# Secure Cross-Domain Data Validation with Field Programmable Gate Arrays

**Abbie Cliche[1], James Brock[1], Ryan Longwell[1], Jason Dahlstrom[1] and Stephen Taylor[2]**
[1]Web Sensing, LLC., Lebanon, USA
[2]Thayer School of Engineering at Dartmouth College, Hanover, USA

abbie.cliche@websensing.us
jim.brock@websensing.us
ryan.longwell@websensing.us
jason.dahlstrom@websensing.us
Stephen.taylor@dartmouth.edu

**Abstract:** Cross-domain systems have traditionally employed virtualization to isolate security domains, providing communication though standard TCP/IP networking stacks coupled with access permissions and credentials to enforce isolation. This deep and complex chain of trust typically depends upon a hardware base, such as a Trusted Platform Module (TPM) chip combined with a secure bootstrapping process. This paper describes a novel and high-performance alternative, Secure Transfer Link (STL), leveraging the unique architectural characteristics of the AMD UltraScale Multi-Processor System-on-Chip (MPSoC) device family: CPU affinity, an on-chip field programmable gate array (FPGA), and bus-mastering. These architectural characteristics make it possible to construct a secure data transfer path within the FPGA that can control which virtual machines may access and transfer data, enforcing isolation. The abstraction can be extended to include deep packet inspection and validation, such as parsing that checks adherence to the JavaScript Object Notation (JSON) protocol. Validation is achieved through the combination of formal grammars with a pushdown automata (PDA) parser and automatic transformation into an FPGA hardware configuration, resulting in a formally verifiable and hardened intellectual property (IP) called the Data Validator. The Secure Transfer Link is constructed by combining this Data Validator with another IP, the Memory Guard, which enforces access controls. These hardware Ips, together, comprise a system which prevents malicious software resident on a processor from undermining access policies or transferring malicious data. The presented IPs are performant. Their throughput improvement over traditional UDP/IP networking stacks is dramatic: speedups of up to 7x for tactical length messages and up to 4x for larger messages. The IPs are created using High-Level Synthesis (HLS), making it possible to formally specify a broad range of alternative policy and enforcement options then automatically include them in the Secure Transfer Link, constituting a novel isolation enforcement solution that is higher throughput than state of the art alternatives and enables selective domain access contingent upon formal verification of data.

**Keywords**: Hardware, Cybersecurity, Cross-domain, Data validation, Privacy, Secure data transfer, FPGA, SoC

## 1. Introduction

Ensuring the integrity, validity, and security of information transfers between security domains is becoming more important as combat arenas become saturated with high-value network-connected devices. This need is not unique to defense applications. Modern healthcare and critical infrastructure may be designed with separate privileged and public facing domains. A single device may be required to store information from multiple security classifications and incorporate information from a lower classification into higher classification domain to complete mission objectives. It is critical that the directionality of cross-domain transfers can be ensured, along with the credentials of the invoking process and the validity of the data itself. A typical approach to cross-domain transfers includes the Linux networking stack. However, when implemented in software, these mechanisms have been shown to be slow and cannot keep up with the needs of sophisticated real-time systems. Additionally, software is vulnerable to a number of API, buffer, and pointer attacks. Faster hardware-based mechanisms offer greater assurance but do not include robust payload filtering required to fully protect receiving domains. There remains a need for cross-domain data transfer inspection and validation technologies with high enough throughput and security assurance for critical systems. This paper presents two cross-domain transfer mechanisms. One is composed of custom parsing, validation, and authentication hardware IPs which are capable of performing deterministic transfer validation at throughputs of up to 7x traditional software alternatives. The other provides a network-based link through the FPGA fabric with the same validation IP. Prior work on automated hardware parser generation was leveraged in conjunction with the novel FPGA IP and cross-domain transfer methods presented in this paper to comprise the Secure Transfer Link.

## 2. Related Work

Many high value systems hold information at multiple classification levels and must enforce strict boundaries between them or require the ability to share information securely with stakeholders from different

organizations or nationalities (Dahlstrom *et al.*, 2019; National Security Agency (NSA), no date). It is vital that the direction of the flow of information can be guaranteed, that access policies can be enforced, and that data integrity can be guaranteed during these transfers. Cross-domain solutions are a class of technology which address this need, enabling controlled transfers of information between information domains with different security classifications or stakeholders (a *transfer* CDS) (General Dynamics Mission Systems, Inc, 2025; NSA, no date) or access to multiple information domains simultaneously (an *access* CDS) (Sundaravarathan *et al.*, 2024). An example of an access CDS is an application like SecureView (Dahlstrom *et al.*, 2019). According to the Air Force Research Lab (AFRL) (2021), SecureView is an application that allows a user to view multiple security domains in different windows on a single screen. Access and transfer cross-domain systems can either be a boon to productivity in enterprise environments or can provide critical inter-agency communication links in operational environments. Transfer CDSs can provide situational awareness through applications like a common operating picture (Sundaravarathan *et al.*, 2024). *Tactical* CDSs deployed in these environments require particularly low data transmission and processing latencies (General Dynamics Mission Systems, Inc, 2025).

A prototypical access CDS architecture may consist of a hypervisor running guest OSs connected by a network, coupled with a secure boot process, and rigorous mandatory access controls (MACs) (Daughety *et al.*, 2021; BAE Systems, 2022; AFRL, 2021; BAE Systems, no date; Everfox (2024)). A hardware root of trust could take the form of a TPM, a specialized cryptographic processor which can ensure the integrity of software used during the boot process (Pamnani and Matarazzo, 2024). Hypervisors are well established as a mechanism for providing isolation between guest OSs (Goldberg, 1974; Barham *et al.*, 2003; Thyagaturu *et al.*, 2022) on a single host and therefore can protect two domains collocated on a single chip. Doing so can reduce the size, weight, and power (SWaP) and latency of a system compared to an alternative architecture requiring physically separate machines. For access CDS applications in tactical domains, minimizing these metrics is always beneficial.

Successful domain isolation using virtual machines relies on privilege-based resource access and sharing. Hypervisors protect resources like memory ensuring that one application cannot surreptitiously gain access to another's memory space (Sundaravarathan *et al.*, 2024). Hypervisor-based CDSs are typically built on Type 1 hypervisors which behave like bare metal to the operating systems they support (AFRL, 2013; AFRL, 2021; Dahlstrom *et al.*, 2019; Sundaravarathan *et al.*, 2024). Formally proven secure hypervisors, such as the BlueRock Hypervisor (BHV) can provide additional assurance (Bedrock Systems Inc. (BSI), 2022).

Most hypervisors allow guest OSs to communicate with each other via either a virtual network or a shared memory mechanism. However, virtual networks do not typically come with data inspection and validation out of the box (Ren *et al.*, 2016; Green Hills Software, 2020; BSI, 2022). Use of network functions, in generals, is computationally expensive(Ren *et al.*, 2016; Thyagaturu *et al.*, 2022). Additionally, network-dependent CDSs like AFRL SecureView are vulnerable to network-based attacks from external interfaces (Sundaravarathan *et al.*, 2024). A single compromised virtual machine can compromise the other virtual machines on the host through either privilege abuse or steganographic attacks embedded in virtual network packets that pass all IP, port, and MAC checks (The MITRE Corporation, 2018). On the other hand, shared memory links between virtual machines can reach throughput exceeding 5Gbps but are highly coupled to a platform's fixed silicon architecture (Ren *et al.*, 2016; Li *et al.*, 2023). This means that there are no opportunities to insert security features, such as filtering, into the shared memory datapath.

A transfer CDS enables unidirectional or bidirectional secure transfers between domains. It may be implemented as software or hardware module (Sundaravarathan *et al.*, 2024) and may, like an access CDS, rely on a hardware base of trust (Daughety *et al.*, 2021). Transferring high bandwidth data from embedded sensors in tactical arenas is a key challenge in transfer CDS design because it requires performing filtering functions, which can be computationally expensive, at line rate (Dahlstrom *et al.*, 2019; Sundaravarathan *et al.*, 2024). Recently, transfer CDS architectures have favored filtering in hardware over filtering in software as the NSA has formally recognized hardware as part of robust system design in its Raise the Bar mandate (Campbell, 2019).

Syntactic filtering capabilities are a critical component of transfer CDSs. Syntactic filtering can stop attacks in which a file's contents don't match its extension and thwart attacks which depend on hiding malicious data between valid fields. When applied to data in transit, filtering can protect against vulnerable APIs in receiving software that do not adequately validate data (The MITRE Corporation, 2020). The challenge historically has been accomplishing syntactic filtering on data payloads with low latency (Swarup, 1994). Consequently, payload parsing is sometimes executed as post-processing or is only applied to a subset of packets (Hu *et al.*, 2023).

There are several hardware-enforced cross-domain solutions on the market (BAE Systems, 2020; "XTS® Diode One Way Transfer Solution," 2022; Everfox, 2024). Some claim to reach 10Gbps of throughput ("XTS® Diode One

Way Transfer Solution," 2022), but this value reflects the maximum throughput under laboratory conditions, and it is unlikely that it includes content filtering. Others offer semantic data inspection and sanitization but do not advertise throughput (Everfox, 2024). One experimental application of hardware filtering reaches 3Gbps, but it employs payload inspection only on packets that have failed header inspection, was only evaluated on packets up to 512 bytes in length, and did not document clock speeds used in the FPGA fabric (Hu *et al.*, 2023). Several software-based systems, such as a secure RTOS separation kernel, do not provide content filtering for transfers between domains (Green Hills Software, 2020; Daughety *et al.*, 2021). Even if content filtering were included, it would still be slower and more power intensive than equivalent hardware-based content filtering. As little as 3Mbps of data throughput would be sufficient for transmission of video, so a hardware-based syntactical filter with performance exceeding that threshold, like the work we present here, would constitute a step forward in the state of the art.

CPU affinity is a guest VM setting supported by most hypervisors. VMs can be locked to a particular CPU core by the hypervisor and therefore predictably linked to other hardware resources on an SoC with virtualization support (Dahlstrom *et al.*, 2019; "Zynq UltraScale+ Device Technical Reference Manual," 2023). In the case of the UltraScale+ MPSoC, this is accomplished through the use of AXI side channels, TrustZone technology, and the XMPU (McNeil *et al.*, 2021). These technologies enable VM resource isolation which extends to the FPGA fabric, forming a trusted connection between a VM and IP blocks residing in the FPGA fabric. Together, these mechanisms make it possible to create a shared memory between virtual machines with hardware-enforced access controls. The speed and flexibility of the FPGA fabric open up new options for adding deep packet inspection and validation capabilities to a system. Formally defined parsers can be constructed quickly and easily using Vitis HLS and then deployed to provide deterministic data filtering (Taylor *et al.*, 2022). Even when validation processes add latency to a system, those run in FPGA fabric will outperform their software counterparts.

## 3. Methods

We have established the general notion that routing cross-domain transfers through the FPGA fabric could lead to improved performance over the network stack. By timing data transfers between two Linux virtual machines (VM0 and VM1) running on the BlueRock Hypervisor, we collected two sets of data to establish a baseline to compare our solution against. The first is the latency of transferring data through BlueRock's vSwitch and the second is the latency of transfers routed through a hard-wired 1Gbps ethernet link through the FPGA fabric, referred to as the *fixed link* from here on out. We then measure the latency of data transfers on our novel cross-domain transfer mechanism, known as the *Secure Transfer Link (STL)* from this point forward. The STL comprises custom memory-mapped IPs in the FPGA fabric responsible for gating each virtual machine's access to memory regions by checking user permissions and validating transfers between memory regions with a data payload validator. Figure 1 shows the three different data transfer pathways within the FPGA SoC.

All of our experiments were performed on the Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit using a programmable logic (PL) fabric clock speed of 100MHz and 1G Ethernet links (shown in by the gigabit media independent interfaces [GMII] in Figure 1). The test data used in the experiments was a set of text files containing JavaScript Object Notation (JSON) structures generated by a fuzzer. The set contained a uniform distribution of structures with depths ranging from 1 to 15.

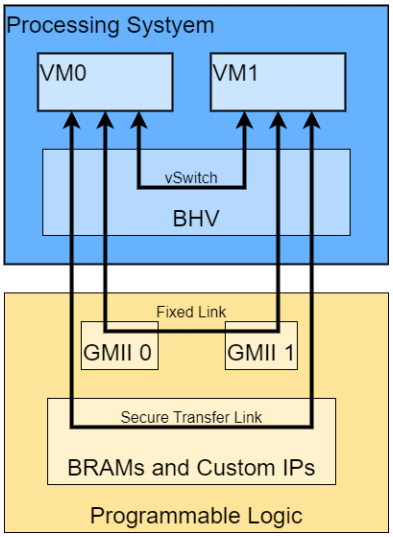


**Figure 1: The three cross-domain transfer mechanisms tested in this paper**

### 3.1 Hardware Design

This work includes two encapsulations of previous a previously developed automatic grammar parser and validator generation flow (Taylor *et al.*, 2022; Taylor and Pope, 2024). This parser was constructed by transforming a formal JSON grammar definition into parsing tables which were compressed and inserted into a generic C++ push-down automata (PDA) implementation. Vitis HLS was leveraged to automatically generate a grammar validator hardware IP module from this source code. This process is generic, and validators for other grammars can be constructed from a number of supported grammar tables which are available as an open-source software library (Taylor *et al.*, no date). The PDA can currently reach a JSON validation throughput of 24Mbps (Taylor *et al.*, 2022).

The fixed link shown in Figure 1 connects two of the MPSoC's integrated Ethernet blocks in a bidirectional direct 1Gbps configuration (VM0 to VM1 and VM1 to VM0). The fixed link also has two modes, *passthrough* and *validation*. In the passthrough mode, the Ethernet link contains no additional logic. In the validation configuration, the Ethernet link contains an instance of our PDA parser encapsulated in a network packet parser. The network packet parser validates the IP, Ethernet, and TCP/UDP packet headers and then extracts the JSON payload for the PDA to validate. The encapsulating IP can be configured in validation mode, when packets whose payloads do not pass the grammar check are dropped, or in passthrough mode, in which all packets pass through the link.

The Memory Guard IP is a build-time configurable hardware block capable of enforcing complex access rules on a variety of memory mapped regions held within the FPGA PL fabric. The Memory Guard can separate the memory in its address map into many segments defined by the user. Read/write permissions can be granted per memory segment and per originating CPU core. Because the VMs in our experiments are pinned to specific CPU cores, permissions can be set based on the privilege level of the VM. Enforcement of privileges is accomplished by observing AXI side-channel signals to detect which CPU core initiated each memory transaction. There is an additional reserved "honeypot" memory region, which can be enabled and configured. Memory transactions which fail permissions checks can be diverted to this region so that they appear to have succeeded but don't allow unauthorized access to data. The Memory Guard can be placed on any memory mapped processing system interface, enabling programmable access controls and responses at any security barrier in the system.

The data validator (DV) is the hardware module developed to validate payload data in flight as it is transferred between memory regions in two modes: validation and passthrough. The Data Validator IP has two 32-bit wide AXI memory mapped interfaces, each of which is connected to a BRAM (block random access memory) within the FPGA fabric. The data validator reads data from the *origin* BRAM and passes it into the PDA JSON parser (if validation mode is enabled). If the JSON parser check fails at any point, the data is dropped and the BRAMs cleared. Otherwise, the data is written into the second *destination* BRAM. In passthrough mode, all data is transferred to the destination BRAM. The Data Validator and processing system use mailbox bits for synchronization. If data is dropped, a status indicator is raised and can be consumed by software as an interrupt so that application level logic can handle retransmission of messages if applicable for a given use case.

The Zynq UltraScale+ MPSoC ZCU102 Evaluation board contains an SoC with a full quad-core ARM A53 processing system (PS) co-located with the programmable logic (PL) resources of a large FPGA. The large number of available direct hardware interfaces between the processor and this FPGA include Ethernet and AXI memory mapped interfaces. The Secure Transfer Link shown in Figure 1 utilizes the AXI memory mapped interfaces for mapping PL BRAMs to the VMs in the processing system. The Secure Transfer Link itself is composed of two individual data diode links configured in opposite directions, one transferring data from VM0 to VM1 and the other transferring data from VM1 to VM0, as shown in Figure 2. Each link consists of two BRAM blocks connected by an instance of the Data Validator IP, and PS access to each BRAM is gated by an instance of the Memory Guard IP.

Our experimental platform is meant to represent a system with multiple domains of varying levels of privilege. The two VMs isolated by the BlueRock Hypervisor represent those domains. The shared memory blocks are first protected by our Memory Guard (MG) IP with permissions similar to the shared memory region in Figure 2. The difference in our Secure Transfer Link test setup in Figure 2 is that we have shared memory regions moving data bidirectionally through two separate links to use the round-trip timing in our results.

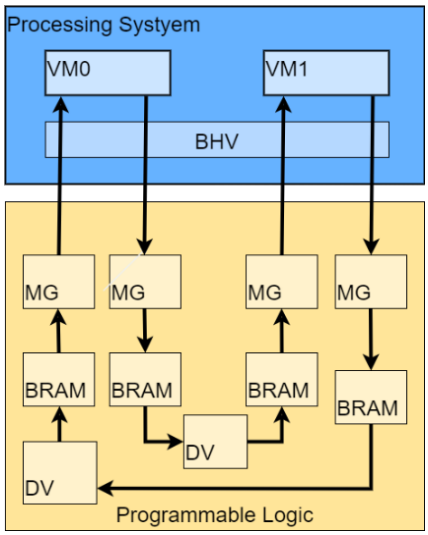


**Figure 2: The secure transfer link**

### 3.2 Test Applications

VM0 and VM1 run on top of the BlueRock Hypervisor (BHV). We used the BHV's built-in virtual switch (vSwitch) as a pure software transfer link to compare against our FPGA fabric mechanisms. To test either the vSwitch or the fixed link, a network client program on VM0 sends a UDP packet out over the relevant interface. The UDP packet is received and sent back by the server program to the original source VM to generate a complete round-trip time for data transfers. When measuring the fixed link, the elapsed time includes overhead incurred by the Linux network stack as well as overhead from the Ethernet frame, IP and UDP headers, and the interframe gap. When measuring the vSwitch mechanism, overhead is only incurred by virtual switching.

A separate application was used to test the Secure Transfer Link. It drives raw data to the Memory Guard, which drives the data onward through the rest of the Secure Transfer Link. The corresponding application running on VM1 polls its shared memory for data and reassembles the received data as a txt file. It then writes this file back onto the Secure Transfer Link in the other direction. The test program running on the source VM collects the elapsed time between the first chunk of JSON data being written to the origin shared memory and the last chunk being read from the destination shared memory.

We computed latency and throughput from the measured elapsed time of a full return trip. We ran tests transferring data sizes from 1 to 1472 bytes to gauge the performance of each transfer link as the transfer size increases. Each test was repeated 10 times to get an average performance and account for random software anomalies in any given run. The latency data presented is the measured latency divided by two to compensate for the inclusion of a return trip. Throughput was computed by dividing the number of bits in each test file by the latency of the transfer.

Latency measurements were taken for the following data transfer mechanisms: Secure Transfer Link passthrough mode, Secure Transfer Link validation mode, fixed link passthrough mode, fixed link validation

mode, and BHV vSwitch. The VSwitch does not have any payload parsing capabilities, so it is only configured as a passthrough.

Each of the JSON data files used in these experiments was generated using a fuzzer. The files ranged in size from 1 to 1472 bytes and had random key-value pairs and nested structures. The maximum JSON structure depth used in these experiments was 15 because it was impractical to create deeper structures in the smallest files. Only valid JSON structures were used for performance testing since invalid structures would introduce non-deterministic latency decreases as the result of halting processing after reaching an error in the data structure and not from any advantage of the secure transfer link.


```
{
    "LabIV8E": -2006,
    "q3": false,
    "1CP": 724.8507268121102,
    "cOPY4K": "VwxslZatE",
    "2Mfz6": 424
}
```


**Figure 3: Sample JSON structure with a depth of 1 used in the experiment**

Randomized raw JSON, as shown in Figure 3, was used for the Secure Transfer Link tests, but it was encapsulated in a UDP packet for tests on the Fixed Link and vSwitch link. UDP was selected over TCP to minimize non-payload overhead as much as possible. A maximum payload size of 1472 was used because common tactical and "command and control" traffic using protocols such as MAVLink fit within packets of this size, and it is the maximum data size for a single UDP packet (Taylor *et al.*, 2022).

## 4. Results

The bandwidth graph in Figure 4 compares the performance of the BlueRock vSwitch data transfer mechanism with that of our fixed link passthrough configuration. The measurements of the two mechanisms are nearly identical, with a root mean squared error of 60µs between the two sets of latency measurements and a root mean squared error of 23Kbps between the two sets of throughputs. Through this comparison, we have established the feasibility of routing data transfers through the FPGA fabric and shown that there is not a significant overhead or slowdown incurred from doing so. We have shown that including the FPGA fabric in the data path enables packet validation logic is enabled at no cost.

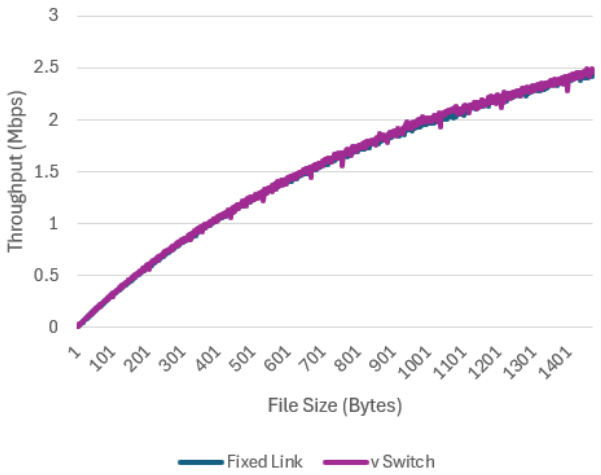


**Figure 4: Fixed link vs. vSwitch passthrough bandwidth**

We now compare the Secure Transfer Link performance against the hardware fixed link, which we have established to have similar performance to the BHV vSwitch method of cross-domain transfer. We first compare the two links with no parsing inline to establish baseline performance improvement. Latency of the fixed link is greater than 2ns and up to 3.5ns more than the latency of the Secure Transfer Link and twice as responsive as that of the Secure Transfer Link to the size of the JSON data. This shows that cross-domain transfers of tactical length messages over the Secure Transfer Link can be performed with about a quarter of the latency by bypassing the overhead of the network stack.

Figure 5 shows the bandwidth of same set of tests. The bandwidth of the fixed link remains low for all sizes of JSON message due to the significant overhead involved in the software network stack. Tactical messages are not large enough relative to the overhead to make the fixed link an efficient method of cross-domain data transfer. In contrast, the bandwidth of the Secure Transfer Link increases as the JSON data size grows due to the small latency overhead of the hardware mailbox signaling and low-latency data movement. This figure shows that bypassing the network stack using the Secure Transfer Link leads to up to 7x bandwidth improvement for cross-domain transfers. A novel shared memory mechanism such as our Secure Transfer Link can yield greater throughput and lower latency when the message size is small enough that the network stack overhead would not be amortized away.

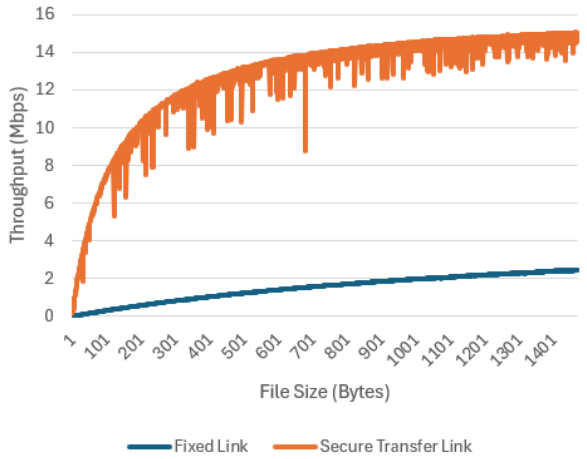


**Figure 5: Fixed link vs. secure transfer link passthrough bandwidth**

Now we introduce the Data Validator with formal JSON parsing into the Secure Transfer Link data path. The latency resulting from including JSON validation in the Secure Transfer Link is greater than the latency without for all data sizes and increases linearly with data size. However, the maximum latency increase for tactical messages is only 407µs. Figure 6 shows that the cost of validation with including the JSON validation is, at worst, a 28% decrease in bandwidth. The resulting 10Mbps throughput is still more than sufficient for applications like streaming video calls and well exceeds the performance required of a tactical CDS.

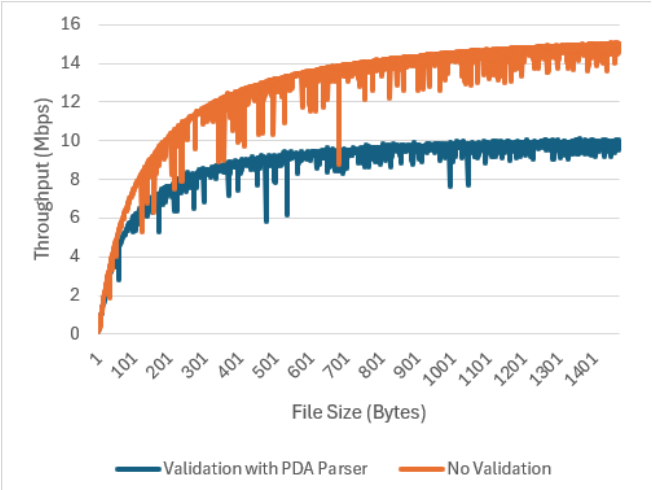


**Figure 6: Secure transfer link passthrough vs. secure transfer link validation bandwidth**

## 5. Conclusion

The number of cross-domain systems employed in modern tactical environments is growing, and performance and SWaP (size, weight, and power) constraints are tightening. This research demonstrates a novel combination of cross-domain assurance through syntactical data validation and a high throughput hardware channel. Our Secure Transfer Link can protect against data injection attacks through deterministic grammar checkers and guard against memory operations initiated by virtual machines in the system which are not whitelisted,

preventing information from leaking between security domains. Additionally, automatically generating FPGA IP from C++ source enables this solution to be easily tailored to other grammars and deployed to extremely low-SWaP environments without FPGA design expertise.

We showed that routing cross-domain transfers through the fabric, in general, is at least as performant as routing through a hypervisor's virtual switch by comparing our fixed-link mechanism to the BHV VSwitch. We have also demonstrated that the Secure Transfer Link attains as much as a 7x speedup over links that include the network stack for tactical length messages. Finally, we showed that data validation mechanisms resident in the FPGA fabric provide assurance without undermining the performance gains of our shared memory mechanism by demonstrating a throughput of up to 10Mbps. This performance beats all other cross-domain links demonstrated in this paper, including the vSwitch which does not include data validation. It also beats the throughput of state-of-the-art software-base syntactic parsing demonstrated in related work (Hu *et al.*, 2023).

## Acknowledgements

This research is supported by the Defense Advanced Research Projects Agency (DARPA).

DISTRIBUTION A : Approved for Public Release; Distribution is unlimited.

**Ethics Declaration**: Ethical clearance was not required for the research carried out in this paper.

**AI Declaration**: AI tools were not used in producing this paper.

## References

Air Force Research Laboratory (2013) "SecureView Overview", Air Force Research Laboratory. Available at: https://www.ainfosec.com/wp-content/uploads/2015/09/SecureView_Overview_Master_PA_Cleared_7Oct13.pdf.

Air Force Research Lab (2021) "AFRL_SecureView_FS_0922", Air Force Research Lab. Available at: https://afresearchlab.com/wp-content/uploads/2021/04/AFRL_SecureView_FS_0922.pdf (Accessed: November 5, 2024).

BAE Systems (no date) "STOP™ High Assurance GPOS" *BAE Systems*. Available at: https://www.baesystems.com/en/product/stop-os (Accessed: November 9, 2024).

BAE Systems (2020) "Data Diode Maximum assurance for unidirectional throughput", BAE Systems.

BAE Systems (2022) "XTS® Diode One Way Transfer Solution", BAE Systems. Available at: https://www.baesystems.com/en-media/uploadFile/20230504165402/1434650747473.pdf.

Barham, P. *et al.* (2003) "Xen and the art of virtualization," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*. *SOSP03: ACM Symposium on Operating Systems Principles*, Bolton Landing NY USA: ACM, pp. 164–177. Available at: https://doi.org/10.1145/945445.945462.

Bedrock Systems Inc. (2022) "Bedrock Hypervisor (BHV) User Guide" (2022), BedRock Systems Inc.

Campbell, S. (2019) "A Brief Note on Raise the Bar and One-Way Transfer," *OWL Cyber Defense*, 18 September. Available at: https://owlcyberdefense.com/blog/a-brief-note-on-raise-the-bar-and-one-way-transfer/ (Accessed: January 15, 2025).

Dahlstrom, J. *et al.* (2019) "Hardening Containers for Cross-Domain Applications," in *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*. *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*, Norfolk, VA, USA: IEEE, pp. 1–6. Available at: https://doi.org/10.1109/MILCOM47813.2019.9020992.

Daughety, N. *et al.* (2021) "vCDS: A Virtualized Cross Domain Solution Architecture," in *MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM)*. *MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM)*, San Diego, CA, USA: IEEE, pp. 61–68. Available at: https://doi.org/10.1109/MILCOM52596.2021.9652903.

Everfox (2024) "Datasheet High Speed Verifier (HSV)" (no date), Everfox. Available at: https://cms-assets.everfox.com/app/uploads/2024/03/19161730/Datasheet_High-_Speed_Verifier_HSV-en-021324.pdf.

General Dynamics Mission Systems, Inc (2025) "Frequently Asked Questions For Cross Domain Solutions", *General Dynamics Mission Systems*. Available at: https://gdmissionsystems.com/cross-domain-solutions/faqs (Accessed: January 12, 2024).

Goldberg, R.P. (1974) "A survey of virtual machine research," *Computer*, 7(6), pp. 34–45.

Green Hills Software (2020) "Creating a Trusted Embedded Platform for MLS Applications", Green Hills Software. Available at: https://www.ghs.com/download/whitepapers/GHS_Trusted_Platform_MLS.pdf.

Hu, Z. *et al.* (2023) "Realtime Malicious Traffic Detection Targeted for TCP Out-of-Order Packets Based on FPGA," *IEEE Access*, 11, pp. 112212–112222. Available at: https://doi.org/10.1109/ACCESS.2023.3323853.

Li, Z. *et al.* (2023) "RTISM: Real-Time Inter-VM Communication Based on Shared Memory for Mixed-Criticality Flows," in *2023 IEEE Real-Time Systems Symposium (RTSS)*. *2023 IEEE Real-Time Systems Symposium (RTSS)*, Taipei, Taiwan: IEEE, pp. 279–290. Available at: https://doi.org/10.1109/RTSS59052.2023.00032.

McNeil, S. *et al.* (2021) "Isolation Methods in Zynq UltraScale+ MPSoCs."

The MITRE Corporation (2018) "CAPEC-122: Privilege Abuse." Available at: https://capec.mitre.org/data/definitions/122.html (Accessed: January 28, 2025).

The MITRE Corporation (2020) "CAPEC-113: Interface Manipulation." Available at: https://capec.mitre.org/data/definitions/113.html (Accessed: January 28, 2025).

National Security Agency (no date) "National Cross Domain Strategy & Management Office", *National Security Agency/Central Security Service*. Available at: https://www.nsa.gov/Cybersecurity/Partnership/National-Cross-Domain-Strategy-Management-Office/ (Accessed: November 9, 2024).

Pamnani, V. and Matarazzo, P. (2024) "Trusted Platform Module Technology Overview," *Microsoft Learn*, 10 July. Available at: https://learn.microsoft.com/en-us/windows/security/hardware-security/tpm/trusted-platform-module-overview (Accessed: November 2, 2024).

Ren, Y. *et al.* (2016) "Shared-Memory Optimizations for Inter-Virtual-Machine Communication," *ACM Computing Surveys*, 48(4), pp. 1–42. Available at: https://doi.org/10.1145/2847562.

Sundaravarathan, V. *et al.* (2024) "Cross-Domain Solutions (CDS): A Comprehensive Survey," *IEEE Access*, pp. 1–1. Available at: https://doi.org/10.1109/ACCESS.2024.3483659.

Swarup, V. (1994) "Automatic Generation of High Assurance Security Guard Filters", *Proceedings of the 17th National Computer Security Conference (NCSC*, Volume 1), pp. 123–131.

Taylor, S. *et al.* (2022) "Automatic Construction of Hardware Traffic Validators," *European Conference on Cyber Warfare and Security*, 21(1), pp. 60–69. Available at: https://doi.org/10.34190/eccws.21.1.200.

Taylor, S. *et al.* (no date) "thayer_parsers." Available at: https://github.com/lvln/thayer_parsers (Accessed: February 10, 2025).

Taylor, S. and Pope, G. (2024) "Hardware Sequence Combinators," *International Conference on Cyber Warfare and Security*, 19(1), pp. 358–365. Available at: https://doi.org/10.34190/iccws.19.1.1965.

Thyagaturu, A.S. *et al.* (2022) "Operating Systems and Hypervisors for Network Functions: A Survey of Enabling Technologies and Research Studies," *IEEE Access*, 10, pp. 79825–79873. Available at: https://doi.org/10.1109/ACCESS.2022.3194913.

"Zynq UltraScale+ Device Technical Reference Manual" (2023).