Point-and-Click Analysis and Hardening to Enhance FPGA Bitstream Security

David Torres
FPGA Security Lead
Red Balloon Security, Inc.
New York, NY USA
davidt@redballoonsecurity.com

Grant Skipper
Naval Surface Warfare Center, Crane
U.S. Navy
Crane, IN USA
grant.a.skipper.ctr@us.navy.mil

Andrew Taub
Commercial Lead
Red Balloon Security, Inc.
New York, NY USA
andrew@redballoonsecurity.com

***Abstract*—Securing a Field Programmable Gate Array (FPGA) bitstream is critical for protecting against cyber vulnerabilities and attacks. Researchers who focus on exploitation of FPGAs study and develop offensive techniques targeting bitstreams. FPGAs deployed in critical infrastructure and other sensitive systems could be configured with unsecured bitstreams and it may be impossible to regenerate those bitstreams and apply recommended security settings without source code. Until recently, there had been no third-party tools to unpack, modify, or repack a generated bitstream developed with vendor software. However, it is now possible to unpack, modify, and harden a bitstream without using vendor software. We devised a method that includes such functionality, augmenting bitstream security and delivering assurance. Under guidance from the National Security Agency's Joint Federated Assurance Center (JFAC) Hardware Assurance Lab, we developed a point-and-click software tool called Bitwise to identify whether a bitstream is vulnerable to Threat Descriptions (TDs) as defined in the Department of Defense's Levels of Assurance (LoA) guidelines, specifically TD-1 (Adversary utilizes a known FPGA platform vulnerability) and TD-6 (Adversary swaps configuration file on target). Bitwise can be used as an evaluation tool that identifies, analyzes, and modifies bitstream header part configuration data, allowing for the successful assessment and redeployment of developed bitstreams. The tool does not require vendor software or source code, as well as other constraint files that are typically necessary to recompile a successful bitstream. Additionally, Bitwise can identify whether Common Vulnerabilities and Exposures (CVEs) are present in a bitstream and provide automated mitigation.**

I. Introduction

The FPGA market was valued at $12.1 billion in 2024 and is estimated to reach $25.8 billion by 2029 [1]. FPGAs are used in a range of critical industries including industrial, aerospace, telecommunication, and automotive. Supplying to both government and commercial markets, FPGA vendors dynamically meet customers' needs by developing proprietary software and hardware. Current designs are integrated to support sophisticated FPGAs that have modernized security features while legacy designs, which are not hardened against different types of targeted attacks, may be vulnerable and could be overlooked. Implementing FPGA bitstream security involves enabling specific vendor tool settings during the design flow which can be daunting and confusing to designers. Through collaboration with the National Security Agency's Joint Federated Assurance Center (JFAC), we present the ability to ensure that security settings are correctly implemented in a FPGA designs without the use of vendor software.

Bitwise (Bitstream Wizard for Intelligent Security Enhancements) is a security verification tool for FPGA bitstreams capable of mitigating against known CVEs, while also positioned to mitigate and defend against future CVEs. By analyzing the bitstream, it is technically feasible to understand a FPGA's security configuration. Using this capability, Bitwise is able to:

- **Transform**: Take an unsecured bitstream and apply best practice security settings, such as encryption and authentication. Deliver a rapid utility for maintaining and updating security settings over the lifespan of a design, such as updating keys without having to regenerate the entire design.
- **Harden**: Remove and inject commands to prevent breaking of encryption.
- **Analyze**: Evaluate & validate security configurations, identify risks & mitigations, and generate reporting that maps to Levels of Assurance (LoA) guidelines.

The bitstream header information contains information about the contents of the entire bitstream. Security settings of the bitstream are in the header information and may be set in the constraints file using Tool Command Language (TCL). To modify settings in the bitstream, the project needs to be synthesized, compiled, and a new bitstream generated. The ability to Transform, Harden and Analyze the bitstream using Bitwise reduces the amount of time a designer needs to evaluate and make modifications to the bitstreams header information.

Just like other hardware and software, FPGAs and bitstreams face vulnerabilities and design flaws. One example is the Starbleed vulnerability which affects Xilinx 7 series FPGAs and was disclosed in 2020 [2]. Xilinx published a Design Advisory for the 7 Series and Virtex 6 FPGAs in response to the Starbleed finding [3]. For affected FPGAs with designs that were created before the disclosure of Starbleed and that are already deployed, Bitwise can help mitigate this vulnerability by evaluating the bitstream without having to recreate the project.

An attacker can make certain assumptions about a bitstream once it is generated by Vivado, Xilinx's FPGA development tool. Hardening the bitstream is a technique that can be used to aggravate an attacker's approach to compromising a secure bitstream. Bitwise can easily help mitigate attacks targeting the Starbleed vulnerability through: (i) configuration shuffling (randomizing order of packets); (ii) injecting "no operation" (NOP) commands; (iii) randomizing NOPs; or (iv) removing all write WBSTAR packets. Hardening allows at a minimum 16-bits and up to 27-bits of entropy. In this scenario, 2^{27} creates a very low probability of 1 in 134,217,728 for an attack to be successful.

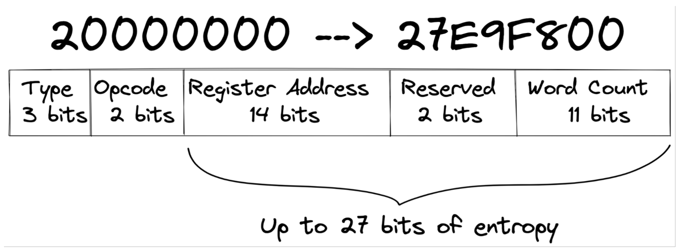


Fig. 1. Example effect of applying static bitstream hardening.

Bitwise provides the end-user the ability to open and analyze a bitstream. The end-user can rapidly modify the bitstream's configuration settings. With no knowledge of vendor software, the bitstream can be evaluated, modified, and redeployed on the FPGA. This helps to reduce the time needed to implement mitigation techniques for known CVEs or ones protected by a vendor's Non-Disclosure Agreement (NDA).

Researchers are consistently publishing innovative techniques to compromise bitstream security. FPGAs are prone to crude and sophisticated attacks, and bitstreams are attractive attack vectors. Intellectual property (IP) theft, harm to FPGA-based systems, and significant data loss are all associated with FPGA security threats [4]. IP theft is a cause for major financial and reputational damage, reportedly in the range of hundreds of billions of dollars annually in the U.S. alone [5]. Protecting against IP theft begins with properly securing the bitstream by properly configuring the FPGA. As vulnerabilities arise, Bitwise provides an automated solution to validate and mitigate known attacks. Cloning and hardware trojan insertion are a few possibilities inherent to compromising a secure bitstream. Cloning involves copying and utilizing the bitstream without permission of the design's owner. Once the device's security is compromised, a malicious actor can insert a hardware trojan, and the bitstream's authenticity and confidentiality may be lost. These attacks are directed towards the FPGA's configuration engine. To help defend against such types of threats, the National Security Agency developed and publicly published a Cybersecurity Technical Report titled, "Department of Defense (DoD) Microelectronics: FPGA Level of Assurance (LoA) 1 Best Practices." This document provides recommendations for hardware assurance strategies, and Bitwise has the ability to analyze, manipulate, harden, and report on the security strength of FPGA bitstreams as outlined in LoA1.

LoA1 has nine Threat Descriptions (TDs) which describe different ways for how an adversary could compromise or attack a FPGA [6]:

1. Utilizes a known FPGA platform vulnerability
2. Inserts malicious counterfeit
3. Compromises application design cycle
4. Compromises system assembly, keying, or provisioning
5. Compromises third-party soft intellectual property (IP)
6. Swaps configuration file on target
7. Substitutes modified FPGA software design suite
8. Modifies FPGA platform at family at design
9. Compromises single-board computing system

Bitwise provides assurance for TD1 (an adversary utilizes a known FPGA platform vulnerability) and TD6 (an adversary swaps the configuration file on target). Referring to TD1, this attack is when an adversary utilizes a vulnerability in a FPGA platform or vendor development software package to initiate an attack. For the LoA1 guidelines, a vulnerability is an unclassified published weakness in the design of a specific FPGA platform or software program that would allow the attacker the ability to use it for malicious purposes [6]. Referring to TD6, this attack is when an adversary obtains access to the system during or after assembly and can compromise the FPGA device's operation via the configuration data [6].

FPGAs are notoriously difficult to secure. The first FPGAs could only hold simple designs. The chips did not have very many logical cells, whereas modern chips contain millions. With the growth of hardware, vendor tools used for development of FPGA designs have become increasingly complex while providing flexibility to accommodate user needs. Many attacks are published specifically in attacking the configuration engine of the FPGA. Digital circuit designers may not fully understand security characteristics for one FPGA family, let alone many families. The intersection between FPGA designers and people who know FPGA security is very small. Most engineers view FPGA configuration as "black magic" and do not have the expertise to actually secure their bitstreams. Bitwise automates this process. During analysis, Bitwise provides a report detailing the security posture of the bitstream. If the device does not meet security requirements, recommendations are provided in the report. An engineer who does not have the knowledge or experience to properly secure the FPGA can do so with an automated approach.

II. Motivation

The National Security Agency's JFAC Hardware Assurance Lab developed and released four Cybersecurity Technical Reports. The purpose of these documents is to help the Department of Defense protect FPGA-based systems. The following reports are: the overall assurance process, best practices, LoA1, and Third-Party IP Review Process for LoA1.

Reviewing the FPGA strategy of outcomes, DoD uses FPGAs in critical systems. Vulnerabilities exist that were discovered after designs have been implemented and fielded [7]. There is no easy way to assess and or modify existing bitstreams in the wild. It is common during development to need project files and specific tools with source code in order to recreate a bitstream and evaluate security settings before developing the new bitstream. Even if a bitstream is developed, there is no easy way to verify the security settings of the specific FPGA bitstream in use. Being able to perform analysis via an automated report examining an FPGA bitstream's security and configuration data is a powerful approach that is implemented in our tool. Devices being used in the wild may be obsolete and the projects that designed not obtainable. Bitwise provides a solution to transform, evaluate, and harden those bitstreams.

III. Threat Model

Intended consequences from attacking the configuration engine result in compromising bitstream's authenticity and confidentiality. Defining the threat model, we focus on the following three attack vectors: (i) reverse engineering; (ii) cloning; and (iii) trojan insertion of the design. Breaking the bitstream's confidentiality, designs can easily be cloned. This provides an environment to easily counterfeit applications negatively affecting the supply chain. Another scenario is if the bitstream is compromised, skilled engineers can reverse engineer and gain important details to the design. These details can be used maliciously to plan attacks on the device. Another attack is the insertion of a hardware trojan. Moreover, it can be used to understand the functionality of the design. Hardware trojans are based on tampering with the bitstream. Attackers can insert logic to leak design details, mainly crypto keys from a design. By the same token, an attacker could disable a design by making critical modifications that could render an FPGA useless. Another possible attack is replacing the existing bitstream with a new bitstream which requires no reverse engineering of the existing bitstream. This demonstrates that protecting the authenticity and confidentiality of the bitstream is critical, especially to national security and DoD Systems.

IV. Meet Bitwise

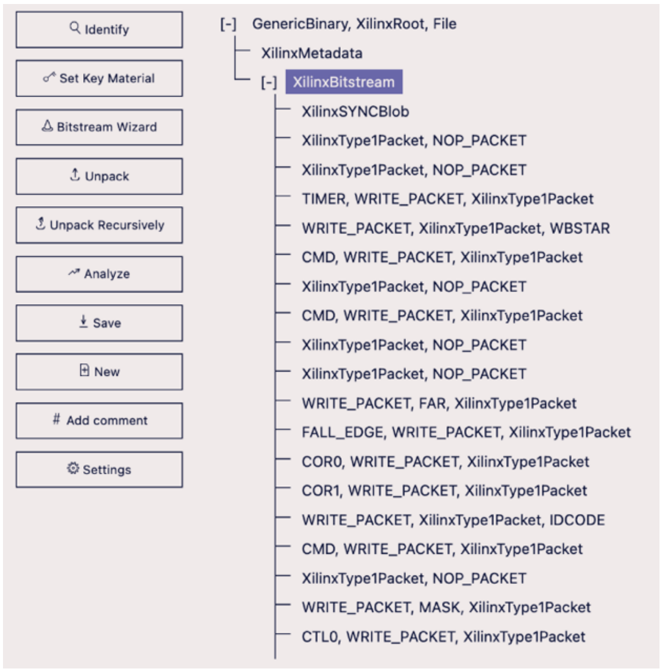
Bitwise is a FPGA assurance tool powered by the Open Firmware Reverse Analysis Konsole (OFRAK) which is a source available Python platform for binary analysis and modification [8]. Bitwise currently supports Xilinx UltraScale and UltraScale+ parts and we are actively expanding support to other FPGAs.

TABLE I.

Bitwise Compatibility		
Manufacturer	***FPGA Family***	***Est. Timeline***
Xilinx	UltraScale, UltraScale+	Complete
Xilinx	7-Series, 6-Series	1H 2024
Intel	Agilex, Stratix, Cyclone	2024
Microchip	PolarFire and/or IGLOO	2024

A. Analysis

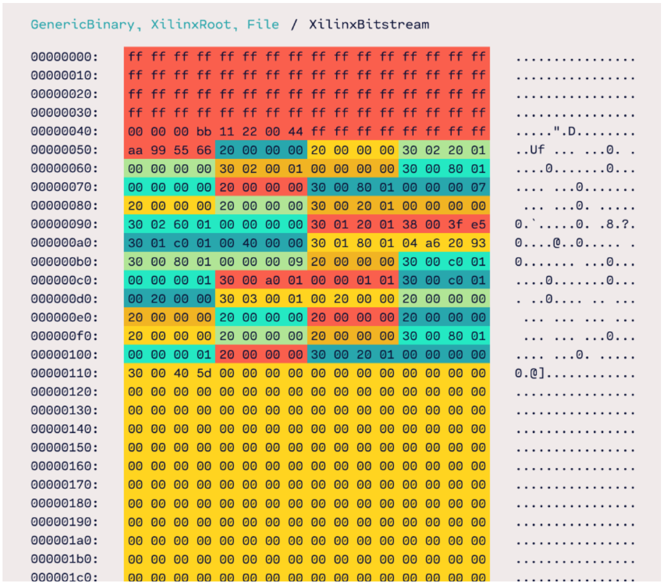
Bitwise and OFRAK allow a user to conduct bitstream verification for hardware assurance. Bitwise is designed to be a drag-and-drop, single push button solution to assist in bitstream analysis and verification. The Graphical User Interface (GUI) displays two panes: the Tree View as shown in Fig.2, and the Hex View as shown in Fig.3. "Set Key Material" allows a user to encrypt and re-encrypt a bitstream with an encryption key. For example, a designer can encrypt a bitstream with a key and later remove or change the key. The "Bitstream Wizard" allows a designer to obfuscate and provide further entropy to a bitstream. "Unpack" allows a user to unpack the bitstream's header information into a list describing its subcomponents which detail write packets, no operation packets, and more. "Analyze" provides information on the security posture of the bitstream, including a detailed report on the security of the bitstream mapping to JFAC's LoA1 guidelines.



```
[-] GenericBinary, XilinxRoot, File
    XilinxMetadata
    [-] XilinxBitstream
        XilinxSYNCBlob
        XilinxType1Packet, NOP_PACKET
        XilinxType1Packet, NOP_PACKET
        TIMER, WRITE_PACKET, XilinxType1Packet
        WRITE_PACKET, XilinxType1Packet, WBSTAR
        CMD, WRITE_PACKET, XilinxType1Packet
        XilinxType1Packet, NOP_PACKET
        CMD, WRITE_PACKET, XilinxType1Packet
        XilinxType1Packet, NOP_PACKET
        XilinxType1Packet, NOP_PACKET
        WRITE_PACKET, FAR, XilinxType1Packet
        FALL_EDGE, WRITE_PACKET, XilinxType1Packet
        COR0, WRITE_PACKET, XilinxType1Packet
        COR1, WRITE_PACKET, XilinxType1Packet
        WRITE_PACKET, XilinxType1Packet, IDCODE
        CMD, WRITE_PACKET, XilinxType1Packet
        XilinxType1Packet, NOP_PACKET
        WRITE_PACKET, MASK, XilinxType1Packet
        CTL0, WRITE_PACKET, XilinxType1Packet
```

Fig. 2. Unpacking a bitstream's header information and listing its subcomponents using Bitwise's Tree View.

In the Hex View, a user can visualize the bitstream header information, including the size and types of packets that are in the header information. In addition to this high-level GUI based wizard, Bitwise offers an API for modifying and updating granular aspects of the configuration bitstream file. After modifications are made, the Hex View displays those changes.



```
GenericBinary, XilinxRoot, File / XilinxBitstream

00000000:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff   ................
00000010:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff   ................
00000020:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff   ................
00000030:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff   ................
00000040:  00 00 00 bb 11 22 00 44 ff ff ff ff ff ff ff ff   .....".D........
00000050:  aa 99 55 66 20 00 00 00 20 00 00 00 30 02 20 01   ..Uf ... ...0. .
00000060:  00 00 00 00 30 02 00 01 00 00 00 00 30 00 80 01   ....0.......0...
00000070:  00 00 00 00 20 00 00 00 30 00 80 01 00 00 00 07   .... ...0.......
00000080:  20 00 00 00 20 00 00 00 30 00 20 01 00 00 00 00    ... ...0. .....
00000090:  30 02 60 01 00 00 00 00 30 01 20 01 38 00 3f e5   0.`.....0. .8.?.
000000a0:  30 01 c0 01 00 40 00 00 30 01 80 01 04 a6 20 93   0....@..0..... .
000000b0:  30 00 80 01 00 00 00 09 20 00 00 00 30 00 c0 01   0....... ...0...
000000c0:  00 00 00 01 30 00 a0 01 00 00 01 01 30 00 c0 01   ....0.......0...
000000d0:  00 20 00 00 30 03 00 01 00 20 00 00 20 00 00 00   . ..0.... .. ...
000000e0:  20 00 00 00 20 00 00 00 20 00 00 00 20 00 00 00    ... ... ... ...
000000f0:  20 00 00 00 20 00 00 00 20 00 00 00 30 00 80 01    ... ... ...0...
00000100:  00 00 00 01 20 00 00 00 30 00 20 01 00 00 00 00   .... ...0. .....
00000110:  30 00 40 5d 00 00 00 00 00 00 00 00 00 00 00 00   0.@]............
00000120:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000130:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000140:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000150:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000160:  00 01 c0 01 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000170:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000180:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000190:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001a0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001b0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001c0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

Fig. 3. Visualizing a bitstream's header information and the size and types of packets in the header information using Bitwise's Hex View.

B. Security

Bitwise's wizard feature provides a variety of bitstream transformation techniques. Bitwise's bitstream wizard supports plaintext to AES GCM, microbitstream, and report generation. Bitwise has the ability to convert a non-encrypted bitstream into an encrypted bitstream. By specifying the AES key and the initialization vector (IV), a user may convert the bitstream from plaintext to GCM.

Once the bitstream is encrypted, Bitwise can detect the bitstream state. Bitwise can unpack the bitstream while it is encrypted. The bitstream wizard has different selections with an encrypted bitstream: GCM to plaintext, Re-encrypt GCM, Harden, and Generate a report. The hardening of the bitstream allows a user to mitigate against known CVEs pertaining to the configuration engine. One example is the Starbleed attack [2], [9]. Bitwise has the following selections: perform WBSTAR autonomy, shuffle commands, inject NOPs, and Salt NOPs. WBSTAR autonomy will remove the write to the WBSTAR register command. Where shuffling commands makes it harder for an adversary working with an encrypted bitstream. This makes it harder to modify certain write commands. NOPs being added modifies the target bitstream changing the actual footprint of the configuration data making it harder to attack specific write commands. Salting NOPs is a way to further obfuscate the configuration header data in the bitstream. Hardening techniques do not prevent an attacker from attacking the bitstream; it makes an attack much more difficult to perform, especially brute force attacks.

C. Reporting

For security reporting, Bitwise can automatically generate a report providing information to the end-user. In the security report, Bitwise details general metadata, details on the security, and configuration details. The general metadata displays design, part, and creation details. Design details explain the design name, whether it has been compressed, and the tool version that was used to design the bitstream. Part information will give the exact part number for the bitstream design. The configuration details in the report display packet types and a raw count of those packets in the configuration header of the bitstream. Based on the security report, Bitwise recommends modifications to the bitstream. In the report, Bitwise can rapidly evaluate the bitstream's security posture according to the DoD's Level of Assurance (LoA) guidelines, stating the compliance-level for the target bitstream and providing actionable feedback that can be used to achieve compliance.

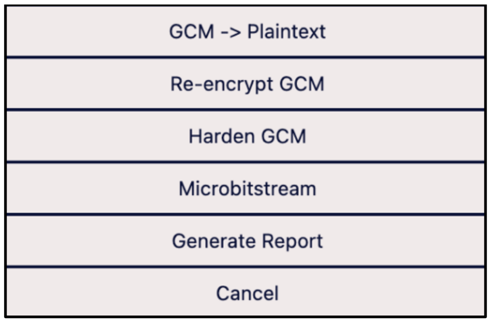


Fig. 4. Apply point-and-click transformation, hardening, and analysis all through Bitwise's wizard.

D. Evaluation

Lastly, Bitwise has the ability to evaluate bitstreams that are encrypted in RSA. This ability allows the user to have the same evaluation as the AES encrypted bitstream. Also, Bitwise supports Per Frame CRC32 bitstream configuration. This takes the actual bitstream as one big blob of configuration data and splits it into individual type1 packets. Bitwise is designed to look at the security of the FPGA through auditing the configuration settings in the header information of the bitstream. The user does not need to know or use vendor software. No source code is needed to recreate and synthesize the project. providing actionable feedback that can be used to achieve compliance.

V. New Technology Integration

Bitwise makes available, for the first time, the Microbitstream enhanced compression technology previously announced by the Naval Sea Systems Command (NAVSEA) Crane during GOMACTech 2023 [10]. Bitwise allows users to compress bitstream configuration to the minimum necessary components necessary to program a FPGA. As an example, the advantage of Microbitstream compression over the standard MFWR (Multiple Frame Write/Read) compression natively supported in Vivado is that Microbitstreams will always result in higher compression ratios. The effect of Microbitstream compression is that both file size and total programming time are dramatically reduced, especially for smaller RTL designs or partial reconfiguration scenarios. Microbitstreams are explicitly supported for bitstreams generated with Per Frame CRC.

VI. Design And Operation

Bitwise is designed to provide assurance capabilities for FPGAs in support of the Joint Federated Assurance Center (JFAC) for verifying and securing FPGA configuration bitstreams. Bitwise is developed on top of the Open Firmware Reverse Analysis Konsole (OFRAK), which provides the capability to unpack, analyze, modify, and repack the bitstream without vendor software tools – an essential set of features.

It is imperative to validate FPGA configuration settings by applying an automated FPGA assurance capability. Bitwise can perform automated bitstream reduction for Xilinx fabric-based FPGA configurations. Reduced bitstreams contain the minimal content required to successfully program a bitstream. Operationally, existing bitstreams can be analyzed for post-process reduction. Bitwise provides the capability of encrypting reduced bitstreams using the existing AES-GCM crypto-system on Xilinx devices. Encryption of reduced bitstreams is a configurable option. To mitigate against known attacks on the FPGA's configuration engine, a security technique is available in the form of encrypted command shuffling. Command shuffling offers a mitigation technique against known CVEs. This feature randomizes, replaces, and removes bitstream commands found at the beginning of encrypted Xilinx bitstreams. This randomization provides a fast, automated method for increasing the effort and technical requirements necessary to exploit known CVEs on FPGAs. This technique facilitates a way to submit pre-existing encrypted bitstreams to Bitwise and automatically patch configurations.

Bitwise can also receive an existing bitstream and generate a report that verifies existing security features. Following JFAC's guidance, Bitwise incorporates recommended security features available to the identified parts. Recommendations are based on current best practices according to user-reported risk tolerance guidance. There is an option to automatically update the submitted bitstream with the security recommendations through Bitwise's wizard. Independently, encryption capabilities may be implemented with Bitwise to allow encryption, decryption or re-encryption of the bitstream without requiring regeneration through the vendor provided integrated development environment (IDE). Bitwise is also capable of importing independent encryption material or allowing to locally generate randomized cryptographic material necessary for bitstream encryption. Cryptographic content generated locally adheres to current best practices for key-generation and incorporates the industry standard OpenSSL cryptography library.

VII. Use Cases

As FPGAs are becoming more sophisticated, so are the tools that create bitstreams and firmware. Encryption, authentication, obfuscation, and secure boot are developed using design suite features specific to vendor applications. In a niche field, each vendor has its own proprietary software providing security to individual devices. Embedded Systems Engineers may not focus on or may overlook cyber threats due to vigorous project deadlines. Bitwise is intended to mitigate and provide a level of assurance currently not available in vendor software or easily executed by an FPGA design team in a cost effective manner. Researchers targeting FPGA exploits publish papers annually on real world threats that can be detrimental to the security of FPGAs. As Common Vulnerabilities and Exposures (CVEs) are published, it can be research intensive to find whether those CVEs may affect a design, since a CVE can be published after a design is implemented. Bitwise allows for the rapid deployment of mitigation techniques that a security expert can apply in response to the CVE. Bitwise provides this capability without use of vendor software or source code. Bitwise uses hardening techniques that enable obfuscation of a bitstream configuration data. This allows for designers to have full control over the bitstream in an easily applied manner. With FPGA tools, it is important to understand vulnerabilities that may reside in the FPGA design tool with application of remedial actions. Depending on the application, many design cycles occur over long periods of time and it can be challenging to apply best practices. New exploits may have been discovered as a result of research efforts, either private or public. Once the final bitstream is developed, it is packaged in a vendor specific file format. This file configures the part to be operational and initializes a custom design on silicone fabric. The header information acts as a first stage boot loader utilizing commands that configure security settings.

VIII. Future Work

Bitwise enhancements are being developed with intentions to be the premier FPGA design security settings evaluation tool. These enhancements will encompass the entire FPGA design by adding the ability to access and assess System on Chip (SoC) firmware. An FPGA bitstream's header information used in the device's configuration engine defines the device's security posture. FPGAs with SoCs are similar and the First Stage Boot Loader (FSBL), typically writen in C code, replaces the bitstream header information. In a bootable image, an FSBL configures the part using a partition header an array of structures containing information related to each partition. These additional features evaluating the FSBL would typically require a high-level of technical expertise and extended labor when performed manually.

Since Bitwise automates the evaluation of the bitstream's security framework, SoC FPGAs will be assessed similarly without using vendors' design software. Bitwise will provide an automated bootable image input file evaluation of the files in the SoC boot image, and users will be able to modify and customize the FSBL. An analyzer will be provided to analyze the bootable image. Bitwise will have complete control of the FSBL and its functionality. Known attacks can be prevented with an evaluation of the FSBL and the Partition Header Table (PHT). The PHT is utilized by the FSBL to acquire information about each partition during the boot procedure. Bitwise will be able to analyze the contents in the bootable image to include PHT and FSBL settings. A report will be provided that identifies the contents of the bootable image: security settings, data in image contents, data in the FSBL and PHT contents, and through following the API, providing a list containing the configuration data to mitigate a known vulnerability.

Bitwise will be able to encrypt, decrypt, sign, and verify the bootable image and its contents without using vendor software. For SoC devices, the BootROM and the FSBL decrypt partitions during the booting cycle. The BootROM reads the FSBL from flash, decrypts, loads, and hands off the control. After the FSBL starts executing, it reads the remaining partitions, decrypts, and loads them. AMD Xilinx Zynq SoCs use the embedded Programmable Logic (PL) hash-based message authentication code (HMAC) and an advanced encryption standard (AES) module with a cipher block chaining (CBC) mode. Additionally, if secure boot is not desired, then software can at least be validated with a simple checksum. Bitwise will be able to support key generation, encrypting device partitions, operational key, rolling keys and checksum as defined in User Guide (UG) UG1400 [11]. This functionality will mitigate if an adversary were to swap the configuration file on the target, addressing Threat Description-6 (TD-6).

Bitwise will be able to audit the bitstream, fsbl.elf, user code for bare-metal applications, and files containing user keys. Original hardware definition files, the bitstream, and keying files can be compared to the files Vivado creates before the file is reported to Vitis. A comparison analysis will be provided by comparing the bitstream with the original bitstream to verify it has not been modified. Vitis or SDK create the FSBL and user code to configure and run on the processor. Bitwise shall be able to provide a comparison analysis in a report using the files in Vitis before the bootable image is created. This functionality facilitates detecting if an adversary were to substitute modified FPGA software design suite by auditing or verifying plaintext contents of programming images against insertions and/or modifications during encryption or build by vendor tools, addressing TD-7.

The FSBL can contain unnecessary code or debug-code that causes the image to be larger in size. Through reduction, Bitwise will be able to remove unnecessary code and modify code in the FSBL. Like bitstream reduction, the FSBL can also be reduced. The ability to modify the FSBL is a hardening technique that will allow the user to insert or remove code from the FSBL without using vendor software. This functionality will be able to mitigate against current CVEs with the flexibility to be used against future CVEs, as well as protected CVEs that only specific contractors know about through vendor NDAs. This functionality will facilitate mitigation of a known vulnerability, addressing TD-1.

Some designs using a Kernel instead of a bare-metal application use a Second Stage Boot loader (SSBL). The SSBL will be evaluated simular to the FSBL. If the design utilizes a Kernel, without using any kernel development tools analysis of the Kernel shall be provided. The abillity to unpack the kernel and evaluate its contents will ensure the desired functionality is implemented. The user shall be able to modify the kernel, even write code that will be functional on the device.

REFERENCES

[1] Markets and Markets. "FPGA Market Market Forecast to 2029." [Online]. Available: https://www.marketsandmarkets.com/PressReleases/fpga.asp (access Jan. 16, 2024).

[2] M. Ender, A. Moradi, and C. Paar, "The Unpatchable Silicon: A Full Break of the Bitstream Encryption of Xilinx 7-Series," 29th USENIX Security Symposium, August 2020.

[3] 73541 - Design Advisory for 7 Series/Virtex-6 FPGAs: Defeating Bitstream Encryption. [Online]. Available: https://support.xilinx.com/s/article/73541?language=en_US (access Jan. 25, 2024).

[4] H. Palmer, "FPGA Security Vulnerabilities and Countermeasures," Electrical Design, March 2023.

[5] S. Klix, N. Albartus, J. Speith, P. Staat, A. Verstege, A. Wilde, D. Lammers, J. Langheinrich, C. Kison, S. Sester, D. Holcomb, and C. Paar, Stealing Maggie's Secrets -- On the Challenges of IP Theft Through FPGA Reverse Engineering," December 2023.

[6] DoD Microelectronics: FPGA Level of Assurance 1 Best Practices, U/OO/230110-22, PP-22-1270, Ver. 1.0, December 2022.

[7] R. Shanahan, "Field Programmable Gate Array (FPGA) Assurance," 20th Annual NDIA Systems Engineering Conference, October 2017.

[8] Open Firmware Reverse Analysis Konsole (OFRAK) on GitHub. [Online]. Available: https://github.com/redballoonsecurity/ofrak (access Jan. 22, 2024).

[9] M. Ender, G. Leander, A. Moradi and C. Paar, "A Cautionary Note on Protecting Xilinx' UltraScale(+) Bitstream Encryption and Authentication Engine," IEEE 30th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), New York City, NY, USA, 2022 pp. 1-9.

[10] M. C. Sozio, G. Skipper, D. Hansen, A. Lukefahr and A. Duncan, "MicroBitstreams: Reducing Configuration Time of Encrypted Bitstreams," *2023 IEEE Physical Assurance and Inspection of Electronics (PAINE)*, Huntsville, AL, USA, 2023, pp. 1-7.

[11] Vitis Unified Software Platform Documentation: Embedded Software Development (UG1400). [Online]. Available: https://docs.xilinx.com/r/en-US/ug1400-vitis-embedded (access Jan. 25, 2024).

APPENDIX

The following table highlights Bitwise's full feature set.

TABLE II.

BITWISE FEATURES
TRANSFORMATION
• Decrypts and re-encrypts a bitstream with a new key
• Converts a plaintext bitstream into an AES-GCM encrypted bitstream
• Converts an AES-GCM encrypted bitstream into a plaintext bitstream
• Verifies, re-signs, and re-encrypts RSA authenticated bitstream
• Performs microbitstream compression with the option of encryption
HARDENING
• Remove commands for hardening purposes (e.g., WBSTAR)
• Salt-injected NOPs to make encryption harder to solve
• Shuffle commands to prevent breaking of encryption
• Inject NOPs to prevent breaking of encryption
SECURITY VALIDATION
• Detects method of authentication of a bitstream
• Detects encryption used on a bitstream
• Detects whether a Warm Boot Start register is used within a bitstream
• Detects whether BRAM or eFUSE key storage is used
• Detects encryption scheme of encrypted bitstream
LEVELS OF ASSURANCE (LOA) ASSESSMENT
• Provides LoA compliance assessment for specific threat descriptions
• Provide security score with regard to LoA compliance
• States criteria to become compliant
• Recommends steps (when applicable) to achieve compliance
• Provides information extracted from metadata when available