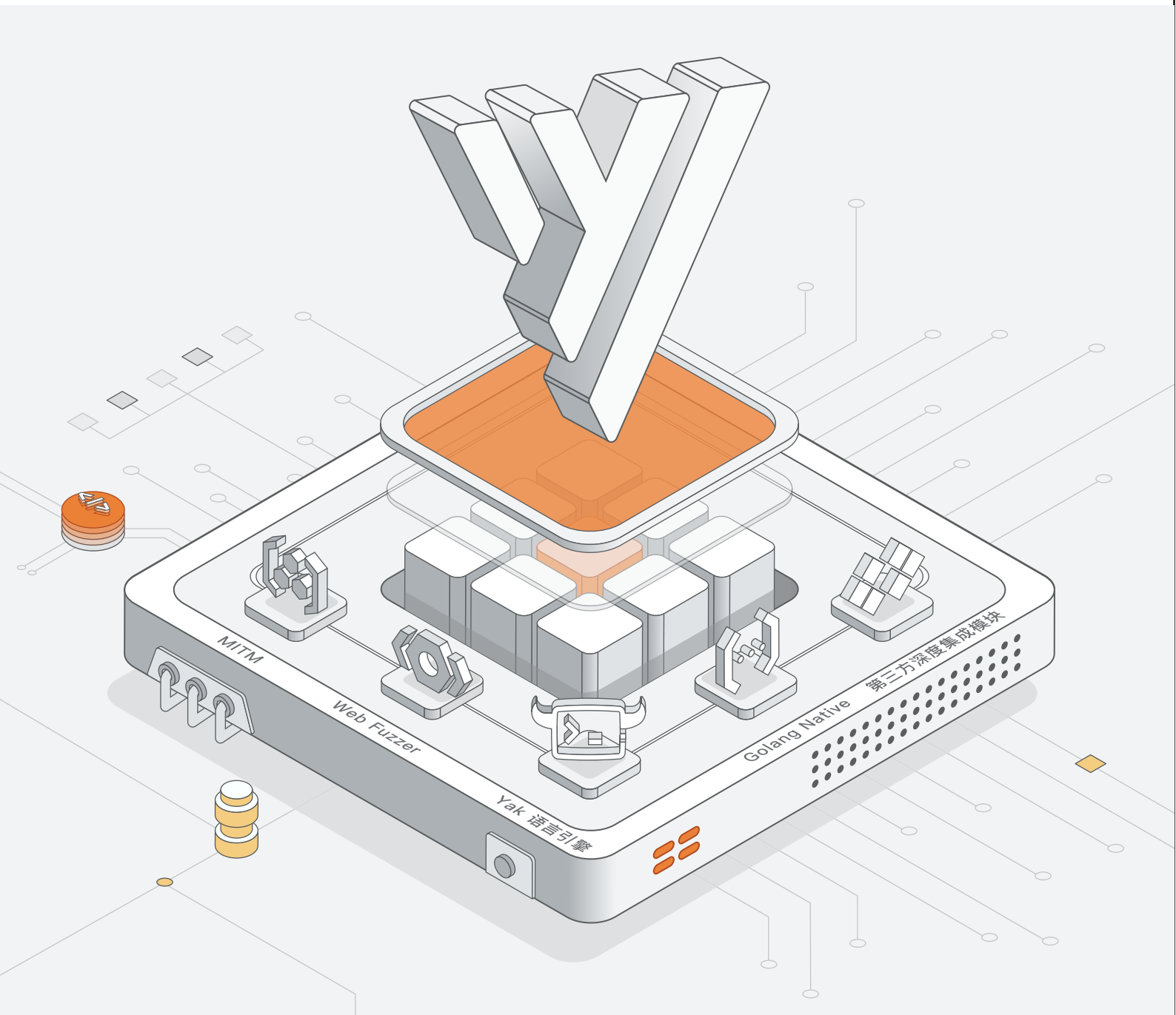
Yakit

企业版:Yakit交互式应用安全测试平台

产品技术白皮书



四维创智（成都）科技发展有限公司

目录

引言

1.1 安全能力“割裂”现状

近年来，随着数字化速度的加快和深入，企业面临的安全挑战日益增加，包括勒索病毒、网络钓鱼、数据泄漏等多种形式的网络威胁给企业造成了直接的经济损失，也严重威胁到用户的隐私和数据安全。针对这一现状，国家高度重视，并在2021年相继出台了《中华人民共和国个人信息保护法》与《中华人民共和国数据安全法》，和2016年的《网络安全法》共同构建了中国信息安全领域的法律框架。

对于企业而言，信息安全能力建设成为了一项不可或缺的任务。从传统的防火墙规则匹配到现代的零信任安全平台，企业需要采购或自研各种安全产品以应对不同的安全威胁。然而，目前市场上的网络安全产品多样且复杂，不同产品之间往往缺乏有效的互操作性，导致安全能力“割裂”和资源分散。这种情况下，企业在进行业务场景测试时常面临困难，因为产品间的不兼容性使得安全方案难以整合，从而削弱了整体的安全防护效果。

1.2 安全融合的必要性和紧迫性

面对安全能力“割裂”的现状，安全融合成为了解决这一问题的关键。安全融合不仅是技术层面的整合，更涉及到安全策略和管理流程的全面优化。通过实现不同安全产品和工具的有效融合，企业可以构建一个更加协同和高效的安全防御体系，提高对外部威胁的响应速度，同时优化内部资源配置和管理流程。

随着企业数字化转型的深入，信息安全不再是单纯的技术问题，而是涉及到企业运营的各个方面。在这个背景下，实现安全能力的融合，不仅可以提高安全防护的效率，还能为企业带来更大的业务价值和竞争优势。因此，从长远的发展角度看，安全融合不仅是企业应对当前网络安全挑战的必要选择，也是推动企业可持续发展的重要策略。

行业概况

2.1 安全融合：新常态下企业安全建设新方向

在过去的很多年中，企业想要端口扫描的能力，需要使用 Masscan + Nmap 全家桶，各种安全产品只能提供一个或两个能力来解决安全问题。企业还需要对许多零散的安全产品进行评估，并考虑它们在当前情况下是否仍然具有实际效用。每种产品都有资本支出和运营支出。

在数字化和网络化日益深入的今天，企业面临的安全挑战变得更加复杂和多元。安全融合成为了新常态下企业安全建设的新方向。这种融合不仅仅是将不同安全产品和服务简单堆砌在一起，而是需要在更深层次上，实现技术、策略和管理的有机结合。安全融合使企业能够更加灵活地应对各种网络威胁，提高安全防护的效率和有效性。

在安全融合的过程中，企业需要考虑到如何将传统的安全措施与新兴的技术趋势相结合，如云计算安全、大数据安全分析、人工智能在安全防护中的应用等。此外，安全融合也意味着需要跨部门协作，打破信息孤岛，实现安全信息的共享和联动。只有通过这样全面的融合，企业才能建立起一个更加坚固、灵活的安全防护体系。

因此，安全融合是新常态下企业的一项安全运营与安全管理的改革，也是新常态下企业安全建设的新方向。

2.2 安全融合的最优解：CDSL-Yaklang

近年来，网络安全编程领域经历了显著的发展。Python 以其简单的语法和丰富的生态工具成为了广泛使用的编程语言。然而，随着网络安全从业人员技术水平的提升，Python 在执行效率和工具规范性方面的不足逐渐显现，导致网络安全产品和工具的工程化和规模化需求日益增长。

安全从业人员使用的工具并不仅限于 Python。例如，使用 Golang 编写的安全工具占有重要比重，此外，像 Nmap、Wireshark、Burp Suite、Metasploit 等基础安全工具也使用了各自不同的编程语言。开发者通常倾向于使用自己熟悉的语言来降低编程成本并加速开发过程。这种做法虽然有效，但也导致了安全产品在维护、融合甚至二次开发方面的割裂。

为了解决这些问题，安全从业人员开始寻求一种更适合安全工具和产品研发的新语言生态。在这个过程中，“安全业务和安全能力研发的分离”理念被广泛接受。基于这一理念，结合“领域限定语言”的思想，我们构建了CDSL（Cybersecurity Domain Specific Language）的概念。CDSL 是专为网络安全领域设计的编程语言，它以 Yaklang 语言为核心，构建了基础设施和语言生态，从而使安全能力研发变得更加容易和高效。

产品概述

3.1 Yaklang 语言的角色和价值

在安全融合的背景下，Yaklang.io 团队开发了一种专门为网络安全领域设计的领域特定编程语言（CDSL）：Yaklang。Yaklang 的目标是解决安全产品融合过程中出现的技术挑战，如不同产品间的互操作性问题，以及安全工具开发中的效率和一致性问题。

Yaklang 作为 Yakit 平台的核心组成部分，扮演着至关重要的角色。这种专为网络安全领域开发的领域特定编程语言为 Yakit 平台提供了强大的后端支持。Yaklang 的设计旨在简化安全产品的集成过程，提高开发效率，并确保各种工具和服务能够无缝协作。Yaklang 通过提供一套统一的编程框架和丰富的库，使得安全产品的开发、集成和维护变得更加高效和简便。它支持企业重新构建或优化现有的安全产品，使它们能够更好地融入统一的安全架构中。

Yaklang 的价值在于它为企业提供了一种新的方式来思考和实施安全策略，推动了安全思维的转变。在 Yaklang 的支持下，企业可以从一个全新的视角审视安全问题，通过更加灵活和创新的方式来构建和优化其安全策略。同时，Yaklang 强大的自定义功能，允许企业根据自身的特定需求，开发专属的安全解决方案。

Yakit上层支持

Db	Yakit 数据库
Report	Yakit 报告系统
Risk	漏洞 验证辅助
Yakit	插件 辅助工具包
Hook	插件 辅助调用

上层能力模块

Spaceengine	网络空间引擎支持 Shodan/ FOFA/ Hunter/ Quake
Subdomain	子域名扫描/ 域传送/爆破/ 互联网收集
Crawler	传统爬虫/ 支持表单识别/ 链接爬取/ 资源限制
Nuclei	Project Discovery 风格的 Yaml PoC
Yak Bridge 机制	DNSLog/ RMI/HTTP/ LDAP/穿透
Wsm	XieCat 风格的 WebShell 管理（实现）

HTTP Fuzz模块

Fuzz	模糊测试支持
PoC	HTTP 底层支持
Fuzztag	模糊标签渲染
Httpool	HTTP 请求池

主机扫描能力

synscan	SYN 端口探测
services-can	服务指纹扫描
ping	ICMP/ TCP-Ping 主机存活扫描
brute	基础协议爆破
MITM	中间人劫持协议

跨语言协议

JS	JavaScript 执行
JAVA	序列化与字节码协议
LDAP	LDAP 协议支持
T3	Weblogic T3 协议
YSO	YSoSerial 支持
Facades	综合服务支持

基础能力

cil	命令行处理
codec	编解码/加解密
context	上下文管理
DNS	DNS 请求
file	文件读写
sync	同步管理
JSON	JSON 支持
judge	格式辅助
SMB	SMB 协议
log	日志输出

str	字符串处理
dyn	动态加载
env	环境变量
exec	命令执行
Gzip	压缩协议
xhtml	Html 词法解析
mmdb	GeoIP
jwt	Token 认证
re	正则处理
TCP/UDP	传输层协议

x	函数式编程反射辅助
HTTP	基础协议
time	时间辅助
HTTP server	服务器
IO	I/O 辅助模块
OS	操作系统
TLS	TLS 协议支持
XPATH	HTML 处置
ZIP	Zip 协议
…	…

语言基础执行环境

YAK Grammar（Golang 风格）　Yaklang 字节码　Golang Native Reflection 技术

Windows　MacOS　Linux

（Yaklang 能力图谱）

3.2 Yakit 与 Yaklang 的协同作用

为了让 Yaklang 本身的安全能力更贴近实际使用场景，我们为 Yaklang 编写了 gRPC 服务器，并基于 gRPC 服务器构建了客户端：Yakit。

实际上就是通过 Yakit 的 GUI 去操控引擎的能力。Yakit 的 gRPC 服务器，让用户部署更加方便快捷，与平台无关，可选择远程部署或直接本地启动在主机中使用。

Yakit 平台和 Yaklang 语言的结合提供了一个强大的网络安全框架。Yakit 作为前端用户界面，为用户提供直观的操作体验和丰富的功能选择。而 Yaklang 则作为后端驱动，保证了这些功能的高效执行和良好的集成性。

这种前后端的协同作用使得 Yakit 能够提供一站式的安全解决方案，覆盖了整个渗透测试流程和安全生命周期。企业用户可以通过 Yakit 快速高效地发现业务安全问题，同时依托于 Yaklang 的强大能力，实现复杂的安全任务和流程自动化。

3.3 Yakit 产品介绍

Yakit，作为高度集成化的 Yak 语言安全能力输出平台，通过 gRPC 服务器与 Yak 编程语言的结合，基于 Electron 为用户提供了一个高度灵活的可视化且功能丰富的安全测试平台。

同时，Yakit 也是一款集成化的渗透测试工具，它提供了一系列安全工具和功能，包括 MITM 劫持操作台、Web Fuzzer、Yak Cloud IDE、ShellReceiver 等，能够更好地完成对企业级应用的安全测试。

Yakit 的用户界面是基于 Electron 和 gRPC 技术构建的，提供了一个直观且易于使用的交互式应用安全测试平台。用户可以通过这个界面轻松地访问 Yaklang 的强大功能，执行各种安全测试和分析任务。此外，Yakit 还集成了诸如 nmap 指纹库、wapplayzer 指纹库、nuclei yaml poc 体系等成熟的外部生态，为用户提供了广泛的资源和工具。

使用 Yakit，我们可以做到：

01 类 Burpsuite 的 MITM 劫持操作台

02 查看所有劫持到的请求的历史记录以及分析请求的参数

03 全球第一个可视化的 Web 模糊测试工具：Web Fuzzer

04 Yak Cloud IDE：内置智能提示的 Yak 语言云 IDE

05 ShellReceiver：开启 TCP 服务器接收反弹交互式 Shell 的反连

06 第三方 Yak 模块商店：社区主导的第三方 Yak 模块插件，你想要的应有尽有

07 ……

产品特点

4.1 一体化的安全测试工具

Yakit 平台的核心优势在于利用我们统一的 Yak 语言和安全架构集成一体。这种架构使得安全能力变得更加集中和高效，减少了因使用多个不同安全产品而产生的复杂性和兼容性问题。统一的架构还意味着更快的响应时间和更高的数据准确性。

Yakit 能够统一管理和执行各种渗透测试任务的工具。工具覆盖信息收集、爬虫、交互式 WEB 流量分析、端口扫描、口令破解等，能够满足各种渗透测试的需求，并且能够在工具之间进行有效的协同。在一个项目中产生的所有业务流量均会被记录到单独的数据库中，该数据库可以自动上传至统一的离线 Server 之上，实现数据的追踪和管理，也可以为后续的质量分析提供数据支撑。

同时，Yakit 中原生集成自研网络安全大模型“万径千机 (ChatCS)”，在测试过程中遇到的所有问题均有专业的 AI 机器人进行高效回复。

4.2 高效的协同作战平台

Yakit 平台通过集成的协同模块和团队共享资源库，实现了团队成员之间的无缝协作，确保在安全测试的过程中，成员能够保持信息实时共享和一致性。

测试团队之间可以通过分享密令随时共享对某一个漏洞的测试细节数据包。方便在需要帮助时，通过共享测试数据快速解决问题。

在隔绝内网中，通过统一协作服务器的配置，多个用户之间可以随时完成远程协助的功能，类似在内网部署一个可即时分享屏幕的向日葵远程协助工具。

所有的工具都能够在隔离的内网环境中正常工作，并且能够进行有效的协同。包括测试工具、通信工具、数据管理工具等。

4.3 推动企业安全建设的“粘合剂”

作为企业安全建设领域的革新者，得益于 Yak 语言天生的安全融合特性，可以将其视为企业安全建设的“粘合剂”，通过丰富的 API 以及插件体系，可以巧妙地将多元化的安全工具和服务等整合在一个统一的平台上。这种集成性不仅打破了传统工具间的壁垒，还能将各个分散的单位紧密结合。

在灵活性方面，可以根据每个企业其独特的安全需求进行相关插件的开发，因此提供了允许企业根据自己特定情况来定制和扩展安全功能的能力。

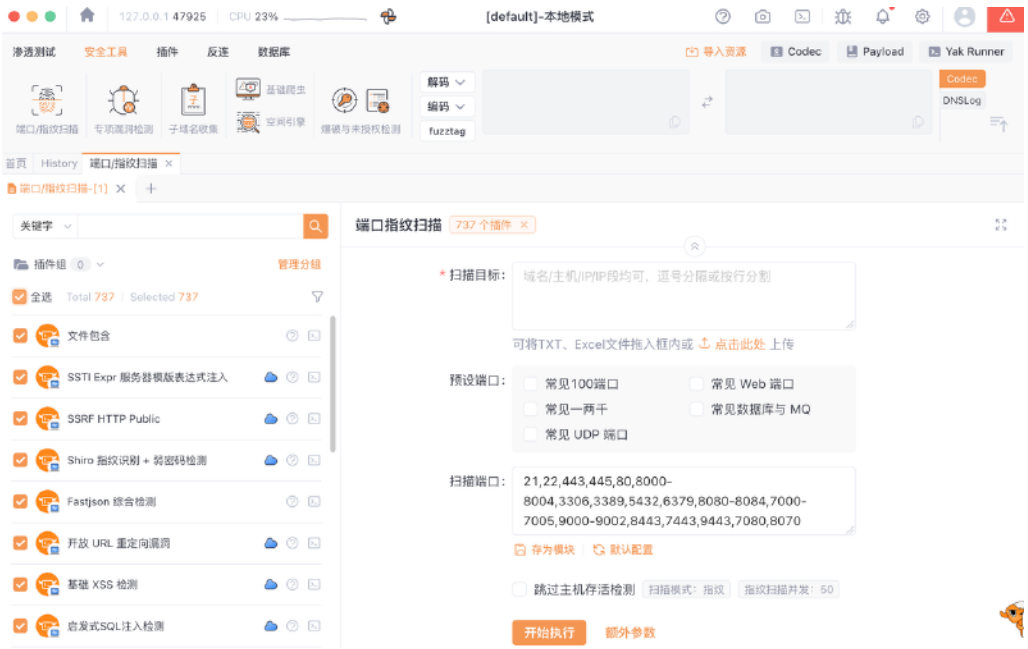
以 Yak 语言为核心，依托衍生产品，通过构建底层安全能力基座，协助企业进行网络安全体系建设，推动企业安全运营与安全管理的改革，提高企业高位安全能力。

功能简介

5.1 产品核心功能

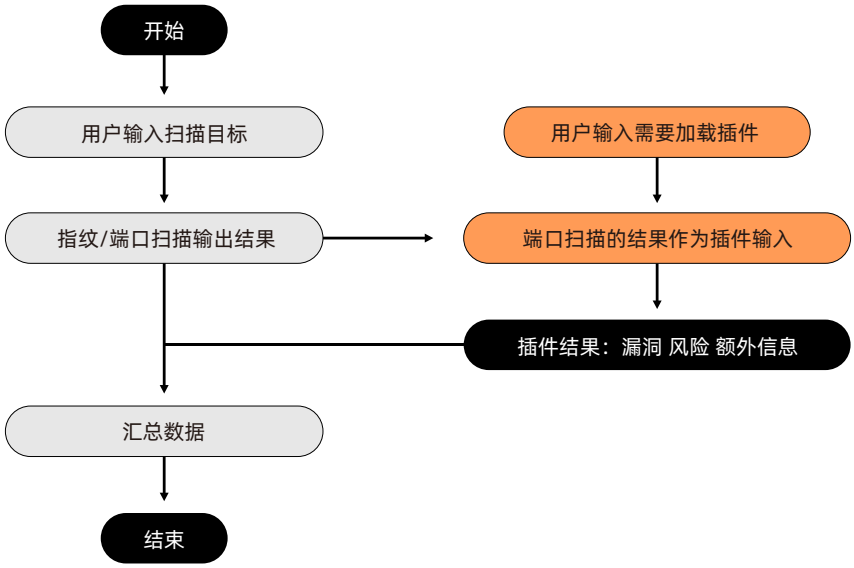
5.1.1 可拓展和编排的基础安全能力

在 Yakit 平台中，我们坚持安全融合的核心理念，为企业提供了一个全面且综合的基础安全产品和工具套件。这个套件覆盖了从资产识别、指纹识别到漏洞扫描、协议爆破、未授权检测等多个关键环节，使企业能够对其网络环境进行全方位的深入安全评估。这样，安全团队能够对潜在的攻击面有一个充分且清晰的了解，帮助企业及时地识别并修补软件的缺陷和配置错误，有效地减少攻击面。



Yakit 平台的一个独特之处在于其基于 Yak 语言的扩展能力。除了提供这些基础的安全工具，Yakit 允许用户基于 Yak 的特性，对这些工具进行定制和扩展。这意味着在满足日常基础安全功能的同时，用户还可以根据自己的具体需要，添加新的功能或改进现有功能。这种扩展能力大大增强了工具的适用性和灵活性，使其能够更好地适应不同企业的独特需求和不断变化的安全威胁环境。

Yakit 的基础安全能力本质上是通过一段脚本或一个 Yakit 插件实现的“动态加载”过程。这个过程非常简便和灵活，用户可以轻松地加载或更新所需的安全功能，无需进行复杂的配置。这种动态加载机制不仅提高了安全工具的灵活性和响应速度，还降低了内部安全开发的的复杂性和成本。



以几个具体的应用场景举例，展示了如何利用Yakit的动态加载和扩展能力来实现高效的安全策略：

• 端口扫描 => 按指纹调用 nuclei => 保存漏洞：

在这个场景中，Yakit 首先执行端口扫描以发现开放的服务。随后，根据扫描结果的指纹信息，动态调用 nuclei（一种安全工具）来进行更深入的漏洞检测。最终，所有发现的漏洞都会被自动保存记录。这个流程简化了从发现服务到识别漏洞的过程，形成了一个简单、暴力且有效的漏洞扫描方案。

• 端口扫描 => Web 爬虫 => 分析页面包含“参数”的点 => 自动保存“参数”相关的 URL 或请求：

这个场景利用了 Yakit 的灵活性来构建一个自动化和定制化的“打点”系统。首先进行端口扫描，接着用 Web 爬虫遍历网站内容。在爬虫的过程中，系统会特别分析那些包含“参数”的页面，自动识别并保存与这些参数相关的 URL 或请求。这为后续的安全测试或攻击模拟提供了一个精确的目标集合，大大提高了安全工作的效率和针对性。

• 端口扫描 => 特定服务开启 => 调用爆破模块，使用简短有效字典来爆破服务：

在这一场景中，Yakit 首先识别开放的端口和运行的服务。如果发现某个特定服务（如 FTP、SSH 等）正在运行，它会自动调用一个密码爆破模块，并配合一个精选的有效字典来尝试破解服务的认证。这个过程完全自动化，使得安全团队能够快速识别和利用那些弱密码配置的服务。

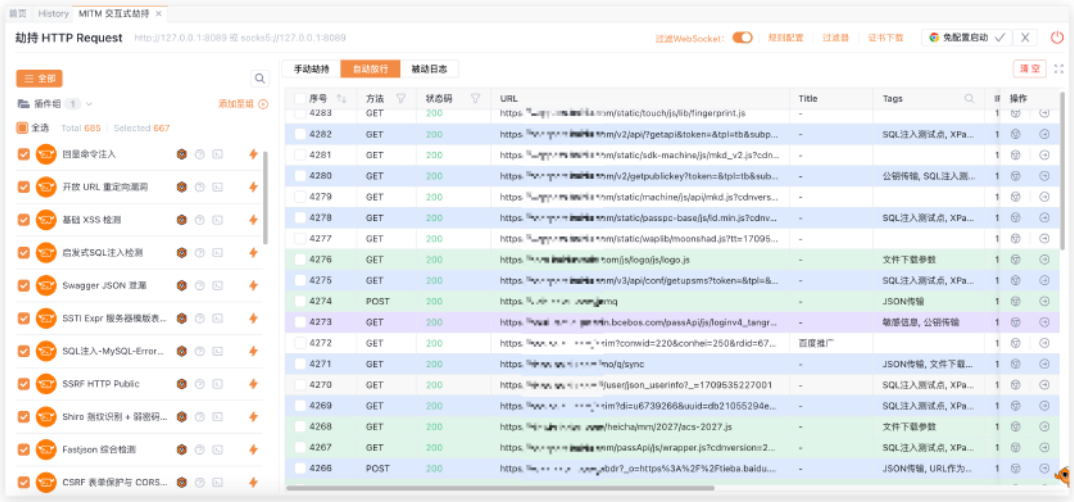
5.1.2 MITM 劫持，却不止于劫持

MITM 操作台可百分百替代 BurpSuite，下载并安装证书、劫持请求、响应、编辑劫持到的数据包等。并且提供一整套顺畅的工作流，劫持 => History => Repeater / Intruder，劫持到的数据，在 History 可以查看历史数据，选择需要“挖掘”的数据包，发送到 WebFuzzer 进行 Repeater / Intruder 操作。除了这些典型的操作场景外，MITM 还提供了插件被动扫描、热加载、数据包替换、标记等更灵活的功能。

被动扫描与热加载

被动扫描支持在劫持流量的过程中，使用插件对流量进行扫描。 插件可由用户灵活定义，比如：

- 可以在劫持的过程中进行漏洞检测、敏感信息检测等等；
- 兼容用户在使用过程中的多种场景；
- 使用插件可随时进行热加载。（也就是在劫持过程中可随时调试插件或根据劫持的情况随时调整插件，通过嵌入 Yaklang 脚本来实现 MITM 动态调试流量，随时动态执行代码。）

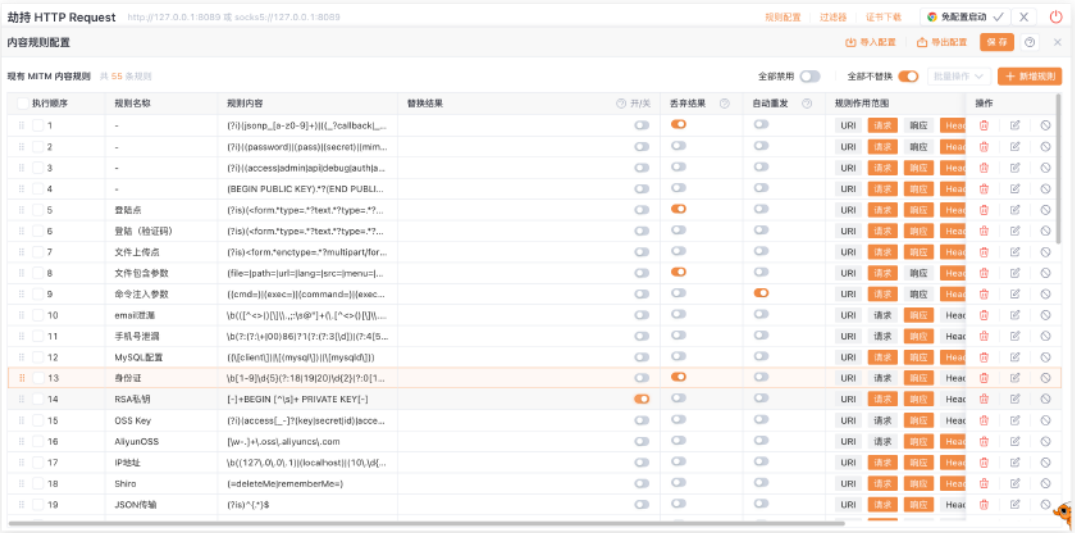


匹配/替换/标记

在劫持过程中可匹配、替换或标记数据包中的信息，比如匹配敏感数据，标记危险 JS 操作（DOM / XSS），替换敏感信息等，用户可以根据自己需要去写规则，从而达到自己的目的。

这是一个非常考验 MITM 技术细节实现的功能。

- 需要正确识别并解码 chunked 数据包，正常解码 `gzip / flate / zlib-flate` 等数据；
- 在解码之后，正确修复数据包的 Headers；
- 在修复了数据包之后，可以对 Response 进行替换；
- 在替换之后，合理修复数据包 `Content-Length` ；
- 在 UTF8 作为通用编码的情况下，灵活处理遇到 `GBK / GBxxx` 的情况等等。



HTTP/2.0 劫持

Yakit 支持在 HTTP/2.0 通信过程中实施劫持。HTTP/2.0 是一种先进的网络协议，相比于其前身 HTTP/1.x，它提供了更高的效率和速度。Yakit 可以劫持、修改或重定向数据包，支持处理 HTTP/2.0 流量，并识别潜在的安全漏洞。关闭后自动降级为 HTTP/1.1，开启后 HTTP2 协商失败也会自动降级。

适配国密算法

国密算法是中国自己研发的一系列加密标准，包括 SM2、SM3 和 SM4 等。Yakit 的适配国密算法功能意味着它可以在 MITM 攻击中识别和处理使用这些国密算法的加密通信。

双向认证

双向认证，也称为双向 SSL/TLS 认证，是一种安全协议，在这种协议中，客户端和服务器都必须互相验证对方的身份。Yakit 的双向认证劫持功能允许它在进行 MITM 攻击时处理这种复杂的认证机制，能够有效地插入到通信过程中，即使在使用双向认证的高安全环境下也能进行监控和分析。

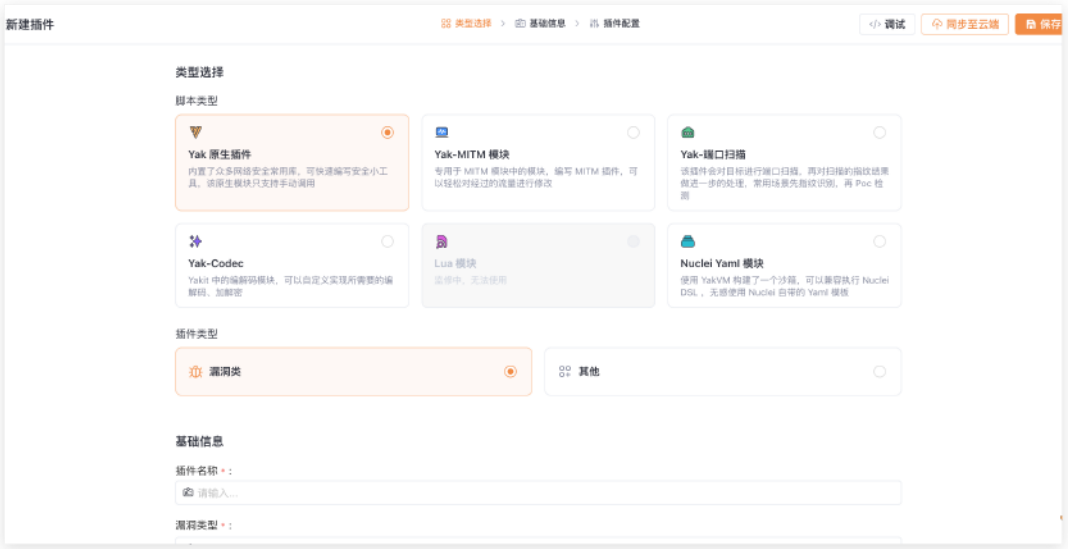
5.1.3 灵活多变，且方便编写的插件系统

灵活多变

Yaklang 的“零依赖”，“嵌入执行”，“自动代码补全”等特点，为 Yakit 的插件系统提供了优秀的可扩展能力与可实践特性。Yakit 的插件系统实质上是将编写的 Yak 脚本发送到 Yak 引擎，引擎将会启动一个进程来执行，执行过程的输出将会通过 gRPC 的双向数据流返回。这就意味着，用户可以根据自己的需要在平台编写各种符合自己实际使用场景的插件。

开放兼容

Yakit 的插件系统具有高度的兼容性和开放性，特别是在支持多种脚本语言和框架方面，除了用 Yaklang 编写的插件之外，还支持使用的 NASL（Network Security Auditor Language）、Lua 以及 nuclei 的漏洞模板插件。这意味着用户可以无缝地将现有的安全脚本和漏洞模板整合到 Yakit 平台中，无需重新编写或进行复杂的转换。这种兼容性使得 Yakit 能够利用广泛的社区知识和资源，提供更多样化和强大的安全检测能力。同时，这也极大地扩展了 Yakit 的适用范围，使其能够更好地适应不同用户的特定需求和场景，从而成为一个真正开放和灵活的安全平台。



自带UI与绘图API

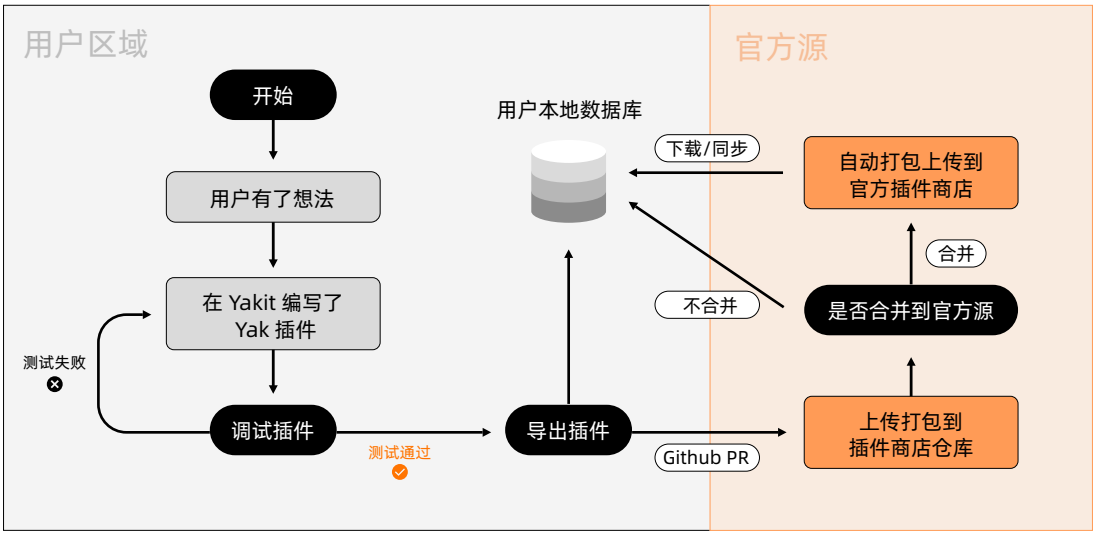
Yakit 提供了一整套插件的使用页面，无需用户思考如何编写“前端”。只需将参数预设好并写好 Yak 脚本，将参数与脚本保存在引擎中，填写相对应的参数，就可以做到在 Yak 引擎中执行。Yakit 除了有提供配套的插件 UI 外，也提供了超强的绘图能力，只需要调用 Yakit 的绘图 API 仓库，直接执行，即可在插件输出中看到绘制的图表，用户无需消耗精力去制作图表。

插件联动

插件的灵活性也与 Yakit 本身的功能进行了结合，比如可以在使用 MITM 劫持、端口扫描的过程中去灵活使用插件，达到漏洞扫描、敏感信息检查等等各种各样的能力。且插件与插件间也可进行联动，只需在编写插件时启用联动 UI，则在使用该插件时，也可使用其他的插件，达到多重使用的目的。

插件生态

Yak 的生态不是完全封闭的，教程与公开源码都是为了鼓励个人或者技术团队把自己的知识/经验沉淀下来，可以用于团队分享或开放供互联网用户使用和学习。随着插件数量和种类的增加，可以完成的细节操作也会越来越多，直到网络安全常见的操作和能力模块都可以在 Yakit 中找到解决方案。

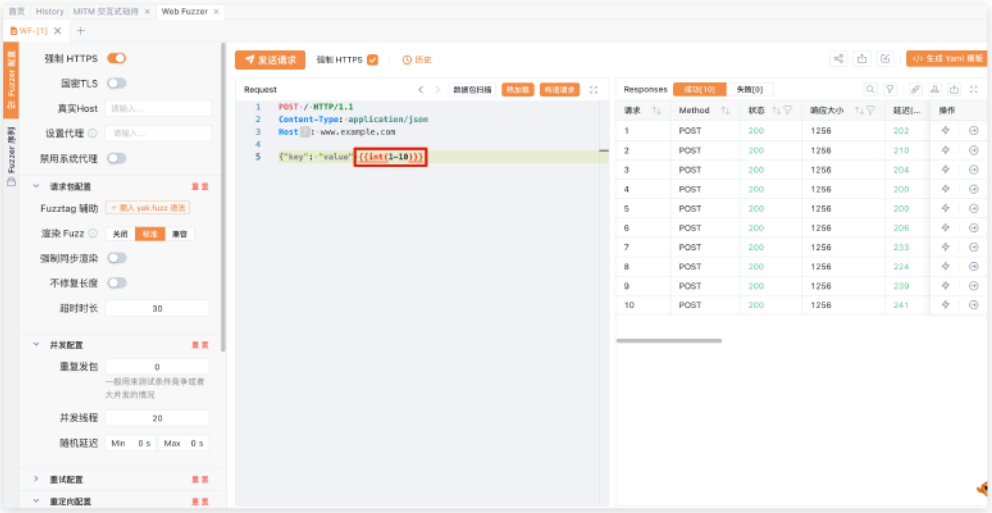


5.1.4 更先进易用的 WebFuzzer

MORE than Repeater + Intruder

Yakit 使用 Yaklang 的核心模糊测试标签语法，实现了对 Repeater 和 Intruder 的完美整合，可以免配置实现批量发包模糊测试 。作为 Yakit 平台的核心组件之一，已经超越了传统的“重放数据包”的功能，成为了一个多面的工具，支持更加丰富的功能。

在使用 WebFuzzer 进行爆破时，通过一个特殊的 Fuzz 标签，标签简单灵活并且可以支持非常多的种类。一般的FuzzTag格式如下： `{{x(user_top10)}}` ，其中 `{{` 为 FuzzTag 标签的开始，`}}` 为 FuzzTag 标签结束。标签内容 `x(user_top10)` 会交由 FuzzTag 执行引擎解析。执行引擎编译过程中会根据小括号将标签内容解析为标签名 `x` 和参数 `user_top10` ，然后再交给标签函数执行，生成数据。如 `x` 函数收到参数 `user_top10` 后，按行读取文件，将读取结果返回。如要进行一个参数遍历 1-10，在 Yakit 中只需要 {{int(1-10)}} 即可把这个位置用数字 1-10 依次替换。



内置常见标签鼠标悬停即可插入，也可自定义，极大化地简化了操作步骤，提供更便捷的测试方式。

除此之外对比 burp suite 的几种模式，Yakit都可以实现支持。

模式	案例	字典数量	理解	应用场景	Fuzztag
Sniper（狙击手）	a= a &b= b	1	轮流替换	fuzz请求参数	轮流替换
Battering ram（攻城锤）	a= a &b= b	1	替换全部	多个字段参数相同	同步渲染
PitchFork（草叉）	a= a &b= b	多个	同步替换	有对应关系的字典	同步渲染
Cluster bomb（集束炸弹）	a= a &b= b	多个	笛卡尔	无对应关系的字典	笛卡尔渲染

其中 Battering ram 模式和 PitchFork 模式是等价的，只是在 burp 中选择字典的数量不同。

热加载 Fuzzer

往往在实战中，需要面对的参数是非常复杂的，上述简单的标签只能覆盖百分之六十的情况，为了解决更复杂的场景，最好的方案就是让标签“图灵完备”。

在 Yakit MITM 插件中，热加载是一个非常有意思的设定，在劫持过程中，任何用户输入的 Yak 代码均可以随时加载到 MITM 过程中执行。基于这项技术，在 WebFuzzer 执行数据包修复和渲染之前，让用户输入一段 Yak 代码，利用 MITM 插件的“热加载技术”，编写回调“函数”，让 Yak 成为 WebFuzzer 和用户自定义代码中的桥梁，在恰当的时候执行这个回调函数，就可以很好地实现标签核心代码“图灵完备”。

匹配器和提取器

为了更加精确地分析数据和处理网络响应，Webfuzzer 模块提供了匹配器和数据提取器功能。

匹配器通过定义一系列的匹配类型（如关键字、正则表达式、状态码等）和匹配位置（如状态码、响应头、响应体等），对网络响应进行深入分析。用户可以结合使用这些匹配类型和位置，创建出复杂的匹配规则来针对性地检测特定的内容或行为。

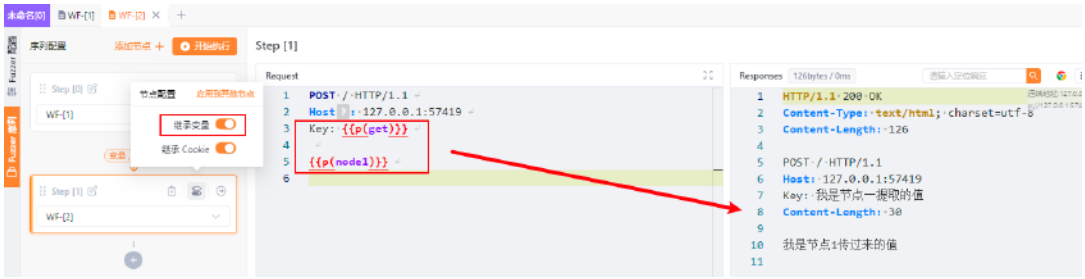
数据提取器则提供了多样的提取方式（包括正则表达式、XPath、键值对等），使用户能够从网络响应中提取关键信息，并将其赋值给变量以供后续使用。

Fuzz 序列器

在实际的安全测试中，往往会面临更加复杂的测试场景，通过 WebFuzzer 序列器，Yakit 用户可以模拟真实世界中的用户行为和应用逻辑，进行更加复杂和深入的逻辑测试。

Web Fuzzer 序列功能允许用户将多个 Fuzzer 节点串联起来，创建复杂的测试序列以模拟实际操作流程。通过简易的拖拽和设置，用户可以构建包含多个步骤的逻辑链，每个步骤可继承前一节点的变量和数据，实现从登录到执行特定操作等一系列动作。这种方法不仅提高了测试的灵活性和深度，也极大地扩展了自动化测试的能力，使得复杂场景下的安全分析变得更加高效和直观。

例如，如果需要模拟一个用户登录并进行后续操作的场景，用户可以首先创建一个进行登录操作的节点，从响应中提取出需要的会话信息或其他关键数据，并将这些信息作为变量保存。随后，在序列的下一个节点中，用户可以使用这些变量来模拟已登录用户的操作。

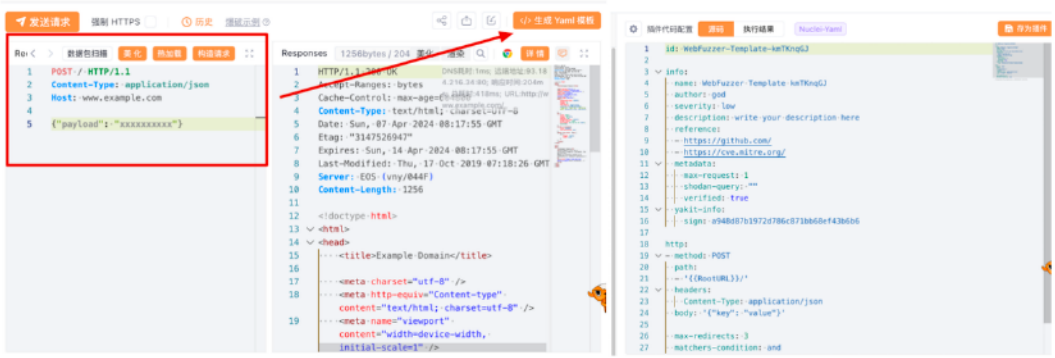


一键生成 PoC

在网络安全领域，PoC（Proof of Concept）是验证概念或漏洞存在的重要工具，它在安全研究、漏洞发现和漏洞利用等方面扮演着关键的角色。传统上，攻击方需要经历从手动测试编写 PoC、调试其正确性，到最终批量部署的繁琐流程，这个过程往往耗时且缺乏效率，因为各个阶段之间缺乏紧密的关联性和自动化的支持。

Yakit 平台针对这一问题提出了创新的解决方案。在 WebFuzzer 功能中，提供了“一键导出 PoC yaml”以及紧密联动的调试功能。

这意味着用户可以在 Yakit 平台上进行 PoC 的设计和测试，并且通过一键导出功能，迅速将 PoC 转化为可用的 yaml 格式文件，极大地简化了 PoC 的编写和部署过程。此外，联动的调试功能确保了 PoC 在实际使用前的正确性和有效性，减少了从业人员在调试和验证上的工作量。



5.1.5 多场景兼容的反连管理

多协议复用端口

人为设置一个子域名，任何查询到这个子域名的请求被集合展示，这就是 DNSlog 的核心原理。目前实现了记录 RMI / LDAP / TLS / HTTP / HTTPS 反连时携带的相关详细信息。有趣的是，这些协议都监听在了一个端口，通过识别不同的协议头来判断是哪一个协议，实现了拟态反连服务器。协议复用除了使用方便以外，更重要的是，通过反连平台返回的未知协议，能让用户知道目标此处真实存在一个反连问题。

内网穿透

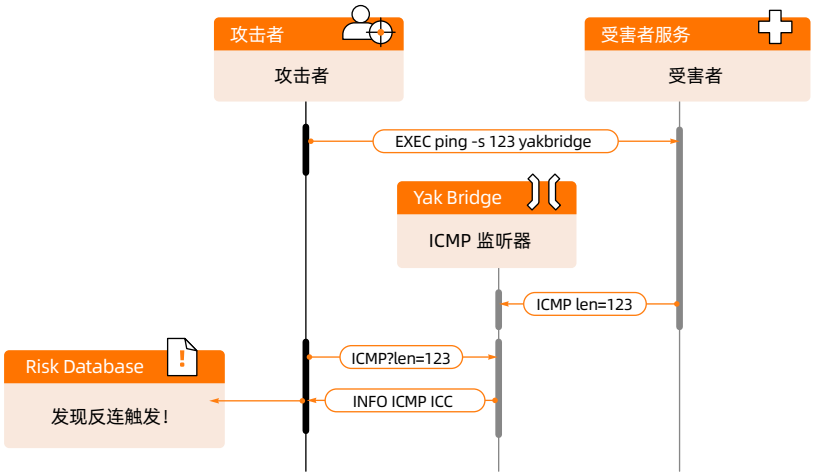
在 Yak 中实现了一套机制，叫 CyberTunnel，使用 gRPC 构建了一套端口穿透技术，可以实现映射本地 TCP/UDP 协议端口到远端。甚至可以映射本地 DNSlog 服务器到远端（远端需要域名提供商配置解析记录）。

服务器上安装了 Yak 核心引擎，使用 YakTunnel 映射本地任何端口出去，包括 Yak gRPC，在内网进行渗透的时候，只需运行一个 Yak gRPC，再通过 Yak Tunnel 穿透出去，就可以使用 Yakit 连接，达到内网穿透的目的。

除了 Yak 支持的任意端口的映射之外，也可以支持 Yakit 把反连服务器映射到远端。对于 Yakit 来说，减轻用户的操作负担是核心诉求之一。

ICMP Size Logger

ICMP 最简单的触发是 ping 命令，如果在受害主机上 ping 自己控制的公网服务器，如果 ping 通了，ICMP 就会马上尝试测试 TCP 反连之类的各种操作。使用 ping 命令的 `ping -s [len] yakbridge` 可以 ping 到目标主机，在用户 ping 命令执行后，可以自动观测到“谁 ping 了我”。

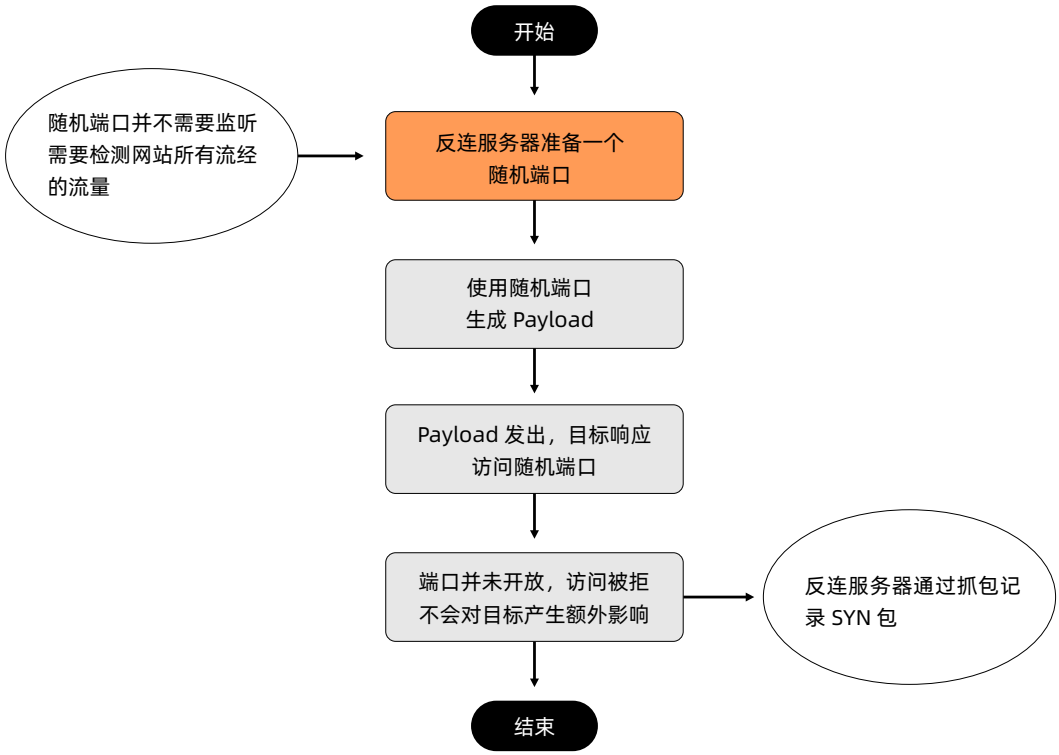


TCP Random Port Logger

实际上，除了 DNS 外的大多数场景来说，绝大多数反连的“检测”都是以 TCP 作为基础协议的，通过对 TCP 的协议实现，从而实现多种复杂应用的兼容：

- 常见的 HTTP 反连检测，一般使用 Path 作为 Token 的标注位置，来识别“到底是哪个漏洞触发”：`http://example.com/[token]`
- 常见的 RMI 类似 `jndi:rmi://example.com/[token]`
- ...

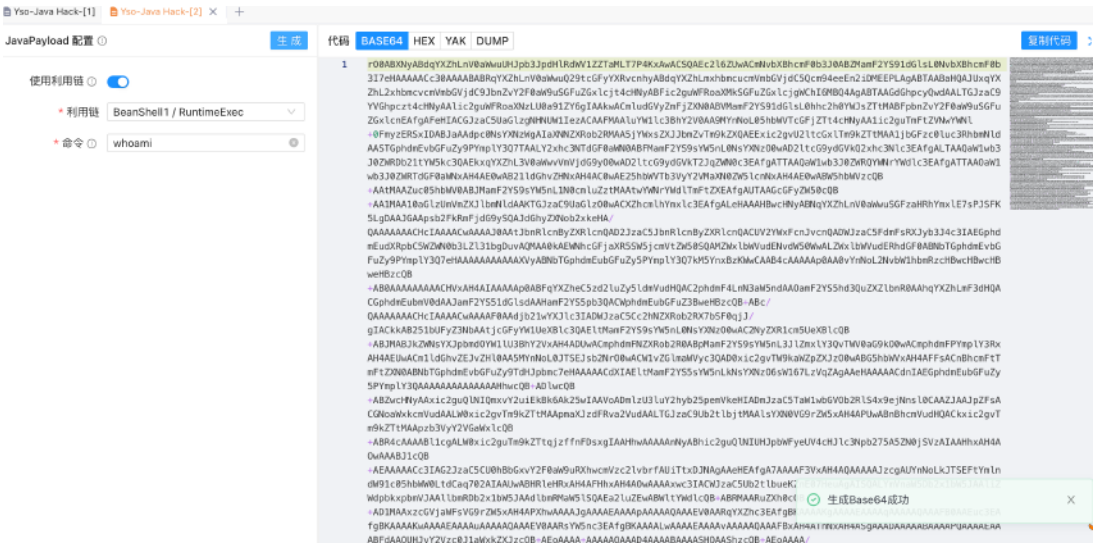
但实际上，上面的各种用例都需要用户具体监听某个端口来实现，经过思考，Yakit 实现了一种更通用的检测手段，并成功把它进行了工程化实现：



Java 反序列化

Java 漏洞的利用相当繁琐，尤其是对于复杂的安全漏洞来说。通常涉及到多个步骤和环境，包括但不限于配置 Java 环境、使用多种工具进行测试和利用、在远程服务器（如 VPS）上监控回显以及记忆和执行各种命令。每一个环节都可能遇到问题，从而增加了整个过程的复杂性和时间消耗。

对此，Yakit 提供了 Java hack 功能，可以大幅简化这一过程。通过提供一站式的解决方案，它可以整合多种工具和流程，将繁琐的环节自动化，提供用户友好的界面和直观的操作流程。例如，提供图形化界面来替代复杂的命令行操作，提供反连平台管理回显，支持自定义恶意类等，大大降低了技术门槛和操作难度。



5.1.6 全方位的权限管控

在企业的实际应用中，全方位的权限管控是维护系统安全的关键环节。Yakit 通过其高级权限管理功能确保只有授权用户才能访问敏感数据和重要功能，从而保护系统免遭未授权访问和潜在内部威胁。这一功能对于维护组织的数据安全性和符合相关法规标准至关重要。Yakit 的权限管理系统提供细致的权限和角色控制，确保每个用户能根据其职责和需要获得适当的访问权限。

用户认证

Yakit 采用认证机制，确保所有访问请求的合法性和安全性。从基本的用户名和密码到多因素认证，Yakit 支持一系列认证方法，确保每次访问都是经过严格验证的。此外，Yakit 后续将集成现有的企业认证系统，如 LDAP 或 Active Directory，以实现无缝的用户管理和认证。

角色定义与权限分配

在 Yakit 中，管理员可以根据组织的具体需求和安全策略，定义多种角色，并为每个角色分配一组特定的权限。这些角色可能包括管理员、审计员、普通用户等，每个角色都具有不同的访问级别和权限。通过将用户分配到适当的角色，Yakit 能够实现精确的权限控制，同时简化了权限管理过程。

访问控制

Yakit 采用先进的访问控制技术，如访问控制列表（ACLs）和基于策略的控制机制，来细致管理用户对系统资源的访问。这些技术确保 Yakit 能够根据用户的角色和权限来决定其是否可以访问特定的资源或执行特定的操作。此外，Yakit 还支持条件访问控制，提供更灵活和安全的访问控制。

审计与报告

为了确保系统的安全性和合规性，Yakit 提供全面的审计和报告功能。它记录所有用户的访问和操作尝试，无论成功与否，为安全分析和合规性审计提供了详细的数据。管理员可以通过 Yakit 的审计日志来监控系统使用情况，识别潜在的安全问题，并生成合规性报告。这有助于组织及时发现和响应安全事件，以及帮助组织满足各种法规和行业标准的要求。

5.1.7 高效的团队协作体系

在渗透测试领域，尤其是在隔离的内网环境中，多人协同成为提升测试效率和质量的关键因素。Yakit 平台通过引入一系列先进的功能，极大地促进了团队之间的协作和信息共享，确保测试工作的连贯性和高效率。通过协作功能，Yakit 不仅解决了隔离内网环境下的通信和协作问题，还显著提升了渗透测试的整体效率和效果。它简化了传统的协作流程，确保了信息和资源的高效共享，使得团队能够更加快速、准确地应对各种安全任务及工作。借助 Yakit，安全团队可以实现更紧密、更高效的协作，有效提升安全测试的效率和应对速度。

实时数据共享与远程协助

Yakit 通过统一的 gRPC 服务端允许用户实时共享测试数据和进程，无论团队成员身在何处，都可以即时掌握其他同事的测试进度和细节。这种实时共享机制显著提升了团队协作的同步性和透明度。此外，Yakit 的远程协助功能使多个用户可以在隔离的内网中完成即时的屏幕共享和远程控制，就像使用向日葵远程协助工具一样，大大简化了团队协作过程，尤其在解决复杂问题和共同完成任务时能更突显其价值。

WebFuzzer 数据共享与工具协同

在特定漏洞测试中，Yakit 允许团队成员通过分享密令实时共享测试细节和数据包，当团队成员面临困难或需要帮助时，可以通过共享测试数据快速寻求协助。此外，所有的测试工具、通信工具和数据管理工具在 Yakit 中都能在隔离的内网环境中正常工作并有效协同，包括测试工具的数据输入输出、通信工具的信息交换以及数据管理工具的资料共享和访问控制等。

文件共享、文档管理

Yakit 的文件共享和协作注释功能使团队成员能够轻松共享文档，并在文件上进行实时讨论和标注。这不仅提高了沟通效率，也确保了所有团队成员对任务文档内容有清晰的理解和共识。同时，Yakit 的文档共享与管理功能使得安全团队可以集中管理所有相关文档，确保信息的一致性和最新状态。此外，Yakit 的权限控制模块也确保了每个团队成员只能接触到对其任务必要的资源，有效地防止了敏感信息泄露或失误操作。

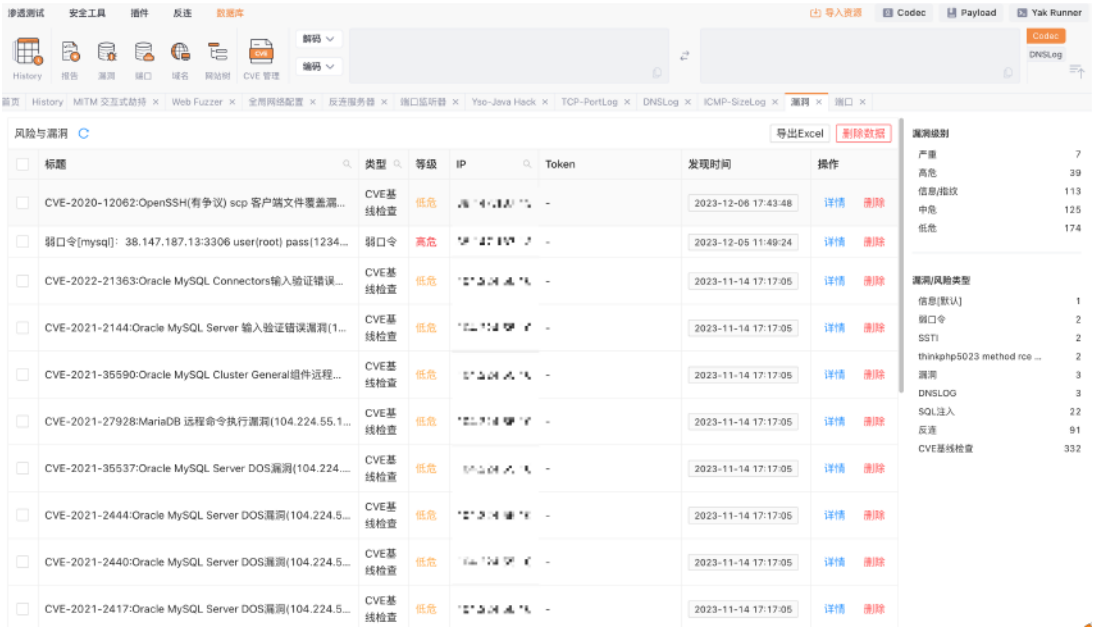
Yakit 支持及时共享关键信息，如实时威胁情报、安全漏洞更新和团队内部的策略调整。这种信息共享机制确保团队成员能够及时获得重要的安全信息，以便更快速地响应和适应不断变化的威胁环境。

5.1.8 项目管理和数据管控

项目管理功能同样是为了增强团队的组织和协作效率而设计的。在这个框架下，每个项目中产生的所有业务流量都被详细记录，并存储到单独的数据库中。这一设计不仅使数据的追踪和管理变得更为高效，也为后续的质量分析和数据审计提供了坚实的数据支撑。

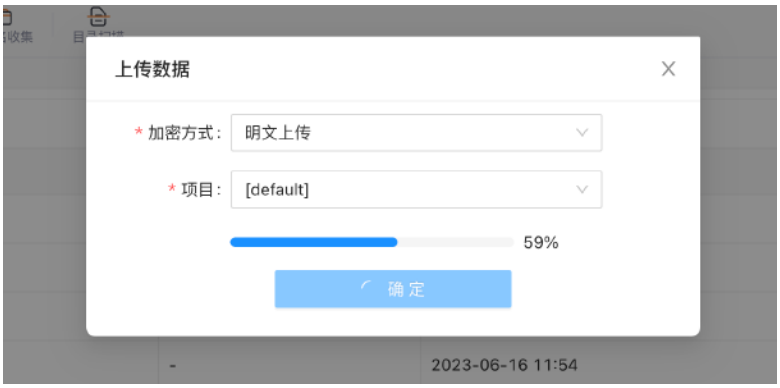
数据追踪和管理

Yakit 中的项目管理功能确保了所有业务流量数据的完整性和可访问性。每个项目的数据都被自动记录并隔离存储，使团队能够针对特定项目轻松地回溯和分析历史数据。



数据上传至离线 Server

为了进一步增强数据管理的灵活性和安全性，Yakit 支持将项目数据库自动上传至统一的离线 Server。这不仅为数据提供了一个安全的备份，也使得团队可以在需要时随时访问和分析这些数据，无论他们身在何处。客户端的数据通过安全的通道上传至服务端，使用高级的加密协议来保障传输过程中的数据安全。所有数据在存储前均进行加密处理，采用行业领先的加密技术确保数据的机密性和完整性，防止未经授权访问。即使面对数据被非法访问的情况，加密存储也能确保信息的安全，保护关键数据不受损害。



可视化展示与报告

Yakit 将分析结果通过直观的图表和报告形式呈现，使用户能够快速理解和评估当前的安全状况。提供可定制的仪表板和报告，满足不同用户的特定需求，提供关键信息和洞察。实时的可视化工具帮助用户迅速识别并响应最新的威胁和事件，加快决策和反应速度。

数据复用与协同

Yakit 支持分析得到的数据提供给其他安全设备和平台，实现跨系统的信息共享和协同。通过标准化的接口和协议，确保数据能够在不同的系统间安全、有效地流动和利用。数据的复用不仅增强了整个安全生态系统的协作和响应能力，也极大提升了对复杂威胁的防御效率。

5.1.9 高级功能和定制化选项

自定义插件和客户端

用户可以根据自己的安全需求和政策定制规则和策略，确保安全措施与组织的具体要求相符合。提供灵活的规则配置界面，用户可以轻松添加、修改或删除规则。

定制化数据分析和报告

提供高级的数据分析报告，用户可以进行深入的数据探索，发现潜在的安全风险和趋势。定制化的报告功能允许用户创建符合自己需求的报告，无论是定期的安全摘要还是特定事件的深度分析。

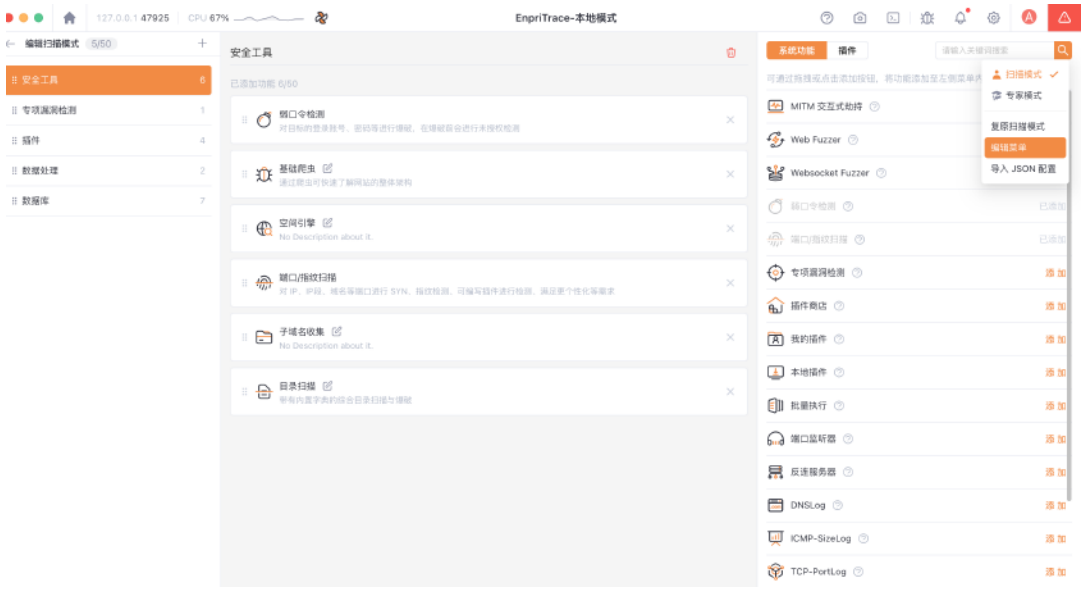

目录
1、项目概述
1.1 测试目的
1.2 安全测试原则
1.2.1 标准性原则
1.2.2 规范性原则
1.2.3 最小影响原则
1.2.4 保密原则
2、测试方法
2.1 概述
2.2 风险等级说明
3、测试结果概述
3.1 总体安全现状
漏洞汇总
风险IP汇总
3.2 端口扫描统计
3.3 插件扫描状态统计
3.4 风险统计
3.4.1 漏洞统计列表
3.4.2 风险资产图
3.4.2 合规检查风险列表
4、后续整改建议
附录：
漏洞详情与复现依据
严重漏洞详情
FOUND Oracle WebLog...
FOUND Oracle WebLog...
80
60
40
20
0
严重漏洞
高危漏洞
中危漏洞
低危漏洞
风险IP汇总
IP地址	严重风险	高风险	中风险	低风险	总计
[redacted]	4 个	36个	115个	4个	159 个
[redacted]	0 个	0个	0个	0个	0 个
[redacted]	0 个	0个	0个	0个	0 个
[redacted]	0 个	0个	0个	0个	0 个
3.2 端口扫描统计
[redacted]	端口	指纹	网站标题
[redacted]	22	linux_kernel/openssh[8.2p1]...	
[redacted]	8004	apache[2.4.25]/debian/debi...	
[redacted]	8010	http	
[redacted]	8011	http	
[redacted]	8009	bootstrap/http/java/jsp	Login Page
[redacted]	8001	apache_tomcat[1.1]/coyote[1	Apache Tomcat/7.0.77

API接口与集成

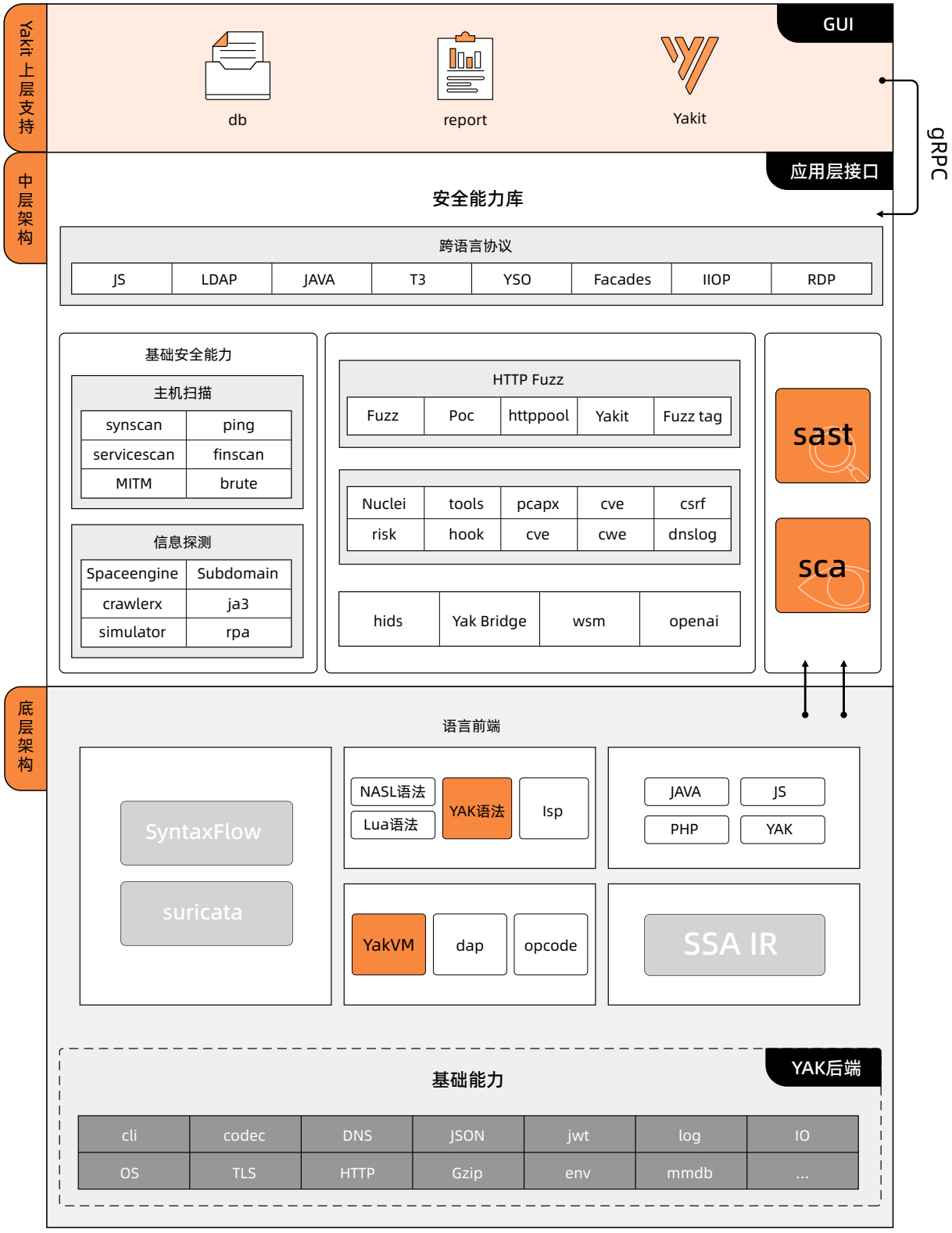
提供 API 接口，支持与其他安全工具和系统的集成，如 SIEM 系统和其他安全产品。通过集成，用户可以构建一个协同的安全生态系统，实现数据和资源的共享。

用户界面和体验定制

用户可以根据自己的偏好和工作流程定制界面和操作方式，如定制仪表板、设置快捷操作等。定制化的体验使得操作更加直观高效，提高了用户的工作效率。



技术特性

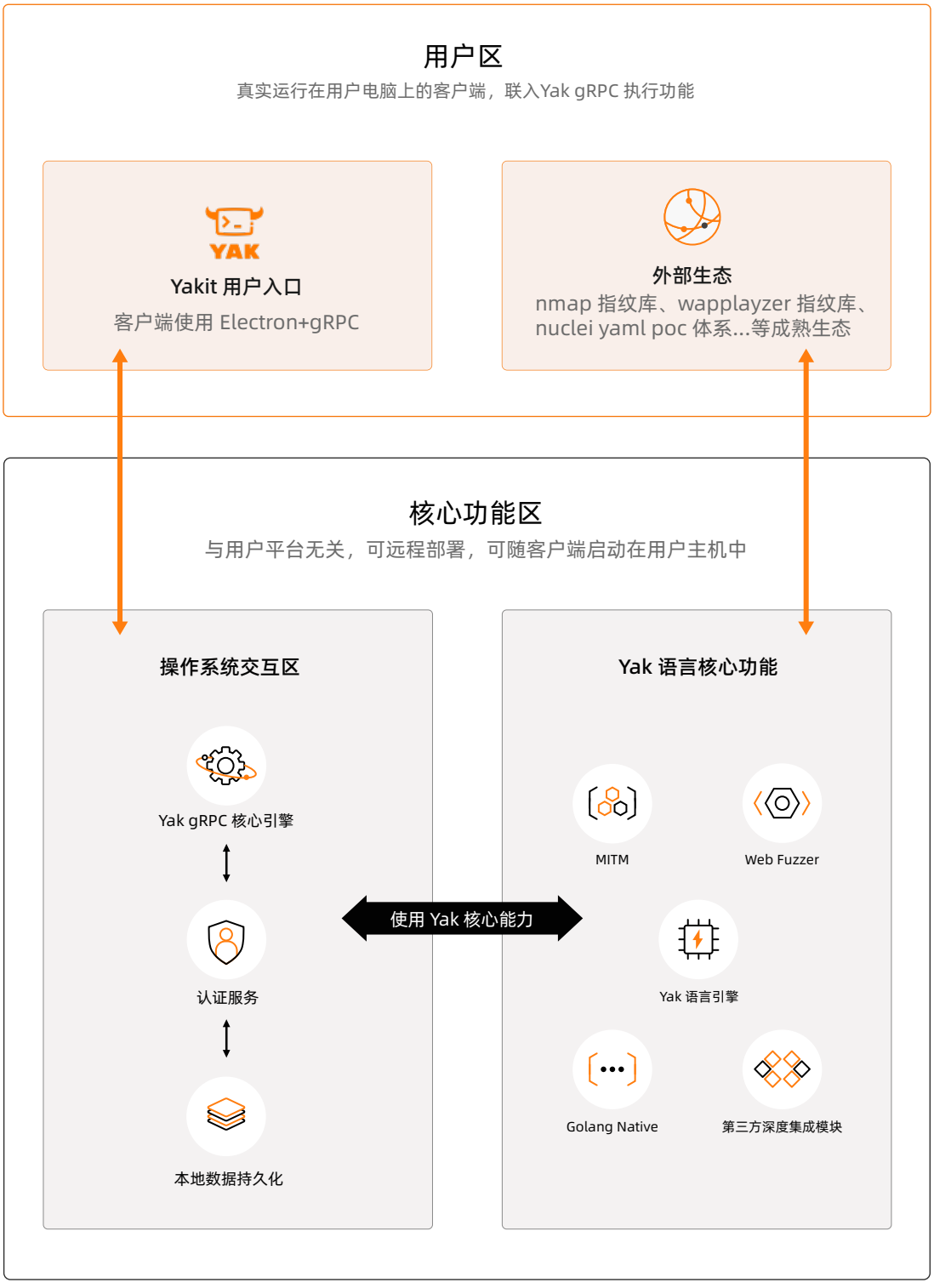

Yakit 上层支持
中层架构
底层架构
GUI
db
report
Yakit
gRPC
应用层接口
安全能力库
跨语言协议
JS
LDAP
JAVA
T3
YSO
Facades
IIOP
RDP
基础安全能力
主机扫描
synscan
ping
servicescan
finscan
MITM
brute
信息探测
Spaceengine
Subdomain
crawlerx
ja3
simulator
rpa
HTTP Fuzz
Fuzz
Poc
httppool
Yakit
Fuzz tag
Nuclei
tools
pcapx
cve
csrf
risk
hook
cve
cwe
dnslog
hids
Yak Bridge
wsm
openai
sast
sca
语言前端
SyntaxFlow
suricata
NASL语法
Lua语法
YAK语法
lsp
JAVA
JS
PHP
YAK
YakVM
dap
opcode
SSA IR
YAK后端
基础能力
cli
codec
DNS
JSON
jwt
log
IO
OS
TLS
HTTP
Gzip
env
mmdb
...

Db: 数据库操作，用于存储和检索数据。
Spaceengine: 支持网络空间搜索引擎（如 Shodan、FOFA）的集成，用于信息收集和资产发现。
Fuzz: 实现模糊测试的功能，模糊测试数据生成，用于发现潜在的安全漏洞。
Report: 生成和管理安全测试报告。
Subdomain: 子域名的发现和分析，支持域传送、爆破和互联网数据收集。
PoC: 漏洞验证和利用脚本的开发支持。
Risk: 漏洞评估和验证辅助。
Crawler: 网络爬虫功能，支持表单识别和链接抓取。
Fuzztag: 用于测试数据的模糊标签渲染。
Nuclei: 提供类似 Project Discovery 的 YAML PoC 框架。
Hook: 插件和扩展功能的辅助调用。
Yak Bridge: 提供 DNSLog、RMI、HTTP、LDAP 等协议的支持和日志记录。
Wsm: 计划中的 WebShell 管理工具。
MITM: 实现中间人攻击的相关功能。
synscan: 网络端口的SYN扫描。
servicescan: 网络服务的指纹识别和扫描。
codec: 数据的编码和解码，包括加解密操作。
JSON: 处理 JSON 格式数据。
YAML: 处理 YAML 格式数据。
xhtml: HTML 内容的解析和处理。
XPATH: 对 HTML 文档进行 XPATH 查询。
judge: 数据比较和判断。
mmdb: 处理 MaxMind DB 格式的地理位置数据。
Gzip: GZIP 格式的压缩和解压缩。
HTTP: 处理 HTTP 协议的请求和响应。
Httpool: 管理 HTTP 请求的连接池。
DNS: 处理 DNS 请求和解析。
LDAP: 处理 LDAP 协议的请求和通信。
T3: 处理 Weblogic T3 协议的通信。
YSO: 处理 Java 序列化和 YSoSerial 的攻击。
ping: 实现网络的 ping 操作，检测主机的在线状态。
log: 实现日志的记录和管理。
exec: 执行系统命令和脚本。
os: 操作系统层面的功能，如文件系统操作。
env: 管理和访问环境变量。
IO: 实现输入和输出操作。

ZIP: ZIP 格式文件的压缩和解压缩。
time: 时间和日期的处理。
str: 字符串处理功能。
re: 正则表达式的创建和匹配。
X: 函数式编程辅助
cli: 命令行界面的创建和管理。
js: 执行和管理 JavaScript 代码。
java: Java 相关的操作，如序列化和字节码处理。
SMB: 处理 SMB 网络文件共享协议。
jwt: 处理 JSON Web Token 的生成和验证。
bot: 实现自动化机器人功能。
context: 管理程序执行的上下文环境。
file: 文件的读写操作。
sync: 实现同步原语，如互斥锁和等待组。
tcp / udp: TCP 和 UDP 网络协议的实现。
brute: 实现基础协议的爆破攻击。
crawlerx: 高级网络爬虫功能。
ja3: 实现 SSL / TLS 客户端指纹识别。
pcapx: 网络包捕获和分析。
sca: 静态代码分析工具。
yakit: 插件辅助工具包，向 yakit UI 传输日志等信息。
systemd: 系统服务的管理。
hids: 主机入侵检测系统。
openai: 集成 OpenAI 相关功能。
tls: 处理 TLS 安全协议。
redis: Redis 数据库的操作和管理。
git: Git 版本控制的操作和管理。
simulator: 模拟器功能，模拟浏览器操作。
facades：综合服务支持。
rpa：模拟浏览器进行爆破操作。
bufio：缓冲 I/O 处理，用于高效的文件和数据流操作。
re2：支持回溯的正则匹配。
suricata：网络安全监控，用于网络流量分析和流量生成。
cve：用于处理 CVE 漏洞库相关信息。
rdp：用于远程桌面协议（Remote Desktop Protocol）的爆破。
bin：二进制处理，可能用于二进制数据的处理或解析。

httpserver：HTTP 服务器，用于构建和运行 HTTP 服务。
csrf：跨站请求伪造（Cross-Site Request Forgery）漏洞相关的处理。
dnslog：DNS 日志处理，用于 DNS 查询的记录和分析。
iiop：weblogic IIOP 协议支持和处理。
dictutil：用于处理和操作字典类型数据。
finscan：FIN 扫描，支持 fin 端口扫描技术。
regen：基于正则表达式反向生成 fuzz 数据。
tools：漏洞利用实现工具集。
xml：XML 处理，用于解析和处理 XML 文档。
math：数学函数库，提供数学计算和操作。
cwe：CWE 漏洞库相关，用于处理 CWE 相关信息。
dyn：用于动态代码加载。

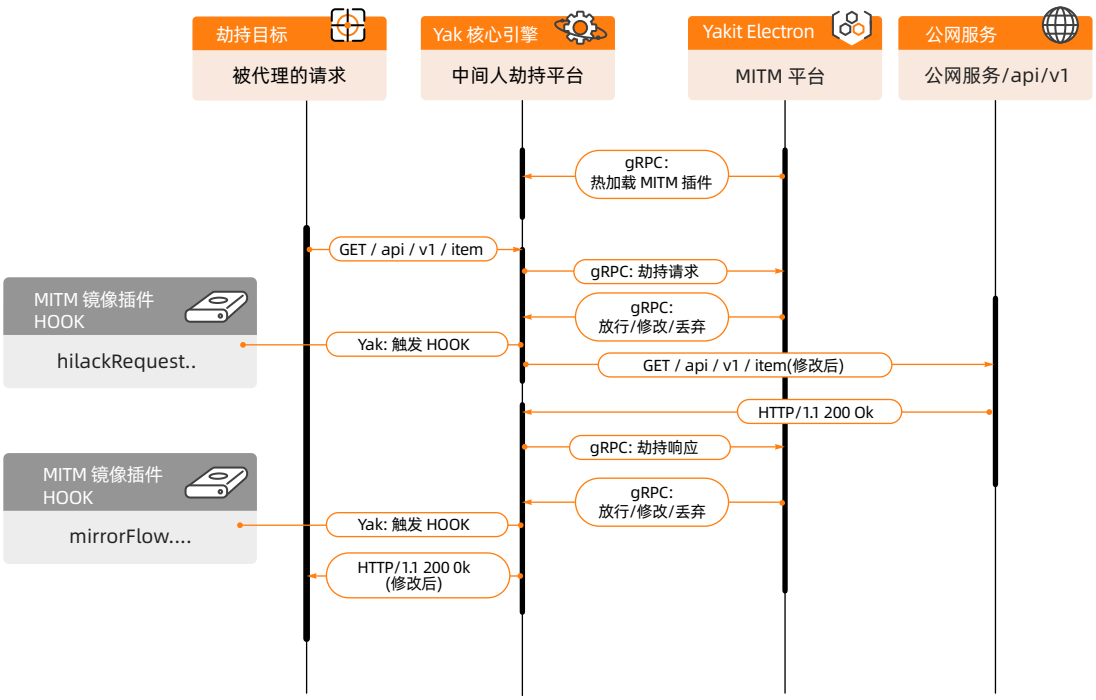
6.1 技术架构详解


用户区
真实运行在用户电脑上的客户端，联入Yak gRPC 执行功能
YAK
Yakit 用户入口
客户端使用 Electron+gRPC
外部生态
nmap 指纹库、wapplayzer 指纹库、
nuclei yaml poc 体系...等成熟生态
核心功能区
与用户平台无关，可远程部署，可随客户端启动在用户主机中
操作系统交互区
Yak gRPC 核心引擎
认证服务
本地数据持久化
使用 Yak 核心能力
Yak 语言核心功能
MITM
Web Fuzzer
Yak 语言引擎
Golang Native
第三方深度集成模块

6.2 专利技术与核心创新

6.1.1 MITM 交互式劫持技术

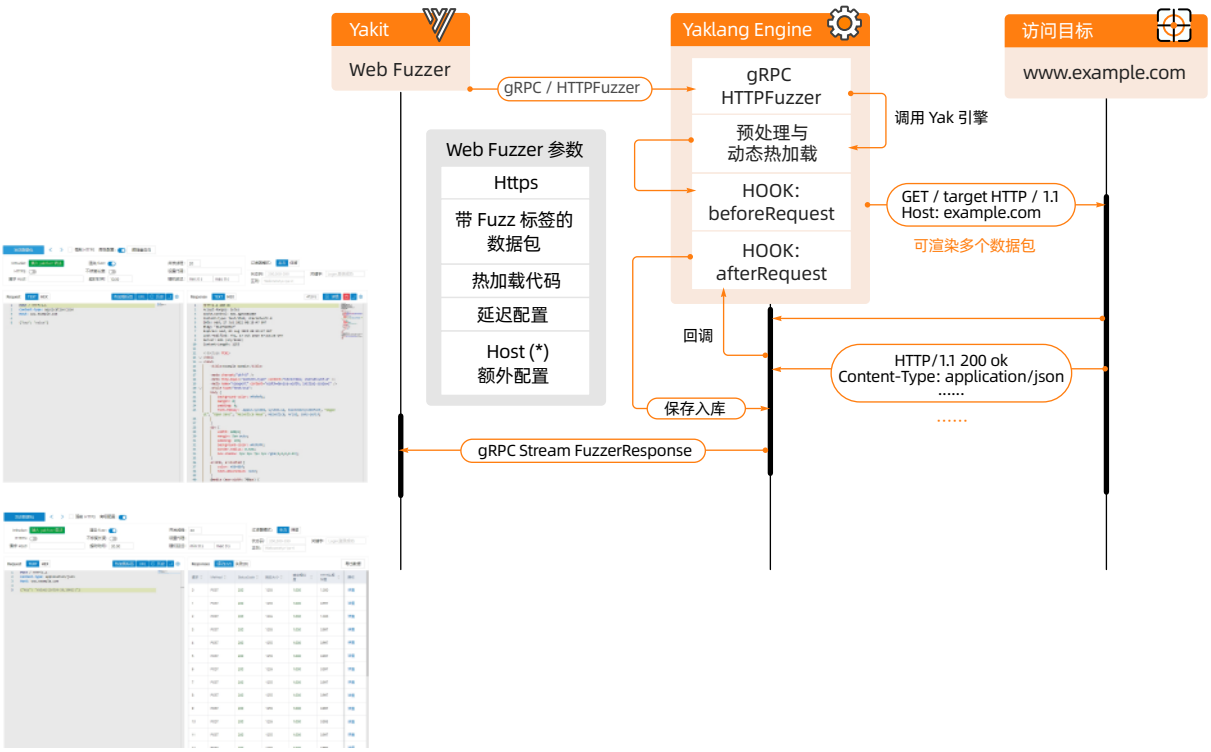
Yakit 的 MITM 模块原理是启动一个 HTTP 代理，自动转发流量，当用户启动手动劫持后，会停止自动转发，阻塞请求，并将请求出栈，做解 Gzip、处理 chunk、解码等处理，让请求变得人类可读，并显示在用户前端，用户可以对请求做查看、修改或重放。重放时，会对用户构造的 HTTP 请求数据包做修复，保证请求包的有效性。Yak 引擎手动实现了HTTP 库，所以用户可以自定义畸形的请求包、响应包，应用于一些特殊场景下的漏洞利用。



6.1.2 Web 应用交互式流量重放与模糊测试

Web 应用交互式流量重放

WebFuzz 模块支持用户自定义 HTTP 原文发送请求。为了让用户使用简单，符合直觉，只需要关心数据相关信息，Yakit 后端做了很多工作。HTTP 原文中一些保证数据传输和解析的信息都是由 Yakit 后端修复补全的，例如修复 CRLF，补全 Content-Type、通过 chunk 方式传输、补全 boundary、修复 Content-Length 等等。

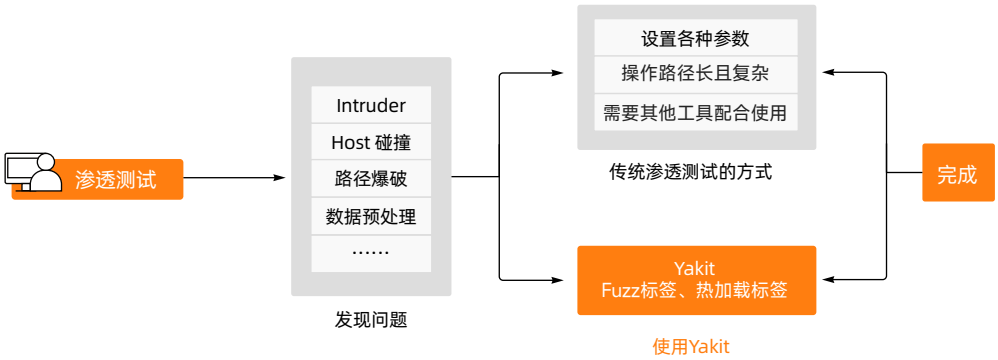


Fuzzer 标签

WebFuzzer 模块支持通过 Fuzz 标签，自然且完美整合了 Host 碰撞、Intruder、目录爆破等功能。例如单参数爆破场景，以爆破用户 id 为例，可以使用 `{{int(1-10)}}` 标签自动生成爆破的 id。面对多个参数爆破的场景，采用笛卡尔乘积的结果作为爆破参数，相较于 BurpSuite 的 Intruder 模块，免去了选择爆破方式，导入字典等步骤，极大的减少了用户的操作路径，符合用户使用习惯。当然 WebFuzz 模块除了使用标签自动生成参数，也支持导入外部字典，例：`{{file(/tmp/username.txt)}}` 。面对一些更复杂的数据场景，WebFuzz 模块支持插入热加载标签，例如需要爆破某地区的身份证号，可以直接在 WebFuzz 模块插入 Yak 脚本生成数据进行爆破。而 BurpSuite 的 Intruder 模块，面临这种场景，需要编写代码生成字典，再导入 Intruder 模块。

爆破优化

WebFuzzer 模块支持并发，提高爆破效率，这是 BurpSuite 不具有的。爆破结果的处理上，WebFuzzer 模块支持多维度对响应包筛选，可以清晰简单快捷地筛选出所需要的信息。在爆破过程中，由于请求次数过多，导致 ip 被 ban 的情况也很常见，WebFuzz 模块支持设置代理解决此问题。



6.1.3 端口协议复用技术

许多漏洞利用场景需要用到不同协议服务的反连，传统漏洞利用工具需要在公网服务器为每一个服务监听一个端口，如 LDAP 类型的漏洞利用，需要启动 HTTP 服务、LDAP 服务，然后发送攻击请求，才能完成一次漏洞利用。传统服务需要为每一个服务分配唯一端口，而 Yaklang 的端口协议复用技术可以监听一个端口，识别请求的协议，作出相应响应。Yaklang 端口协议复用技术优势除了节省系统资源，便于手工测试等以外，由于底层原理是手工识别了各种协议头，基于协议标准对各种协议进行手工实现，从而可以灵活构造各种协议的数据包。例如可以构造一些畸形的协议进行漏洞利用，或通过 DNS 协议、ICMP 协议等携带数据，可以通过这种方式作为后门的权限维持或绕过一些 TCP 协议不出网的情况下的漏洞利用。

丰富的协议支持

传统漏洞利用工具由于语言的特性，不同语言对不同协议支持或使用便捷性不一，在某些场景下的漏洞利用需要多种语言工具结合使用，如常见的 Fastjson 漏洞利用流程：

（1）用Java语言编译一个恶意 class，

（2）在公网通过 python 启动一个 HTTP 服务，将恶意 class 放在 HTTP 服务器根目录下，

（3）通过 LDAP 利用工具启动 LDAP 服务，输入 python 的 Web 服务地址作为参数，

（4）启动 Java 利用工具，输入攻击目标和 LDAP 服务器地址，完成漏洞利用，当漏洞存在时，RMI 服务或 HTTP 服务可以看见目标主机的回连记录。

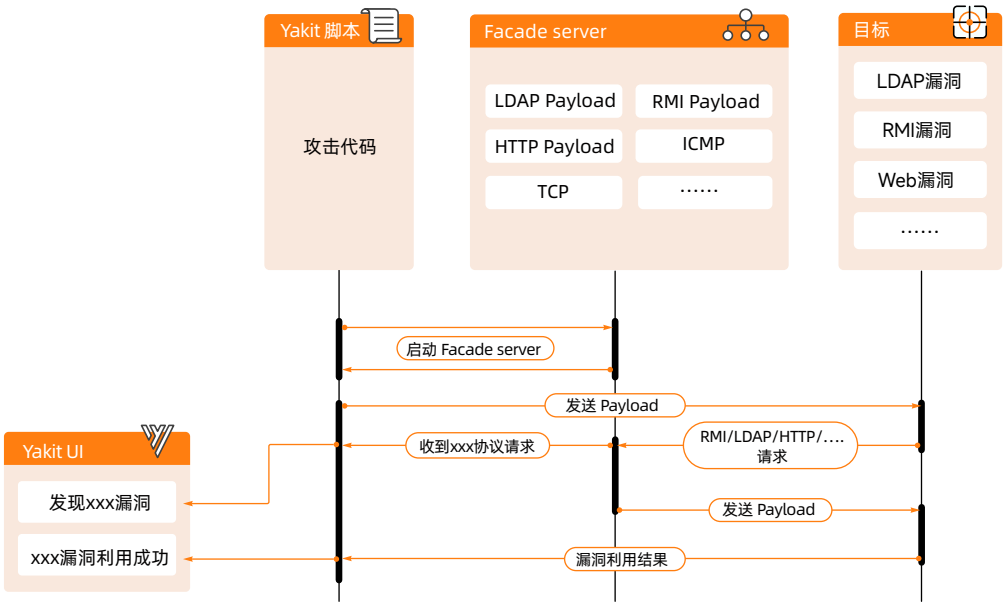
由于是通过多种语言结合使用，攻击目标在 Java 工具输入，而回显记录在 RMI 工具或 HTTP 服务的主机上。此类工具利用过程繁琐，需要在服务器配置多种语言环境，漏洞检测也需要人工检查，所以很难实现自动化，更不能实现批量漏洞检测。

Yaklang 提供了一些对各种协议支持的内置库，如 HTTP、ICMP、LDAP、RMI、T3、IIOP 等协议。通过"端口协议复用"技术，只需监听一个端口，就可以自动识别连接目标的协议，作出响应，可支持各种协议的反连。上文的 Fastjson 漏洞场景利用，Yak 插件中只需要监听一个端口，设置 LDAP 资源、HTTP 资源，并用 HTTP 协议发送 Paylaod，即可实现漏洞利用。而且可以通过 HTTP 资源访问记录，实现漏洞检测自动化，甚至此类漏洞的批量检测。通过 Yaklang，一个脚本即可实现批量自动化检测或利用。

随着 Java 语言的兴起，Java 应用漏洞随之暴露。一些 Java 漏洞利用需要使用 Java 官方库或 Java 框架等内置的协议，使用非 Java 语言编写利用工具，甚至需要手动实现协议。这些情况导致耗时耗力，适用场景也有限，且回报率低。所以市面上流传的 Java 应用漏洞的利用工具都是基于 Java 语言的，因此，此类工具不能构造畸形的协议，在面对一些场景下的 Java 漏洞，传统安全工具无能为力。Yaklang 为了实现对Java 漏洞的利用，提供了对 RMI、T3、IIOP 等协议支持的内置库，并为用户提供了简洁的调用库，编写漏洞利用工具时用户可以专心于漏洞利用逻辑而不是协议相关内容，漏洞利用工具编写效率和插件执行效率大大提高。由于 Yaklang 编写的 Java 漏洞利用插件不受标准库限制，所以可随意构造畸形的 Payload，解决了 Java 漏洞利用工具由于受标准库限制而无法解决一些漏洞场景的弊端。

反连核心技术

Yakit 基于端口协议复用技术实现了反连模块，其中包括 Reverse Shell、反连利用、反连检测功能。Reverse Shell 可以监听指定端口，作为反弹 shell 的接收端，收到反弹 shell 后，可以和 ssh 一样控制远端服务器。传统渗透测试的反连需求中需要使用 nc 监听端口，但一些按键如退格键、方向键会有乱码的情况，反弹的 shell 使用起来和原生的ssh还有很大区别，而 Yakit 的 Reverse Shell 可以做到类似原生 ssh 的使用体验。反连利用部分可以通过监听一个端口，为不同协议的回连设置 Payload，收到请求后便会自动识别请求的协议，返回向 Payload，实现漏洞利用。反连检测功能提供了 TCP、DNSLog、ICMP 反连的检测，可用于一些命令执行的检测。



6.1.4 基于 Yaklang 热加载的插件技术

MITM 模块前后端使用流式传输，当前端勾选或取消新插件时，后端会受到前端请求，更新加载的插件，以达到热加载的目的。所以 MITM 模启动时，可随时勾选或取消被动扫描插件且立即生效，除此之外，用户可以自己编写或在已有插件基础上对插件进行修改调试。

解决方案和应用场景

· 客户案例 ·

xx银行：打造一体化测试平台，提升渗透测试质量管控

背景与挑战

xx银行是一家国内处于头部的金融机构，随着数字化服务的不断扩展，面临着日益复杂的网络安全挑战。传统的渗透测试流程碎片化、效率低下，且渗透测试和漏洞挖掘依赖于公司自身安全团队和第三方安全服务机构，参与的专业渗透测试人员数量少、整体水平参差不齐、使用的工具种类繁多，测试手法各式各样，最终的渗透测试结果越来越难以满足当下业务信息系统快速发展需求。渗透测试过程xx银行急需一个一体化的测试平台来统一管理渗透测试流程，提升测试的质量和效率。

Yakit解决方案

为了解决xx银行的挑战，Yakit 提供了一个全面的一体化渗透测试平台，集成了从任务规划、执行、数据管理到结果分析的全流程服务：

01 精细化任务管理

Yakit 的项目管理功能使XX银行能够将复杂的安全任务细分为可管理的单元，每个任务都有明确的目标和时间表，确保整个测试过程的有序和透明。

02 实时数据共享与协作

通过 Yakit 平台，XX银行的安全团队能够实时共享测试数据和进度，即时协作解决问题，显著提升了团队协作的效率。

03 安全数据管理

所有测试生成的数据都安全地上传并存储在 Yakit 服务端，采用先进的加密技术保证数据安全，同时支持深度分析和可视化展示，帮助团队深入理解安全状况。

04 高效率的工具集成

Yakit 集成了多种渗透测试工具和协作工具，确保XX银行的安全团队可以在一个平台上完成从数据收集、漏洞扫描到结果报告的所有工作，大幅提升了测试效率。

成果

采用 Yakit 后，xx银行的渗透测试流程得到了显著优化。团队协作更加紧密高效，数据管理更加安全可靠，测试结果的准确性和深度都有了大幅提升。最终，XX银行成功提升了渗透测试的整体质量和效率。

资质认证


NERCIS

CNAS

Yaklang 编程语言系统

软件检测报告


科技查新报告


软件产品登记
测试报告

样品名称 YAKIT网络安全半兵测试系统（简称：YAKIT工具）V2.0

开发单位 四维创智（北京）科技发展有限公司

委托单位 四维创智（北京）科技发展有限公司

报告日期 2022 年 04 月 19 日

北京软件产品质量检测检验中心


(19) 中华人民共和国国家知识产权局

(12) 发明专利申请


(19) 国家知识产权局

(12) 发明专利申请


(19) 国家知识产权局

(12) 发明专利


(19) 中华人民共和国国家知识产权局

(12) 发明专利申请

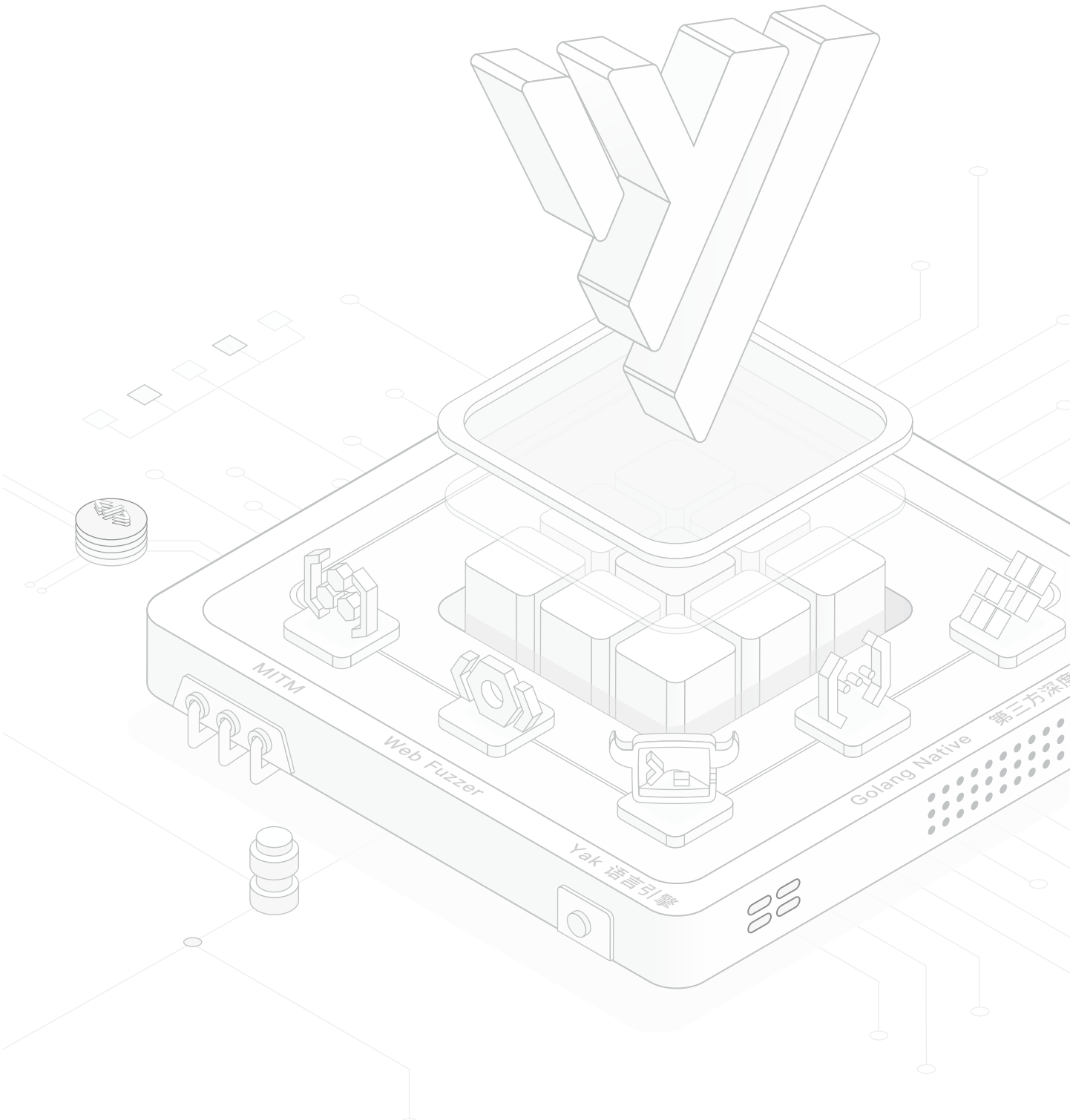
关于我们

万径安全作为全球最早提出网络安全生态共建的高新企业，首次提出使用 DSL(Domain-Specific Language) 模式向行业输出优秀的安全能力基座和技术解决方案。

公司多年来一直在 AI+安全、移动安全、内网安全等领域精耕细作，先后推出多款自研安全产品，并针对行业的不同安全需求，提供不同的行业解决方案。同时也向全行业提供安全检测、渗透测试、风险评估、网络安全培训等服务。与能源（电力）、军工、金融等领域有广泛合作，客户及合作伙伴来自政府、军工、通信、电力等各领域。

立足于攻防一线，以“图灵完备”的 Yaklang 语言作为底层能力，为客户提供攻防一体的产品与服务。未来，万径安全将致力于为安全行业输出专业的基础设施，帮助客户解决安全融合问题，为客户带来全新的安全体系建设思路，打造网络安全生态产品共建体系。

坚持“**做难而正确的事**”！凭借先进的技术理念，帮助企业进行安全能力建设，帮助用户应对变化多端的互联网安全威胁。**让世界更安全，让安全更简单！**


MITM
Web Fuzzer
Yak 语言引擎
Golang Native
第三方深度

Yakit

010-5945 6626（北京）
market@4dogs.cn
四川省成都市希顿国际广场B座1806A
https://www.yaklang.com