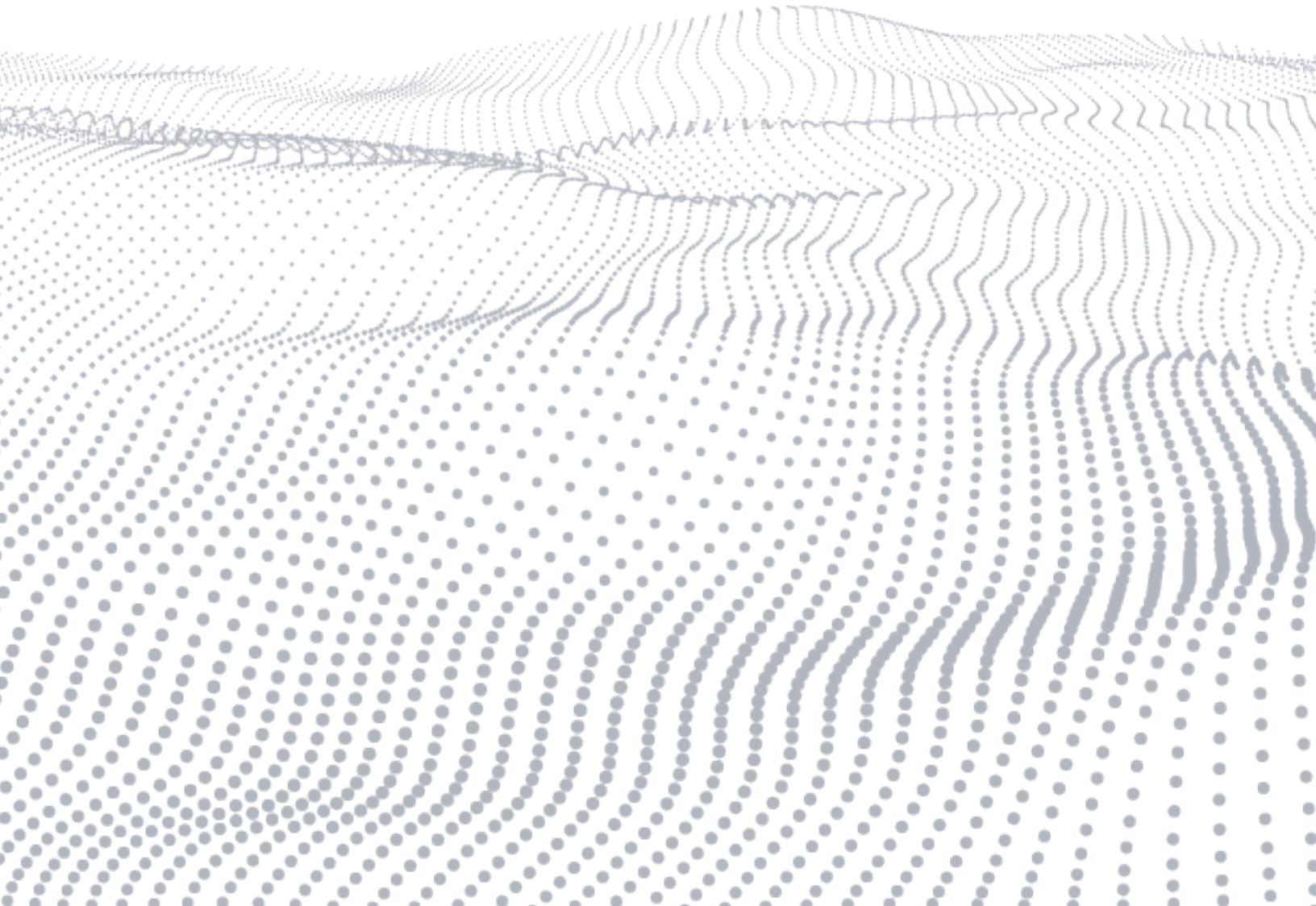
BLACKDUCK® SVS 奇安云

什么是模糊测试:
The Poet, the Courier, and the Oracle

测试用例引擎、测试驱动器、预言机



概括

模糊测试是一种公认的用于定位软件漏洞的优秀技术。基本前提是向目标软件提供故意格式错误的输入并检测故障。完整的模糊器由三个组件组成。诗人创建了格式错误的输入或测试用例。快递员将测试用例交付给目标软件。最后，预言机检测目标中是否发生故障。无论对于构建软件的组织还是使用软件的组织来说，模糊测试都是软件漏洞管理的重要工具。

1. 软件环境中的模糊测试

模糊测试或模糊测试是一种软件测试，其中故意将格式错误或意外的输入传递到目标软件以查看是否发生故障。

在本文中，我们使用的软件是指从源代码编译为在某种处理器上运行的可执行代码的任何东西，包括操作系统、桌面应用程序、服务器应用程序、移动应用程序、嵌入式系统固件、片上系统以及更多的。

当一个软件由于意外或格式错误的输入而意外失败时，这就是一个鲁棒性问题。

此外，各种各样的不法分子积极寻求通过提供意外或格式错误的输入来使软件失败。当软件因故意攻击而失败时，这就是一个安全问题。

导致人类伤害或死亡的软件故障是一个安全问题。

鲁棒性、安全性和安全性是同一个大妖怪（软件错误）的三个面孔。 bug 是开发人员所犯的错误；在适当的条件下，错误会被触发，并且软件会执行不应该执行的操作。提高稳健性、安全性和安全性就是发现和修复错误的问题。

1.1 积极和负面测试

从历史上看，软件测试一直关注功能。该软件是否按照预期的方式工作？在功能测试（一种积极测试）中，测试开发人员创建代码和框架，向目标软件提供有效输入并检查正确的输出。例如，如果我们按下红色大按钮（提供输入），软件是否会打开城市的电网（正确输出）？

在传统的软件开发方法中，软件设计是目标软件的需求列表。测试开发团队的任务相当简单，即将设计需求转化为测试用例，以验证软件是否按照规范中的描述执行。

功能测试当然很重要——目标软件在提供有效输入时必须按预期运行。然而，仅经过积极测试的软件在发布到混乱和敌对的世界时很容易失败。

现实世界是一团糟。它充满了意想不到的条件和格式错误的输入。软件必须能够处理其他软件和人员，这些人员会提供格式不良的输入，以意外的顺序执行操作，并且通常会滥用软件。负面测试是向软件发送不正确或意外的输入并检查故障的过程。

请注意，不同的负面测试工具对于同一测试目标会产生不同的结果。每个工具的工作方式不同，并且会在目标软件上测试不同类型的格式错误的输入。

1.2. 软件漏洞

错误也称为代码漏洞。在软件世界中，漏洞分为三种类型：

1. 设计漏洞是软件本身设计的问题。例如，一个不需要用户进行身份验证的银行网站就存在严重的设计漏洞。一般来说，设计漏洞必须由人类来寻找和消除——在这个级别上根本不存在自动化工具。
2. 当某个软件的设置暴露了漏洞时，就会出现配置漏洞。例如，使用默认（工厂安装的）管理凭据部署数据库就是一个配置漏洞。虽然有一些自动化工具可以帮助定位配置漏洞，但大部分查找和销毁工作必须由人类执行。
3. 代码漏洞是错误。使用手动编码的测试用例进行积极测试可用于查找和修复与功能相关的错误。负面测试可以高度自动化，可用于提高软件的稳健性和安全性。

此外，软件漏洞是未知的、零日的或已知的。

1. 未知的漏洞处于休眠状态。它还没有被任何人发现。
2. 一个人、一个团队或组织已经发现了零日漏洞。零日漏洞尚未发布。受影响软件的构建者和用户很可能不知道该漏洞。没有可用的修复或对策。
3. 已发布已知漏洞。负责任的供应商为其软件发布新版本或补丁以解决已知漏洞。虽然模糊测试通常用于定位未知代码漏洞，但它也可能触发因设计或配置不当而导致的漏洞。

1.3. 黑盒、白盒和灰盒测试

在黑盒测试中，测试工具不了解目标的内部结构。该工具仅通过外部接口与目标进行交互。

相比之下，白盒工具利用目标的源代码来搜索漏洞。白盒测试包括静态技术，例如源代码扫描以及动态测试，其中源代码已被检测和重建，以实现更好的目标监控。

灰盒工具结合了黑盒和白盒技术。这些工具通过其外部接口与目标进行交互，但也利用源代码来获得额外的洞察力。

模糊测试可以是黑盒测试或灰盒测试。这种灵活性使得模糊测试成为测试软件的极其有用的工具，无论源代码或详细的内部信息是否可用。作为一种黑盒技术，模糊测试对于任何想要了解他们正在操作或计划部署的系统的现实稳健性和可靠性的人来说都是有用的。这也是为什么模糊测试成为黑帽特工和黑客用来查找软件漏洞的首要技术的原因。即使没有源代码，更密切地监视目标软件的重要部分的能力也可以提高测试的质量。日志文件、进程信息和资源使用情况提供了有价值的信息，可在模糊测试期间使用这些信息来了解异常输入如何影响目标系统。

如果源代码可用，则可以使用调试器工具和其他工具使模糊测试成为灰盒技术。这提高了测试的准确性，并可以更快地解决所定位的缺陷。

1.4. 静态和动态测试

使用静态和动态测试技术来追踪漏洞。成熟的开发过程应该利用各种静态和动态技术，以将风险降低到所需的水平。本节重点介绍常见的静态和动态软件测试技术，其中之一是模糊测试。

可以在不实际运行目标软件的情况下使用静态技术。它们包括以下内容：

1. 源代码审查由开发人员进行。他们通读源代码并寻找未知的漏洞。虽然这可能有效，但速度非常慢，并且成功很大程度上取决于审稿人的技能。
2. 自动源代码分析工具扫描目标源代码并报告可能存在漏洞的编程模式。人类开发人员必须检查此扫描的结果。
3. 静态二进制分析工具扫描编译的（可执行）代码并报告所包含的库和相关的已知漏洞。

成熟的开发过程应该利用各种静态和动态技术，以将风险降低到所需的水平。

动态测试技术在目标软件运行时执行，包括以下内容：

1. 负载测试向目标软件提供大量输入，以查看它是否因容量过大而失败。
2. 互操作性测试验证两个实现是否能够使用指定的协议、语言或符号进行交互。
3. 一致性测试验证被测系统及其行为是否符合相关规范。
4. 模糊测试向软件提供异常输入以查看是否发生故障。这是定位未知漏洞的绝佳方法。

1.5. 什么是模糊测试？

模糊测试是向软件发送故意格式错误的输入以查看其是否失败的过程。每个格式错误的输入都是一个测试用例。失败表示发现了错误，然后可以修复该错误以提高目标软件的稳健性和安全性。模糊器是一种测试目标软件的软件。合适的模糊器由三个组件组成：

1. 诗人负责创建测试用例。诗人也称为测试用例生成器、测试用例引擎或异常器。
2. 快递员将测试用例发送到目标。快递员也称为注射器、递送机构或测试驱动器。
3. 预言机确定目标是否失败。并非所有模糊器都实现所有三个部分，但为了利用自动模糊测试的强大功能，所有部分都应该存在。

当使用模糊器来提高鲁棒性和安全性时，最终目标不仅仅是发现错误，而是修复错误。有用的模糊器必须保存记录，生成可操作的报告，并提供顺利的修复过程来重现故障，以便可以修复它们

> 最终目标不仅仅是发现错误，而是修复错误。

1.6. 缩小范围：漏洞管理

模糊器并不能解决您的所有问题。它必须是工具库的一部分，也是软件漏洞管理流程的一部分。您还可以用于软件漏洞管理的其他工具如下：

- 手动安全审查
- 逆向工程
- 静态二进制代码分析
- 已知漏洞扫描
- 补丁管理工具
- 模糊测试

1.7. 全程放大：风险管理

软件漏洞管理是更大的图景（风险管理）的一部分。寻求降低或至少了解其总体风险的组织将结合使用软件漏洞管理和其他风险管理技术。模糊测试是一种用于评估软件稳健性和安全性的强大技术，这与风险直接相关。

现在您已经了解了谁使用模糊测试、模糊测试与其他软件测试技术的关系以及模糊测试在漏洞管理领域的应用，我们将继续讨论模糊测试中使用的技术和算法。

2. 测试用例引擎

模糊测试是一个无限空间问题。对于任何软件，无效输入的集合都是无限的。一个有效的测试用例引擎必须足够聪明，能够制作最有可能触发目标软件错误的测试用例。从本质上讲，这归结为有一位测试用例引擎创建了接近目标期望的测试用例，但在某些方面存在缺陷。

生成测试用例的方法对测试用例材料的质量有着深远的影响。

2.1. 随机的

最简单但效率最低的模糊测试方法是随机模糊测试。测试用例引擎只是使用随机数据作为测试用例。

随机模糊测试通常是无效的，因为测试用例与有效输入完全不同。目标检查并快速拒绝测试用例。在大多数情况下，测试用例无法渗透到目标代码中。

理论上，随机的测试用例最终会产生一个类似于有效输入的测试用例。然而，即使使用相对较短的有效输入，产生基本正确的测试用例的概率也非常低，这使得等待这样的测试用例所需的时间非常长。

2.2. 模板

模板测试用例引擎将异常引入有效输入以创建测试用例。一般来说，模板比随机测试用例更有效，因为它们大多是正确的。目标软件将处理测试用例，异常情况会锻炼目标以安全、稳健和可靠的方式处理意外或畸形输入的能力。

模板测试引擎的入门通常很快，因为合适的模板文件或网络捕获通常很容易获得。此外，模板可以轻松地针对在输入中使用意外或非标准元素的目标软件进行调整。

然而，模板测试用例引擎有很大的局限性。首先，对于包含某种完整性验证（例如消息校验和）的协议，测试有效性将受到限制。模板测试用例引擎会引入异常，但不知道更新和校验。模板测试用例引擎可以为目标创建数百万个测试用例，但如果完整性检查未通过，目标软件将不会尝试解析其余消息，并且相应的解析代码仍然未经测试。其次，对于包含有状态功能的协议，模板测试用例引擎也有同样的问题。模板不理解协议的语义，会盲目地重放其异常的有效消息。它无法正确设置会话标识符等有状态功能，因此其测试此类协议的有效性受到限制。

最后，对于部分或完全加密的协议，不能直接使用模板测试用例引擎。如果可以获得未加密的有效消息，模板作者可以创建测试用例引擎随后可以通过外部加密机制将其传送到目标。

模板通常受到良好模板可用性的限制。所用模板的质量决定了结果的质量。

现代模板测试用例引擎可以克服许多上述限制。例如，Black Duck Traffic Capture Fuzzer (Defensics TCF) 是一个模板模糊器，可基于数据包捕获文件 (pcap) 创建测试用例。 TCF 咨询流量分析器以查看是否可以剖析数据包捕获文件。如果可以的话，TCF 使用协议解析器提供的信息，特别是消息结构和字段边界信息，来创建一组更有针对性的测试用例，从而实现更有效的测试。此外，TCF能够正确计算许多协议的长度和校验和字段，大大提高了其有效性。

Black Duck 通用模糊器（Defensics UF 或 DUF）是增强型模板模糊器的另一个示例。 DUF 使用有效文件的集合（语料库）作为测试用例的基础。 DUF 分析有效文件以推断其结构并为目标创建高质量的测试用例。

语料库提取是一种克服基于模板的模糊测试与模板质量相关的一些局限性的方法。语料库提取查找并选择将用于创建测试用例的模板。基本技术是选择最能代表整体协议或文件格式的有效案例。例如，在 对于具有多种消息类型的协议，选择的模板应包括所有消息类型。对于具有可选部分的文件格式，好的模板应包含所有可选功能。

所用模板的质量决定了结果的质量。

2.3. 分代模糊器

一代测试用例了解其测试的协议、文件格式或 API。这意味着它知道每种可能的结构和消息类型、每条消息中的所有字段以及有关如何交换消息的规则。因为一代测试用例知道所有的规则，所以他可以系统地打破所有的规则。

此外，由于分代模糊器完全了解协议或输入类型，因此它可以正确处理使模板测试用例感到困惑的事情。分代模糊器可以跟踪会话标识符等状态特征，并且可以正确设置校验和等值，这可能是其他模糊测试技术的限制因素。

一代测试用例创造了对目标来说看起来合法的高质量测试材料。对于具有多消息对话的协议，这允许分代模糊器与目标交换多个有效消息，在交付异常测试用例之前将其驱动到特定状态。

截止2024年10月，Black Duck 模糊测试平台拥有超过27大类共 268代/个模糊器，适用于各种网络协议和文件格式。

2.4. 进化论

进化测试用例使用有关目标行为的反馈来影响后续测试用例的创建方式。与目标对测试用例的响应相关的一些测量用于对已发送到目标的测试用例进行评分。测试用例根据之前得分最高的测试用例创建了更多测试用例。

“纯粹的”进化模糊器是黑盒模糊器。进化模糊器提供了目标地址和端口，仅根据目标的响应向目标发送越来越相关的输入。黑鸭模糊器使用进化方法来塑造其测试材料。互操作性扫描会发现目标中实现了哪些功能，然后配置测试套件，以便仅测试已实现的功能。

3. 测试驱动器

测试驱动器负责将测试用例创建的测试用例传递到目标软件。模糊测试涵盖多种学科，每个学科都有自己的挑战。

3.1. 网络协议模糊测试

模糊测试的一种常见应用是网络协议测试。协议是关于不同软件如何通过网络进行通信的一组规则。解释协议消息的代码是攻击向量。模糊测试是一种用于定位协议处理代码中未知漏洞的出色技术。

测试网络软件进一步分为模糊测试不同角色和类型的组件。许多协议都包含客户端和服务器的概念，其中客户端发起连接，服务器响应。

在服务器测试中，测试驱动器的工作很简单。目标软件侦听传入连接。测试驱动器所要做的就是连接到目标服务器并发送测试用例。

客户端测试通常更加复杂。预言机必须像服务器一样，监听传入的连接。每次目标客户端与测试驱动器建立连接时，测试驱动器都会用测试用例进行响应。通常，需要鼓励客户反复与测试驱动器建立联系，以便测试驱动器能够继续交付测试用例。

端到端或直通测试是另一种类型。某些网络组件无法直接寻址，但它们会在网络流量通过时对其进行检查。一个很好的例子是具有网络地址转换 (NAT) 的防火墙和应用层网关 (ALG) 支持 IP 语音 (VoIP) 交互。为了测试这样的组件，将模糊器放置在一侧，并将一些终止软件放置在远端。模糊器信使将测试用例通过目标传送到终结器。终结者需要对信使做出适当的响应，并且需要足够强大以处理测试用例材料而不会失败。

3.2. 文件模糊测试

在文件模糊测试中，故意将格式错误的文件传递给软件。测试驱动器的工作很难定义，因为不同的软件并不以任何标准化的方式使用文件。有时，最好的方法是编写自定义包装器或代码，以帮助有效地将测试用例提供给目标。

Black Duck 文件模糊器提供各种开放式交付选项。测试用例可以写入文件系统以供稍后交付，可以通过网络连接发送，可以交付给特定命令，也可以由内置 HTTP 服务器提供服务。

3.3. API 模糊测试

API 模糊测试是一种不同的方法，其中测试应用程序编程接口 (API) 的各种方法或功能，以了解它们如何响应格式错误的输入。在这种情况下，快递员必须从测试用例创建源代码，编译（如果适用）并运行代码，然后查看是否发生故障。

请注意，远程过程调用 API（例如 DCERPC、SunRPC 和 Java RMI）属于网络协议模糊测试。类似地，XML/SOAP 和更现代的基RESTful/JSON 的远程 API 应被视为协议模糊测试。

3.4. 用户界面模糊测试

许多软件都提供用户界面 (UI)，以便人们可以查看信息并提供输入。可以使用键盘、小键盘、鼠标或某种其他方法来提供输入。 UI 模糊测试是提供意外或畸形的结果的过程

输入到 UI。例如，在传统桌面上的应用程序中，模糊器可能几乎同时在屏幕的两侧进行鼠标点击。

大多数现代操作系统和编程环境都提供了一种编程方式来传递输入事件，以便此类测试可以完全在软件中完成。然而，对于一些较小的设备，模糊用户输入的唯一方法是使用能够按下按钮的机械设备。

4. 预言机

预言机确定测试用例是否通过。它检查目标以查看是否发生故障。了解何时发生故障对于模糊测试的成功至关重要。如果你不能让你的目标软件出现可重现的故障以便能够修复它，那么导致它出现故障并没有任何好处。

使用多种方法检查目标故障可以增加在发生故障时检测到故障的可能性

本节描述了应对预言机挑战的几种方法。好消息是可以组合多种方法。使用多种方法检查目标故障可以增加在发生故障时检测到故障的可能性。

4.1. 故障类型

当软件的行为方式超出了其创建者的意图或预期时，它就会失败。在传统应用的模糊测试中，故障模式分为四类：

- 崩溃
- 无限循环
- 资源泄漏或短缺
- 意外行为

这些故障模式根据所测试的系统或软件的类型、底层操作系统等而有所不同。崩溃可能只是崩溃，也可能导致目标拒绝服务、性能下降、信息泄露、安全受损或其他情况。后果取决于软件的目的和功能、操作地点和时间等。

预言机的工作是检测故障。故障模式如此广泛，这不是一件容易的事！

4.2. 传统模糊测试

模糊测试领域持续快速发展，当前的创新领域之一是预言机。本节介绍在模糊测试期间监视目标的“传统”方法。

4.2.1. 观察

即使在高度自动化测试的时代，人类观察仍然具有很高的价值，不应被低估。了解目标软件的人可以观察其功能并监控日志文件和资源使用等生命体征，以一种很难自动化的方式注意到事物并得出结论。当然，人类观察成本高昂，并且无法很好地扩展到数十个目标或数周的测试。

对于测试设置和初始测试，人类观察是一种出色的预言技术。

4.2.2. 有效案例或功能

有效的案例预言机通过发送有效输入并寻找有效且及时的功能响应来检查目标的运行状况。这可以在快递员交付每个测试用例后完成。如果测试用例导致失败，导致目标无法响应有效输入，预言机将立即发现。

有效的案例预言机具有以下优点：

- 它简单且易于自动化。
- 这是一种黑匣子技术；它可以在不了解目标源代码或内部工作原理的情况下使用。
- 它是专注的。例如，当在具有多个开放端口和协议的 Web 服务器上模糊 HTTP 时，使用具有有效 HTTP 消息的有效 case oracle 是确保 HTTP 服务器进程仍在运行的好方法。有效的案例预言机会让您知道您的目标是否发生崩溃或冻结。然而，有效的案例预言机无法“看到”更微妙的问题迹象，例如内存泄漏、异常资源消耗、断言失败等等。

Black Duck 模糊测试工具在其测试套件中包含对有效案例预言的广泛支持。

4.2.3. 资源监控

另一种可以针对多种类型的目标实现自动化和通用的有用技术是资源监控。当向目标提供模糊测试用例时，预言机会检查内存、磁盘空间、处理能力和其他资源的使用情况，以查找任何异常情况。

对于支持简单网络管理协议（SNMP）的目标，预言机可以在模糊测试期间检索与资源使用情况和其他生命体征相关的 SNMP 值。这样的预言机允许直接使用任何支持 SNMP 的目标。

Black Duck 模糊器支持浏览和选择 SNMP 值，以及使用 SNMP 预言机的内置自动支持。该图（参见图1）显示了模糊测试期间的资源消耗图。

图1：资源监控

4.3. 预言机

更好地监控目标可以更准确地检测漏洞并更快地解决问题。本节列出了一些应对模糊测试中的预言挑战的强大方法。

4.3.1. 外部的

由于软件目标有各种形状和大小，因此不可能创建一个适用于所有情况的通用预言机。有效案例预言机运行良好，因为它使用与正在测试的相同的接口。

通常必须根据具体情况创建针对特定目标的预言机。发生这种情况时，重要的是您的模糊器能够轻松集成您创建的预言机。

Black Duck 模糊测试工具提供了一种称为外部检测的机制，其中它将调用用户提供的脚本来确定目标上是否发生故障。在每个测试用例之后都会调用此机制，以便当发生故障时，可以识别导致故障的测试用例并再次运行。外部检测脚本可以检查日志文件、监视资源使用情况、检查进程或执行任何其他可以编写脚本的操作。

4.3.2. 动态二进制

模糊测试的优点之一是它是一种黑盒技术——即使没有源代码，您也可以对软件进行模糊测试。

即使没有源代码，也可以使用一些高级方法来检查正在运行的目标软件。

- 例如，strace 等工具可以显示目标软件如何与底层操作系统交互。
- 动态二进制检测（DBI）是一种很有前途的技术，可以动态修改可执行代码，以便在不需要源代码的情况下详细检查其工作原理。
- Microsoft 的 PageHeap 是一个可以对任何可执行代码进行堆分配监视的工具。

4.3.3. 源代码检测

如果您确实有可用的源代码，则可以使用其他先进的神秘技术来检测故障。根据您的模糊器和所使用的技术，您也许能够将这些预言机集成到模糊测试过程中以精确检测故障。

图 2：功能和行为检查

表明也可能不表明某种失败。此外，在调试器中运行软件时，由于 try-catch 结构使用不当而隐藏的错误可能会被暴露出来。

- 同样，使用调试符号编译的目标软件可以使用 valgrind 的 memcheck 工具运行，以分析内存使用情况并检查错误。
- 使用 AddressSanitizer (ASan) 构建的目标软件将快速暴露内存使用错误。
- 其他类型的目标仪器也可能可用。

4.3.4. 功能和行为检查

可以在模糊测试期间检查某些攻击向量以检测功能故障。例如，在模糊 TLS 消息时，如果模糊器提供错误的身份验证凭据，但目标无论如何都允许模糊器访问，则这是明显的功能故障。

根据攻击向量，可以实施各种功能检查，包括身份验证绕过、代码注入、消息放大等。

为了能够进行功能检查，模糊器必须已经具有被测试协议或文件格式的完整模型和语法，这意味着模糊器必须是分代的。

现在属于 Black Duck 的 Defensics 团队在向其 TLS 模糊器添加此类功能检查（称为 SafeGuard）时发现了 Heartbleed。以下屏幕截图（参见图 2）显示了 Defensics 测试日志的一部分，其中在目标软件中检测到了 Heartbleed 漏洞。

5. 总结

模糊测试是一种用于定位软件漏洞的出色技术。基本前提是向目标软件提供故意格式错误的输入并检测故障。完整的模糊器由三个组件组成。诗人创建了格式错误的输入或测试用例。快递员将测试用例交付给目标软件。最后，预言机检测目标故障。

不同的模糊测试技术对模糊测试的有效性有显着影响。在大多数情况下，当诗人能够创建几乎正确但在某些方面异常的测试用例时，它会更加有效。不同的预言机技术提供不同级别的故障检测能力。多种预言机技术可以一起使用来帮助检测最大故障数。

模糊测试可以省钱，因为尽早修复错误比稍后修复错误要便宜得多。

模糊测试是软件漏洞管理的重要工具，无论是对于创建软件的组织还是使用软件的组织来说都是如此。模糊测试必须作为流程的一部分进行部署。构建者使用模糊测试作为安全开发生命周期的一个组成部分，而买家则使用模糊测试作为验证和确认的关键工具。从财务上来说，模糊测试可以省钱，因为尽早修复错误比稍后修复错误要便宜得多。在部署或产品发布之前修复的错误没什么大不了的。在生产场景中定位并可能利用的错误可能会付出巨大的代价。

在更广泛的风险管理背景下，通过模糊测试主动发现和修复错误可以防止其他类型的损害。如果您的产品中的错误导致灾难性故障或大规模数据泄露，您的声誉可能无法恢复，并且您的法律责任可能无法克服。如果您提供医疗保健、电力、通信或其他关键基础设施等服务，您所使用的产品中的错误可能会导致人类伤害或环境破坏。

查找并修复错误可以节省金钱，保护您的客户和声誉，并且在许多情况下可以挽救生命。

探索 Black Duck 模糊测试工具如何

可以帮助您构建更安全的软件。

关于BLackDuck

Black Duck®提供业界最全面、最强大、最值得信赖的应用安全解决方案组合。我们在帮助世界各地的组织快速保护其软件、将安全性有效地集成到其开发环境中以及利用新技术进行安全创新方面拥有无与伦比的记录。作为软件安全领域公认的领导者、专家和创新者，Black Duck 拥有您在软件中建立信任所需的一切。