Your Security Software Runs on the Machine It Is Trying to Defend.

Every endpoint agent, every EDR product and every antivirus engine shares a structural weakness that no amount of engineering removes: it executes on the host it is protecting, under an operating system that an attacker with sufficient privilege controls. **When the host is compromised, the thing watching the host is compromised at the same moment, and it will continue to report that everything is fine.**

We designed and built an FPGA-based threat detection system to answer that problem in hardware. A Xilinx ten-layer dense board, custom logic written in VHDL and Verilog, attached to the host over PCIe and operating as a dedicated security co-processor with its own clock, its own memory and its own execution path. It watches the host. It does not depend on the host.

The system detects DDoS activity through traffic pattern analysis, ransomware through behavioural analysis of file encryption patterns, malware through signature detection with dynamic code analysis, and backdoor communications to hidden services. In validation it achieved **95% detection accuracy across threat categories, detection latency below 10 microseconds, and identification of ransomware encryption patterns within the first 30 seconds of infection** — while consuming none of the host's resources to do it.

THE ARCHITECTURE



SILICON DOES NOT NEGOTIATE WITH THE MALWARE RUNNING ABOVE IT

WHAT MAKES THIS DIFFERENT

01

Out-of-band by construction

The detection logic runs in silicon on a separate board. Compromising the operating system does not grant access to it, because it is not running under the operating system.

02

Custom logic, not software on a chip

Detection implemented as VLSI logic blocks in VHDL and Verilog, synthesised through Vivado. Parallel by nature rather than sequential by necessity.

03

Microsecond decisions

Below ten microseconds end to end. Fast enough to act on a packet rather than to write a report about one that already went through.

04

Zero host overhead

DDoS mitigation demonstrated without measurable impact on host performance. The board has its own DDR4; it borrows nothing.

05

Four detection classes in parallel

DDoS, ransomware, malware and backdoor communications, each with dedicated logic running simultaneously rather than competing for a scheduler.

+

Reprogrammable

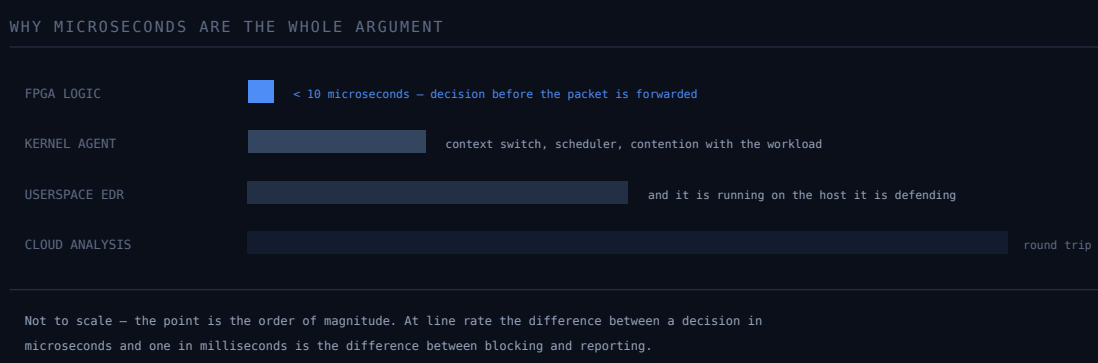
An FPGA is not an ASIC. The logic can be updated as threats change, without replacing the board.

WHY SOFTWARE CANNOT SOLVE THIS

This is not an argument that software security is badly built. It is an argument that it is `structurally` positioned to fail against a sufficiently privileged adversary, and that no amount of engineering effort changes the structure.

Consider what a kernel-level rootkit actually does. It does not defeat your endpoint agent by out-programming it. It defeats it by sitting underneath it and controlling what the agent is able to observe. The agent asks the operating system what files exist; the operating system, now under adversary control, answers. The agent reports clean. It is not malfunctioning — it is working exactly as designed, on top of a foundation that is lying to it.

The same structure produces the performance problem. Security software competes with the workload for the same CPU, the same memory and the same scheduler. This is why deep inspection gets tuned down in production, why agents get exclusion lists, and why the most sensitive systems in a bank frequently run the lightest monitoring — precisely inverting where the scrutiny should be.



MICROSECONDS VERSUS MILLISECONDS IS THE DIFFERENCE BETWEEN BLOCKING AND REPORTING

THE STRUCTURE

The watcher runs under the thing it watches. Privilege escalation defeats observation without defeating the observer.

THE INVERSION

Move detection off the host entirely, onto silicon with its own clock and memory, reached over PCIe.

THE CONSEQUENCE

Host compromise no longer implies detection compromise. The two are now independent failures rather than one.

WORKSTREAM 01

Hardware Design and Detection Logic

Detection algorithms that are trivial in software become interesting engineering problems when they must be expressed as logic gates, run at line rate, and fit in the resources a board actually has.

WHAT WE ACTUALLY DID

The board is a Xilinx ten-layer dense design with high-speed data processing, a PCIe connector for direct integration with the host motherboard, and onboard DDR4 for threat signatures and transient state. Detection logic was written in VHDL and Verilog and synthesised through the Vivado Design Suite. Each detection class — traffic anomaly, encryption behaviour, code signature, covert communication — is a dedicated logic block, running concurrently.

That concurrency is the point of choosing an FPGA over a faster CPU. On a processor, four detection engines contend for cores. In fabric, they are four independent circuits that happen to share a die. Adding the fourth costs area, not time.

THE CHALLENGES WE ENCOUNTERED

- **Machine learning models do not fit in fabric as written.** A model that runs comfortably in Python assumes floating-point arithmetic, dynamic memory and unbounded latency. Hardware offers none of the three.
- **Resource constraints are absolute.** Software that needs more memory gets more memory. Logic that does not fit does not fit, and the failure arrives at synthesis rather than at runtime.
- **Timing closure across four parallel engines.** Every path must meet timing at the target clock. One long combinational path in one detection block constrains the entire design.
- **Signature storage against fabric limits.** Signature libraries grow without bound. Onboard memory does not, and a PCIe round trip to fetch a signature would surrender the latency advantage that justifies the board.
- **A false positive in hardware is expensive.** Software can be patched in an afternoon. Reflashing fielded boards is an operation, so the accuracy bar before deployment is higher.

HOW WE RESOLVED THEM

- Models were quantised to fixed-point and restructured for fabric rather than ported to it, with the arithmetic reformulated so that the operations available in hardware are the operations the model performs. Accuracy loss was measured against the floating-point reference rather than assumed negligible.
- Detection logic was partitioned so that the latency-critical path — the decision that must complete before the packet moves — lives in fabric, while deeper analysis that tolerates microseconds runs in a slower stage behind it.
- Timing was closed by pipelining each detection block independently, accepting additional latency in stages where the budget allowed in order to keep the critical path short where it did not.
- Signatures were stored in DDR4 with the hot set held in on-chip memory, so that the common case never leaves the board and the uncommon case degrades to a memory access rather than a bus transaction.
- Accuracy was validated across threat categories before deployment rather than tuned after, reaching 95% across categories in the simulated environment.

HOW WE PROVED IT WORKED

The board was tested in a simulated environment against a range of cyberattack scenarios covering each detection class. Accuracy was measured per category rather than as a single blended figure, because a system that is excellent at DDoS and poor at ransomware has an average that describes nothing. Latency was measured end to end rather than as logic delay, and the DDoS case was validated specifically for host impact — mitigating an attack while the host continued to serve its workload without measurable degradation. **The claim is not that the board is fast. It is that the board is fast while the host is unaffected.**

> **RISK AVOIDED**
> A compromised kernel presenting a clean picture to the security stack running above it — detection that reports green precisely because the adversary has already won, which is the failure mode that makes dwell times measured in months possible.

> **WHAT A US BANK SHOULD TAKE FROM THIS**
> Ask what your detection layer would report on a host whose kernel is under adversary control. If the honest answer is that it would report normally, then your detection and your host are a single point of failure wearing two names.

WORKSTREAM 02

Ransomware as a Behavioural Problem

Signature detection finds ransomware you have seen before. The ransomware that matters is the one you have not, and by the time a signature exists for it, the sample that generated the signature has already encrypted someone.

WHAT WE ACTUALLY DID

The board detects ransomware by behaviour rather than by identity — observing the pattern that encryption produces regardless of which family produces it. Bulk sequential access across a file tree, read followed by write of high-entropy content, and a rate profile that no legitimate workload exhibits. In validation, the system identified encryption patterns within the first 30 seconds of infection.

Thirty seconds is the number that matters and it is worth being precise about why. A ransomware campaign is a race between the encryption and the response. At typical rates, thirty seconds is a small number of files. The same detection arriving five minutes later — a perfectly respectable figure for a cloud-analysed alert — is a restore operation instead of an interruption.

THE CHALLENGES WE ENCOUNTERED

- **Legitimate encryption exists.** Backup software, disk encryption, database compaction and archival all produce high-entropy sequential writes. A naive entropy detector fires on the backup window every night.
- **Entropy alone is insufficient.** Compressed archives are high-entropy. Encrypted files are high-entropy. Media files are close enough. Entropy is a hint, not a verdict.
- **Ransomware adapts to detection.** Families deliberately throttle their rate, encrypt partial files, and interleave with benign activity specifically to defeat behavioural thresholds.
- **The board sees traffic and behaviour, not intent.** Hardware has no notion of which process is authorised to encrypt what — that context lives in the host, which is the thing we have deliberately refused to trust.

HOW WE RESOLVED THEM

- Detection combines the dimensions rather than thresholding any one of them. Entropy, access pattern, rate and breadth across the file tree together describe something that legitimate encryption does not do, even though each dimension individually is unremarkable.
- Known-legitimate patterns were modelled explicitly, including their timing. Backup activity is expected behaviour with a shape and a schedule; the same shape at the wrong hour is the signal, not the noise.
- Rate thresholds were set from the observed profile of the environment rather than from a vendor default, on the understanding that slow ransomware is a real evasion and that a detector tuned only for fast encryption is a detector the adversary can walk past.
- The board's independence from host context was accepted as a limitation and stated as one. It is a complement to host-level context, not a replacement for it, and any vendor claiming otherwise is selling.

HOW WE PROVED IT WORKED

Detection was validated against multiple ransomware behaviours in the simulated environment, and separately against legitimate encryption workloads to establish the false positive rate under conditions the system would actually meet. The thirty-second figure was measured from infection to pattern identification rather than from the first suspicious write, because the honest measure is when the system reached a conclusion, not when the evidence first became available.

RISK AVOIDED

Ransomware detected at the point the ransom note appears — which is not detection at all, but notification, and which converts a security control into a recovery time objective.

WHAT A US BANK SHOULD TAKE FROM THIS

The FFIEC expects you to evidence ransomware recovery capability to your regulator. Detection at thirty seconds and detection at thirty minutes produce very different conversations about what your recovery time objective actually needs to survive.

WHERE THIS BELONGS IN A BANK

Not everywhere. A security co-processor is a considered choice for systems where the consequence of compromise justifies dedicated hardware, and an extravagance everywhere else. **The question is not whether hardware security is better. It is which of your systems cannot afford to have their detection layer compromised alongside them.**

WHERE IT FITS	WHY THERE SPECIFICALLY	MAPS TO
Payment switch	Availability is the product and the latency budget is brutal. Detection that costs microseconds is affordable; detection that costs milliseconds is not.	FEDNOW / RTP
HSM and key infrastructure	The systems where host compromise is categorically unacceptable, and where an independent observer is worth its cost.	PCI / KEY MGMT
Privileged admin hosts	The adversary's actual target. Detection that survives the compromise of the host is the only detection that matters here.	OCC SRP SPRING 2026
Core ledger boundary	Ransomware detection at thirty seconds rather than thirty minutes changes what your recovery objective has to withstand.	FFIEC
AI inference hosts	Where model weights and live data sit together, an observer the host cannot see or disable is the difference between containment and exfiltration.	SR 11-7 / AI RISK
Sovereign or air-gapped	Environments where cloud-analysed detection is not an option and the decision must be made locally, at line rate.	DATA RESIDENCY

95% · <10µs · 30s

Detection accuracy across threat categories. End-to-end detection latency. Ransomware encryption patterns identified within the first thirty seconds of infection. DDoS mitigated with no measurable host impact. Measured in a simulated environment against defined attack scenarios, which is what these figures are and what they are not — they are validation results, not a production deployment across a fleet.

THE HONEST LIMITATIONS

A document that only lists strengths is a brochure. The constraints here are real and a bank's engineers will find them in the first technical session, so they are better stated now.

- **It does not see host context.** The board's independence from the operating system is its central virtue and its central limitation. It cannot know which process is authorised to do what. It complements host telemetry; it does not replace it.
- **Deployment is physical.** A PCIe card is a rack visit, a change window and a hardware lifecycle. This is a real cost that a software agent does not carry.
- **Validation was in a simulated environment.** The figures are from defined attack scenarios, not from a fleet under production traffic. We would expect to re-establish them in your environment rather than ask you to accept ours.
- **Logic updates are an operation.** Reprogrammability is genuine, but reflashing fielded boards is not a software patch cycle. The roadmap includes firmware update mechanisms precisely because this is the constraint that matters at scale.

WHERE THIS GOES NEXT

> Every security architecture eventually rests on something the adversary cannot reach. For most institutions that thing is a software agent running on the machine the adversary has already taken.
>
> We build the other kind. If you have systems whose compromise you genuinely cannot tolerate, the conversation worth having is which of them warrant an observer that lives outside the blast radius.

TECH4BIZ SOLUTIONS | CODENUK | AUDITTRAX LABS

tech4bizsolutions.com codenuk.com auditraxlabs.com

contact@tech4biz.io