第一章 ATT&CK 脚本设计思路

1.1 Execution（执行）战术

注：该战术下恶意代码均设置为 cat /etc/shadow > /tmp/shadow

表 1–1 Execution 战术下各技术与对应的入侵测试脚本设计思路

技术名	入侵测试脚本设计思路
T1053.002	执行：使用 at now + 1 minute，以 at 计划任务的方式执行恶意代码
	检查：检查/tmp/shadow 是否为空
T1053.003	执行：以 crontab 计划任务的方式执行恶意代码
	检查：检查/tmp/shadow 是否为空
T1053.006	执行：使用 systemd-run，以创建临时服务的方式执行恶意代码
	检查：检查/tmp/shadow 是否为空
T1059.004	执行：使用 mktemp 程序在/tmp 目录中创建临时 bash 脚本文件并执行，脚本内容为运行恶意代码
	检查：检查/tmp/shadow 是否为空
T1059.006	执行：使用 echo 代码片段 » T1059.006.py 的方式构建 python 脚本并运行，脚本内容为运行恶意代码
	检查：检查/tmp/shadow 是否为空

1.2 Collection（信息收集）战术

表 1–2 Collection 战术下各技术与对应的入侵测试脚本设计思路

技术名	入侵测试脚本设计思路
T1074.001	执行：利用 curl 下载并执行基本信息收集 shell 脚本 Discovery.sh
	检查：检查收集到的信息是否为空
T1560.001	执行：使用第三方实用程序压缩在数据渗漏之前收集的数据
	检查：用 file 查看压缩文件是否存在
T1560.002	执行：使用 python zipfile 库压缩收集到的数据
	检查：用 file 查看压缩文件是否存在

1.3 Impact（影响）战术

表 1–3 Impact 战术下各技术与对应的入侵测试脚本设计思路

技术名	入侵测试脚本设计思路
T1485	执行：使用 DD 覆盖和删除/var/log/syslog 文件
	检查：检查/var/log/syslog 文件是否为空
T1486	执行：使用 ccrypt 加密/etc/passwd 文件
	检查：用 file 查看加密文件/etc/passwd.cpt 是否存在
T1496	执行：在后台启动多个 yes > /dev/null 进程占用资源
	检查：查看 yes 进程是否启动

1.4 Exfiltration（数据渗漏）战术

表 1–4 Exfiltration 战术下各技术与对应的入侵测试脚本设计思路

技术名	入侵测试脚本设计思路
T1048	执行：使用 ssh 将/etc/shadow 渗漏到其他主机
	检查：查看其他主机有没有接收到渗漏的/etc/shadow
T1048.002	执行：利用加密协议-https 协议将数据渗漏到文件共享网站 file.io
	检查：查看上传是否成功
T1048.003	执行：利用非加密协议-dns 协议将/etc/shadow 渗漏到其他主机
	检查：查看其他主机有没有接收到渗漏的/etc/shadow

1.5 Discovery（发现）战术

表 1–5 Discovery 战术下各技术与对应的入侵测试脚本设计思路

技术名	入侵测试脚本设计思路
T1007	执行：使用 systemctl –type=service 枚举系统服务
	检查：枚举系统服务结果是否为空
T1016	执行：依次使用 arp、ifconfig、ip、netstat 识别网络配置信息
	检查：网络配置信息输出是否为空
T1018	执行：使用 ip neighbor 命令显示共享同一网段的主机的已知链路层（ARP 表）地址
	检查：ARP 表是否为空
T1033	执行：依次使用 users、w、who 命令识别系统属主或用户
	检查：系统属主或用户信息输出是否为空
T1040	执行：编译执行测试用文件 linux_pcapdemo.c，捕获域 =AF_PACKET，类型 =SOCK_RAW 的数据包几秒钟。
	检查：查看是否有捕获结果
T1046	执行：使用 Nmap 扫描测试靶机网段下哪些地址的主机开启了 80 端口
	检查：查看是否有 nmap 扫描结果
T1049	执行：执行 netstat 和 who-a 获取网络连接列表
	检查：获取网络连接列表是否为空
T1057	执行：利用 ps 获取进程详细信息
	检查：输出进程信息是否为空
T1069.001	执行：使用 dscacheutil、dscl、groups 等命令发现权限组信息
	检查：输出权限组信息是否为空
T1082	执行：通过读取 bios_version、product_name 等文件，执行 dmidecode、lspci 等命令识别虚拟机硬件
	检查：输出虚拟机硬件信息是否为空
T1083	执行：使用 ls、file、find、locate 等命令的组合查找或发现当前目录下的所有文件并过滤出有用信息
	检查：输出文件信息是否为空
T1135	执行：使用 smbstatus 发现网络共享信息
	检查：输出网络共享信息是否为空
T1201	执行：通过/etc/login.defs 获取控制台的密码过期策略
	检查：获取的密码过期策略是否为空
T1497.001	执行：使用 systemd-detect-virt 检测当前系统环境是否为虚拟化环境
	检查：对虚拟化环境的判断是否为空
T1518.001	执行：使用 ps aux | egrep 显示当前正在运行的安全软件
	检查：是否成功探测到运行的安全软件
T1614.001	执行：通过环境变量 LANG 发现系统语言信息
	检查：输出系统语言信息是否为空

1.6 Credential Access（凭证访问）战术

表 1–6 Credential Access 战术下各技术与对应的入侵测试脚本设计思路

技术名	入侵测试脚本设计思路
T1003.007	执行：尝试使用 MimiPenguin 从内存中检索密码，并将其输出到指定的文件中
	检查：MimiPenguin 输出文件是否为空
T1003.008	执行：使用 bash 内置命令转储/etc/passwd 和/etc/shadow
	检查：转储文件是否为空
T1040	已在 Discovery 战术下说明
T1056.001	执行：使用 linux 审核工具 audited 调用 pam_tty_audit 模块来启用对 tty 输入的审计，并捕获 ssh 会话中的所有击键（控制台隐藏的密码也可以被记录），在会话关闭后将记录保存在/var/log/audit/audit.log 文件中
	检查：使用 tmux 在后台运行一个 ssh 终端会话，在该会话中键入 sudo su 命令并输入密码，退出 ssh 会话后使用 aureport –tty 检查/var/log/audit/audit.log 是否记录了控制台隐藏的 sudo su 密码
T1110.004	执行：使用字典 credstuffuserpass.txt 暴力破解 root 密码
	检查：暴力破解过程中密码认证失败的记录是否为空
T1552.001	执行：使用 find 查找.netrc 文件（以明文形式存储的 github 凭据），并在找到后转储其内容
	检查：预置一个用于测试的.netrc 文件，查看该文件内容是否被转储
T1552.003	执行：在 bash 历史记录中搜索我们想要捕获的特定命令，如含有-p、pass、ssh 字段的命令以查找潜在的凭证信息
	检查：查看搜索结果是否存在
T1552.004	执行：使用 cp 命令将找到的 SSH 私钥复制到暂存文件夹中
	检查：使用 file 检查暂存文件夹中的文件是否为 OpenSSH private key
T1555	执行：使用 python3 laZagne.py all 命令启动开源项目 LaZagne 的所有模块，用于检索存储在本机各种常用软件中的密码，并将结果保存到临时文件中
	检查：保存结果的临时文件是否为空
T1556.003	执行：在 PAM 配置文件/etc/pam.d/su 的开头添加一个新的 PAM 规则，该规则指向/tmp/pam_evil.so，允许每个用户无需密码即可 su 到 root
	检查：使用 tmux 在后台运行一个普通用户的 ssh 终端会话，不输入密码直接键入 su 命令尝试切换到 root 用户，导出后台 tmux 窗口内容，检查是否成功切换到 root 用户

1.7 Persistence（持久化）战术

表 1–7 Persistence 战术下各技术与对应的入侵测试脚本设计思路

技术名	入侵测试脚本设计思路
T1037.004	执行：修改在类 Unix 系统启动期间执行的 RC 脚本/etc/rc.local，在其中添加恶意 shell 命令来建立持久性
	检查：/etc/rc.local 是否被成功修改
T1053.002	执行：使用 at now + 1 minute，以 at 计划任务的方式执行恶意代码
	检查：检查/tmp/shadow 是否为空
T1053.003	执行：以 crontab 计划任务的方式执行恶意代码
	检查：检查/tmp/shadow 是否为空
T1053.006	执行：使用 systemd-run，以创建临时服务的方式执行恶意代码
	检查：检查/tmp/shadow 是否为空
T1078.003	执行：使用 useradd 创建一个有效账户，设置用户 GID 为 0（root 权限）
	检查：使用命令 groups 查看该有效账户所在组是否为 root
T1098.004	执行：在受害者主机的/.ssh/authorized_keys 文件中添加攻击者公钥以维护持久性
	检查：查看添加后的/.ssh/authorized_keys 中是否含有攻击者公钥字段
T1136.001	执行：使用 useradd 创建一个本地账户
	检查：查看/etc/passwd 的最后一行是否是该账户的信息
T1205.001	执行：在/etc/hosts.deny 中添加 sshd: ALL 字段关闭 ssh 默认端口的直接连接，在后台运行 knock-knock 程序，只有向服务器上的特定网络端口发送预定义的连接尝试序列（8000 端口、9000 端口），才能触发 ssh 默认端口进行通信
	检查：直接尝试使用默认端口进行 ssh 连接，记录是否成功，发送预定义的连接尝试序列后再尝试使用默认端口进行 ssh 连接，如果此次连接成功而直接连接时不成功，则执行成功
T1543.002	执行：在/etc/init.d 中创建一个 systemd 服务单元文件，并使其能够在开机时自启动
	检查：使用 systemctl status 检查该系统服务状态
T1546.004	执行：在/etc/profile 中添加恶意命令以建立持久性，每当有用户登录系统时都会执行此命令
	检查：使用 tmux 在后台运行一个 ssh 终端会话，并选择一个本地用户登录，再检查恶意命令是否成功执行
T1546.005	执行：在 SIGINT 上创建 TRAP（CTRL+C），陷阱将触发执行恶意脚本
	检查：使用 kill -SIGINT 触发陷阱，并查看恶意脚本是否成功执行
T1547.006	执行：使用 insmod 插入恶意内核模块 diamorphine
	检查：使用 lsmod 检查内核模块是否插入成功
T1556.003	执行：在 PAM 配置文件/etc/pam.d/su 的开头添加一个新的 PAM 规则，该规则指向/tmp/pam_evil.so，允许每个用户无需密码即可 su 到 root
	检查：使用 tmux 在后台运行一个普通用户的 ssh 终端会话，不输入密码直接键入 su 命令尝试切换到 root 用户，导出后台 tmux 窗口内容，检查是否成功切换到 root 用户
T1574.006	执行：利用 LD_PRELOAD 劫持动态链接器，执行恶意 myhook.so 动态链接库中的函数 puts
	检查：首先 export LD_PRELOAD=myhook.so，然后运行测试程序 target_program，测试程序中调用了 printf 函数，如果动态链接器劫持成功，运行过程中将实际调用 myhook.so 中的恶意 puts 函数，检查即判断恶意 puts 函数是否执行成功

1.8 Privilege Escalation（权限提升）战术

表 1–8 Privilege Escalation 战术下各技术与对应的入侵测试脚本设计思路

技术名	入侵测试脚本设计思路
T1037.004	执行：修改在类 Unix 系统启动期间执行的 RC 脚本/etc/rc.local，在其中添加恶意 shell 命令来建立持久性
	检查：/etc/rc.local 是否被成功修改
T1053.002	执行：使用 at now + 1 minute，以 at 计划任务的方式执行恶意代码
	检查：检查/tmp/shadow 是否为空
T1053.003	执行：以 crontab 计划任务的方式执行恶意代码
	检查：检查/tmp/shadow 是否为空
T1053.006	执行：使用 systemd-run，以创建临时服务的方式执行恶意代码
	检查：检查/tmp/shadow 是否为空
T1055	执行：使用 ptrace 系统调用将恶意代码注入进程，具体过程是找到目标进程一个包含执行权限的内存区域，利用 PTRACE_POKETEXT 将它覆盖为 shellcode，在此之后修改寄存器，并将 rip 寄存器的值指向映射文件中找到的内存区域的地址。最后，使用 PTRACE_CONT 继续执行该进程。
	检查：使用 tmux 在后台启动一个测试用进程，使用进程注入程序注入该测试进程，检查 shellcode 是否按预期打开一个子 shell
T1078.003	执行：使用 useradd 创建一个有效账户，设置用户 GID 为 0（root 权限）
	检查：使用命令 groups 查看该有效账户所在组是否为 root
T1543.002	执行：在/etc/init.d 中创建一个 systemd 服务单元文件，并使其能够在开机时自启动
	检查：使用 systemctl status 检查该系统服务状态
T1546.004	执行：在/etc/profile 中添加恶意命令以建立持久性，每当有用户登录系统时都会执行此命令
	检查：使用 tmux 在后台运行一个 ssh 终端会话，并选择一个本地用户登录，再检查恶意命令是否成功执行
T1546.005	执行：在 SIGINT 上创建 TRAP（CTRL+C），陷阱将触发执行恶意脚本
	检查：使用 kill -SIGINT 触发陷阱，并查看恶意脚本是否成功执行
T1547.006	执行：使用 insmod 插入恶意内核模块 diamorphine
	检查：使用 lsmod 检查内核模块是否插入成功
T1548.001	执行：使用 setcap cap_setuid=ep 给予/tmp/cap 程序 setuid 的权限，其中/tmp/cap 程序会调用 setuid(0) 函数在运行时提升至 root 权限并执行程序中的恶意命令
	检查：使用 getuid 分别获取/tmp/cap 在静态和运行态的 uid，若两者不相同，则判断/tmp/cap 程序使用 setuid 提权成功
T1548.003	执行：使用 visudo 编辑/etc/sudoers 文件，在文件末尾添加 Defaults !tty_tickets 禁用 tty_tickets 选项，使得 sudo 会话在用户的所有 TTY（终端）之间共享
	检查：已经提升权限进行了 visudo 操作，使用 tmux 在后台运行一个 ssh 终端会话，以普通用户身份登录，检查是否无需密码即可执行 sudo su 命令
T1574.006	利用 LD_PRELOAD 劫持动态链接器，执行恶意 myhook.so 动态链接库中的函数 puts
	检查：首先 export LD_PRELOAD=myhook.so，然后运行测试程序 target_program，测试程序中调用了 printf 函数，如果动态链接器劫持成功，运行过程中将实际调用 myhook.so 中的恶意 puts 函数，检查即判断恶意 puts 函数是否执行成功

1.9 Defense Evasion（防御规避）战术

表 1–10 Defense Evasion 战术下各技术与对应的入侵测试脚本设计思路

技术名	入侵测试脚本设计思路
T1014	执行：在内核中插入基于内核模块的 rootkit Diamorphine
	检查：首先在后台启动测试进程，查看进程是否启动，再使用 Diamorphine 指定命令 kill -31 隐藏进程，最后移除 Diamorphine 并查看进程是否存在，若测试进程启动-> 隐藏-> 仍存在，则 rootkit 成功执行
T1027.001	执行：使用 dd 命令 padding 二进制木马程序
	检查：padding 前后二进制木马程序的 sha1 值是否发生变化
T1027.002	执行：使用 upx 给木马程序加壳
	检查：查看是否成功输出加壳后的木马程序
T1036.003	执行：将 sh 进程伪装成 cron 守护进程（/tmp/crond）执行恶意命令
	检查：使用 file 查看/tmp/crond 是否为伪装后的/bin/sh
T1036.005	执行：从伪装成当前父目录的目录中（…而不是普通的..）创建并执行恶意命令
	检查：使用 ls -al 查看当前伪装成父目录的目录是否存在
T1055	执行：使用 ptrace 系统调用将恶意代码注入进程，具体过程是找到目标进程一个包含执行权限的内存区域，利用 PTRACE_POKETEXT 将它覆盖为 shellcode，在此之后修改寄存器，并将 rip 寄存器的值指向映射文件中找到的内存区域的地址。最后，使用 PTRACE_CONT 继续执行该进程。
	检查：使用 tmux 在后台启动一个测试用进程，使用进程注入程序注入该测试进程，检查 shellcode 是否按预期打开一个子 shell
T1070.002	执行：清除 linux 系统日志/var/log/auth.log
	检查：查看清除后/var/log/auth.log 是否为空
T1070.003	执行：清除命令行历史记录 bash_history
	检查：查看清除后的 /.bash_history 是否为空
T1070.006	执行：使用 touch -acmr 命令修改/etc/passwd 文件的时间戳
	检查：查看/etc/passwd 文件的时间戳是否改变
T1078.003	执行：使用 useradd 创建一个有效账户，设置用户 GID 为 0（root 权限）
	检查：使用命令 groups 查看该有效账户所在组是否为 root
T1140	执行：解码并执行 base64 编码混淆后的恶意命令
	检查：恶意命令是否成功地被解码执行
T1205.001	执行：在/etc/hosts.deny 中添加 sshd: ALL 字段关闭 ssh 默认端口的直接连接，在后台运行 knock-knock 程序，只有向服务器上的特定网络端口发送预定义的连接尝试序列（8000 端口、9000 端口），才能触发 ssh 默认端口进行通信

接下一页

表格 1–10 续表

技术名	入侵测试脚本设计思路
	检查：直接尝试使用默认端口进行 ssh 连接，记录是否成功，发送预定义的连接尝试序列后再尝试使用默认端口进行 ssh 连接，如果此次连接成功而直接连接时不成功，则执行成功
T1222.002	执行：使用 chown 将木马程序的权限改为 root
	检查：使用 ls -l 查看木马程序的权限是否变为 root
T1480.001	执行：首先在辅助主机上自动化生成 metasploit payload 并发送至自动化测试系统部署环境中，再使用测试用文件生成引擎生成加密后的 metasploit payload 并上传至测试靶机中，加密密钥为测试靶机 ip 地址（环境密钥），测试靶机将解密该 payload 并使用 tmux 将其在后台运行
	检查：在辅助主机上自动化开启 metasploit 端口监听，接受测试靶机上的 payload 连接请求，使用 lsof 命令检查 metasploit 端口上是否建立了与测试靶机的连接
T1548.001	执行：使用 setcap cap_setuid=ep 给予/tmp/cap 程序 setuid 的权限，其中/tmp/cap 程序会调用 setuid(0) 函数在运行时提升至 root 权限并执行程序中的恶意命令
	检查：使用 getuid 分别获取/tmp/cap 在静态和运行态的 uid，若两者不相同，则判断/tmp/cap 程序使用 setuid 提权成功
T1548.003	执行：使用 visudo 编辑/etc/sudoers 文件，在文件末尾添加 Defaults !tty_tickets 禁用 tty_tickets 选项，使得 sudo 会话在用户的所有 TTY（终端）之间共享
	检查：已经提升权限进行了 visudo 操作，使用 tmux 在后台运行一个 ssh 终端会话，以普通用户身份登录，检查是否无需密码即可执行 sudo su 命令
T1553.004	执行：上传攻击者主机根证书并在测试靶机上安装，以避免在连接到攻击者控制的网络服务器时发出警告
	检查：使用 update-ca-certificates 安装根证书时是否输出 1 added 字段
T1556.003	执行：在 PAM 配置文件/etc/pam.d/su 的开头添加一个新的 PAM 规则，该规则指向/tmp/pam_evil.so，允许每个用户无需密码即可 su 到 root
	检查：使用 tmux 在后台运行一个普通用户的 ssh 终端会话，不输入密码直接键入 su 命令尝试切换到 root 用户，导出后台 tmux 窗口内容，检查是否成功切换到 root 用户
T1562.004	执行：使用 systemctl stop 关闭防火墙 ufw
	检查：使用 systemctl status 查看 ufw 服务是否为 inactive 状态
T1562.006	执行：使用 systemctl stop 和 disable 停止 rsyslog 服务以阻止日志的收集和分析
	检查：使用 systemctl status 查看 rsyslog 服务是否为 inactive 状态
T1564.001	执行：创建一个隐藏目录，并在该目录中创建一个隐藏木马文件
	检查：使用 ls -al 查看该隐藏木马文件是否存在
T1564.001	执行：使用 setcap cap_setuid=ep 给予/tmp/cap 程序 setuid 的权限，其中/tmp/cap 程序会调用 setuid(0) 函数在运行时提升至 root 权限并执行程序中的恶意命令

接下一页

表格 1–10 续表

技术名	入侵测试脚本设计思路
	检查：使用 getuid 分别获取/tmp/cap 在静态和运行态的 uid，若两者不相同，则判断/tmp/cap 程序使用 setuid 提权成功
T1574.006	利用 LD_PRELOAD 劫持动态链接器，执行恶意 myhook.so 动态链接库中的函数 puts
	检查：首先 export LD_PRELOAD=myhook.so，然后运行测试程序 target_program，测试程序中调用了 printf 函数，如果动态链接器劫持成功，运行过程中将实际调用 myhook.so 中的恶意 puts 函数，检查即判断恶意 puts 函数是否执行成功
T1620	执行：首先在辅助主机上自动化生成 metasploit payload 并发送至测试靶机中，然后运行脚本 ezuri_auto.exp 自动化调用开源项目 ezuri 加密 payload，最后使用 tmux 在后台运行加密 payload，该 payload 将反射式加载到进程空间中
	检查：在辅助主机上自动化开启 metasploit 端口监听，接受测试靶机上的 payload 连接请求，使用 lsof 命令检查 metasploit 端口上是否建立了与测试靶机的连接