拟态防御基准功能实验

从拟态防御理论预期效果出发的测试称为基准功能实验，也可称之为从拟态防御定义出发的效果测试。拟态防御通常分为两种测试，一种是传统的安全性测试，主要测试漏洞呈现情况和攻击可达性情况；另一种是注入测试例实验。也称为“白盒插桩”实验，要求在不依赖实验者经验和技巧的情况下，依据产品测试规范用例就能准确甄别被测对象是否具备基本的拟态防御功能，以及应对攻击逃逸的相关性能。本节将重点讨论白盒插桩实验问题。

前提与约定

本实验是以被测目标给定拟态构造的威胁分析和安全性设计为基本依据，以拟态括号提供的输入通道、语法语义、规则方法等作为归一化的攻击表面，实验目标对象在“白盒条件下”，通过拟态括号表面攻击通道注入给定的测试例或测试例集合，检定其安全性、可靠性、可用性设计是否满足声称的可量化指标。需要指出的是，拟态构造内通常包含两大类功能和性能，白盒注入测试方法只用于检定拟态构造的内生安全功能和相关性能，不应当造成被测对象其他服务功能和性能的不可恢复损坏，也不包括对拟态构造外的安全性、可靠性、可用性进行测试的内容。

实验约定：

(1)一个注入测试例，由嵌入被测对象源程序的测试接口代码和通过攻击可达路径注入的测试代码两部分组成。

(2) 凡是嵌入被测对象执行体或运行场景中的测试接口代码功能相同，包括通过攻击表面通道与合规方法接收指令代码和数据包或更新测试内容的功能，以及激活或关闭测试例的功能。

(3) 通过测试接口注入的测试代码属于内存驻留型的可执行代码，且不应当导致被测对象功能不可恢复的损坏。

(4) 测试代码应当具有影响所在执行体或运行场景输出矢量内容或控制其输出的能力。

(5) 凡是测试代码功能两两不相同的测试例称为“差模测试例”，两两相同的称为“共模测试例”。

(6) 所有执行体或运行场景 F 内原则上都还可以设置测试接口，但是同时驻留测

试代码的数量 f 需满足 n≤f≤F/2，n 为拟态括号当前服务场景的余度数。

(7) 实验阶段需开启系统管理界面以便整个实验过程可观察。

测试例构造：

由于被测对象的异构执行体或运行场景软硬件支撑环境常常只有可执行目标代码甚至只是物理器件，测试例不仅构造困难而且往往不可注入，即使能够植入，功能验证也颇具挑战性。所以，在应用程序源代码层面上构建测试例接口、设计调用功能是可能的选择。测试接口应设计成具有通过拟态括号攻击表面输入通道及合规方法请求激活的“后门功能”，能够借助攻击表面接收上传的测试代码，并可通过应用程序实施在线控制注入测试代码的执行。当然，有条件情况下也不排除在其他层面（例如操作系统层面）设置测试接口及相关功能的做法。需要指出的是，除了测试接口代码外，上传的目标对象测试代码应当是内存驻留型的可执行代码，以便灵活的定义或改变测试功能，增强注入测试的完备性。

需要说明的是，在应用层设置实验接口以及通过攻击表面注入测试代码还可达成另外二个目的：一是实验效果具有置信度。事实上，无论基于什么层面漏洞后门的攻击，只要能精确控制对象执行体或运行场景输出矢量的表达，也就是攻击所能达成的最高目标了。因而只要保证应用软件按约定的要求调用测试接口功能，激活注入的测试代码，就能直接或间接的威胁系统设定的安全目标；二是可通过系统管理界面观察执行体或运行场景异构资源配置情况。变化注入测试代码可以验证运行环境的 CPU 类型、OS 版本或其他的相关环境信息等，并可为测试代码确定目标执行环境。

差模测试例注入实验

前提：假如一个被测目标功能函数满足 I【p1,p2,p3,…,pn】0，其中，所有功能 pi 相同，即 p1=p2=…=pn。但是所有 pi 的实现算法都不相同，即 pc1≠pc2≠…≠pci。如果存在一个测试例 e1 可以使 p1 产生正常响应序列之外的输出矢量 s1。以此类推，测试例 e2 e3 ei 可以使 p2 p3 pi 产生输出矢量 s2 s3 si 且 s1≠s2≠s3≠si。那么，按照拟态防御效果定义，将相互之间没有配合关系的测试例 e1 e2 e3 ei 通过攻击表面分别注入 p1 p2 p3 pi 执行体或防御场景，在 I【P】0 的拟态界上不应该出现任何 si。被测对象的拟态防御功能应该能够清晰的表明，除了不能自动恢复的停机-制瘫事件之外，从机理上说，只要是不存在

协同关系的“独狼式攻击”差模测试例，且该测试例一旦产生裁决器可感知的动作，并能在下一个测试例激活前可被策略性的清除或移除，对差模测试例的抑制效果应能达到 100%。

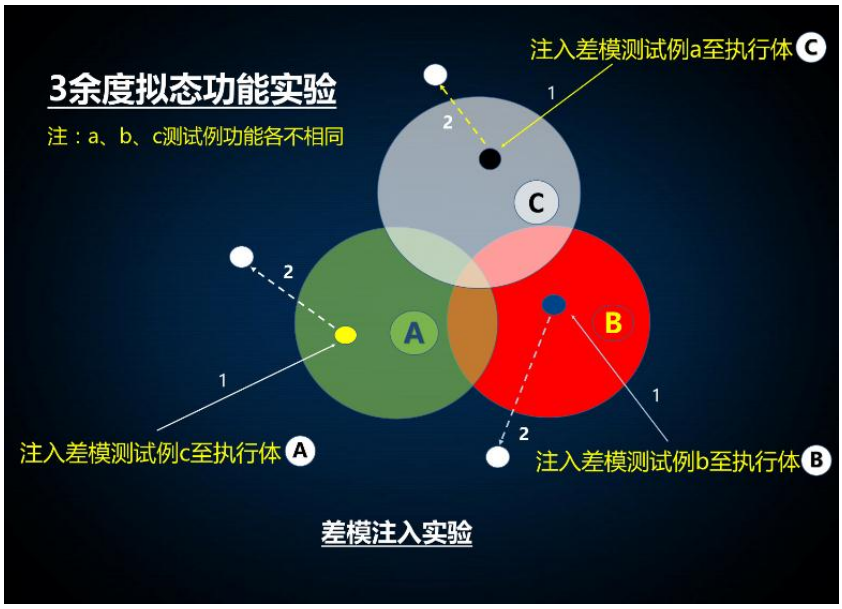

图 1　差模测试例注入实验

图 1 是一个 3 余度拟态功能的实验场景，所有落在 A、B、C 三个执行体区域内的差模（功能各不相同的）测试例都应该设计成能被拟态裁决环节感知的场景。按照拟态构造的反馈控制策略和迭代收敛后向验证机制，该实验的另一个预期结果是：问题执行体或运行场景本身要么被替换或重构重组，要么附着有注入测试例的执行体或运行场景被清洗复位或重启。相关实验场景、操作过程及响应结果应该是可观察的。

时间协同差模测试例实验

该实验的目的是，在差模测试例注入实验的基础上，将 f 个差模测试例均匀分布在拟态构造内 n 个执行体的 m 种防御场景上（m≥n），差模测试例数量 f 与 m 的占比 F=f/m * 100%，满足 34%≤F≤50%。实验者通过攻击表面连续地发出 f 个测试例的激活指令，观察目标对象在服务功能和性能上会有什么样的表现。重点关注， 1.实验过程中是否会产生差模逃逸； 2.实验对象在此过程中，其服务功能和性能的劣化程度； 3.反复地发送 f 个测试例的激活指令，测量实验对象的问题规

避机制需要多长时间才能使相关防御场景不再被调用。

实验方案：①抽样测试执行体防御场景恢复时间以便确定 f 个测试例的连续激活策略；②在占比 F 为 34%、40%、50%的情况下，确定 f 个测试例在 m 种场景中的部署策略（例如均匀分布）；③在实验过程中，测量目标对象服务功能和性能，验证可靠性与可用性设计指标，重点是测量同一组时间协同差模测试例从“攻击可达到攻击不可达”的时间，检验目标对象问题场景规避机制的有效性；④其他方法和操作参照差模测试例注入实验。需要指出的是，F 的相关取值均应高于经典异构冗余构造（DRS）故障容忍极限 f≤(n-1)/2。例如，当 n=m=3 时，DRS 允许 f≤1,对应的占比 F≤33.3%。而时间协同差模测试例的实验取值,F=34%、40%、50%则分别对应 DRS 在 n=3、5 以及冗余度无限大时的最坏情况。

n-1 模测试例注入实验

该实验是为了检验拟态界内一旦产生 n-1 共模有感逃逸(n 为当前服务集冗余度)，拟态构造应当具有从逃逸状态下恢复的能力，证明拟态构造即使出现共模逃逸事件仍然具有预期的概率属性。尽管 n-1 共模逃逸在安全性量化设计中属于小概率或极小概率事件，但是目标对象工程实现中如果不能达成共模逃逸的解脱功能，则发生首次逃逸事件后，该攻击经验将可复制且不再属于概率问题了。n-1 共模逃逸实验不仅要验证拟态功能的正确性还要测试恢复过程的持续时间，后者可量化评价抗共模逃逸性能高低。同理，整个实验过程和效果应当是可观察的。

n-1 模测试例注入实验规定，如果拟态构造当前服务集内的执行体或防御场景数量为 n，假定存在一个测试例 ti 可分别注入到 n-1 个执行体或防御场景内，并能以某种方式通过攻击表面激活且可产生 n-1 个相同输出矢量 si。按照拟态防御定义，在 I【P】O 拟态界上，此时虽然能出现大概率的共模逃逸现象，但是裁决器却可感知到多模输出矢量间存在不一致状态。换言之，n-1 共模逃逸属于可感知的逃逸。于是，反馈控制环节会根据预先设计的后向验证策略（为了区别差模场景和 n-1 共模场景间的不同需要引入辅助判决策略），决定怎样通过渐进或迭代收敛方式改变拟态括号内的防御场景，使之退化成图 1 的差模实验形态，并最终被移除或清除。

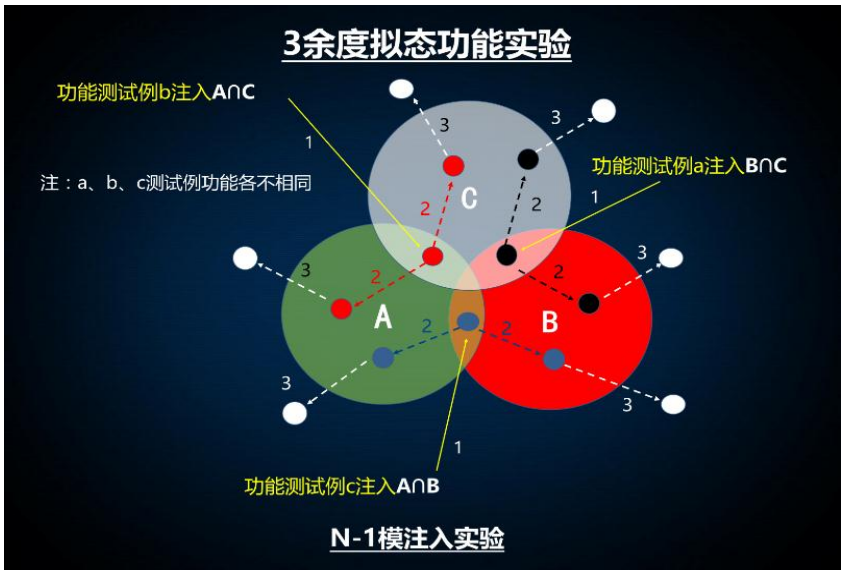

图 2　n　1 模测试例注入实验

图 2 是一个三余度拟态构造抗 n–1 共模逃逸场景示意。显示了共模测试例 a、b、c 分别注入到执行体（防御场景）A 和 B 的交集区、A 和 C 的交集区、B 和 C 的交集区的情况，以及这些测试例从执行体或运行场景内移除或清除的路径（不同颜色虚线所示）。假定 n–1 共模测试例能够注入任意 2 个执行体或运行场景的功能交集内，且可通过攻击表面输入通道合规的激励消息使之产生相同的输出矢量。按照拟态防御定义，裁决器此时能感知到多模输出矢量中存在不一致的情况，但不能直接甄别出问题执行体或运行场景，需要通过后向验证机制区分并消除注入的测试例影响：首先，对输出矢量不一致的执行体或运行场景作清除重启或替换操作，如果裁决器状态仍未改变，则在输出矢量相同的执行体或运行场景中按照某种策略选择一个对象实施前述操作。其次，观察裁决器状态，如果发生状态逆转则就会退化为图 1 所示的差模状态，此时只要再对服务集内未更新过的执行体重复上述操作，注入的测试例将会被从当前服务集内清除或移出。因此，按照拟态防御定义，即便 n–1 共模攻击成功，其逃逸状态也不具有稳定鲁棒性。

需要指出的是，即使 n–1 模测试例在攻击表面意义上能被同时激活，由于拟态构造固有机制原因可能会被识别为差模攻击，因而不能保证 n–1 模逃逸状态的稳定呈现，需要反复注入和激励直到出现实验期望的 n–1 模逃逸状态为止。整个实验过程应当是可观察的。

由此不难看出，拟态构造的系统具有“即使逃逸成功也无法稳定维持”的特性，虽然不可能替代传统信息安全手段的全部作用，但是却具有后者所不具备的柔韧性或弹性功能。尤其是对企图利用差模攻击获取敏感信息或者破坏信息完整性的攻击者而言，拟态防御可能比一般的加密措施更具有比较优势，因为拟态防御不属于可计算问题，所以也不会陷入“一旦被暴力破解就全线崩溃”的困境。

n 模测试例注入实验

按照拟态构造定义，拟态构造存在极小概率的 n 模逃逸情况且可量化设计。但与 n–1 模逃逸情况不同，从机理上说拟态构造对于 n 模逃逸是不能感知的，需要通过外部或内部的某种策略，扰动反馈控制环路，改变拟态括号内的当前运行环境，使之转变为 n–1 模的可感知形态，并进入相应的解脱或恢复进程，确保 n 模逃逸即使发生，仍属于概率性事件。这一功能需要通过注入测试例的白盒方式进行检验，并根据设定的环路扰动策略，测量验证解脱恢复过程的标称时间。

如图 3 所示，假定注入的 n 模测试例如果能在 3 个执行体 A∩B∩C 的功能交集内产生一致的输出矢量，则拟态裁决环节理论上应当无感。但按照拟态防御定义，即使裁决器未发现输出矢量异常，当前服务集内的执行体或防御场景也可能因为外部控制指令发生强制的、非确定性的替换或清洗重启操作，这意味着 n 模注入测试条件下的逃逸状态一定是不稳定的，当执行体或防御场景自身具有清洗重启或可重构重组功能时，n 模无感逃逸的情况应当随着恢复过程的推移自动转变为 n–1 模有感逃逸状态，并最终退化为差模情景并被无感移除。换言之，通过攻击表面注入服务集内的 n 模测试例会因为宿主执行体或运行场景被外部指令策略性的清洗重启或重构重组操作而移除。假如后向验证策略约定，执行体 C 作例行清洗后再重新加入当前服务集(也可直接重构或替换执行体 C)，当 C(或替换执行体)输出矢量与 A 和 B 仍旧不同时，反馈环路则会优先清洗或替换运行时间最长的执行体(譬如 A)。

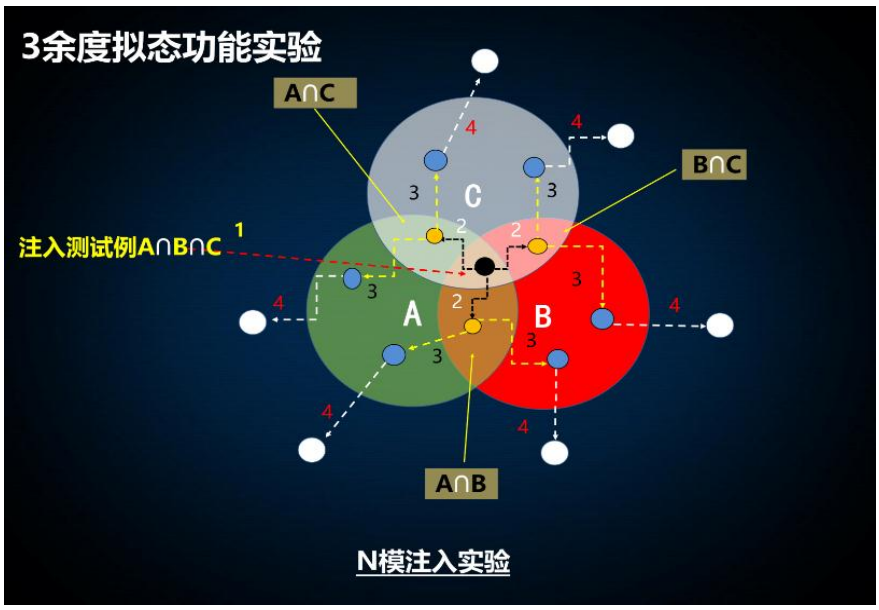

图 3　n 模测试例注入实验

于是，当裁决器发现 AC 输出矢量不同于 B 的情况时，表明前述场景中出现过逃逸现象（即图 2 的场景），再对 B 实施上述操作直至裁决器不再有感。按上图虚线所示迁移轨迹，n 模测试例场景会退变为 n–1 模测试场景，最后会退变到差模测试场景直至全部移除。由此可见，在拟态防御环境内即使攻击者有能力实施 n 模或跨域协同攻击构成一时的逃逸状态，但是从机理上因为无法获得保持稳定逃逸的能力，从而使攻击成果的鲁棒性利用成为难以克服的挑战。需要指出的是，即使 n 模测试例在攻击表面意义上能被同时激活，由于拟态构造固有机制原因可能会被识别为差模或 n–1 模攻击，因而不能保证 n 模逃逸状态的稳定呈现，需要反复注入和激励直到出现实验期望的 n 模逃逸状态为止。

反馈控制环路的注入测试

拟态反馈控制环路包括输入分配与代理、输出裁决与代理及反馈控制三个部分。按照拟态防御定义，反馈环路与执行体和拟态括号的输入/输出通道之间只存在“单向联系机制”，且允许反馈控制环路内存在漏洞（事实上也很难杜绝）但不存在恶意代码之前提条件（低复杂度时，工程实践上通常可以满足），如图 4 所示。

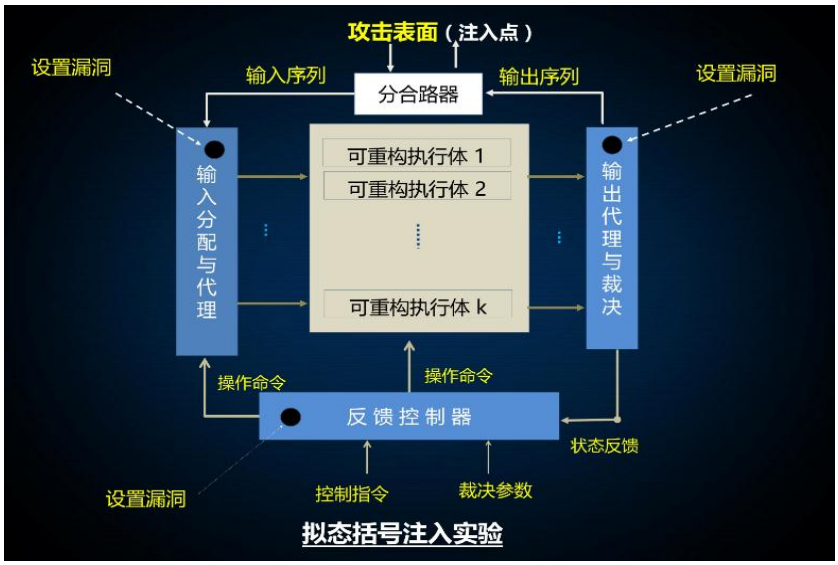

图 4 拟态括号单向机制验证

按照严格的单向联系机理，执行体内的病毒木马等应当无法利用反馈环路中的漏洞注入攻击代码或实现隧道穿越。同理，拟态括号功能对于外部攻击者而言通常是“透明的”，即无论是输入通道上的输入分配与代理环节，还是输出通道的输出裁决与代理环节，或者仅用于内部策略调度的反馈控制环节都应当是“不可见的”。因为理论上，拟态括号既不解析输入激励序列内容也不关心多模输出矢量的语法和语义，所以应当具有威胁目标的不可达性。然而，工程实现上，有时不得不关心输入激励序列的隔离导入和负载均衡等功能而必须引入代理机制，也常常因为拟态裁决无法回避异构冗余执行体多模输出矢量某些可选项值域、通信序列号等不确定性，甚至是计算精度上的差异而使之必须对其语法、语义等作非透明性的预处理。因此，随着括号部件功能复杂性或智能处理能力的增强，未知漏洞存在的可能性也随之增大。但是，如果在设计上能保证拟态括号部件即使存在漏洞也无法被利用，则可以认为满足基本安全性要求。 “白盒验证”方式，就是要检验能否在相关部件上设置“漏洞测试接口”，能否利用“漏洞接口”上传“测试代码”，以及能否利用上传的测试代码实现拟态逃逸。不过，这种方式有时会因为目标对象的实现技术形态而不能实施，例如，用无源光分配器或布线逻辑器件作为输入代理部件时，就无法设置“漏洞测试接口”以及“上传并执行测

试代码”。倘诺如此，可以认为输入代理部件具有“漏洞不可利用”的固有属性。同理，如果无法借助从拟态括号规定的输入通道，途径输入代理部件和异构执行体输出，上传“测试代码”到输出代理部件或裁决器上的“测试漏洞”并执行之，则可以认为输出代理部件或裁决器也具有“漏洞不可利用”的属性。显然，这一属性并非是这些部件的固有属性，很大程度上是由于拟态构造效应带来的。需要强调指出的是，任何测试例即使能从拟态界外注入拟态括号相关部件的“测试漏洞”，但只要不能实现拟态逃逸之目的，仍可认为该括号满足“漏洞不可利用”的安全性假设。总之，工程实践上常常需要综合应用相关的安全技术，以确保拟态括号上即使存在设计漏洞也不能使之成为既定安全目标的“防御短板”。

性能度量。

在上述各项测试中要度量测试例从激活到失能的时间，以此来衡量反馈控制环路的收敛速度、作动性能或场景规避精准度是否达到系统设计指标的要求。

需要强调指出的是，不论何种形式的共模测试例，拟态防御从机理上都应该能从逃逸状态下自行解脱出来。也许不同设计方案导致的解脱时间有所不同，在实现代价方面也可能有所差异，但共模逃逸的解脱功能则是不可或缺的，因为“即使实现攻击逃逸，也不无法稳定维持”是拟态防御追求的高可用性目标。同理，通过差模测试例的设置也可以检验目标对象的可靠性设计。作者声明，在拟态构造注入测试例中设置机密性检查功能也是可行的，但必须排除基于“声光电磁热”等物理效应或与其他非拟态构造任务共享资源导致的侧信道攻击情况。

Mimic Defense Benchmark Function Experiment

The test based on the expected effect of the mimic defense theory is called benchmark function experiment, also called definition based effect test of mimic defense. There are usually two types of tests for mimic defense, one is the traditional security test, which mainly tests the vulnerability display and attack reachability, the other is the test case injection experiment, also called "white box instrumentation" experiment, which requires to accurately identify whether the tested target has the basic mimic defense function and whether it has the capabilities to deal with attack escape according to examples for product test standards, without relying on the experimenter's experience and skills. This section will focus on the white box instrumentation experiment.

Prerequisite and arrangement

On the basis of the threat analysis and security design of the given mimic structure of a tested object, with the input path, syntax and semantics, and rules and methods in the mimic brackets as the normalized attack surface, the experiment examines if the design of security, reliability and availability of the tested object can meet the alleged quantifiable indicators in the "white box testing" by injecting some given test cases or test case sets through the attack path on the mimic bracket surface. It should be pointed out that the mimic structure usually contains two big types of functions and performance. The white box injection approach can only be used to check the endogenous security function and relevant performance of the mimic structure, without causing unrecoverable damage to other functions or performance of it. The test of security, reliability and availability outside the mimic structure is not included in the experiment.

Experiment arrangement:

(1) One agreed test case consists of two parts, namely, the test interface embedded in the target source program and the test code injected through the attack reachable path.

(2) Any code functions of test interfaces embedded in the tested target executor or operational scenario are the same, including the functions of receiving instruction code through the attack surface path and compliant method, or of updating the test content.

(3) The test codes injected through the test interface are memory resident executable codes, and should not lead to unrecoverable damage to the tested target function.

(4) The test codes shall have the ability to influence the output vector content of the resident executor or operational scenario, or to control its output ability.

(5) The test cases whose every two test code functions are different are called “differential mode test cases”, and those whose every two test code functions are the same are called “common mode test cases”.

(6) All the executors or operational scenario F can, in principle, have test interfaces, but at the same time, the number f of resident test codes shall satisfy $n \leqslant f \leqslant F/2$, and n is the redundancy of the current service scenario in the mimic brackets.

(7) During the experiment stage, open the system management interface to observe the entire experiment process.

Structure of test cases

Usually, the hardware and software supporting environment of the test target’s heterogeneous executor or operational scenarios has only the executable target codes or even physical devices, so the test case is not only difficult to be constructed, but often cannot be injected. Even if the test case can be implanted, its function verification is rather challenging. Therefore, it’s a possible choice to construct the test case

interface and design the invoking function at the application source code level. The test interface should be designed to have a “backdoor function” that is activated by input channels and compliance method of the attack surface in the mimic brackets, and can receive the uploaded test code through the attack surface. It can perform online control of injection test code through applications. Of course, under certain conditions, the test interface and related functions may also be set up at other levels (e.g., operating system level). It should be noted that in addition to the test interface code, the uploaded target test code should be the executable code resident in the memory to flexibly define or change the test function, and enhance the completeness of injection test.

It should be explained that there are two goals for setting up an experiment interface at the application level and for injecting test codes through the attack surface. One is that the experiment result is credible. Any attacks based on vulnerabilities and backdoors at any layer cannot directly or indirectly threaten the security goal of the system until they can precisely control the expression of output vectors of the target executors or operational scenarios (the ultimate goal the attacks can reach), ensuring that the application software invoke the test interface function as per the agreed requirements and activate the injected test code. The other is to observe the resource configuration of the executors or operational scenarios through the system management interface. By changing the injection test code, you can verify the CUP type, OS version and other information of the operating environment, and specify the target execution environment for the test code .

2) Differential mode test case injection experiment

Prerequisite: Assume that the test target function satisfies I[p1,p2,p3,…,pn]O, among which all functions pi are the same, i.e. p1=p2=…=pn but the pi implementation algorithms are all different, that

is, pc1≠pc2≠…≠pci. If there is a test case e1 that could make p1 generate an output vector outside the normal response sequence s1. By analogy, test cases e2 e3 ei can lead p2 p3 pi to generate output vector s2 s3 si and s1≠s2≠s3≠si, then, according to the definition of mimic defense effect, if test cases e1 e2 e3 ei, which do not have a cooperative relationship with each other are injected into executors p1 p2 p3 pi or defense scenarios respectively through the attack surface, then no si should appear on the mimic interface of I[P]O. The mimic defense function of the tested target should be able to clearly indicate that, except for the shutdown paralysis event that cannot recover automatically, the suppression effect of the differential mode test cases can be 100% achieved as long as the "lone wolf attack" differential mode test case does not have a cooperative relationship, and the test case can generate an action perceivable by the arbiter, which can be cleared or removed before the activation of another test case.

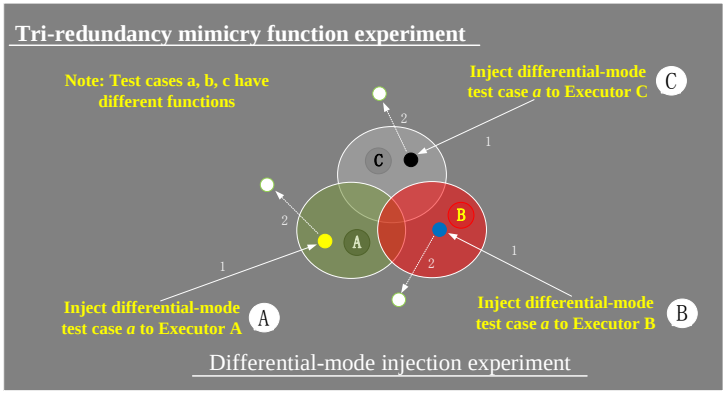

Figure 1: Injection experiment of differential mode test cases

Figure 1 illustrates an experimental scenario with a tri redundancy mimicry function. All the differential mode (with different functions) test cases that fall within the three executor regions A, B, and C, should be designed as scenarios that could be perceived by the mimic ruling phase. According to the feedback control strategy and the iterative convergence

backward verification mechanisms of mimic structure, another expected result of the experiment is that the problematic executor or operational scenario is either replaced, reconfigured or reconstructed, or cleaned, recovered or restarted by latching onto the executor or operational scenario with injected test cases. The related experimental scenarios and operation processes should be observed.

3) Experiment of the time cooperative differential mode test cases

Experiment goal: based on injection experiment of the differential mode test cases, distribute f differential mode test cases evenly among m defense scenarios of n executors in the mimic structure ($m \geq n$), the ratio of DM test cases f to m is $F=f/m * 100\%$, subject to $34\% \leqslant F \leqslant 50\%$. The experimenter sends the activation instruction to f test cases consecutively through the attack surface to observe the functionality and performance of the target object, focusing on (a) possible differential mode escape in the experiment process; (b) degradation of functionality and performance of the target object in the process; and (c) how long it takes for the problem avoidance mechanism to keep relevant defense scenarios from being re invoked.

Experiment plan: ① Test the recovery time of the sampled executor defense scenarios to determine the consecutive activation strategy of f test cases; ② decide on the deployment strategy (e.g., even distribution) of f test cases in m scenarios, given the Ratio F is 34%, 40% or 50%; ③ measure the functionality and performance of the target object in the experiment process to verify the design indicators of reliability and availability, with emphasis laid on the measurement of the time of the same group of time cooperative differential mode test cases from "attack reachability to unreachability", which tests the validity of avoidance mechanism of the target object in problematic scenarios; ④ for other methods and operations, please refer to the injection experiment

of differential mode test cases. Please note that the values of F shall be higher than the failure tolerance limit of the classic DRS, f≤(n 1)/2. For instance, if n=m=3, DRS allows f≤1, and the corresponding ratio F≤33.3%,while the experimental values of the time cooperative differential mode test cases are F=34%, 40% and 50%, which correspond to the worst condition of the DRS at n=3, 5 and infinite redundancy.

4) n-1 mode test case injection experiment

The experiment aims to check that the mimic structure should still have the ability to recover from the escape state in case a n 1 common mode perceivable escape (n is the redundancy of the current service set) occurs in the mimic interface, and prove that the mimic structure still have the expected probability even a common mode escape event occurs. The n 1 common mode escape is a small or tiny probability event in quantifiable design of security, however, failure to get rid of common mode escapes in the engineering implementation of the object will make the attack experience replicable but no longer be a probability event whenever a first common mode escape successful happens. The n 1 common mode escape experiment not only verifies the correctness of the mimic function, but also test the duration of the recovery process, the latter can quantitatively evaluate the performance of resisting common mode escapes. Also, the whole experiment process and effect shall be observable.

The injection experiment of n 1 common mode test cases stipulates that, when the number of executors or defense scenarios in the current service set of the mimic structure is n, suppose a test case ti can be injected into n 1 executors or defense scenarios and can be activated through the attack surface in a way to generate n 1 same output vector si, then, as per the mimic defense definition, mimic escape may probably occur in the I[P]O mimic interface, but the arbiter can perceive the inconsistency

between multimode output vectors. In other words, the n 1 common mode escape is a kind of perceivable escape. As a result, the feedback control phase will decide how to change the defense scenario in the mimic brackets by asymptotic or iterative convergence method according to the pre designed backward verification strategy (the relevant auxiliary ruling strategy is needed to tell the difference between differential mode scenarios and n 1 common mode scenarios), degenerate it into the differential mode experiment form as shown in Figure 1, and finally remove or clear it.

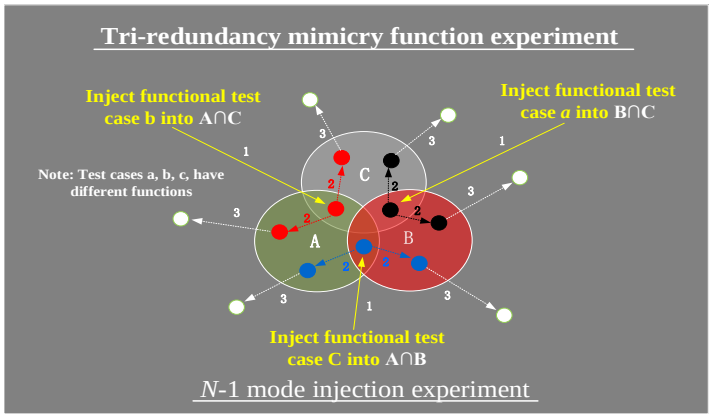

Figure 2: n 1 mode test case injection experiment

Figure 2 illustrates the experiment of an anti n 1 common mode escape in a tri redundancy mimic structure. It shows that common mode test cases a, b, and c are injected respectively into the intersections of executors (defense scenarios) A and B, A and C, B and C. It also shows the path (shown in dotted lines of different colors) from which these test cases were removed or cleared from the executors or operational scenarios. Assume that the n-1 mode common mode test case can be injected into the functional intersection of any two executors or operational scenarios, and be enabled to generate the same output vector by inputting channel compliant activation messages through the attack surface, then, according to the definition of mimic defense, the arbiter can perceive the

inconsistency in the multi mode output vector but cannot directly identify the problematic executor or operational scenario and needs to distinguish and eliminate the effect of the injected test case through the backward verification mechanism. To do this, the first step is to clear, restart or replace the executors or operational scenarios with inconsistent output vectors. If the status of the arbiter remains unchanged, choose one object from among the executors or operational scenarios with the same output vectors according to a certain strategy to repeat the previous step. Secondly, observe the status of the arbiter. If the status reverses, it would degenerate into the differential mode status shown in Figure 1. In this case, repeat the above step on executors not updated in the service set, and the injected test case will be cleared or removed from the current service set. Therefore, according to the mimic defense definition, even if the n-1 common mode attacks succeed, their escape states are not stably robust.

It should be pointed out that, when the n 1 mode test cases can be activated simultaneously, they may be recognized as differential mode attacks due to the inherent mechanism of the mimic structure, they cannot ensure steady display of the n 1 mode escape statue, so they shall be repeatedly injected and activated until the expected n 1 mode escape status appears in the experiment. The whole process of the experimentation should be observable.

It is clear that the mimic structure system has a feature that “an escape cannot be steadily sustained even if it is successful”. Although the system cannot replace all the functions of the traditional information security measures, it has the resilience that the latter does not. Especially when dealing with attackers who attempt to obtain sensitive information or sabotage the integrity of information through differential mode attacks, the mimic defense may be even more

advantageous than general encryption measures. As mimic defense is not a computable problem, it will not fall into the dilemma “where once it is breached by brute force, the whole defense system will collapse”.

5) n mode test case injection experiment

As its definition dictates, the mimic structure supports quantifiable design and features a tiny probability of n mode escape, which, unlike the n 1 mode escape, is mechanically unperceivable to the mimic structure. Therefore, to ensure the potential occurrence of the n mode escape, it is necessary to employ a certain external or internal strategy to disturb the feedback control loop, change the current operating environment inside the mimic brackets into the n 1 mode perceivable form, and enter the relevant disengagement or recovery process, ensuring that the n mode escape, if happens, is still a probability event. The function needs to be checked through the white box test of injected test cases, and measured and verified for the nominal time it takes to finish the disengagement according to the preset loop perturbation strategy.

As shown in Figure 3, assume that the injected n mode test case can produce consistent output vectors within the functional intersection of the three executors A∩B∩C, then it should not be perceived theoretically in the mimic ruling phase. However, according to the definition of mimic defense, even if the arbiter does not find any output vector anomaly, the executors or defense scenarios in the current service set may experience a forced and non deterministic replacement or cleaning and restart operation due to an external control instruction. This means that under the n mode injected test conditions, the escape state is surely unstable, and when the executors or defense scenarios themselves have the cleaning, restart, reconfiguration or reconstruction functions, the n mode unperceivable escape should automatically turn into the n 1 mode perceivable escape status as the recovery progresses, and eventually

degenerate into a differential mode context and be removed unperceivably. That is, the n mode test cases injected through the attack surface into the service set are removed since the host executor or operational scenario is strategically cleaned and restarted, or reconfigured and reconstructed by an external instruction. If the backward verification policy prescribes that executor C goes through a routine cleaning and then rejoins the current service set (may also directly reconfigure or replace executor C), then if the output vectors of C (or its substitute) are still inconsistent with those of A and B, the feedback loop will prioritize cleaning or replacement of the executor with the longest running duration (such as A).

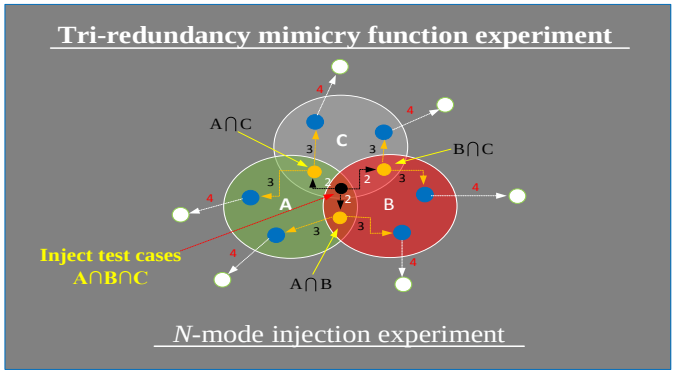

Figure 3: n mode test case injection experiment

Thus, when the arbiter finds that the AC output vectors are different from those of B, it indicates that an escape phenomenon (i.e., the scenario in Figure 2) has occurred in the scenario, and the above steps are performed on B until the arbiter no longer perceives the inconsistency. Tracing the migration trajectory shown by the dotted line in the above figure, the n mode test case scene will regress into the n 1 mode test scenario, and finally to the differential mode test scenario until all test cases are removed. We can see that in a mimic defense environment, even if the attacker has the ability to implement a n mode or cross domain coordinated attack and achieve a temporary escape, he will

not be mechanically capable of maintaining a stable escape, and therefore the robustness of attack effect remains an insurmountable challenge. It should be pointed out that, when the n mode test cases can be activated simultaneously, they may be recognized as differential mode or n 1 mode attacks due to the inherent mechanism of the mimic structure, they cannot ensure steady display of the n mode escape statue, so they shall be repeatedly injected and activated until the expected n mode escape status appears in the experiment.

6) Injection test in the feedback control loop

The mimicry feedback control loop consists of three parts: input allocation and proxy, output ruling and proxy, and feedback control. According to the definition of mimic defense, the input/output channel between the feedback loop and the executors and mimic brackets allow for "one way communication mechanism" only, and for one prerequisite that the feedback control loop may have vulnerabilities (really inevitable) but must not have malicious codes (this can be satisfied engineeringly in low complex cases) (as shown in Figure 4).

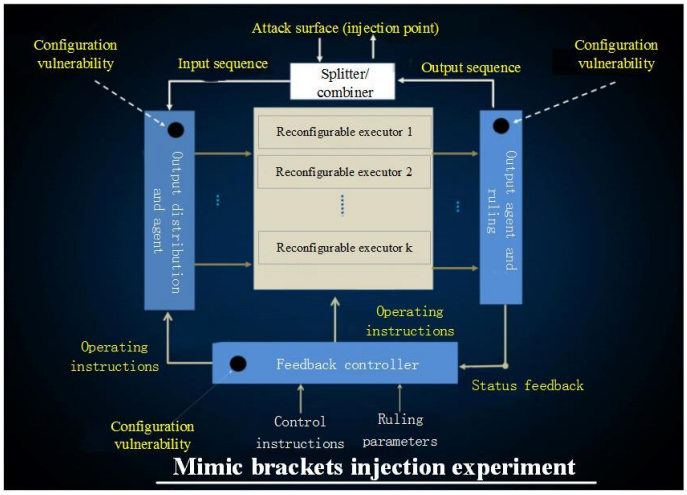

Figure 4: Unidirectional mechanism verification in mimic brackets

According to the strict unidirectional communication mechanism, the

Trojans in the executor should not be able to use the loopholes in the feedback loop to inject the attack code or achieve tunnel through. Similarly, the mimic bracket function is usually “transparent” to external attackers, which means that either the input assignment and proxy on the input channel, or the output ruling and proxy on the output channel, or the feedback control for internal policy and schedule should all be “invisible.”

In theory, the mimic brackets neither resolve the input incentive sequence content nor care about the syntax and semantics of the multimode output vector, so they are unreachable to threat targets. However, in engineering implementation, you have to care about the isolated import and load balance of the input excitation sequence, and have to import the agent system. Oftentimes, the mimic ruling cannot avoid the uncertainty of some optional value ranges and communication serial numbers of multi mode output vectors of the heterogeneous redundant executors. This, together with the difference in computation precision, requires that the mimic brackets must be opaque in syntax and semantics. There will be an increasing probability of the existence of unknown vulnerabilities as the function of bracket parts gets more complex or their intelligent processing capability gets stronger. However, mimic brackets can be considered to be up to the basic security requirements if the design can ensure that their parts will not be exploited even if they contain viruses. The white box testing aims to check if it is possible to set vulnerability test interfaces on some parts, if the test codes can be uploaded through the vulnerability interfaces, and if mimic escape can be achieved via the uploaded test codes. Sometimes, this method may not work due to the technological form of the object. For example, the vulnerability test interfaces cannot be set and the test codes cannot be uploaded and executed when passive optical splitters or wiring logic devices are used as input

agent parts, or the encryption mechanism is introduced into them. If so, it can be considered that the input agent parts are born with an attribute that their vulnerabilities are unavailable. Similarly, failure to upload the test codes through the mimic brackets specified path via the input agent parts and heterogeneous executors to the output agent parts or the arbiter for execution signifies that the output agent parts or the arbiter also have an attribute that their vulnerabilities are unavailable. Obviously, this attribute is not inherent in these parts, it derives mainly from the mimic structural effect. It should be noted that event if the test cases can be injected with test vulnerabilities of relevant mimic bracket parts from outside the mimic interface, the brackets may still be considered to meet the assumption that the vulnerabilities are unavailable on condition that no mimic escape can be achieved. In short, the security technologies are often required in engineering practice to guarantee that the mimic brackets shall not become the "short plate" of defense for the set security goal even if they contain design flaws and vulnerabilities.

7) Performance measurement. In the above mentioned tests, the time from activation of the test case to disabling them should be measured in order to evaluate whether the converging speed, actuating performance or scenario avoidance precision of the feedback control loop have met the requirements of the system design.

No matter what types the common mode test cases are, the mimic defense shall always be able to get out of the escape state. Maybe different design schemes lead to different time of disengagement, and to different cost of implementation. However, the function of disengagement from the common mode escape is indispensable because "failure to maintain a successful attack escape" is the high availability goal of mimic defense. Similarly, you can test the reliability design of the target

object with the differential mode test cases. The author declares that it is feasible to set confidentiality inspection function in the injected test cases in the mimic structure on condition that there are no side channel attacks launched by taking advantage of the physical (acoustic, optical, electrical, magnetic and thermal) effects or the resources shared with other non mimic structure tasks.