The N-Variant System Framework

A System Structure for Secretless Security

University of Virginia

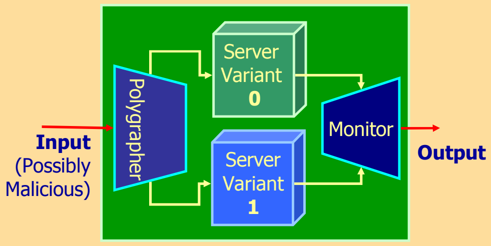
http://www.cs.virginia.edu/nvariant

Framework

The N-Variant System framework is a general mechanism for detecting and preventing classes of attacks on vulnerable servers. The framework consists of:

- A *polygrapher* that replicates input to multiple processes
- A set of *N* variant processes
- A *monitor* that observes the variants and their outputs.

The variant processes all implement the same service but are constructed to be artificially different in ways that detect attacks. The system runs the variants in parallel, giving each variant identical inputs and checks that they behave similarly before forwarding the output to the user. Consequently, any attack unable to simultaneously compromise all the variants in the framework will be detected and stopped before secret information is divulged or damage is done to the server. Other defense mechanisms rely on keeping a secret (e.g. randomization key) from the attacker. Our approach provides provable security that does not rely on secrets.



2-Variant System

Variant Requirements

Normal Equivalence – The variations must exhibit *equivalent* behavior on valid inputs. To achieve this there needs to be a mapping between the states and actions of the variants that accounts for the expected variation.

Detection Property – Any attack instance that could succeed against one variant, must produce detectable behavior on another variant.

Constructing Artificial Variations

Ultimately, the effectiveness of this approach relies on developing variations, or combinations of variations that differ from the perspective of the attacker. For example:

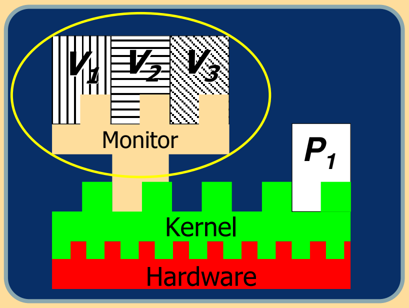
Address space partitioning: variants have disjoint address spaces so that addresses that are valid in one variant are guaranteed to be invalid in another variant. This thwarts attacks that depend on absolute addresses.

Instruction set tagging: the instruction set for one variant is tagged with a 0-bit, and the instruction set for the other variant is tagged with a 1-bit. Any instruction that is valid in one variant, causes an illegal instruction in the other. This thwarts direct code injection attacks.

The effectiveness of the framework, depends on developing variations that alter assumptions on which attackers rely.

To a Great White Shark, all humans are basically alike

Implementations



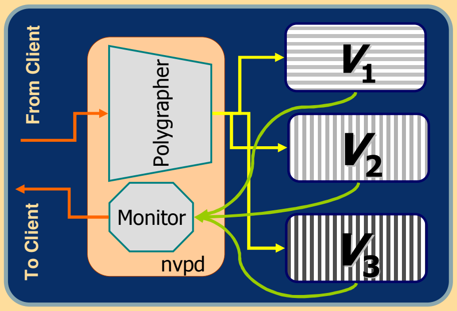
N-Variant System with Kernel Modification (N=3)

Kernel Modification

With the kernel modification implementation, the input splitting and output monitoring are performed at the system call level. When a variant makes a system call, the call is replaced by a wrapper that ensures that all the variants are making the same request, performs the request once and then forwards the result to all the variants.

Divert Sockets

The divert socket implementation splits the input and monitors the output using a user program `nvpd`. This program captures all incoming network traffic and sends adjusted copies of the network packets to all variants. The variants then send their responses back to the monitor, which compares their output and forwards it to the user if all the outputs are sufficiently similar.



N-Variant System with Divert Sockets (N=3)

People

PIs
Jack Davidson
David Evans
John Knight (PI)
Anh Nguyen-Tuong
Jonathan Rowanhill

Research Staff
Adrian Filipi

Students
Benjamin Cox
Michael Crane
Salvatore Guarnieri
Wei Hu
Nathanael Paul
Nora Sovarel
Dan Williams

NSF Award 0524432:
CT-T: A System Structure for Secretless Security