**引言**

恶意软件是经过精心设计，被用来攻击计算机系统或计算机网络并且造成损害的软件，危害不仅限于感染型病毒和蠕虫的自身复制耗尽系统资源，破坏操作系统、后门软件开放系统端口供黑客连接从而形成僵尸网络对服务器发起分布式拒绝服务攻击、木马伪装成正常程序，实则窃取用户的敏感数据和破坏系统，为黑客提供后门、勒索病毒加密文件要求用户支付赎金等。根据统计数据，仅2022年在全球大约发生了55亿次恶意软件攻击事件。

SonicWall

Sonicwall cyber threat report

<https://www.sonicwall.com/medialibrary/en/white-paper/2023-cyber-threat-report.pdf> (2023)

为了应对恶意软件带来的威胁，许多开源以及商业杀毒软件厂商不断升级病毒库，更替杀毒软件版本。目前针对恶意软件的识别主要分为静态分析、动态分析两种。而本实验研究的强化学习模型，是生成针对反病毒软件的静态特征分析的PE可执行程序对抗性样本，目的是用于反病毒软件厂商的机器学习和深度学习反病毒引擎模型训练和逆向分析工作者们学习研究，本实验未研究动态特征分析对抗性样本生成。静态特征分析包括文件Hash（如MD5 SHA256）匹配，这在很多病毒样本分析网站，例如VirusTotal，VirSCAN等被使用，如果上传的文件和病毒库里面存在的样本的Hash值匹配，则反病毒软件判断该文件是病毒。此外，静态分析还包括资源节分析，时间戳检查，数字签名检查，函数导入表检查，特征字符串匹配，DEBUG信息检查等。在过去的十几年中，学术界出现了大量利用机器学习和强化学习的恶意软件查杀模型判断恶意软件的研究成果，甚至可以运用启发式杀毒规则，检测分析软件的代码行为，来判断从未出现过的新型恶意软件。

[5] Hyrum S Anderson and Phil Roth. 2018. Ember: an open dataset for training static PE malware machine learning models. arXiv preprint arXiv:1804.04637 (2018).

[15] George E Dahl, Jack W Stokes, Li Deng, and Dong Yu. 2013. Large-scale malware classification using random projections and neural networks. In 2013 IEEE Inter national Conference on Acoustics, Speech and Signal Processing. IEEE, 3422–3426.

[55] Edward Raff, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, and Charles K Nicholas. 2018. Malware detection by eating a whole exe. In Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence.

[56] Konrad Rieck, Philipp Trinius, Carsten Willems, and Thorsten Holz. 2011. Auto matic analysis of malware behavior using machine learning. Journal of Computer Security 19, 4 (2011), 639–668.

[60] Joshua Saxe and Konstantin Berlin. 2015. Deep neural network based malware de tection using two dimensional binary program features. In 2015 10th International Conference on Malicious and Unwanted Software (MALWARE). IEEE, 11–20.

[61]J.X. Geng, J.F. Wang, Z.Y. Fang, Y.J. Zhou, D. Wu, W.H. Ge

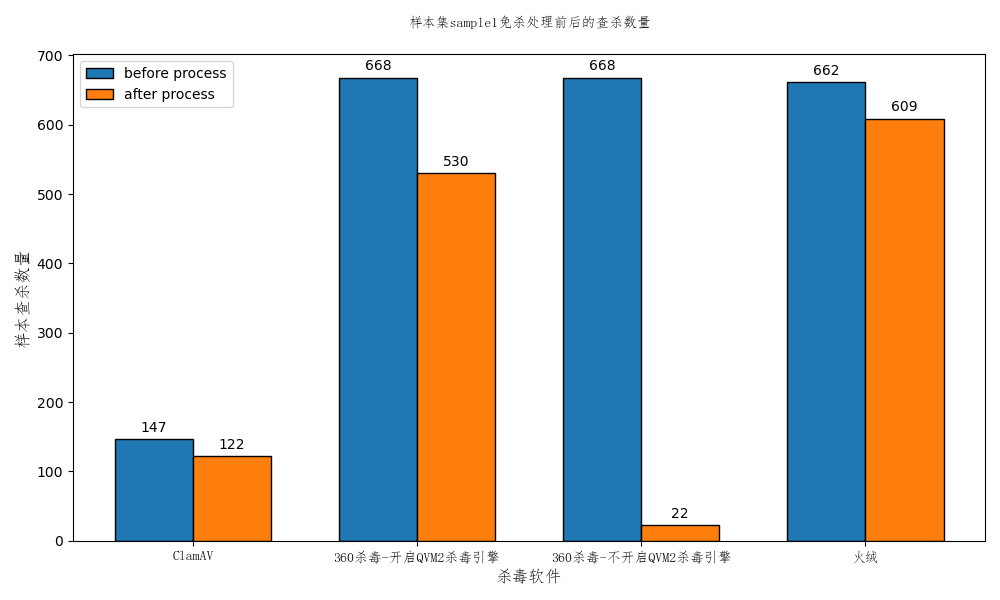
A survey of strategy-driven evasion methods for PE malware: transformation, concealment, and attack

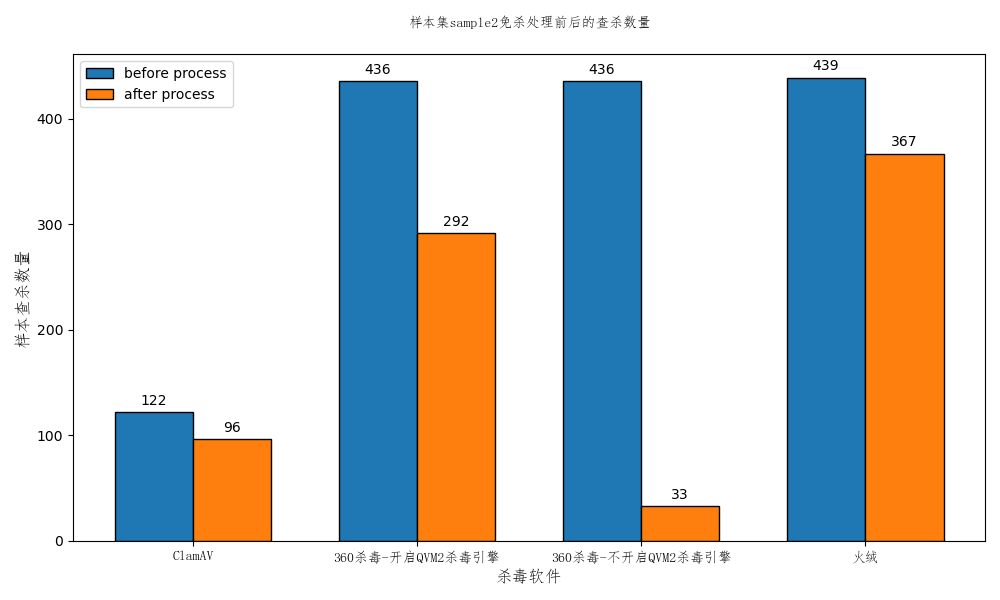
Comput. Secur., 137 (2024), Article 103595, [10.1016/j.cose.2023.103595](https://doi.org/10.1016/j.cose.2023.103595)

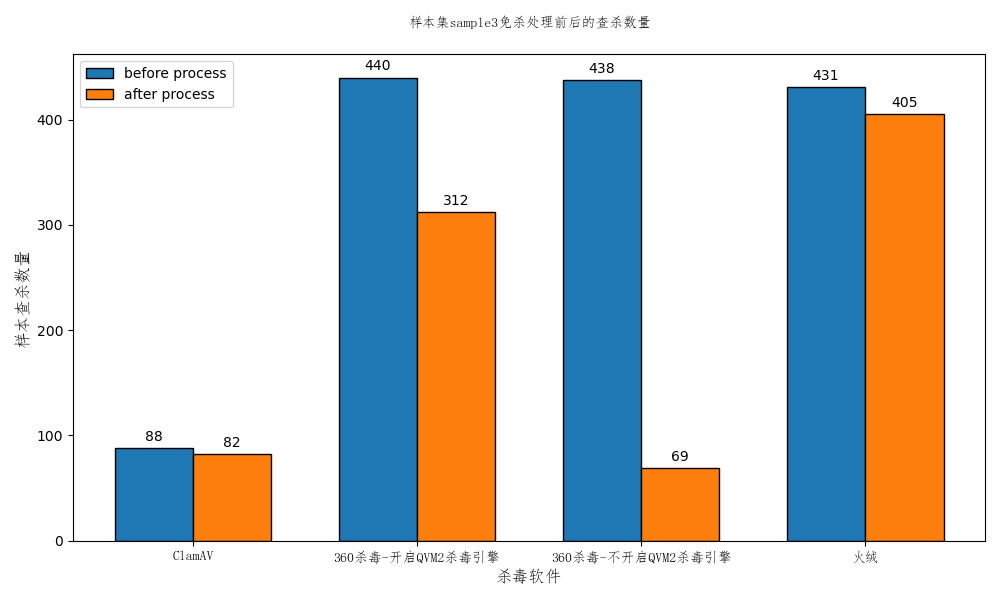
但不幸的是，某些基于静态特征分析的反病毒软件在某些情况下有着很高的误报率，可能因为机器学习模型自身的问题，误判一些正常的软件的行为，认为这些正常软件是恶意软件，例如QVM反病毒引擎误报Microsoft Visual Studio Complier、Clang等C/C++语言编译器。此外，基于机器学习的反病毒软件也存在严重的漏洞，攻击者只需要对恶意软件进行修改，甚至有时只需要增加一个资源文件改变恶意软件的Hash值，就能绕过反病毒软件的检测，恶意软件的编写者的规避技术对于反病毒软件的识别带来了极大的挑战，这促使攻防对抗的双方不断采取更先进的措施。

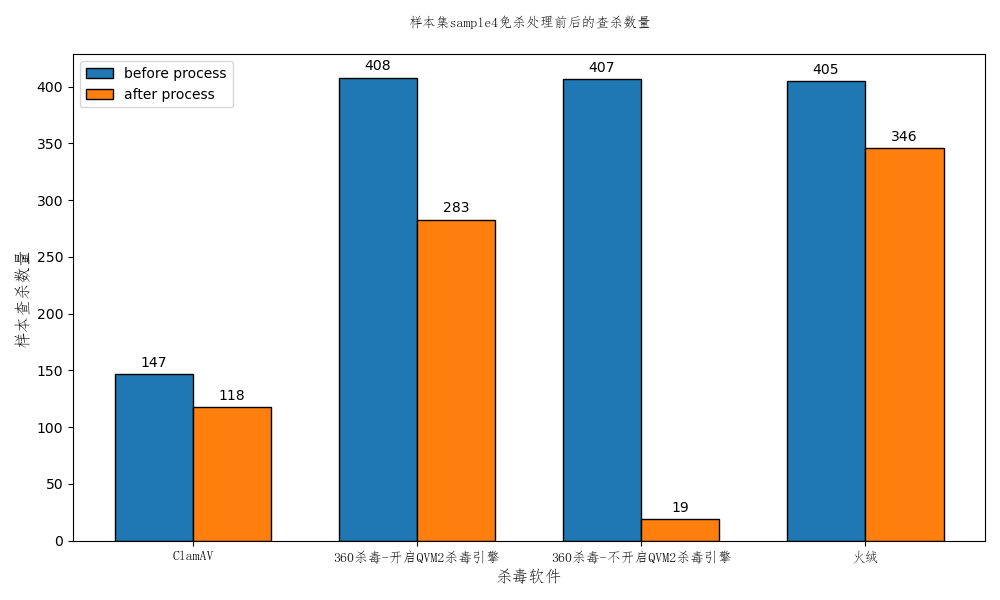
**关于实验结果的讨论：**

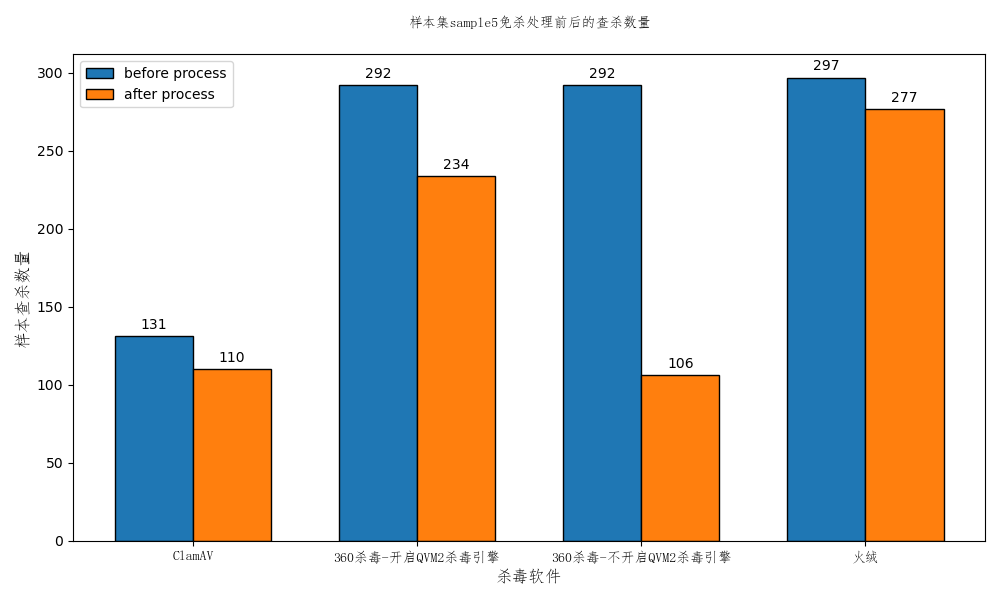
在此实验中，笔者一共使用了五组数据，合计大约2000个样本来进行测试，测试结果如下所示。



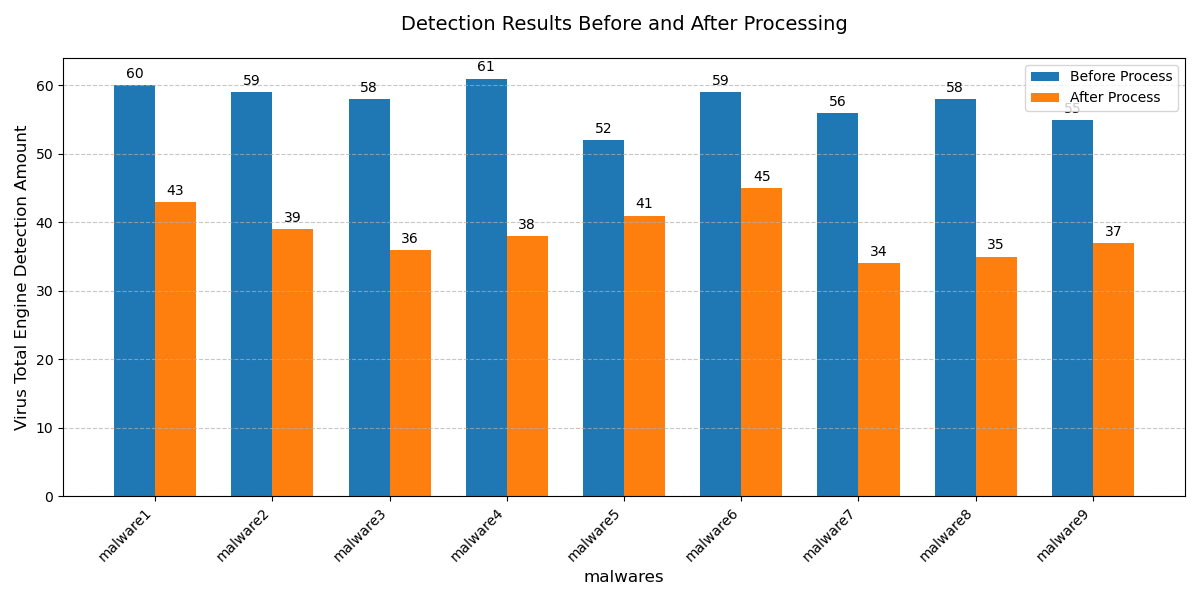








此外，对于某组样本中的部分程序，笔者使用了Virus Total来进行上传扫描，结果如下。



以下是笔者在结果研究中遇到的一些问题的思考以及一些建议。

**问题1：为什么火绒和ClamAV对于免杀处理后的样本和处理前的样本检出差距相比360杀毒比较低？**

笔者发现，处理后的样本对于火绒杀毒来说，逃逸概率低的原因是火绒有一定的反扰动能力，处理后的样本很多都会报毒如下：



说明有一部分反病毒软件的静态特征分析能力，能识别出对抗样本中的扰动。这也是为什么采用SARSA算法不宜设置过大的state数值的原因，一方面是处理前后的PE程序反复读写，当程序大小因为新增节，新增无意义尾部内容，用无意义字节填充空洞等操作多次使用后变的过于庞大（在某次样本处理时，笔者观察到了一个PE程序处理前后，变为原大小大约6.6倍，从2.67 MB变成了17.6MB），如果不限制state值，程序中途会变得很大，不但导致后续免杀操作极其缓慢，因为要先读取原文件，然后再加以修改，最后写入磁盘中，而且会导致根据查杀结果获取result的速度相当慢，因为ClamAV反病毒软件扫描一个样本的时间也和样本大小正相关，当样本很大的时候，判断查杀结果会相当的慢。另一方面，当无意义尾部内容过多，会被反病毒软件认为是疑似恶意程序刻意扰动，从而直接判定此程序为为了绕过反病毒软件检查的恶意程序。而从火绒杀毒对于样本的描述也可以看到这一点，火绒杀毒认为经过处理后的某些恶意程序是代码混淆器。

同样对于ClamAV，可能它的部分检测也能对抗样本中的扰动，尽管经过处理后的恶意程序中有大量良性内容，但ClamAV仍然会查杀这些恶意程序。

**问题2：为什么需要研究QVM引擎开启先后的情况？**

这是因为QVM作为一个自学习的人工智能引擎，这个病毒检测引擎是基于机器学习的模型，通过检测文件的某些特征，来判断该文件是否是恶意程序，如果某程序被QVM引擎检测出是恶意程序，那么360杀毒在查杀结果中会报HEUR/QVM.xx.Malware.Gen字样。这里使用QVM引擎作为检测标准是为了研究论文中的强化学习对抗性样本生成模型产生的处理后样本在基于不同模型的反病毒软件之间的逃逸可迁移性，以及验证机器学习的病毒检测引擎对于样本扰动的对抗能力。但遗憾的是，QVM引擎对于样本扰动的对抗能力比较差，有时甚至仅仅增加了无意义的图标ICO资源，QVM引擎就认为样本是良性的。

**问题3：为什么不测试360杀毒的鲲鹏引擎？**

这是因为大多数反病毒软件，倾向于防御现有的威胁和威胁变种，而不是没有广泛传播的对抗性样本，鲲鹏引擎也如此，它多被360杀毒用于对抗目前广泛流行的恶意样本（例如银狐和某些勒索病毒），而非对抗性样本或是偏旧的样本。这是因为有时需要反病毒软件对未知程序的判断更加精准以及判断速度更快，也就是尽量降低误报的可能性以及判断时间，很多反病毒软件厂商可能会使用白名单数字签名放行，或是使用文件哈希白名单来进行放行。因为例如Intel，AMD的驱动程序，它们会加载驱动，但这跟某些病毒的行为很相似，某些病毒也会使用驱动来提权使自己具有更强的破坏力以及针对反病毒软件，如果这些硬件外设驱动程序因为释放了一些驱动文件被误报没有放行，将会导致某些外设驱动不能正常安装，对用户来说后果无疑是灾难性的。

**问题4：对抗性样本攻击对于使用反病毒软件的用户是否有很大的危害性？**

这里笔者是在VMware创建的Windows7虚拟机中使用360杀毒进行测试静态免杀情况，并且在360杀毒的设置中打开了自动上传发现的可疑文件，尽管部分处理后的样本逃逸了360杀毒的自定义查杀的检出，但是360杀毒仍然将这些逃逸样本归类为可疑文件。虽然这些逃逸样本不会报为高危风险项，但是360杀毒仍然会将这些样本上传到360云安全中心来进行检验，在可疑样本被360杀毒上传一段时间后，再次启动自定义查杀扫描剩余的逃逸样本，有部分样本被360杀毒检出判定为高危风险项。说明对于开启未知程序上报的云查杀（笔者后续得知360云安全中心是有动态行为检测的，并不完全只有静态检测）的安全软件来说，对抗性样本攻击虽然有一定危害，但大多数仍然最终会被检出是恶意程序。同时，360杀毒也具有动态检测功能，例如Behavioral脚本引擎、主动防御模块这些基于动态行为分析的检测，如果用户执行了恶意程序，360杀毒会拦截并且隔离恶意程序。当反病毒软件具有动态检测功能，则静态对抗性样本攻击的效果是有限的，但仍然不可以忽视。究其原因，是因为某些病毒具有反虚拟化策略、反沙箱、识别监控进程的功能。

Yokoyama A., Ishii K., Tanabe R., Papa Y., Yoshioka K., Matsumoto T., Kasama T., Inoue D., Brengel M., Backes M., *et al.*

Sandprint: Fingerprinting malware sandboxes to provide intelligence for sandbox evasion

International Symposium on Research in Attacks, Intrusions, and Defenses, Springer (2016), pp. 165-187

Rudd E., Rozsa A., Gunther M., Boult T.

A survey of stealth malware: Attacks, mitigation measures, and steps toward autonomous open world solutions

IEEE Commun. Surv. Tutor., 19 (2) (2017), pp. 1145-1172

Bulazel A., Yener B.

A survey on automated dynamic malware analysis evasion and counter-evasion: PC, mobile, and web

Proceedings of the 1st Reversing and Offensive-Oriented Trends Symposium, ACM, ACM, New York, NY, USA (2017), p. 2

Apostolopoulos T., Katos V., Choo K.-K.R., Patsakis C.

Resurrecting anti-virtualization and anti-debugging: Unhooking your hooks

Future Gener. Comput. Syst., 116 (2021), pp. 393-405

云查杀技术对这些病毒来说未必有效，因为在被上传到云端沙箱（例如Cuckoo Sandbox）中之后，恶意软件不会表现出预期的恶意行为，从而阻碍了恶意软件行为分类和恶意行为的自动识别检测。过度依赖云查杀仍然可能存在潜在的威胁，同时，云查杀需要联网的问题也是一大隐患，对抗性恶意程序可能在自己被运行时，因为反病毒软件因为扰动导致判断该恶意程序是未知安全性，需要上传后利用云查杀技术进行分析，而这时的恶意程序自身可能会通过修改网络设置导致计算机无法连接到互联网，或利用Windows防火墙规则，本地Hosts文件篡改反病毒软件请求服务器地址到本地环回地址127.0.0.1等方式使反病毒软件无法连接到互联网，从而阻止了反病毒软件从云端获取云端沙箱传回来的信息，这就导致反病毒软件对对抗性恶意程序无法进行精确地判断。更糟的情况是当云查杀技术面对可能出现的感染型病毒变种对抗性样本和蠕虫对抗性样本。

传统的感染型病毒，例如文件型病毒，只感染固定的目标文件，例如PE可执行文件（.exe），动态链接库文件（.dll）。

"Parasitic malware: The resurgence of an old threat", Network Security, vol. 3, pp. 15-18, 2008.

其中的一部分病毒通过将自身的代码后附加在正常程序后，当程序被运行时，其中的病毒代码也随之运行。然而，传统的感染型病毒很容易被反病毒软件查杀，因为反病毒软件只需要扫描受感染的文件，通过静态分析，检查其尾部内容是否有黑名单中的恶意代码段，即可判断这个文件是否是被感染，而且反病毒软件对被感染的文件可以通过匹配相应字符串，如通过正则表达式匹配，直接去除截断掉尾部的病毒代码。因为病毒代码和正常程序代码是物理分离的，所以反病毒软件的操作可以将受感染的恢复到原始文件，即对被感染的文件实行修复而不是直接删除或者隔离。

然而，通过本实验，可以说明，将来很可能存在感染型病毒变种对抗性样本。不同于传统的感染型病毒，它的危害较大，具有感染新文件后，自身也会突变不断制造新变种的新型攻击。这种病毒除了自身的恶意代码，也携带了对抗性扰动操作的相关模块，例如内部包含一个被pyinstaller打包的附带Python编译的PE可执行文件，用于产生对抗性扰动来针对基于机器学习的杀毒引擎以实现逃逸。在病毒因为用户运行导致入侵未被感染的系统后，该病毒会在磁盘中通过遍历文件系统的目录来寻找新的可被该病毒感染的目标文件，将自身代码附加在被感染程序后，并判断产生对抗性扰动操作的程序对应的Environment\_Variable\_α环境变量是否存在，若存在则调用产生对抗性扰动的可执行程序对被新感染的程序进行处理，否则释放可产生对抗性扰动的可执行程序，随后将其移动到安装有操作系统的磁盘的某个位置（可以为系统根目录）下，随后设置文件为隐藏属性，添加该可执行程序到系统的环境变量中，最后调用产生对抗性扰动的可执行程序对被新感染的程序进行静态免杀处理。··

该变种对抗性样本的部分行为可用伪代码和程序流程图表述如下：

start();

scan\_all\_files\_in\_system\_to\_get\_can\_infected();

function1:

if System\_Environment\_Variables.contains(Environment\_Variable\_α):

    for file in files\_virus\_can\_infected:

        append\_malicious\_code(file);

        run\_evasion\_module(Environment\_Variable\_α);

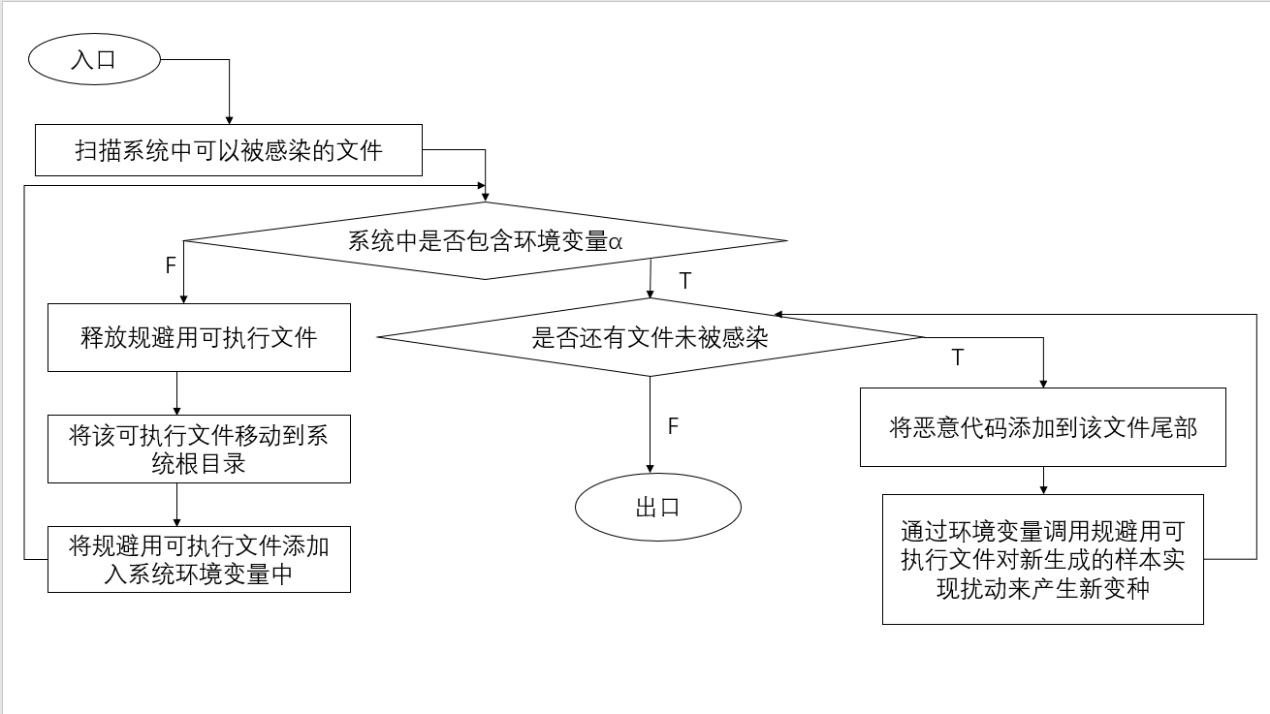
else:

    release\_executable\_file\_for\_evasion();

    move\_evasion\_module\_file\_to\_system\_root\_folder();

    add\_environment\_variable\_to\_system\_environment\_variables(Environment\_Variable\_α);

    go to function1;



这种感染型病毒变种每次新感染的程序的尾部病毒代码将互不相同，因为受到了对抗性扰动可执行程序的修改。基于静态分析的反病毒软件很难查杀这种变种对抗性样本，很容易面临能杀毒，但是查杀不完全的问题，每次杀毒如果不能检测出感染型病毒变种对抗性样本感染的全部文件，遗留下来的受感染文件一旦被用户无意间通过启动，例如通过某些Windows应用程序的开机自启动，用户打开浏览器访问网页但是浏览器主程序已被感染，仍然会导致其他文件被感染。更危险的是，被感染的文件难以像应对传统文件型病毒的修复措施，即精准去掉尾部的病毒代码来恢复原始文件。因为需要处理对抗性扰动新增的节和无效字节，甚至有可能是导入表被篡改，节表被修改的行为，一旦去除有偏差，则会导致受感染文件无法正常运行。因此反病毒软件极有可能无法通过黑名单中的病毒代码来计算线性偏移量，进而直接截断固定长度的尾部从而修复直接受感染文件，只能选择删除，这对用户来说是危害巨大的，因为很多的Windows软件包含了大量的动态链接库（.dll）文件和PE可执行文件（.exe），一旦某些动态链接库被反病毒软件删除后，这些软件极有可能无法正常运行。即使反病毒软件拥有云查杀功能，频繁上传大量变种样本不但消耗了用户带宽资源，而且变种样本因为每个被感染的实例的尾部特征值唯一，难以在第一次扫描中就被反病毒软件检测出是恶意程序，只会被定性为不流行的可疑文件，这导致对感染型病毒的响应延迟，很有可能会面对云查杀查杀病毒并不彻底的问题。幸运的是，即使面对这种将来可能存在的感染型病毒变种对抗性样本，因为它只有不断产生静态规避变种的功能，而没有不断产生动态规避变种的功能，反病毒软件的动态行为检测仍能在病毒程序未执行感染操作时触发，识别出恶意程序并警告用户。同时，PE文件的动态规避目前尚未出现规模化的攻击方法，尽管已有研究证明可通过代码分片来攻击动态行为检测。

[33]2012 IEEE Symposium on Security and Privacy

Smashing the Gadgets: Hindering Return-Oriented Programming Using In-Place Code Randomization

Vasilis Pappas, Michalis Polychronakis, and Angelos D. Keromytis

Columbia University

[50] Jithin Pavithran, Milan Patnaik, and Chester Rebeiro. 2019. D-TIME: Distributed Threadless Independent Malware Execution for Runtime Obfuscation. In 13th USENIX Workshop on Offensive Technologies (WOOT 19). USENIX Association, Santa Clara, CA. https://www.usenix.org/conference/woot19/presentation/ pavithran

[29] Kyriakos K. Ispoglou and Mathias Payer. 2016. malWASH: Washing Malware to Evade Dynamic Analysis. In 10th USENIX Workshop on Offensive Technolo gies (WOOT 16). USENIX Association, Austin, TX. https://www.usenix.org/ conference/woot16/workshop-program/presentation/ispoglou

因此，在特定的条件下，如过度依赖本地静态检测忽视动态上传上报可疑文件、过度依赖云查杀的动态检测沙箱和云端病毒库遭遇网络中断、反病毒软件无条件直接信任某些公司签发的安全证书，很大可能会导致静态规避攻击绕过杀毒软件的防御，构成实际威胁。

**对研究人员的建议：**

本研究使用对抗样本对采用机器学习技术的病毒查杀引擎进行黑盒测试，但需要注意，对某些特征的修改和删除，是否是修改容易被规避的脆弱特征、容易损坏原有恶意文件恶意行为的特征，例如机器码，本研究中尽管代码中保留了修改机器码的函数，但最终并未将其封装为操作函数，而是将其弃用，因为修改机器码在测试中并未通过，测试时使用了一个输出HelloWorld的C语言机器码为X86-64 PE可执行程序，在修改机器码后，导致代码段重定位失败，程序无法执行、以及一些容易导致程序被直接查杀的特征，例如UPX加壳，在测试中，只采取UPX加壳，QVM引擎会直接判定该程序是恶意程序，甚至被UPX加壳的文件只是毫无危害的HelloWorld C语言程序。而采用UPX加壳以及多种方式混合，QVM引擎认为该程序是恶意程序的概率会大幅下降。同时，证明修改后的样本对计算机仍然有危害是至关重要的，但需要注意测试的安全问题，因为几乎没有一种杀毒软件能100%的概率查杀静态处理后的样本，即使在测试中使用虚拟机，仍然可能会存在虚拟机逃逸问题，例如VMware WorkStation的漏洞CVE-2023-34048和CVE-2023-34056，测试前尽量对重要数据进行备份以及设置虚拟机还原点，防止运行恶意程序危害到实体机的意外发生。

**对反病毒软件厂商的建议：**

本研究表明静态分析器容易被攻破，虽然目前没有出现前文中提及到的感染型病毒变种对抗性样本这种对静态分析针对性逃逸的流行性恶意软件，但规避静态分析的研究仍然具有价值，这也是为什么EMBER，MalConv等静态分析模型日益被关注的原因。

MAB-Malware: A Reinforcement Learning Framework for Attacking Static Malware Classifiers

arXiv:2003.03100v3 [cs.CR] 29 Apr 2021

此外尽量不要过度依赖于本地病毒库的Hash值匹配病毒扫描策略，因为静态规避样本相比原样本Hash值不再相同，且达成静态规避的处理步骤不同，也会使不同的静态规避样本的Hash值不相同。同时，对某些异常PE可执行文件，例如尾部具有过长的字符串，可以考虑将它们判断为代码混淆器，疑似风险文件提示给用户，并且上传到反病毒软件的云安全中心进行分析，而不是直接认为是无危害文件，尽管这样来说会有误报的可能，但是可以对抗一部分的规避静态分析的对抗性样本。此外，需要注意加强反病毒软件的自我保护功能和动态行为监测机制。同时，反病毒软件厂商也需要注意静态病毒库和动态病毒库的存放问题，本地可以尽量少存放静态病毒库，只存放一些近期流行的恶意程序样本，并且一段时间进行本地病毒库更新，这样可以为用户节省磁盘空间；而动态高风险动态行为检测库最好存放在本地，这样即使在网络环境不佳或者是离线状态，仍然能拦截病毒的运行，防止病毒危害计算机和造成用户损失。同时，云端病毒库可以对某些Hash值或者携带某些有效签名的程序的动态高风险行为设置白名单，例如一些签名注册工具。上述操作在网络正常的情况下，可以在经过本地高风险行为检测后，再进行云端查询判断程序是否是恶意程序，尽管会有离线模式下无法连接到云端病毒库导致误报某些非恶意软件的敏感操作的问题，但是可以很大程度上规避恶意程序运行造成的损失。在本实验中使用的Clam AV反病毒软件是基于本地离线病毒库，需要手动更新，大约有630MB大小。

**动态检测对于静态对抗性样本的缓解作用：**

尽管本实验的方案无法绕过动态检测，但静态规避研究仍具有重要价值，可以为用户执行潜在的恶意软件提供预判防御。动态规避属于另一个研究方向，目前shellcode的逃逸检测上已经出现了动态规避，通过异或加密shellcode后封装在C/Python代码的数组中，在执行时解密即可实现动态规避。然而动态检测相比静态检测，对某些恶意软件的检测效果并不佳，因为有些恶意软件会在运行时试图破坏反病毒软件，例如关闭相应的防御服务，加载驱动让自己常驻内存来对抗反病毒软件等操作，这就需要杀毒软件的一些自我保护机制来应对。

**总结：**

在本实验中，笔者使用了SARSA算法构建了一个强化学习的静态查杀规避处理框架，用于生成对一些黑盒的反病毒软件和反病毒引擎的对抗性扰动静态规避攻击样本，动作选择可以认为是多臂老虎机问题，在有限的尝试次数下探索较好的行为，尽量规避无效动作，降低修改总次数来实现规避概率最大化。实验表明，处理前后在Clam AV和火绒杀毒下检出率变化较小，这可能是因为这两款杀毒软件自带一定的抗扰动能力，能对抗本实验中对于恶意软件的针对静态检测的扰动处理，然而360杀毒的云查杀引擎在没有云上传功能下，对处理后的恶意软件样本查杀率很低。实验表明了，在采用相似检测机制的反病毒软件之间，该模型具有可迁移攻击特性，且能推测出QVM反病毒引擎的抗扰动能力弱于Clam AV。