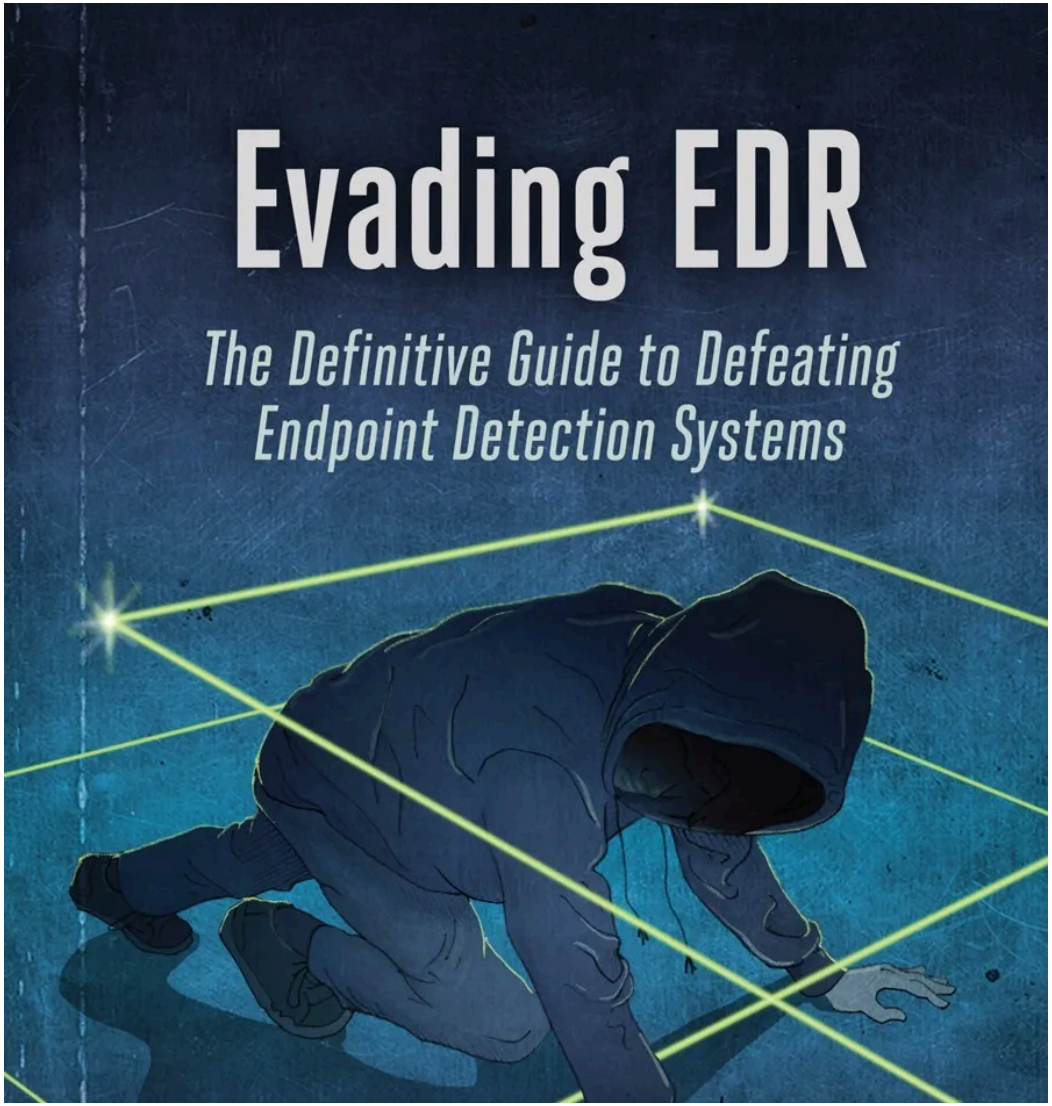
资深红队专家谈EDR的工作原理与规避

原创 Zer0d0y 天御攻防实验室 2024年03月27日 15:22 广东



上个月资深红队专家Matt Hand做了一场EDR原理与规避的分享，Matt Hand（马特·汉德）是一位有着十多年经验的红队操作员，同时也是《规避EDR - 战胜端点检测系统的终极指南》（Evading EDR）一书的作者，他主要关注的领域是漏洞研究和EDR规避，并且也是规避技巧方面的主题专家。

笔者曾学习过一些EDR绕过相关的文章和演讲，但是大多数此类文章，多停留在攻击者视角思考看待EDR。Matt由于其独特的背景（曾在业界网络攻防明星公司SpecterOps担任服务架构师），所以其能够从进攻方和防御方的视角，全面审视终端安全对抗。

通过今天这篇文章，我想再次传达这样一个观点：

> 真正的对抗，是人与人之间的对抗！
> 真正的对抗，是体系对抗！
> ---- Zer0d0y

总之，演讲非常精彩！以下是笔者整理的相关内容分享给大家。

主要内容：

Matt首先介绍了端点检测与响应系统（EDR）的工作原理。EDR是由多个组件构成的系统，通过在终端上部署传感器来收集终端行为数据，并将数据发送到云端进行分析检测。

然后，详细解释了EDR系统的演进过程和不同时期所使用的技术手段，包括内核驱动程序、微过滤驱动、ETW、网络过滤驱动等。

然后，分析了EDR系统检测恶意行为的两种模式：精准检测和广泛检测，并指出需要同时使用这两种检测模式才能构建全面的终端安全防御。

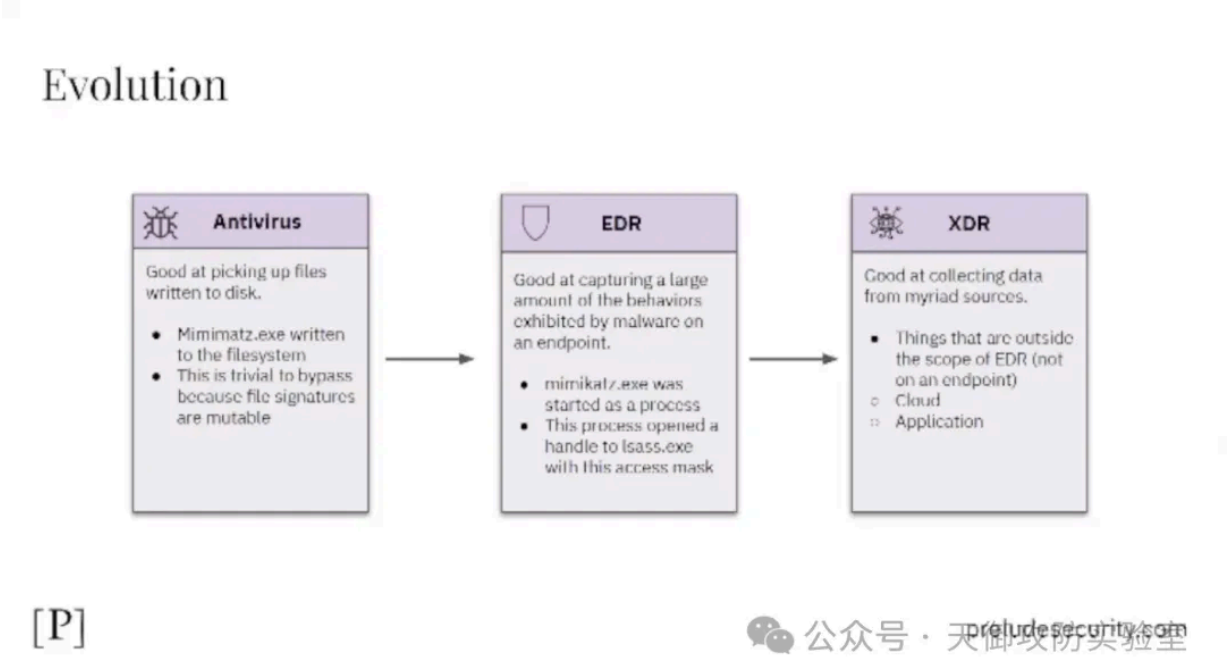
最后讨论了如何评估EDR系统的有效性，以及在实际应用中如何权衡误报和漏报的比例。Matt强调，EDR系统的关键不在于能够将各个数据点关联起来，而在于能够准确检测每一个数据点，为调查人员提供分析线索。

EDR系统内幕

大家好，我是Matt Hand。我在Prelude公司从事安全和研究相关的工作。我整个成年生涯都在从事红队渗透工作，辗转于多个不同的领域，主要以安全顾问的身份。最近，我在SpectreOps公司从事一些有趣的工作。我去年出版了一本关于EDR如何工作以及如何绕过它的书，这是多年努力的结晶。今天，我想带大家了解一下EDR系统的高层架构，它是由哪些组件构成的，它们在底层是如何运作的。

很多人可能认为EDR是一个极其复杂的黑盒子，只知道有恶意行为输入，然后检测结果输出。但实际上EDR虽然比较复杂，却并不神秘。我们将依次讨论EDR的组成部分、系统架构以及实际的运行机制，用一种与厂商无关的方式来审视EDR这个系统本身。然后我们再来看EDR所处理的数据以及如何从更高的层次利用这些数据。

先来回顾一下历史。我刚入行的时候，还在到处"喷"MS-08-067漏洞，用Metasploit做横向移动，PSExec连选项都没有。没错，我入行很多年了。那时候有杀毒软件。去RadioShack买一盒Norton软件，装在电脑上就可以用了。杀毒软件对文件系统上的威胁检测非常在行。它有个特征库，可以识别文件的哈希值和其他属性，判定是否为恶意文件并清除。在当时那是了不起的发明，但攻击者很快就适应了。红队界很快就出现了PowerShell Empire、Veil等规避框架，我们很快就不把杀毒软件放在眼里了。



于是，EDR的理念就应运而生。既然无法可靠检测磁盘上的恶意程序，那能不能关注恶意程序的行为特征呢？比如Mimikatz.exe，它在系统里启动一个进程。单凭Mimikatz.exe存在于磁盘并不意味着存在恶意行为，但它的启动则预示着恶意行为的发生。所以EDR把重点放在"启动"这个行为上。即便将Mimikatz改名为Minidogs，其行为特征依然如故。它仍会以特定的硬编码的访问权限打开LSASS的进程句柄。EDR的职责就是收集终端的各种行为数据。

近几年，业界提出了XDR的概念，很多人误以为这是个全新的东西，但其实不然。终端可以泛指任何能执行代码的系统，并不局限于工作站、笔记本等传统计算机。还有云端系统，应用程序本身也会生成日志。这些不同的数据源其实

都是终端的一种。XDR的任务就是收集所有这些异构数据源的信息，集中到一处，以便关联分析不同系统间的行为。这其实与SIEM的功能更接近。XDR可以检测从内网到云端的攻击活动，在此类场景下大显身手。但本次讨论我们只关注Windows终端上的EDR。

EDR内部原理

市面上的EDR产品可以分成几类。它们目前基本处于同一水平。每个厂商都声称自己有独门秘技。SentinelOne有他们的，CrowdStrike也有他们的，否则大家都去买最便宜的产品了。EDR本身不是单一的东西，而是由不同组件构成的系统，每个组件负责不同的任务。EDR的核心组件包括：

- 传感器：负责收集终端行为数据或生成遥测数据

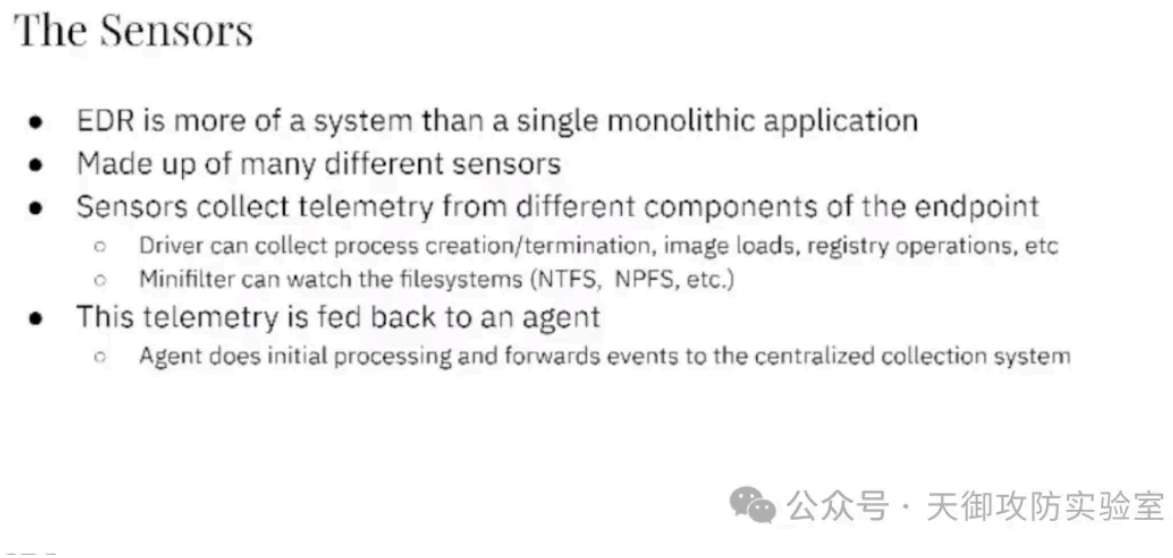
- 代理：接收传感器数据，并发送到云端

- 数据收集系统：接收并存储所有原始事件数据

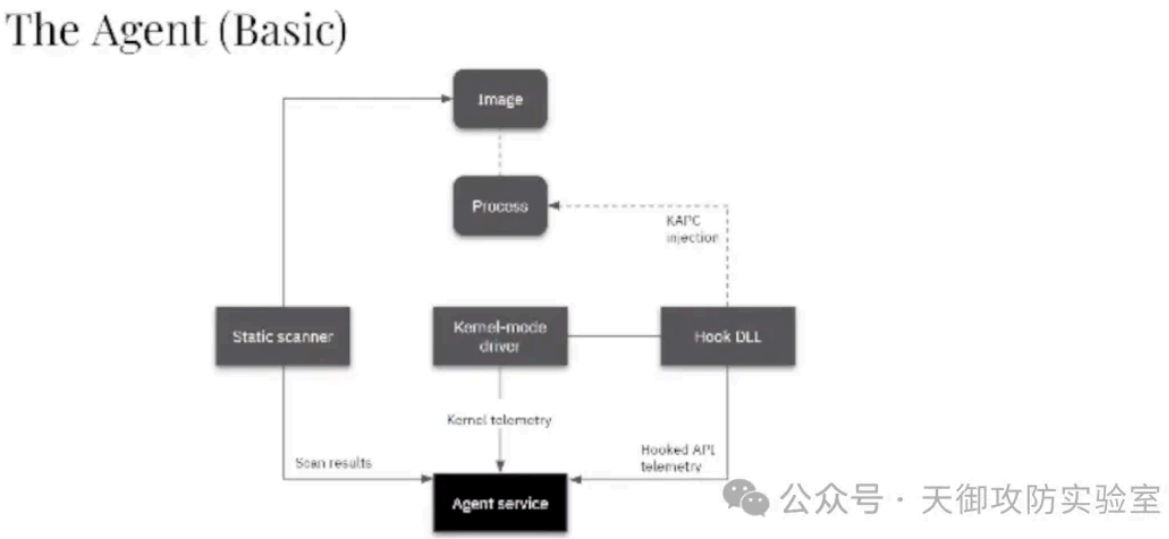
- 关联分析引擎：建立事件间的关联

- 检测引擎：根据规则和模型识别威胁

其中最关键的是驻留在终端上的组件，因为绝大部分的规避机会就存在于此。我们将重点关注终端上的传感器。



EDR传感器是分布在系统各处的一些小程序，根据其设计用途收集特定的行为数据。不同类型的传感器负责从不同的数据源收集数据，尽管来源可能有所重叠，但各司其职。初期的EDR可能只包含少数几个组件。Carbon Black是我最早接触到的EDR之一，它非常简单。在我看来，一个像样的EDR怎么也得有个内核驱动。大多数EDR仍然使用某种杀毒引擎，毕竟识别磁盘上的Mimikatz.exe总归是个优势。我真不理解那些对杀毒嗤之以鼻的人。杀毒在它擅长的领域表现出色，只是覆盖面不够广而已。在EDR产品中，你仍然会看到厂商捆绑自己的杀毒引擎，甚至通过DLL注入到进程以监控某些API函数的调用。



其中最重要的可能要数内核驱动了。驱动加载进内核后，可以订阅感兴趣的系统事件。比如进程创建/终止、线程创建/终止、文件句柄打开/复制、注册表操作、可执行映像加载（EXE、DLL、驱动）等。这些是最常见的事件类型。此外还有一些不那么常用的，例如内核启动前的反恶意软件扫描，负责检查在系统启动前就加载的恶意驱动，但这个功能现在很少使用了。驱动监控这些行为，生成遥测数据，而且很难被篡改。规避用户态的API Hook非常容易，大家这些年来想出了很多办法，直接系统调用、反Hook应有尽有。这就是为什么我们不再把API Hook当作主要的数据源，顶多作为其他数据源的补充。**如果由于种种原因导致遥测数据没有正常到达，倒是能提示EDR使用者可能存在规避企图。渗透人员需要留意这点。**

相比之下，要对付驱动就难多了。要么得有管理员权限强制卸载它，要么得自己写驱动修改它的代码。但无论如何都需要较高的权限，而获取这些权限的过程本身就可能会被EDR察觉。传感器收集到的所有数据，都由代理服务接收、处理并上传。这就是早期EDR的基本构成，跟我们今天见到的相比还略显简陋。

如今的EDR传感器种类就丰富多了。这些新增的传感器让EDR对系统行为的感知更加全面。最常见的有：

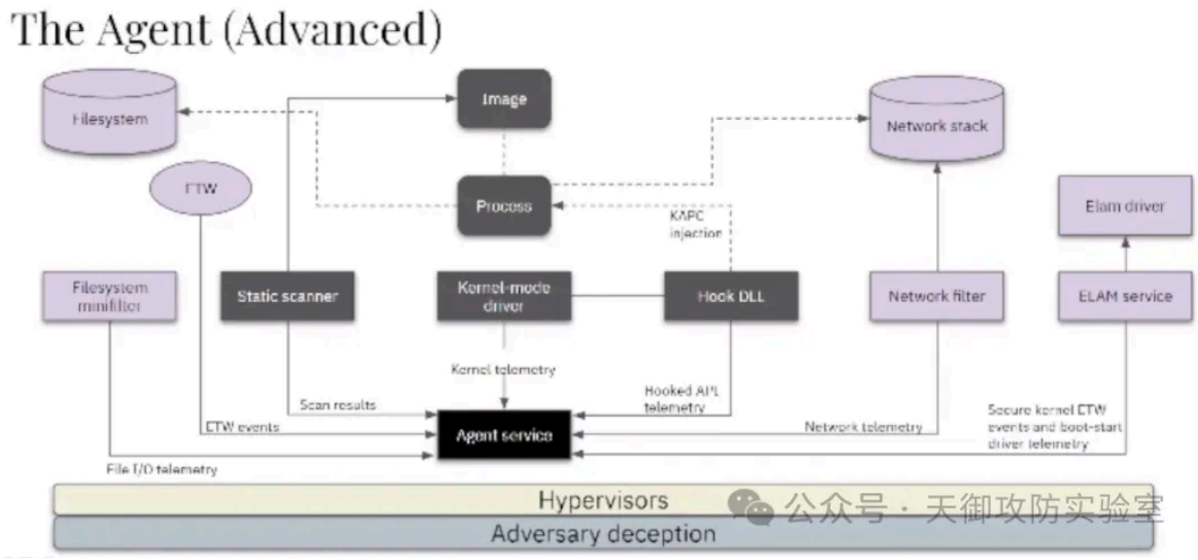
- 微过滤驱动（Minifilter）：监控文件系统活动，不限于NTFS，还包括命名管道、邮槽（mailslot）等

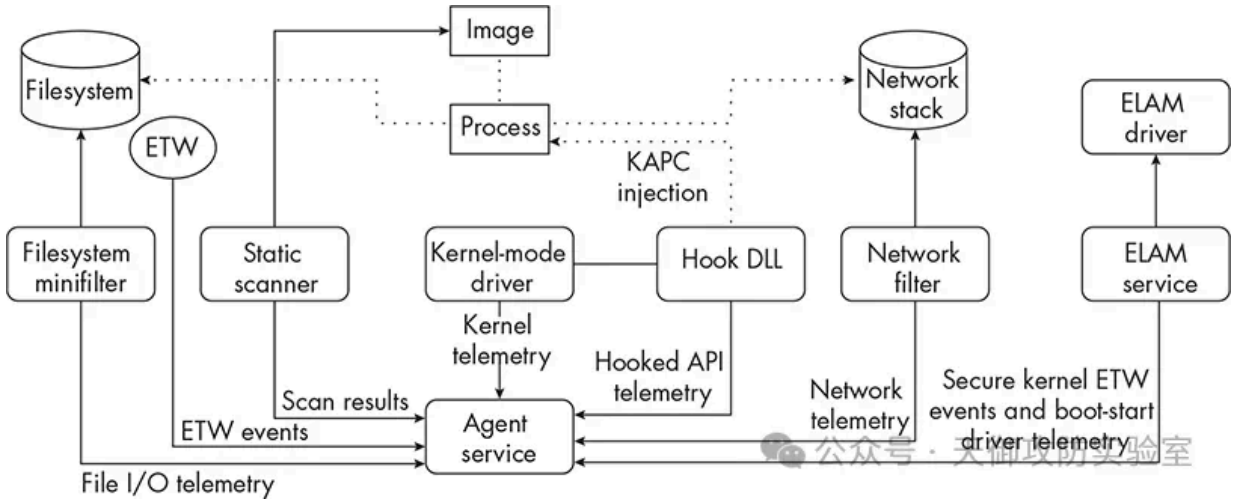
- ETW：系统服务、任务计划程序等组件会通过ETW报告自己的事件

- 网络过滤驱动：利用Windows筛选平台（WFP）实时捕获网络通信数据

- 反恶意软件服务：启动前扫描驱动程序

其中ETW和网络过滤驱动的应用最为广泛。ETW可以采集到进程加载.NET程序集、进程访问其他进程内存等事件；网络过滤驱动可以像窃听器一样监控所有网络连接，当然也可以拦截、重定向流量。现在反恶意软件驱动主要用于订阅系统的威胁情报ETW Provider，而不是扫描驱动了。要使用这个Provider，必须嵌入一个ELM驱动，微软会验证其数字签名。没有这个签名，就别想用威胁情报。



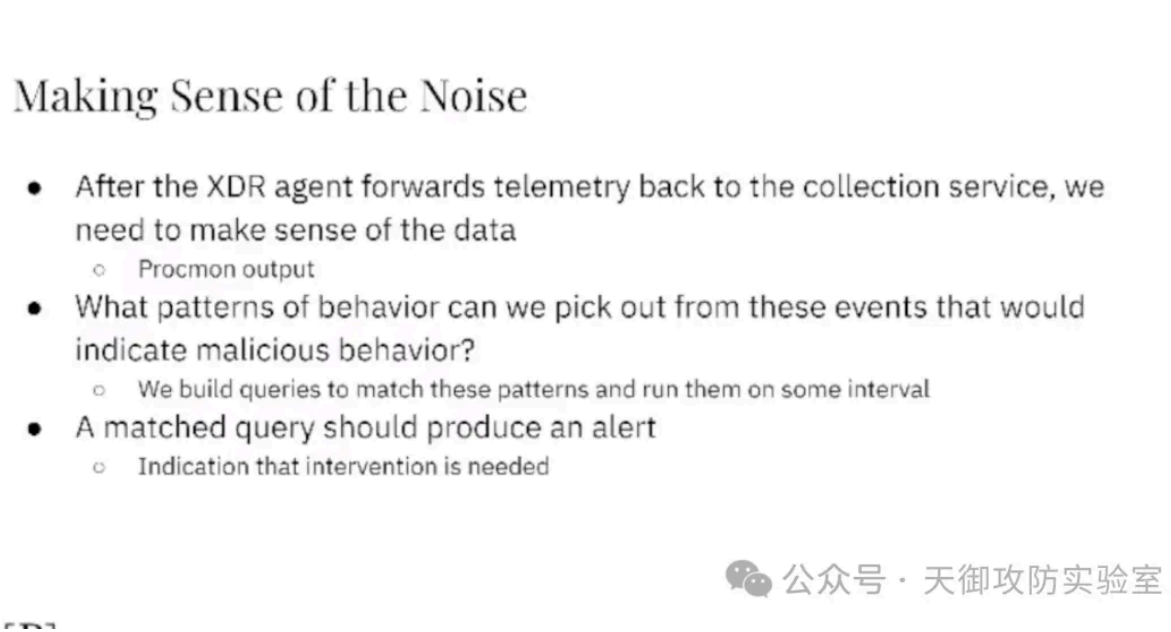


顶级EDR产品甚至会集成更高级的传感器，例如虚拟机监控器。它位于内核之下，可以监视进程在用户态和内核态之间的切换。虚拟机监控器在漏洞防护和检测方面别有用处，目前还在积极研究之中。有兴趣的可以私下找我要几篇论文。

总而言之，今天的EDR本质上就是一组分工明确的传感器，分布在系统的各个层面，收集不同的行为数据汇总到代理服务，再由代理上传到云端，供后续的关联分析和威胁检测。千万别小看这些数据，一个大型企业每天要产生数TB的日志，把这些数据都传到云上可是个挑战。所以云端还得做数据清洗和归类存储。

EDR规避

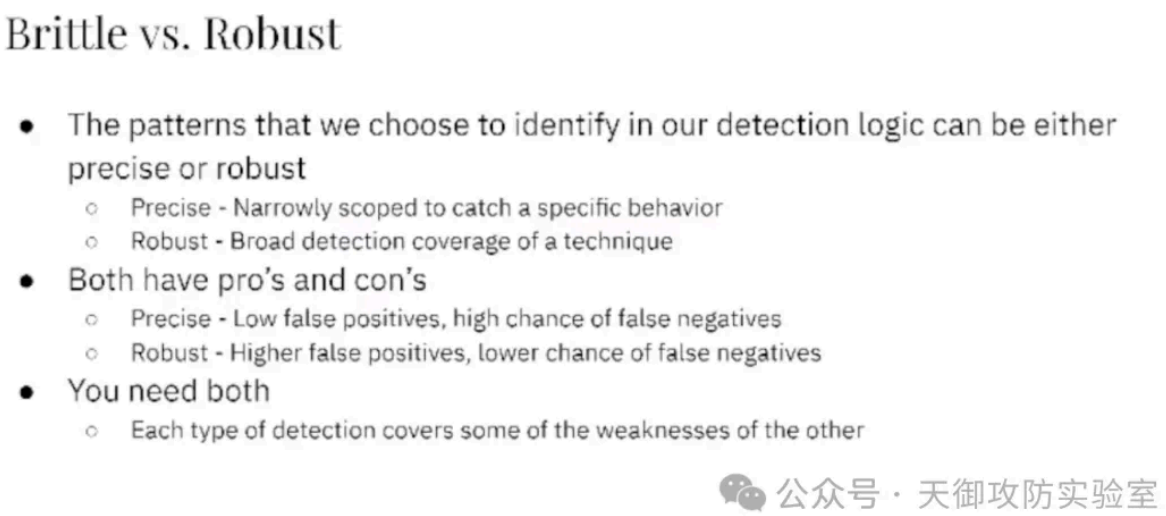
那么问题来了，在茫茫数据海洋中，我们要如何找出能指示恶意行为的数据模式呢？比如某个注册表键被修改，某个进程启动，这些单独的事件毫无意义。我们需要在噪音中提取有价值的信号。一旦发现可疑的行为模式，就相当于制定了一条检测规则。触发规则意味着可能发生了威胁事件，值得分析人员关注。



制定检测规则一般有两种思路：

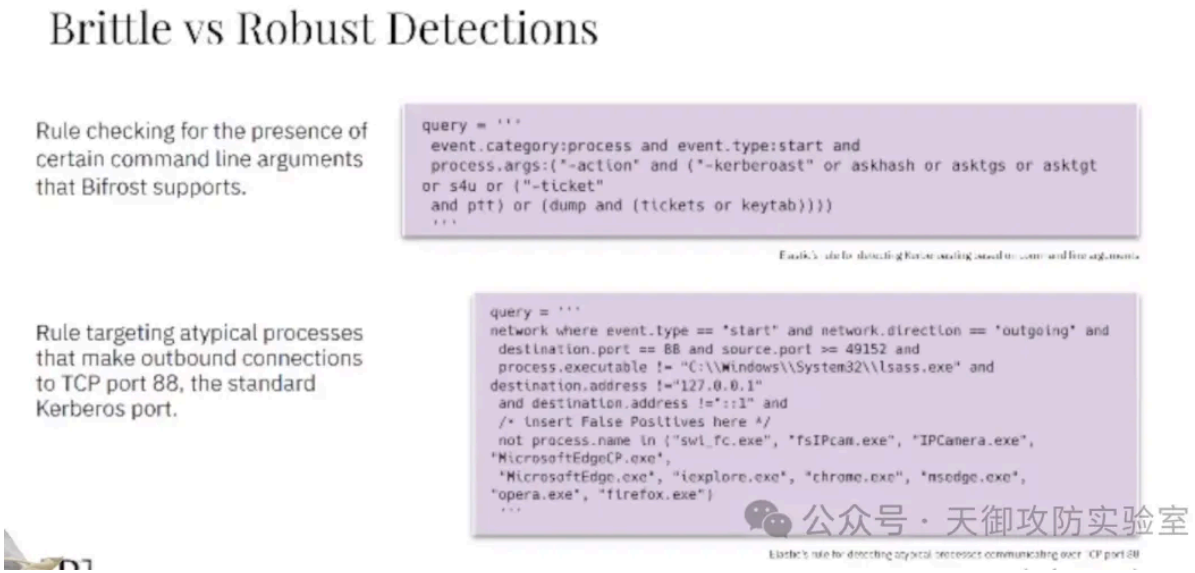
1) 精准检测：高度特异性，只检测某种特定行为，比如进程以Mimikatz常用的参数启动、访问LSASS进程内存等。优点是极少误报，缺点是覆盖面窄，容易漏报。

2) 广泛检测：针对某类威胁的通用检测，比如任何进程访问LSASS进程都预警。优点是覆盖全面，漏报少；缺点是误报多。



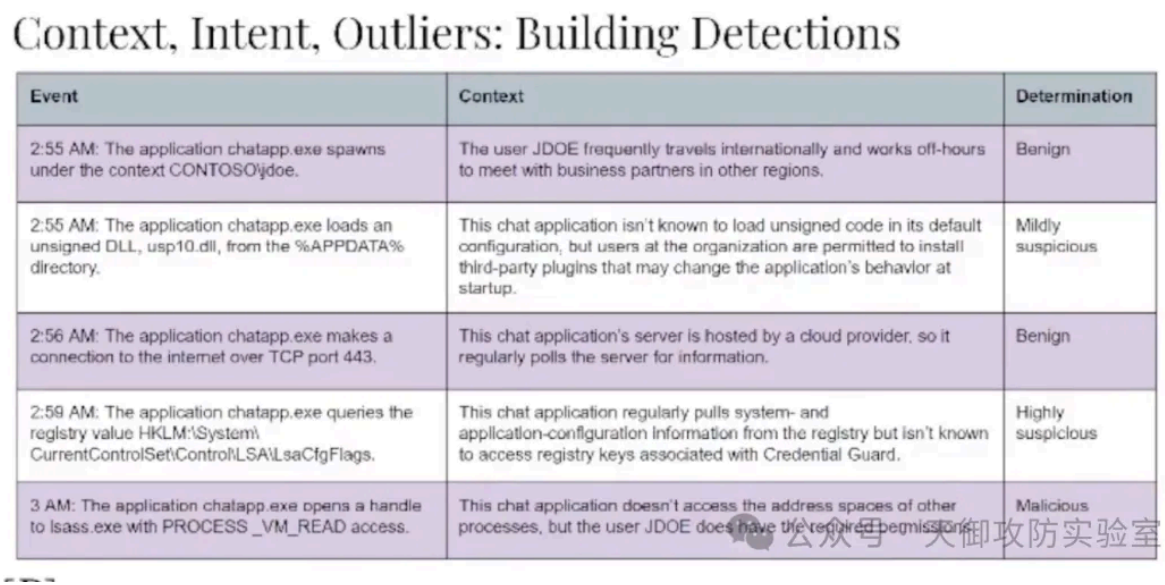
这两种检测方法优缺点分明，企业必须根据自身需求权衡轻重缓急。比如可口可乐公司大概不愿意让配方泄露，所以会倾向于降低漏报，哪怕代价是处理更多的误报。其他行业各有自己的侧重。

打个比方，某个EDR规则要求聊天程序启动时加载了无签名DLL才预警。这个要求太过严格，实际只适用于已知的恶意程序变种。而放宽条件，只要聊天程序加载了任何无签名DLL就预警，又会产生大量误报。所以精准检测和广泛检测必须并重，犹如车之两轮。杀毒软件的签名算法不就是精准检测的典型吗？它在特定场景下的检出率很高。与此同时，我们还需要广泛检测来弥补精准检测的覆盖盲区。两者相辅相成，缺一不可。



举个具体的例子。这是一条检测Kerberoasting攻击的规则。上半部分非常具体，规定进程启动参数必须包含"Kerberos"、"dump"等特征字符串，显然是在检测某些特定的攻击工具。如果攻击者略微修改工具，改成"roast"、"save"之类，规则立刻失效。下半部分则比较宽泛，只要注意进程连接88端口就预警，虽然事后还得排除一些已知的正常程序。如果攻击者把程序伪装成浏览器，就可以从容绕过检测。由此可见，精准规则误报少但易被规避，广泛规则全面但易误报。两者必须结合，互为补充，才能发挥最大效力。

此外，EDR还能按时序串联起若干事件，呈现出入侵的完整过程，辅助调查人员开展分析。但我认为，EDR的关键在于准确检出每一个关键事件，而不是上来就要还原全貌。聊天程序加载无签名DLL，这一个事件本身就值得警惕了。把各种可疑线索拼接起来，还原事件经过，这是调查人员的工作，EDR只需要忠实地记录下各种蛛丝马迹。



总而言之，一个优秀的EDR应该能够完整捕获本文提到的各类关键数据。聊天程序启动（进程创建事件），加载无签名DLL（映像加载事件），连接TCP 43端口（网络连接事件），修改注册表（注册表访问事件），以进程VM读取权限打开进程句柄（句柄操作事件，进程访问事件）。EDR的意义就在于为调查人员提供这些线索，帮他们从噪音中理清思路，最终拼出完整的攻击过程。

问答环节：

问：您个人最喜欢哪款EDR?

答：很抱歉，出于商业合作的考虑，我不便透露具体产品名称。就功能而言，市面上的EDR大同小异。关键还是要看用户如何充分利用手中的EDR。如果把EDR当摆设，肯定发挥不了作用。那些积极研究EDR、不断开发新用法的用户，他们手里的EDR才是真正的利器。

问：现在有没有EDR能关联多个事件形成完整的攻击链?

答：市面上大多数EDR都提供了关联分析的功能，比如CrowdStrike就有个"事件"页面。EDR会根据若干相关事件拼出一个完整的攻击过程。不同厂商的实现方式不尽相同，但基本原理都差不多。

问：理想的误报和漏报比例是多少？

答：这要视企业的实际情况而定。每个企业对数据的敏感程度不同，能够投入的人力物力也不一样。**我们必须跳出"误报率越低越好"的思维定式。单纯压低误报率并非上策。我们要问，这些误报来自哪里？收集这些信息真的有意义吗？能给我们带来哪些启示？然后再去优化数据源，降低无效数据的比例。**每个企业都要自己去寻找这个平衡点。

问：如何有效过滤海量告警，提取真正有价值的预警线索？

答：这是一个复杂的话题，足以单独作为一个议题来探讨。我们可以考虑把一部分分析工作外包给托管服务商（MDR），让他们负责初步鉴定和分流。也可以优化企业内部的分析流程，比如梳理从告警到响应行动之间的每个环节，找出效率低下的步骤并改进。总的原则就是要缩短从告警到判断的链条长度。因为攻击往往在深夜或节假日发生，我们要争取在短时间内做出反应，否则攻击者早就得手了。

问：修改了Mimikatz等工具的显示字符串，EDR会不会不认识了？

答：这要看检测的具体实现方式。如果EDR单纯依赖字符串匹配，那改掉字符串确实可以规避。但大多数EDR同时使用行为检测，这样就没那么容易了。比如我们不看进程的文件名，只关注它调用了什么API、访问了哪些数据，一样可以发现它的真面目。静态特征再怎么变，行为模式骗不了人。

问：如果攻击者把一连串的动作拆开执行，每两步之间拉开很长时间间隔，这样EDR还能关联起来吗？

答：这是个有趣的问题。**EDR在内存里保存事件的时间窗口通常就几分钟。如果攻击的每个步骤间隔太长，中间还夹杂大量无关事件，要把它们完整关联起来确实很难。这是EDR固有的局限性。**要克服这个问题，我们要提高每个关键事件的权重。比如发现了可疑的注册表修改，哪怕后续迟迟没有异常，这件事本身也值得警惕。我们要把重点放在可疑的单个行为上，而不是完整的行为链。

问：面对无文件攻击，EDR还有用武之地吗？

答：EDR可以说就是为无文件攻击而生的。传统杀毒软件检测不了内存中的恶意代码，而EDR可以通过这些代码的行为特征来识别它们。比如一段PowerShell脚本在内存中下载了CobaltStrike的Payload，我们可以不看脚本本身，只盯防Payload的种种可疑行为。对EDR而言，攻击是否落地成文件并不重要，重要的是攻击在运行时会露出什么马脚。无文件攻击只会让EDR更加大显身手。

最后一点讨论，如何判断对EDR的规避是否成功？

1. 作者做了一个调查，大多数人认为成功规避EDR意味着要么不触发任何告警，要么不留下任何遥测数据。但作者指出，红队人员很少有机会接触到客户的EDR系统，因此无法确认这两点。

2. 作者进一步问那些选择"无告警"或"无遥测"的渗透测试人员：你们如何验证是否触发了告警？是否使用了类似Process Monitor这样的工具来模拟真实的遥测数据？

3. 作者认为判断规避的标准取决于如何定义规避本身。是说EDR根本没有采集到入侵痕迹，还是说EDR虽然采集到了痕迹，但没有正确检测出入侵行为？

4. 作者赞同评论中的一个观点：声称规避了某款EDR产品，却没有它的遥测数据和告警日志作为依据，这种判断是不严谨的。攻击者可能会说"反正我已经完成目标撤离了，被发现也无所谓"，但遥测数据本身就很有价值，有助于防御者事后做痕迹分析。

总的来说，作者强调要具体问题具体分析，规避成功与否要看具体的评判标准。在没有日志数据的情况下，仅凭攻击行动没被阻断就武断地宣称规避了防御，这是值得商榷的。遥测数据本身也很重要，在事后分析中大有用处。

参考资料：

视频地址

https://twitter.com/preludeorg/status/1763277444013068553