



GARRISON

HARDSEC


A plea for help...

The background features a complex, abstract geometric pattern of dark, faceted shapes, resembling a crystalline or crystalline structure. The shapes are rendered in various shades of dark gray and black, creating a sense of depth and texture. The overall composition is centered and balanced.

SOMETHING PRACTICAL

“Transform and verify”

https://www.ncsc.gov.uk/guidance/pattern-safely-importing-data

 National Cyber Security Centre
a part of GCHQ

Search

Guidance | Threats | Incident Management | Marketplace | Education & Research | Insight | Press & Media | Topics

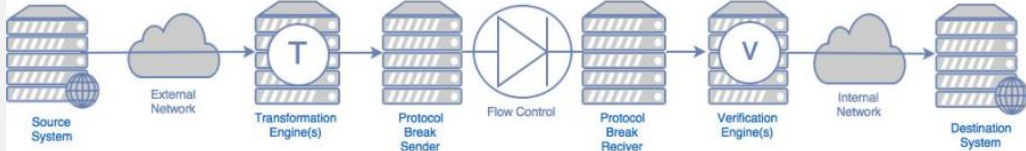
Published guidance | Infographics | NCSC glossary

[Home](#) > [Guidance](#) > [Published guidance](#)

Guidance

Pattern: Safely Importing Data

Created: 16 Jul 2018
Updated: 16 Jul 2018

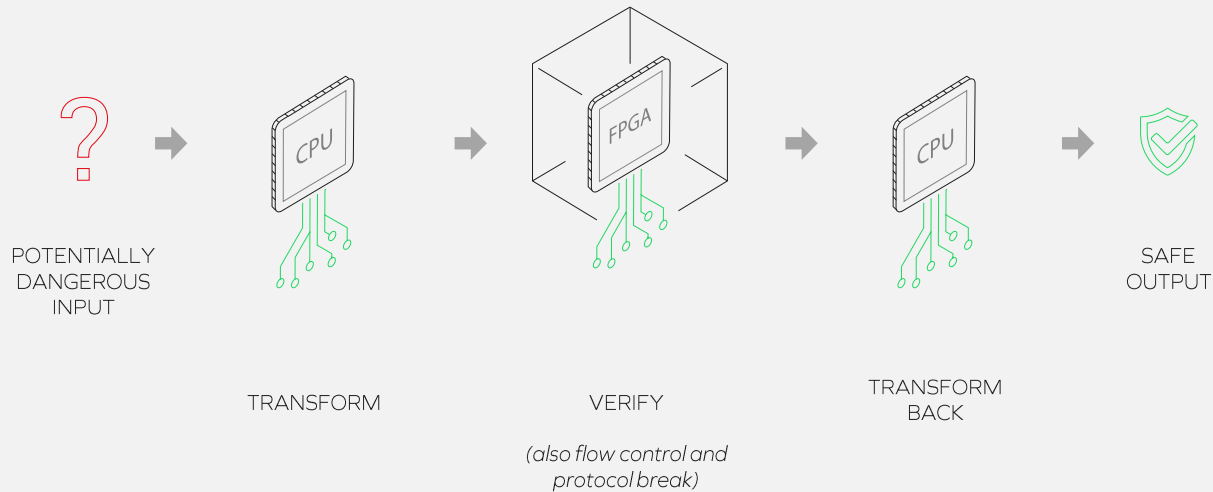


The diagram illustrates the data import process flow:

```
graph LR;
  A[Source System] --> B[External Network];
  B --> C[Transformation Engine(s)];
  C --> D[Protocol Break Sender];
  D --> E[Flow Control];
  E --> F[Protocol Break Receiver];
  F --> G[Verification Engine(s)];
  G --> H[Internal Network];
  H --> I[Destination System];
```

The flow starts with a Source System connected to an External Network. Data then passes through Transformation Engine(s), a Protocol Break Sender, Flow Control, a Protocol Break Receiver, Verification Engine(s), an Internal Network, and finally reaches the Destination System.

Transform and verify...with FPGAs



The background features a complex, abstract geometric pattern of dark, faceted shapes, possibly representing a crystalline structure or a stylized plant. The shapes are rendered in various shades of dark gray and black, creating a sense of depth and texture. The overall composition is centered and balanced.

SOMETHING THEORETICAL

HARDSEC.ORG

← →  https://www.hardsec.org



Hardsec: practical non-Turing-machine security for threat elimination

Sandy Macadam
sandym77@gmx.com
hardsec.org

[PDF](#)

Abstract

Security based on threat detection is a never-ending game of cat and mouse. By contrast, strong cybersecurity eliminates threats without relying on threat detection. Strong cybersecurity requires however that the threat elimination platform itself can be trusted: if an attacker can compromise the threat elimination platform then there can be no confidence that it continues to carry out threat elimination. Software running on a CPU that is (approximately) a Turing machine is inherently hard to secure against compromise, but hardware security suffers from problems due to inflexibility and high up-front costs. We present a practical approach (which we describe as 'hardsec') which uses Field Programmable Gate Array (FPGA) devices to deliver effective security threat elimination using non-Turing-machine implementations.

Introduction

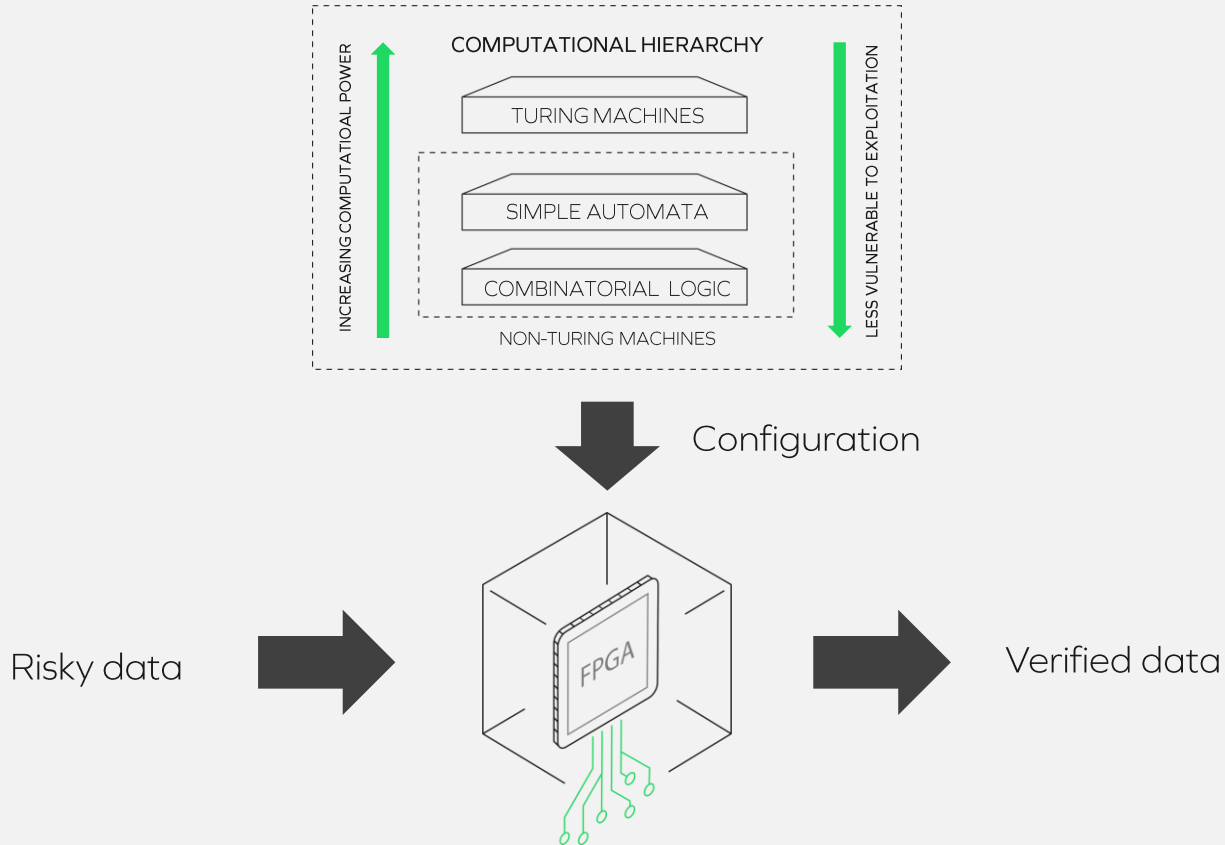
It is well known that maliciously constructed data inputs can be used by an attacker to exploit system vulnerabilities. Security approaches to defending against malicious data inputs fall into one of two camps. In one approach, steps are taken to detect malicious data inputs so that any such data is blocked before it is used as input. An attacker is of course motivated to avoid detection, and this approach therefore ultimately becomes a never-ending game of cat and mouse.

The second approach is to assume that all potential input data is malicious, and thus take steps to sanitise the data before it is used. This approach presents an inherent paradox, because the data must necessarily be used as input to the threat elimination system that carries out the sanitisation, and so could be used to exploit a vulnerability in that threat elimination system. Once the threat elimination system itself is exploited, the attacker can then output malicious data designed to exploit the original system.

One way of addressing this paradox is to run multiple independent threat elimination systems in series: an attacker then needs to prepare and deliver multiple different malicious datasets designed to exploit each of the threat elimination systems in turn. This increases the level of effort and cost that the attacker must expend in order to achieve their ultimate goal, which is to exploit the original system. However, it also increases the level of cost to the defender, because they must procure multiple independent threat elimination systems. It is not clear which of these costs increases the most rapidly: the cost to the attacker or the cost to the defender. In addition to the defender's direct costs in procuring multiple independent threat elimination systems, there is a further cost in some circumstances in that there will be an impact on end-to-end latency which can cause undesirable impact for the end user.

The other way of addressing the paradox is to engineer threat elimination systems using techniques which make them harder to exploit than the systems they are designed to protect. This has historically been the goal of high assurance software engineering techniques. After briefly reviewing the challenges with these techniques, we present an alternative approach based on the use of

THE HARDSEC IDEA





A PLEA FOR HELP