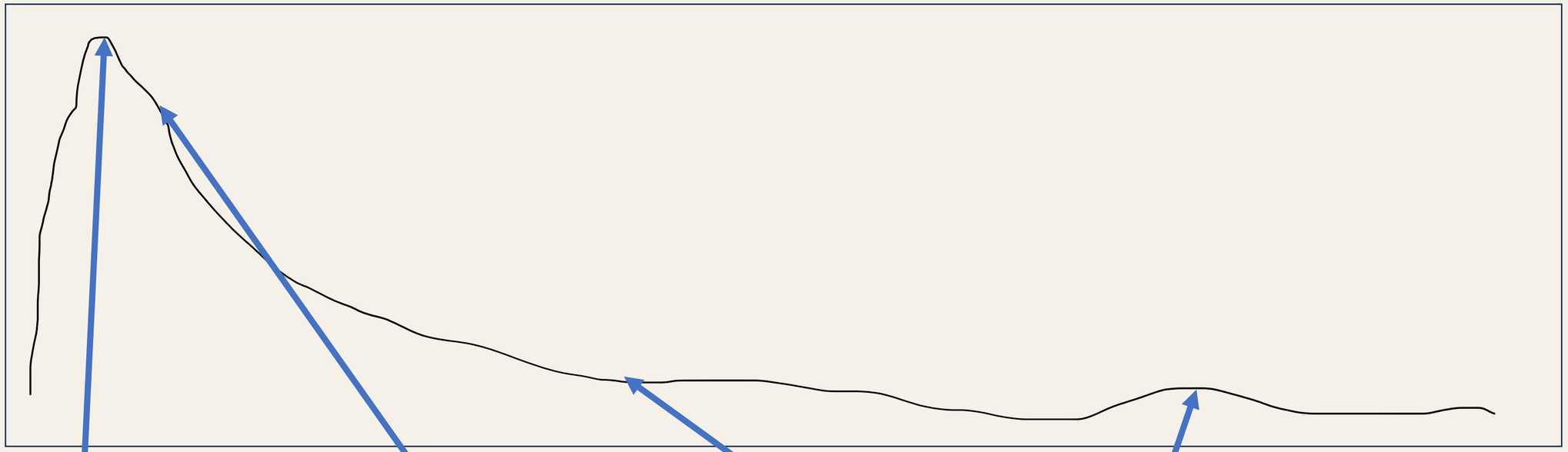
Essentially zero.

SCORE	SOFTWARE	WHY EMBER FLAGGED IT	VERDICT
1.0000	**AdiIRC v2.4.0.0** *freeware IRC client*	IRC = three decades of botnet C2. Network-heavy .NET, unsigned, no major-vendor provenance.	LEGIT
0.9994	**AutoHotkey v1.1.00** *macro / automation runtime*	AHK is the runtime of choice for infostealers and RATs. Runtime signature is poisoned in training.	LEGIT
0.9985	**Unidentified .NET** *1.9 MB · .rsrc entropy 7.49*	Heavy embedded resource section at near-max entropy. Classic 'bundled payload' shape.	AMBIGUOUS
0.9981	**Stripped 64-bit tool** *168 KB · 1 import · Nov 2023*	No version, publisher, or copyright. Reads as custom dropper / loader tooling.	LEGIT
0.9981	**UltiMaker Cura v1.0** *3D printer slicer (Qt)*	Qt apps drag sprawling imports + bundled resources. Pattern-matches installer malware. Unsigned.	LEGIT
0.9977	**Unidentified DLL** *57 KB · .text entropy 7.89 · 2012*	Textbook packer-tier .text entropy. Byte histogram matches packed malware too closely.	AMBIGUOUS
0.9957	**AutoHotkey v1.1.00** *second AHK sample*	Same runtime as #2, different script. EMBER is reacting to the runtime, not the behaviour.	LEGIT
0.9955	**Stripped 64-bit tool** *236 KB · same batch as #4*	Compiled one second before #4. Same pipeline, same missing metadata, same verdict.	LEGIT
0.9937	**conda.exe** *Anaconda Python package manager*	Package managers fork, fetch, write to user-writable paths. Overlaps cleanly with downloader malware.	LEGIT
0.9927	**UPX-packed GUI app** *original identity stripped*	UPX is the strongest packed-malware signal EMBER sees. Plenty of OSS ships UPX'd anyway.	AMBIGUOUS

WIN32 false-positives ~4× more often than WIN64.

FILE FORMAT	N	MEDIAN SCORE	FLAGGED ≥ ≥ 0.5	FPR
WIN32 PE	6,594	0.00108	165	**2.50%**
WIN64 PE	18,767	0.00047	112	0.60%
.NET	5,096	0.00007	49	0.96%
All	30,457	0.00042	326	**1.07%**

99% of goodware binaries below ~0.53



MEDIAN
0.000418

MEAN
0.0168

P95
0.0484

P99
0.5316

MAX
0.999995

Walking it down.

From a stock msfvenom calc loader at 0.9985 down toward the noise floor

msfvenom calc. VirtualAlloc. CreateThread.

EMBER2024 SCORE

0.9985

malicious, extremely

The textbook bad-time payload. We'll keep the structure trivial so any score change should unambiguously the thing we changed.

PREVIOUS SCORE

0.9985

▼ 0.0004

NEW SCORE

0.9981

Manifest. Icons. File properties.

Company name. Product version. A nice icon. The whole "this is totally legit corporate software" costume.

PUNCHLINE

It barely moved.

Costume gets you 0.01. The model is not impressed by your dress-up. On to the actual structure of the binary.

PREVIOUS SCORE

0.9981

▼ 0.0034

NEW SCORE

0.9847

I removed everything.

Emptied the shellcode buffer →

Removed the loader →

Removed the body of `main()`

Result: still malicious.

WHY

It doesn't look like the goodware the model was trained on.

Absence of malicious code isn't presence of benign code. You have to actively look like normal software, not just stop looking like malware.

▼ ~0.40

PREVIOUS SCORE

0.9847

NEW SCORE

0.45 – 0.66

First real movement.

Vibe-coded a pile of benign-looking Win32 calls

Picked imports using DLL export categorisation

```
listDirectoryWithDetails("C:\\");
displaySystemAndConsoleInfo();
registryDemo();
memoryDemo();
threadingDemo();
environmentDemo();
```

```
{
    "title": "CreateThread",
    "description": "Creates a thread to execute within the virtual address space of the calling process.",
    "category": "Process and Thread Management"
}
```

https://mez0.cc/posts/dll-export-category/

▼ ~0.40

PREVIOUS SCORE

0.9847

NEW SCORE

0.45 – 0.66

First real movement.

FORCE_IMPORT macro to pad without calling

```
#define FORCE_IMPORT(DLL, RET, CALLCONV, NAME, ARGS) \
    __declspec(dllimport) RET CALLCONV NAME ARGS; \
    static void* dummy_##NAME = (void*)&NAME;

// ==================== COMCTL32.dll ====================
FORCE_IMPORT(COMCTL32, HWND, WINAPI, CreateStatusWindowW, (LPCWSTR lpszText, HWND hwndParent, int nID))
FORCE_IMPORT(COMCTL32, HIMAGELIST, WINAPI, ImageList_Create, (int cx, int cy, UINT flags, int cInitial, int cGrow))
FORCE_IMPORT(COMCTL32, BOOL, WINAPI, ImageList_Destroy, (HIMAGELIST himl))
FORCE_IMPORT(COMCTL32, BOOL, WINAPI, ImageList_Draw, (HIMAGELIST himl, int i, HDC hdc, int x, int y, UINT fStyle))
FORCE_IMPORT(COMCTL32, BOOL, WINAPI, ImageList_GetIconSize, (HIMAGELIST himl, int *cx, int *cy))
FORCE_IMPORT(COMCTL32, int, WINAPI, ImageList_ReplaceIcon, (HIMAGELIST himl, int i, HICON hicon))
FORCE_IMPORT(COMCTL32, COLORREF, WINAPI, ImageList_SetBkColor, (HIMAGELIST himl, COLORREF cr))
```

▲ 0.4480

PREVIOUS SCORE

0.5102

NEW SCORE

0.9582

Put shellcode + loader code back. Score jumps right back up.

Out-of-band: file on disk, UNC, HTTP

Steganography in something benign

Bluffy (pre.empt), shellcode as CSS rgb() values

https://github.com/preemptdev/bluffy

SIDEBAR · BLUFFY

Shellcode as CSS.

```
/* shellcode bytes 0x55 0x48 0x8B = */
.btn { color:    rgb(85, 72, 139)    ; }
.nav { color:    rgb(72, 137, 248)   ; }
.hdr { color:    rgb(72, 131, 228)   ; }
```

Three shellcode bytes per CSS color. Your loader fetches a stylesheet.

The byte histogram on disk looks like a CSS file. Because it is one.

▼ 0.0806

PREVIOUS SCORE

0.9582

NEW SCORE

0.8776

PREV 0.9582

NOW 0.8776

Function Resolution

Strip the suspicious imports out of the IAT, resolve them at runtime from hashed names.

PLOT TWIST

XOR'ing strings raised the score.

Byte entropy spiked. Strings feature lost its handle. Net: model got *more* suspicious. Beautiful "huh" moment.

PREVIOUS SCORE

0.8776

▼ 0.5276

NEW SCORE

0.35

Complicating main

Real argparse + command dispatch

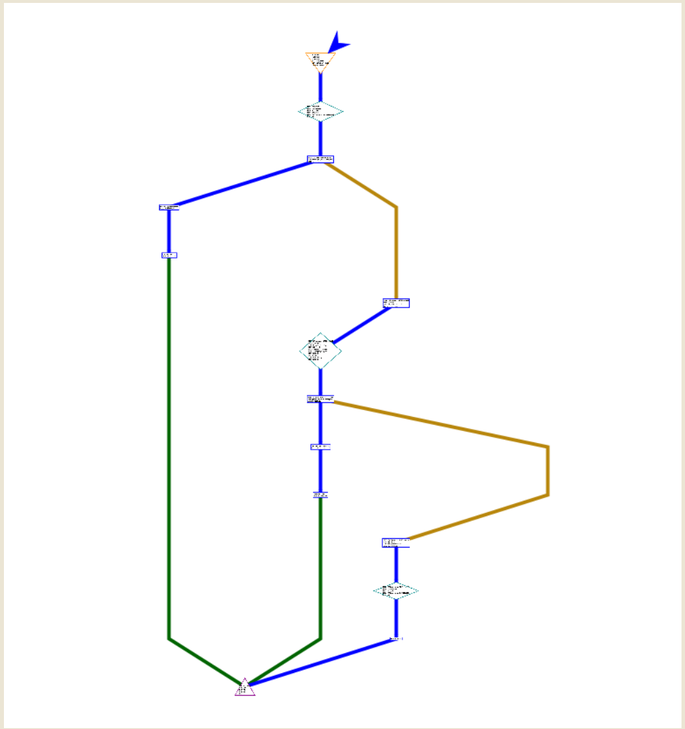
Genuine code paths around the loader

Logging, config loading, the boring stuff

Code Flow

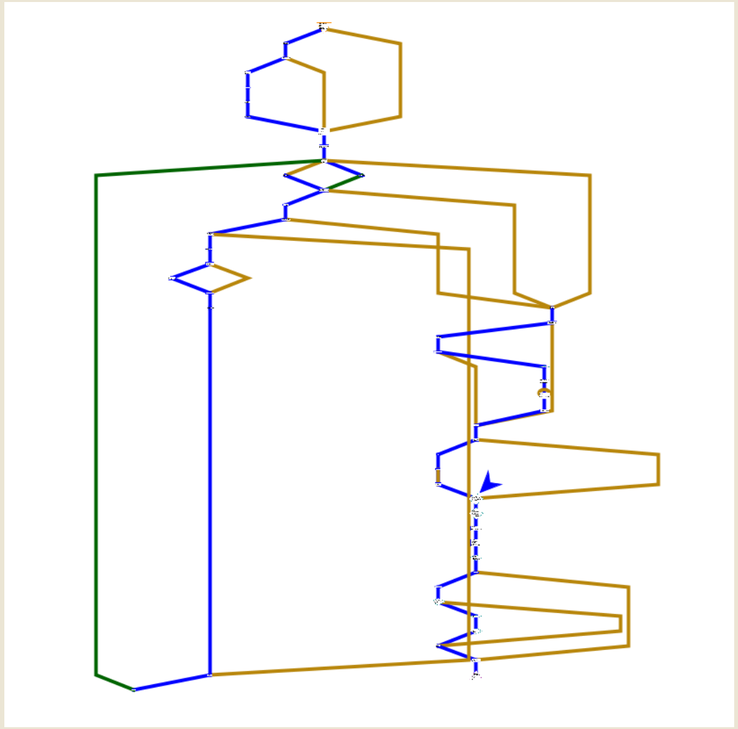
naive

VIRTUALALLOC + CREATETHREAD IN MAIN



fluffed

ARGPARSE + COMMAND DISPATCH AROUND THE LOADER LOADER



PREVIOUS SCORE

~0.35

▲ ~0.20

NEW SCORE

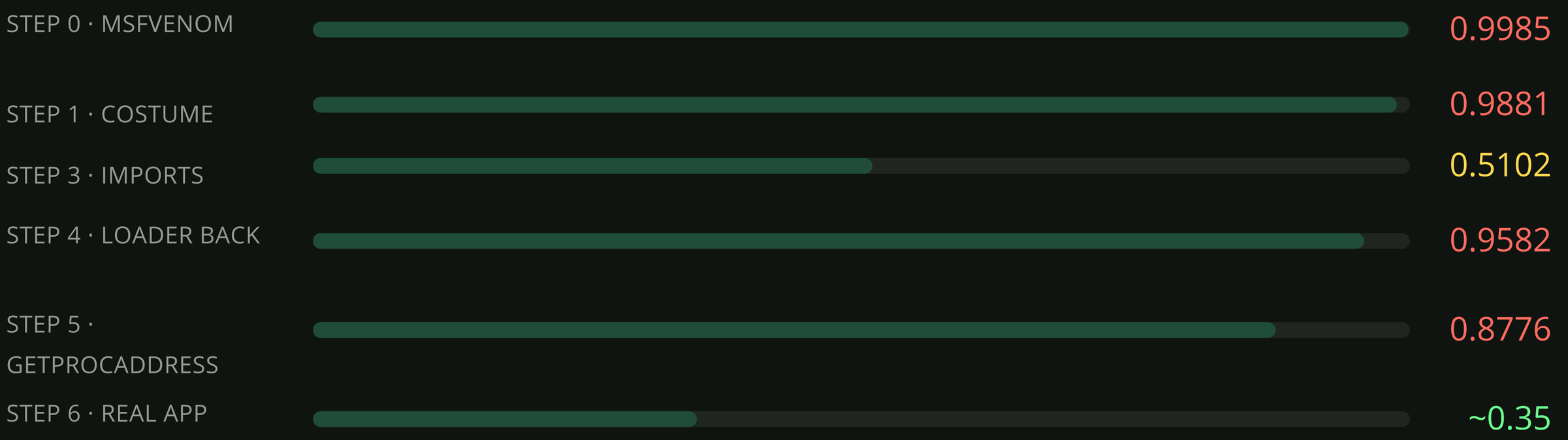
~0.55

Adding EDR Tampering

Adding in VEH Clearing
Slight Increase
Cost of evasion?

https://github.com/rad9800/misc/blob/67bfeb6496efe6b61db6b2de8d99332a5cebc100/bypasses/ClearVeh.c

From 0.9985 0.35.



Findings.

Imports

HOLLOW ANSWER

FORCE_IMPORT puts symbols in the IAT without ever calling them. The model counts them. The score nudges down. The binary still doesn't do anything with them.

HONEST ANSWER

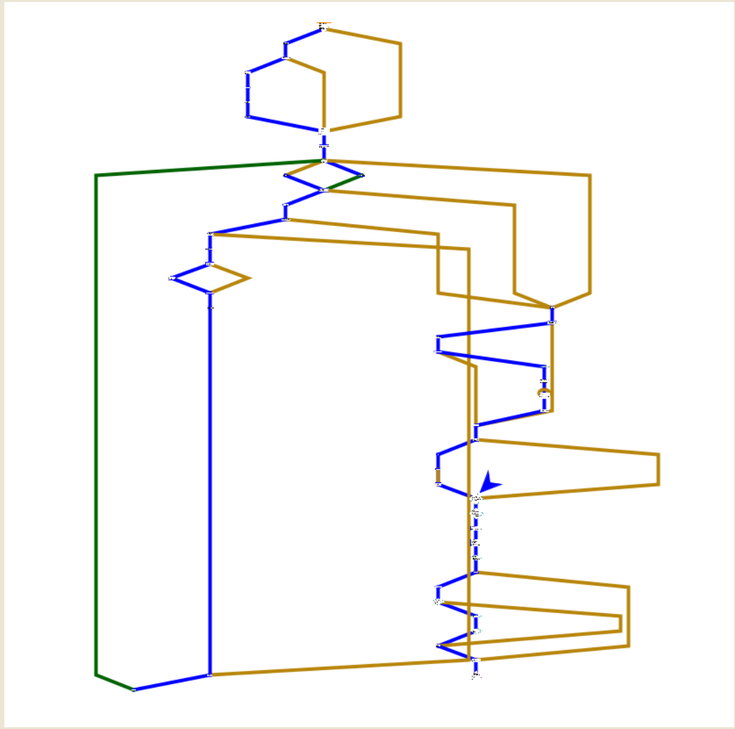
Write code that actually uses what it imports. A GUI app that paints. A CLI tool that parses arguments. The model isn't graded on intent, but its shape.

```
#define FORCE_IMPORT(DLL, RET, CALLCONV, NAME, ARGS) \
    __declspec(dllimport) RET CALLCONV NAME ARGS; \
    static void* dummy_##NAME = (void*)&NAME;

// ==================== COMCTL32.dll ====================
FORCE_IMPORT(COMCTL32, HWND, WINAPI, CreateStatusWindowW, (LPCWSTR lpszText, HWND hwndParent, int nID))
FORCE_IMPORT(COMCTL32, HIMAGELIST, WINAPI, ImageList_Create, (int cx, int cy, UINT flags, int cInitial, int cGrow))
FORCE_IMPORT(COMCTL32, BOOL, WINAPI, ImageList_Destroy, (HIMAGELIST himl))
FORCE_IMPORT(COMCTL32, BOOL, WINAPI, ImageList_Draw, (HIMAGELIST himl, int i, HDC hdc, int x, int y, UINT fStyle))
FORCE_IMPORT(COMCTL32, BOOL, WINAPI, ImageList_GetIconSize, (HIMAGELIST himl, int *cx, int *cy))
FORCE_IMPORT(COMCTL32, int, WINAPI, ImageList_ReplaceIcon, (HIMAGELIST himl, int i, HICON hicon))
FORCE_IMPORT(COMCTL32, COLORREF, WINAPI, ImageList_SetBkColor, (HIMAGELIST himl, COLORREF cr))
```

Code Flow

Changing the flow of the application had significant impact



Code signing.

One signature stripped from one binary. The model treats Authenticode as a strong benign prior, almost everything else is decoration next to it.

Same binary. Same bytes (almost). **+0.5429**.

Resemblance to legitimate software.

01

Imports

What it sees you call. Real apps have busy IATs.

02

Byte distribution

Real code, real strings, real resources. Real binaries are not empty.

03

Signing

Asterisk attached. Acquiring it is the other half of the problem.

Not-redteamers

ML for everyone

FOR RED

Score it before the engagement.

A loader that never hits disk is the worst outcome on a red team. The client paid for impact, not for you to burn three days finding out your dropper was DOA. Score it locally, fix it locally, and spend the engagement actually doing the work.

FOR BLUE

Put it on the analyst's desk.

The blue teams I've enjoyed working with are the ones who take a suspicious file offline and crack it open. Curious, hands-on, fast. Give those people ML scoring in triage. Not just gated behind EDR vendors or VT, on the desk, in the workflow, beside the disassembler.

Thanks!