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 $n \leq f \leq F/2$, 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 $I[p_1,p_2,p_3,\ldots,p_n]O$, among which all functions p_i are the same, i.e. $p_1=p_2=\ldots=p_n$ but the p_i implementation algorithms are all different, that is, $p_{c1}\neq p_{c2}\neq\ldots\neq p_{ci.}$ If there is a test case e_1 that could make p_1 generate an output vector outside the normal response sequence s_1. By analogy, test cases e_2 e_3 e_i can lead p_2 p_3 p_i to generate output vector s_2 s_3 s_i and $s_1\neq s_2\neq s_3\neq s_i$, then, according to the definition of mimic defense effect, if test cases e_1 e_2 e_3 $e_{i,}$ which do not have a cooperative relationship with each other are injected into executors p_1 p_2 p_3 p_i or defense scenarios respectively through the attack surface, then no s_i 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.

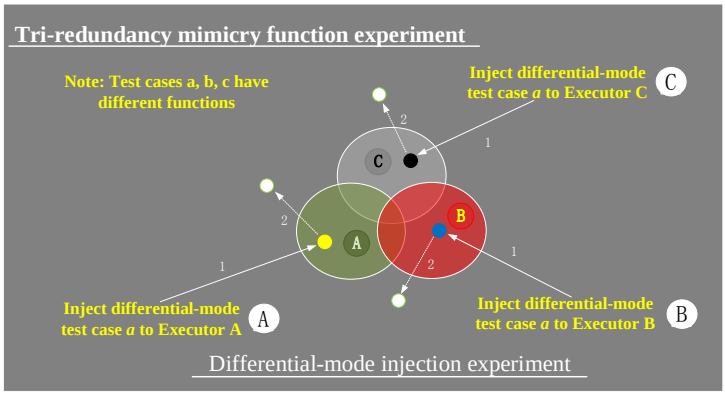

Figure 1: Injection experiment of differential mode test cases

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 $F=f/m \cdot 100\%$, subject to $34\% \leq F \leq 50\%$.

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, $f \leq (n \quad 1)/2$. For instance, if n=m=3, DRS allows $f \leq 1$, and the corresponding ratio $F \leq 33.3\%$,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 t_i 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 s_i, 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.

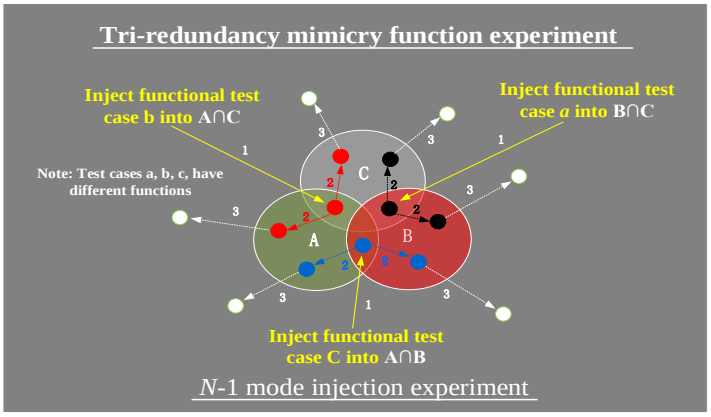

Figure 2: n 1 mode test case injection experiment

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 $A \cap B \cap C$, 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).

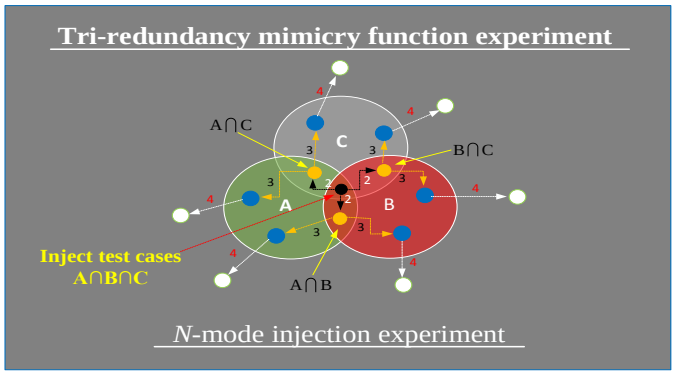

Figure 3: *n* mode test case injection experiment

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).

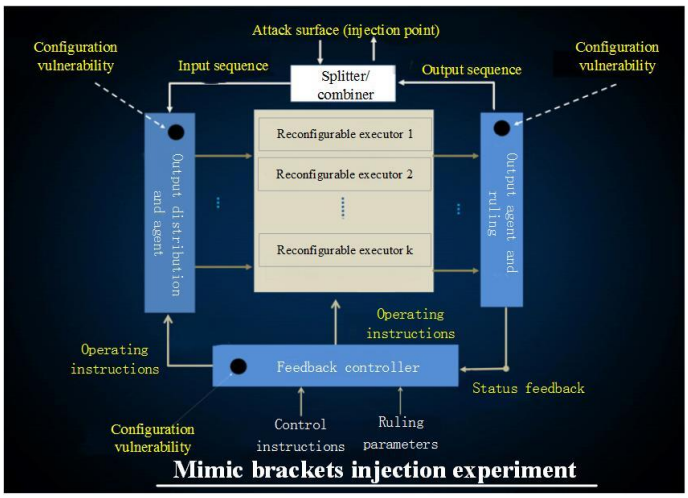

Figure 4: Unidirectional mechanism verification in mimic brackets

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.