



**NOVEMBER 2020**

## **THE PERFECT WEAPON, HIDDEN IN PLAIN SIGHT**

---

A Study on How the Espressif Wi-Fi and BLE Chips and Modules can be Weaponized for Espionage, Disruption, and Destruction

**Authored By:**

Drew Spaniel, Lead Researcher, ICIT

Itzik Kotler, ICIT Contributor & Co-Founder and CTO, SafeBreach

**Contributors Include:**

Joyce Hunter, Executive Director, ICIT

Malcolm Harkins, ICIT Fellow & Chief Security and Trust Officer, Cymatic

Jerry Davis, ICIT Fellow & Founder, GryphonX LLC

## Contents

Introduction .....	2
The ESP8266 and ESP32 Could be Leveraged as “Perfect Weapons” .....	3
Why Examine an Espressif Device? .....	4
This Publication Began with a Year-Old Forum Post.....	5
Do Espressif Devices Pose a Reasonable Risk? .....	6
Anatomy of the Attack Scenario .....	7
Danger Beneath the Surface .....	7
Model Campaign .....	7
Targeting Specific Devices.....	8
Attack Obfuscation.....	8
Device Synchronization .....	8
Attack Potential.....	9
Open Systems Interconnection Model .....	9
TCP/IP:.....	10
Demonstration of Attack Feasibility .....	11
Conclusion.....	15

## Introduction

In an August 20, 2020 webinar, Ellen Lord, the Undersecretary of Defense for Acquisition and Sustainment, commented that her office is considering incentives to bring microelectronic production and testing back to the United States so that the security and reliability of hardware can be more readily assessed. Currently, the vast majority of production and testing occurs outside the United States and poses a substantial risk to supply chain security because adversaries can introduce malware or backdoors onto subcomponents and then laterally compromise sensitive networks. As microelectronic components are vital to emerging technologies such as 5G wireless networks, artificial intelligence, quantum computing, weapons systems, and the Internet of Things (IoT), it is critical we improve our control over supply chains' security. At the Defense Advanced Research Projects Agency's Electronics Resurgence Initiative Summit, Lord commented,

We can no longer identify the pedigree of our micro-electronics, therefore, we can no longer ensure that backdoors, malicious code, or data exfiltration commands aren't embedded in our code. While we develop the ability to identify the technical path to ensure all components, circuits and systems are clean regardless of their manufacturing location, we need to find a path to domestic sources to provide a secure and resilient supply of legacy, state-of-the-present, and state-of-the-art microelectronics.

The Department of Defense is concerned about the security risks to microelectronics due to the potential impacts on national security and critical infrastructure systems. Cyberattacks at the microelectronic level are difficult to detect and can result in the disruption of critical services, the theft of sensitive information, and the destruction of vital systems. Furthermore, the potential impact is not limited to the national security space.

As a leader of critical infrastructure cybersecurity research, ICIT has warned about [supply chain security concerns in the past](#) and we have supported initiatives and frameworks like [Deliver Uncompromised](#). This publication will serve as the first in an on-going series focused on supply chain security. Subsequent publications will vary in content from high-level thought leadership to technical analysis.

Supply chain security is complicated as many systems are proprietary or rely on black-box code. Therefore, any meaningful analysis must rely on:

1. Hypotheses that are founded in logic and
2. Facts that are based on data and experimentation.

Over the course of this series, ICIT is going to provide as much logic and data as possible and as always, our analysis will be peer-reviewed by our fellows prior to publication. However, we recognize that in many cases we are only evaluating an aspect of an issue or a subset of the data. The case study discussed in this publication grew from an intriguing forum post (Appendix A), into a narrative rooted in conjecture and logic-driven hypotheticals, into a proof-of-concept benchmark that is relatively innocuous compared to the viable attack vector. This information is not presented as condemnation of

an individual company. Rather, it is presented in a digestible and actionable narrative form as a model for how stakeholders and consumers should critically evaluate supply chain security. ICIT encourages and welcomes further analysis and feedback from the national security community that supports or refutes any of the suppositions published in this brief case study.

As a final note, the authors would like to address the publication of this information. This publication focuses on a specific family of devices because it was inspired by a forum dialogue from November 2019. The Espressif moderator, who identifies as “a core Espressif engineer”, opens the dialogue by dismissively saying “this conspiracy theory comes up every now and again.” Nothing discussed in this publication therefore should be unknown to Espressif. In fact, the simple benchmark test detailed at the conclusion of the analysis as a proof of concept was achieved using 100 lines of publicly available code from Github and a newly purchased (Summer 2020), off the shelf Espressif ESP8266. This is not a vulnerability disclosure. Based on their engineer’s forum response and their renaming of an API library (which neutralized the Github code, but did not remediate the vulnerability as the new name of the library simply needs to be inserted) Espressif is aware of the attack vector discussed in this publication. In our further evaluation of Espressif’s devices, it made sense for ICIT researchers to profile basic information about the OEM in order to assess size, market position, and an estimation of security practices. In doing so, ICIT encountered some inconsistencies in market position and behavior that we found troubling in the same way that from a national security perspective ICIT, and the national security community, found Huawei and ZTE inconsistencies troubling. It is possible that completely innocuous explanations exist for how Espressif is able to significantly undercut market competitors or why after at least a year of this vector being publicly disclosed, the threat still remains in components that included in more than 400,000 devices per day. ICIT did not find answers for some of these questions because the information required is not publicly available. Instead, we found “smoke.” We leave it to the reader to independently evaluate the information and conclude whether there is “fire.”

## **The ESP8266 and ESP32 Could be Leveraged as “Perfect Weapons”**

This ICIT publication will discuss the risk posed to devices that rely on the ESP8266 or ESP32 Wi-Fi and Bluetooth low-energy (BLE) integrated circuits (ICs) and modules developed by the Shanghai-based Espressif Systems. Impacted devices include IoT thermostats, smart lightbulbs, smart outlets and switches, smart wearables, sensing devices, HVAC systems, home access control systems, telehealth and medical devices, and industrial controls. These chipsets can be found in consumer devices and critical infrastructure systems alike and the associated risks and potential threats permeate across all sectors. This proof-of-concept case study is constructed to highlight the potential risks pertaining to a single subset of devices, the Espressif ESP8266 and ESP32; however, the methodology and conclusions can be applied to many ICs and subcomponents from many manufacturers. This information is presented to inspire stakeholders to critically examine the security practices of all OEMs in their supply chain and to question anything that seems suspicious.

Infected Espressif devices would be a perfect weapon since they have a CPU, native memory, a 2.4 GHz integrated Wi-Fi antenna independent of other system-level services, and the ability download a poisoned update over-the-air. The infected component would only need a power connection from the host device. A similar attack using an intel CPU would not be feasible as the unit lacks the prerequisite independent features native to the ESP8266, ESP32, and similar devices. ICIT Fellow and Cymatic Chief Security and Trust Officer Malcolm Harkins explains, “This risk is the digital equivalent of installing a faulty junction box in your home. Not only is your home at risk of fire, but you could also inadvertently damage every other appliance or device.”

The attack paradigm detailed can be conducted programmatically without the need for any additional hardware other than the Espressif device. The device itself could contain the logical or algorithmic trigger for when to behave maliciously, create radio frequency interference, or otherwise disrupt operations. Instructions embedded in the device firmware could be used to enable malicious behavior under specific conditions, such as timing, geolocation, MAC address of connected devices, instructions delivered from a command and control (C2) server, BLE or WeChat connections, or other built-in functionalities.

Attack campaigns can be measured based on the means, motive, and opportunity of the adversary as well as the viability and potential impact of the attack vector. Overall, the Espressif devices have the potential to act as the perfect malicious tool to provide the means for an adversary to conduct widespread espionage, disruption, and destructive attacks. Indeed, from a hypothetical standing, similar to Huawei and ZTE, Espressif’s suspiciously low pricing could be indicative of a motivation other than profit maximization. Likewise, their increasing market position of approximately 400,000 new devices per day in an expanding subset of markets could prove a dangerous opportunity if the devices are indeed leveraged with malicious intent. Worse, in addition to being a perfect weapon, the attack vector is elegantly simple in its independence from an intentional or unintentional malicious insider. Malware is not positioned on the network via a phishing email nor an infected flash drive. It is plugged in and connected by uninformed consumers who purchased a low-cost device which included the embedded potential threat. If the purchased device behaves as intended most of the time, the consumer, who only wanted a cheap networked device, may never suspect that they inadvertently exposed their network to surveillance or disruption to save on the price of a smart light bulb, switch, thermostat, or other IoT device.

## Why Examine an Espressif Device?

Since its foundation in 2008, Espressif has been steadily increasing its market share by severely undercutting competitor prices. Hundreds of millions of cheap IoT devices on the market rely on the ESP8266 or ESP32 chipsets. Espressif Wi-Fi and BLE modules are popular due to their disproportionately low price point. As a result, the company is annually increasing its market share despite its relatively minute size, geographic origin, and questionable financials. The basic versions of the ESP32 and ESP8266 chipsets market for just under \$2 a piece when bought in 10,000 piece lots and under \$1 per chip in

some high volume applications, like light bulbs and switches, which are possibly their highest volume, most durable installation. Meanwhile, their competitors retail for around triple that price. While not indicative of nefarious activity, their market strategy is puzzling because it does not appear to be focused on generating revenue. In 2019, Espressif sold a total of 143.5 million chips and modules, according to their report to China's science and technology innovation board. Yet, in their 2019 financial report, Espressif declared a total revenue of ¥757.4 million (~\$113.61 million), despite ¥588.5 million (~\$88.28 million) in total operating expenses and resulting in ¥168.9 million (~\$25.34 million) in operating income. Their market strategy on its own is not indicative of any nefarious activity. But, if Espressif were to undercut markets so that the Chinese government could increase its network access or soft-power dominance, the implications could prove significant, especially considering that consumers are blindly enticed by the minimal price points of the ESP32 and ESP8266. In that scenario, profit margins would be subsidized for geopolitical, socio-economic, and technological influence.

### **This Publication Began with a Year-Old Forum Post**

In early November 2019, on the Espressif forums, user greengnu asked a compelling query, "How is it possible that the ESP32 is so cheap?" (Appendix A). They posited,

"Even if the whole wireless function would be left out we'd still have a dual core 240mhz 32bit cpu that can run complex applications for a dollar (soc). I couldn't find any other cpu in the same performance class that comes even close in terms of price performance. How is this pricing possible? Just saying. If I were the Chinese government and wanted to spy on the world through an armada of small IoT devices linked into every private wifi around the globe..."

When learning about the increasingly pervasive ubiquity of the Espressif products due to its extraordinarily low price points, ICIT researchers had the same suspicion.

An Espressif moderator, ESP\_Angus, responded within the hour to denounce the post as a conspiracy theory and to warn that it was off-topic. More importantly, and perhaps more telling, they provided a dismissal of the question, indicating that this query and the associated theory had come up before. The response indicated that Espressif was aware of the potential weaponization of their products. The moderator claimed to be an Espressif engineer, which, if true, is not insignificant for a company of only around 240 employees as they may be one of only a few dozen developers with internal knowledge behind the design of ICs like ESP32 and ESP8266. They defended Espressif's position by citing that:

"all network code in ESP32 above the Wi-Fi MAC layer is open source, the WPA2 integration is open source, and the build system and toolchain are all open source. So the binary Wi-Fi libraries somehow have to implement an entire 'shadow' firmware implementation that could take over the main firmware, no matter what that main firmware happens to do, and without any ability to analyze the real firmware source..."

While the debate surrounding the security of open source code is beyond the scope of this publication, the moderator's defense intentionally misdirects the conversation by omitting the potential for insider



threat or intentional compromise from Espressif itself. If malicious backdoors were added to the firmware, then the code could be operating as intended while still presenting a danger.

The moderator indemnified Espressif from nefarious intent by asserting that a backdoor would be too complex given the differing applications and attached peripherals of each device. However, in the age of big data and artificial intelligence, monitoring the metrics of a few hundred million devices or leveraging them in a coordinated attack is not out of the realm of possibility. After all, a low-level cybercriminal operated the Mirai botnet to leverage over 500,000 devices. A well-resourced organization, like the nation-state sponsored advanced persistent threat (APT) group, would not have difficulty tracking and strategically weaponizing a few hundred thousand or million IoT devices.

ESP\_Angus also claimed that engineering a cheap chip was far easier than the complexity necessary for the suggested conspiracy theory. However, if that were the case, why are Espressif's competitors unable to market chipsets at comparable prices? When further pressed on the price point, ESP\_Angus responded that Espressif can afford their low prices due to their "investors." That argument may sound definitive or convincing at a glance, but it does not withstand critical examination. Businesses exist to maximize profits for their investors. As such, it is illogical that Espressif's hypothetical "investors" would intentionally pay for Espressif to undercut competing firms while capturing diminished profits. ICIT did not delve deeper into Espressif's business model, but the overall response from the moderator evoked suspicion in our research team which inspired us to buy a device and run a simple benchmark test similar to what "greengnu" described.

### **Do Espressif Devices Pose a Reasonable Risk?**

In March 2019, ICIT released a research publication entitled "Did China Just Legalize Espionage: Recent Provisions to Chinese Law Increases Risk to Multinational Organizations Operating in China." The whitepaper summarized China's 2017 National Cybersecurity Law (CSL) and its 2018 "Regulations on Internet Security Supervision and Inspection by Public Security Organs" provision. The original National Cybersecurity Law permitted Chinese authorities the right to evaluate the source code of technologies used by foreign companies operating in China under the guise of identifying vulnerabilities through national security reviews. Some firms, such as Recorded Future, postulated that the law could be abused by Chinese-state agencies to identify zero-day flaws and other vulnerabilities that could later be exploited by state-sponsored attackers to compromise the systems of western companies and exfiltrate sensitive consumer data or intellectual property. The new provisions to the CSL, issued November 1, 2018, gave the Chinese Ministry of Public Security the legal authority to remotely conduct penetration tests on the systems and networks of any Internet-related business with at least five internet-connected computers operating in China. This means that even if companies like Espressif are well-intentioned, Chinese intelligence could deploy backdoors in their source code, disseminate malicious updates, or collect data from remote units. For downstream manufacturers, this means they may be infecting their consumers' networks with devices that are compromised at the original equipment manufacturer (OEM) level. As a result, defense, security, healthcare, and other critical infrastructure networks may be

jeopardized at an IC layer which their information security personnel do not audit and their layered defenses do not protect.

## Anatomy of the Attack Scenario

As mentioned earlier, Espressif markets their ESP8266 and ESP32 products at about a third of their competitor's price. Manufacturers and consumers who do not prioritize security as part of the acquisition process are likely to rely on Espressif components. If you have used an off-the-shelf Wi-Fi wall switch, light, or outlet, then there is significant chance that your device has relied on Espressif's Wi-Fi or Wi-Fi/Bluetooth modules for IoT applications. However, Espressif's reach is broader. Around 12 million Espressif modules are deployed in the supply chain each month. The company develops a wide variety of IoT IC for application in smart homes, healthcare, consumer electronics, and industrial automation. Individually, the compromise of one of these devices may not be significant or even noticed. However, if compromised en masse, an attacker could disrupt mission-critical operation, exfiltrate sensitive data, conduct network reconnaissance, or otherwise leverage the impromptu botnet in targeted attacks reminiscent of the Mirai botnet. A review of the previously disclosed vulnerabilities within Espressif devices is outside the purview of this publication. Instead, this publication will focus on the risk introduced to external parties if the firmware of Espressif devices, intentionally or unintentionally, included malicious code.

## Danger Beneath the Surface

The danger of Espressif devices lie in their firmware. Espressif ICs rely on black box proprietary firmware that could be infected with a malicious backdoor prior to distribution or remotely replaced with malicious code. A single malicious insider could infect a firmware update and compromise tens of millions of devices. Moreover, even if Espressif and its personnel were all well intentioned, the 2018 provision in the Chinese CSL should cause buyers to question the integrity of the underlying source code and conduct penetration tests when able.

## Model Campaign

In the aforementioned forum discourse, Espressif engineer ESP\_Angus commented,

This conspiracy theory comes up every now and again, and as a core Espressif engineer I've tried to figure how it would technically work. And I have no idea. Would all ESP32s ping a single server once a day to check if they should switch into 'backdoor mode'? And what would 'backdoor mode' do exactly, given every chip has different peripherals attached to it in different ways? And no one would notice this additional unauthorized behavior on their Wi-Fi, despite millions of chips in daily use?"

Because the forum moderator identifies as "core engineer" it is worthwhile to break down their arguments (Appendix A) which can be summarized as:



1. Espressif is able to price products at a third of competitors' pricing because their manufacture process is cheaper and their operating costs are lower.
2. ESP32 and ESP8266 designs are more sophisticated and cost effective than market competitors.
3. Maliciously leveraging ESP8266 and ESP32 devices is not technologically possible because it would be too complex to trigger specific devices, obfuscate attacks, or synchronize units en masse.

ICIT will leave the first two points to the determination of the audience and market competitors. The remainder of this publication will focus on the technological feasibility of maliciously leveraging the aforementioned devices.

### Targeting Specific Devices

For a Mirai style attack, where the adversary is leveraging the computational power of the infected device, the specific peripherals are irrelevant. If the adversary utilizes machine learning then they may even be able to activate or leverage devices specifically based on detected peripherals.

Espressif ICs control their own stack and retain the capability to periodically communicate with remote update servers so long as they are networked. Consequently, after deployment, the unit can pass metadata back to a Command and Control (C2) server which could be used to identify peripherals, determine function, or discern potential if weaponized. Further, a backdoor hidden at the firmware level could be used to deliver malware to the integrated device or onto the associated network.

### Attack Obfuscation

If the malicious traffic originates from a trusted network device, organizations may not notice the unauthorized Wi-Fi traffic. Even if they do, what is the likelihood that an organization would recognize a misbehaving IoT device as a firmware level compromise? If their security solutions do not detect malware higher in the application stack, this is highly unlikely. Further, it may not be noticed if the misbehaving electronic is a background IoT device, such as a light switch. In such a scenario, a burst of traffic, nefarious communique, or injection of malicious data onto the associated network can easily go unnoticed.

### Device Synchronization

Internet connections using transport layer security (TLS) connections require a knowledge of the current time to verify if a certificate is within its validity window. Each time a TLS session is initiated, the client and server both need to retrieve the correct time to verify the others' certificate. Connected devices usually go to a pool of NIST-specified Network Time Protocol (NTP) servers to retrieve the time, or they can also be hard coded. Absolutely nothing is stopping hidden code in ESP8266 or ESP32 from randomly checking the time from an NTP server in Beijing. The response will be accurate, and behavior would not be affected if the time format and data are accurate from that Beijing server.

Unfortunately, there is also way an attacker could leverage every single compromised device at the same time. The server's response to an NTP request has many different bitfields, including a bitfield called "stratum." A stratum is an indicator of the level of accuracy of the source. GPS satellites compose Stratum 1, while a source synchronized to those GPS satellites is in Stratum 2. A source synchronized to a device in Stratum 2 source would be Stratum 3, and so on. There are only 15 stratums; anything connected to a Stratum 15 device is not considered accurate enough to qualify for a stratum rank. Thus, unsigned integer values above 15 are never returned by most NTP servers. However, a server with unkind intentions could return a number above 15 as a command mechanism. No one tests for this condition, and if they did notice, it would likely be disregarded as erroneous overhead traffic.

**Table 1: Stratum Ranges for Network Time Protocol (NTP)**

Stratum: This is a eight-bit unsigned integer indicating the stratum level of the local clock, with values defined as follows:

Stratum	Meaning
0	unspecified or unavailable
1	primary reference (e.g., radio clock)
2-15	secondary reference (via NTP or SNTP)
16-255	reserved

If the returned stratum level is in the invalid range of 16-255, typical Wi-Fi clients would disregard the out of range value as an error and make another request. However, if ESP8266 or ESP32 has hidden code, this could act as a logic bomb or could queue specific behavior.

## Attack Potential

Additionally, attackers could exploit the firmware in Espressif devices to conduct a Layer 2 distributed denial-of-service (DDoS) attack that cannot be disrupted by internet service providers or conventional security solutions. Each individual device would have to be physically unplugged to stop the attack. Firewalls and network traffic filters would prove impotent against an attack en masse because the solutions operate at a higher level of the logical stack. Before delving deeper into this attack paradigm, it may be beneficial to explain, at a high-level, the flow of traffic through networked devices.

## Open Systems Interconnection Model

The International Standards Organization Open Systems Interconnection (OSI) model is a conceptualization of the layered activities necessary for network communication.

**Table 2: OSI Model Layers**

Layer	Name	Activity	Protocols
7	Application	User-level data	FTP, HTTP, POP3, & SMTP
6	Presentation	Standardized data appearance, blocking, or compression	Compression and encryption protocols like SSL
5	Session	Session/logical connections within an application, message sequencing, or recovery	Authentication protocols
4	Transport	Flow control, priority assignment, end-to-end error detection and correction	TCP & UDP
3	Network	Blocking message data into uniform sized packets, routing	IP, ICMP, ARP, & RIP
2	Data Link	Reliable data delivery over physical medium; transmission error recovery, packet separation into uniform sized frames	802.3, 802.5, 802.11
1	Physical	Communication across physical media; individual bit transmission	100Base-T & 1000 Base-X, 802.11

Each layer adds its own activity to a communication, like an assembly line that passes data along in three directions. Data is communicated in parallel or across the same layer to another host but communicated abstractly with the layer above or below. Interactions with the above and below layers are actual interactions while interactions with parallel layers (peers) are virtual communications. For a sender and receiver, peer-to-peer correspondence occurs between like layers. When sending, the logical message transmission path operates from Layer 7 to Layer 1, but from Layer 1 to Layer 7 for a receiver. Physical communication always occurs across a medium, like a wire, at Layer 1. In this way, each layer performs the same activity for a sender and receiver, just in a reverse order. For instance, if the sender's Layer 4 affixes a header to a message that designates the sender, receiver, and relevant sequence information, then the receiver's Layer 4 reads the header and removes it after verification that the receiver is the intended recipient. In other words, peer-to-peer correspondence occurs between the same layers.

The layer an adversary leverages in a DDoS attack depends upon what type of traffic is employed and how the traffic is generated. Application traffic is a Layer 7 DDoS attack, routers communicate traffic at Layer 4, and packet floods occur at Layer 3. A data-link, or Layer 2, DDoS attack, such as what is possible through the exploitation of Espressif devices, is high unprecedented.

### TCP/IP:

The OSI model has too much overhead for megabit-per-second, or greater, communications. Consequently, the transmission control protocol/internet protocol (TCP/IP) stack was invented to manage the Internet. TCP/IP is conceptualized in four layers, but defined by protocols. Despite its name,

TCP/IP actually contains three protocols: TCP, IP, and the user datagram protocol (UDP). The transport layer receives messages of variable lengths from the application layer which it then parses into units of manageable size, transferred in packets. The internet layer transfers packets as datagrams to different physical connections, based on the destination of the data. The physical layer consists of the drivers and devices that perform the actual bit-by-bit data transfer.

**Table 3: Internet Connection Layers and Services**

Layer	Layer Characteristics		Layer Services	
	Action	Responsibilities	TCP Protocols	UDP Protocols
<b>Application</b>	Prepare messages from user interactions	Addressing, user interactions	SMTP, HTTP, FTP, Telnet, etc.	SNMP, Syslog, Time, etc.
<b>Transport</b>	Convert messages to packets	Sequencing, integrity, error correction	TCP	UDP
<b>Internet</b>	Convert packets to diagrams	Routing, flow control	IP	IP
<b>Physical</b>	Transmit diagrams as individual bits	Data communication	Data communication	Data communication

The TCP protocol ensures the correct sequencing of packets and the integrity of the data within the packets. The protocol also calls for the retransmission of missing packets and fresh copies of damaged packets. The TCP service can build up overhead as computational resources are expended to record and check sequence numbers, verify the integrity of data, and request and wait for the retransmission of faulty or missing packets. TCP packets include a sequence number, an acknowledgement number for connecting packets, flags, and source and destination port numbers. UDP lacks the error-checking and correcting features of TCP. In most DDoS attacks, adversaries flood the victim system with malformed, unrequested, or recursive TCP, UDP, or ICMP traffic. In a DDoS attack that leverages Espressif devices, the adversary could flood nearby devices with unresolvable traffic requests. Worse, because many Espressif devices seem innocuous, the victim may not discover the attack vector or realize the need to physically disconnect the compromised device. In all likelihood, they would only suspect their smart devices of malfeasance after ensuring their router is not malfunctioning, their computer is not infected, and their internet connection is stable.

### Demonstration of Attack Feasibility

Approximately 400,000 new Espressif devices connect to Wi-Fi access points (AP) each day. Many of these APs have a catalogued MAC address, or basic service set identifiers (BSSID), that can be linked to a close latitude/ longitude location using publicly available databases. Just from an espionage perspective, each Espressif device reporting a BSSID to C2 infrastructure could prove devastating if an adversary aggregated the data into a global heat map of networked devices. The below heat maps from WiGLE.net represent the hundreds of millions of Wi-Fi APs that have been cataloged by their BSSID along with the approximate coordinates of each installed AP. Every Espressif Wi-Fi device connected to a Wi-Fi network

has knowledge of the BSSID to which it is connected. These devices can easily report this BSSID back to China where its location can be correlated to that BSSID by using a publicly available database, even though GPS is not native to the Espressif devices.

**Figure 1: BSSID Heat Maps for Geolocation**

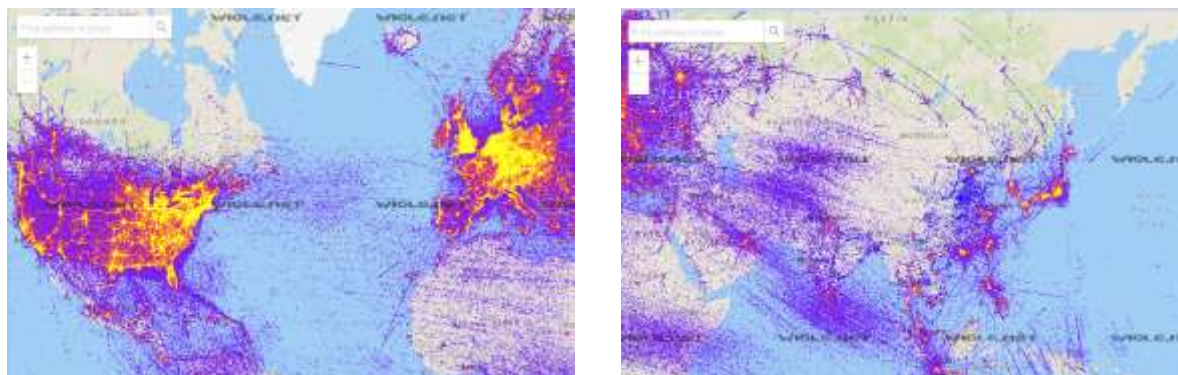


Figure 1 shows the publicly available mapping of BSSID to latitude and longitude. Even if infected embedded devices only reported back the MAC addresses of nearby devices, and never launched a malicious attack, the gathered data could still prove useful or intelligence operations or subsequent cyber campaigns.

However, the disruptive and destructive potential of the attack vector is even more troubling. While almost all manufacturers of embedded Wi-Fi devices do not expose the ability to create and send raw frames, the Espressif devices have an exposed application programming interface (API) command, `Wi-Fi_send_pkt_freedom()`, that allows for the construction and transmission of any packet, including management frames like BEACON. This binary-level packet injection could be leveraged as a coded logic bomb or a strategically triggered malicious feature. While it seems this API has been deprecated in recent releases, it is coming back, renamed as `esp_Wi-Fi_80211_tx()`, in the upcoming release. While the API call has temporarily disappeared, this is alarming because its underlying functionality remains fully accessible by the underlying Espressif binary code. An API name change without significant functional change is suspicious, and generally not best practice, as it prohibits backwards compatibility. Regardless of the API, the ability to create raw and malicious frames is always accessible by the underlying binary.

[Using code publicly available from GitHub](#), ESP8266 and ESP32 can be weaponized into beacon injectors capable of disrupting all 2.4 GHz Wi-Fi dependent devices in range. For instance, here is a benchmark Wi-Fi scan of networked devices:

**Figure 2: Sample Wi-Fi Range of a Home Network**



Figure 2 depicts a sample home network Wi-Fi range. AP MAC addresses were captured during the scan but were anonymized for publication and replaced with white boxes

Here is the same spectrum after running a single beacon injector device for one minute:

**Figure 3: Sample Beacon Injector Attack on a Home Network**

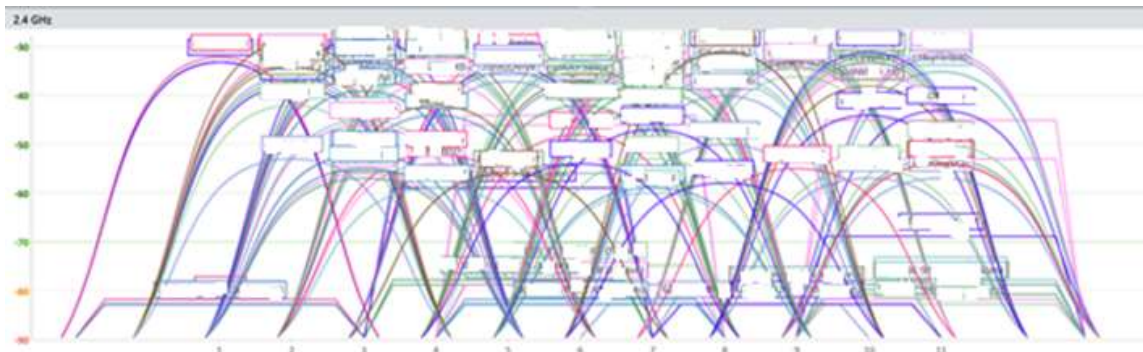


Figure 3 depicts a the same home network Wi-Fi range as Figure 2 after a publicly available, simple beacon injector code (~100 lines) was run on an off the shelf ESP8266. The white boxes represent real and spoofed AP MAC addresses. As you can see, the beacon injector floods the network and all nearby devices. In fact, when running it effectively disrupted nearby laptops and game consoles that were connected to the Internet. The only way to stop the attack was to physically disconnect the device. If this code or more sophisticated malware were run on an ESP8266 that was embedded in a home smart device (lightbulb, wall outlet, etc.), the user would have to somehow recognize that it was the IoT device disrupting every nearby systems and then physically disconnect the device which could be as complicated as unscrewing and removing a smart wall outlet.



This demonstration is relatively pedestrian compared to the potential attack vector. Imagine if more sophisticated malware was distributed to every Espressif device via a malicious update. Now add the potential impact if the malware were more complex than the hundred lines of code used in this simple beacon injection attack. Next, consider the option to trigger specific devices via native BLE capabilities or embedded WeChat command functionality. Finally, combine that with the ability to geolocate devices and either individually, logically, or synchronistically trigger malicious activity.

This proof of concept was conducted in Summer 2020 using code from about five years ago. It is worth noting that over the course of our writing this, the API was removed from the SDK releases; however, API `esp_wifi_80211_txt` will return in the upcoming release. The API `esp_wifi_80211_txt` can be used to send the beacon, probe requests, probe responses, send action frames, and send non-QoS data frames.

ICIT used this outdated publicly available code to demonstrate that a Wi-Fi deauthentication attack remained possible on a newly purchased device, even if the most recent versions of *esp\_iot\_sdk* removed the ability to send Wi-Fi control frames with *wifi\_send\_pkt\_freedom*. Additionally, [code is available](#) to re-enable *wifi\_send\_pkt\_freedom* in newer versions of *esp\_iot\_sdk*. For a sophisticated adversary, it would be trivial to decrypt traffic from other devices on the same network thanks to the promiscuous mode or they could use an extension of the Github code to deauth a device (on the same network) and capture the resulting 4-way handshake to derive the temporal pairwise key. Doing this would be no harder than computing your own pairwise keys when connecting to an Access Point. If the attacker can install the keys in the hardware, then they would be able to capture plaintext data, pretend to be an access point, or launch buffer overflow attacks, denial of service attacks, or remote code execution. It may also be feasible to attack any WPS-enabled access points using this same approach. If the pairwise key for stations on the same network could be determined, the devices may also be able to launch attacks against other WPA2 access points. Future publications will explore these scenarios and other more technical attack vectors in greater depth.

There are two explanations for why these devices appear to be designed to be exploited. The first is that adage that “it is a feature, not a bug.” If a nation-state adversary were to use Espressif’s remote update features to pass backdoors or malicious code to embedded devices, the potential impact would be significant. In this case, the API would have been left publicly exposed so that various attacks could be implemented without having to rerelease the SDKs and would free the SDK engineering team from developing such ‘attack applications’. The second explanation is “Hanlon’s razor” which bluntly states, “never attribute to malice that which can be adequately explained by stupidity”. The existence of this API certainly suggests a complete disregard for security or lack of awareness by the engineers. Combined with the markedly low price point per unit, it is logical to conclude that Espressif must not dedicate significant budget or attention to securing their products. We leave it to the reader and Information Security stakeholders to form their own conclusions, pen test their devices containing Esp8266 or Esp32 components, or otherwise test the assertions in this case study.

Espressif markets to consumers, healthcare companies, and industries seeking automation. What happens if code in smart homes spies on or disrupts members of the current tele-workforce, identified through a subset of other devices on their home networks? What if a beacon injection attack goes off in a healthcare environment and overwhelms critical devices with a flood of Layer 2 traffic? What if industrial operations of firms in competition with Chinese manufacturers are subtly disrupted or rendered inefficient? The dangers posed by Espressif devices, and ones like it, are limited only by the imagination of the attacker.

## Conclusion

It has often been said that if you are not paying for a product, then you are the product. By flooding the market with over 400,000 devices per day, Espressif has enticed consumers to adopt devices that are either negligently unsecured or designed with malicious exploitability. Given that their price point is nearly a third of their competitor's price, it is unlikely that the firm is operating to maximize a profit. Keeping in mind the two recent Chinese National Cybersecurity Laws and current Chinese five-year plan, it is possible that these underpriced devices are being spread globally for nefarious purposes in the same way that Huawei and ZTE are believed to operate to advance the posture of Chinese state interests. Regardless of which theory proves true in the long-term, the systemic problem remains the same. Manufacturers and consumers need to care more about supply chain security. Suspiciously cheap components should concern the technology community and should remind us all of the overwhelming need for greater supply chain security. Malcolm Harkins adds, "Our society has become economically dependent on technology, but we also take for granted that we rely on devices to sustain our quality of life. In the absence of critical analysis and hard questions, vulnerabilities such as this arise and put us all at risk, with increasingly grave consequences." Espressif is not the only OEM whose security should be critically tested, but we should be concerned with their influx of potentially compromised products, just like we are with Huawei or ZTE.

# Appendix A

🏠 Board index / ESP32 English Forum / Explore /

Search...



General Discussion

## How is it possible that the ESP32 is so cheap?

🔒 Locked



Search this topic...



7 posts • Page 1 of 1

### How is it possible that the ESP32 is so cheap? (#p51486)

🗨 by **greengnu** » Thu Nov 07, 2019 9:34 am

greengnu

Posts: 24

Joined: Wed May 08,  
2019 8:45 pm

Even if the whole wireless functionality would be left out we'd still have a dual core 240mhz 32bit cpu that can run complex applications for a dollar (soc). I couldn't find any other cpu in the same performance class that comes even close in terms of price performance.

How is this pricing possible?

Just saying. If I were the chinese government and wanted to spy on the world through an armada of small IoT devices linked into every private wifi around the globe....

### Re: How is it possible that the ESP32 is so cheap? (#p51489)

🗨 by **ESP\_Angus** » Thu Nov 07, 2019 10:32 am

ESP\_Angus

Posts: 2181

Joined: Sun May 08, 2016  
4:11 am

Folks, discussing conspiracy theories is entirely off-topic for this forum and if necessary posts in this thread will be moderated.

“ greengnu (./memberlist.php?mode=viewprofile&u=10548&

[sid=a0f64d3ecd7ca0abf75d2e77fca2c917](#)).wrote: ↑(./viewtopic.php?p=51486&

[sid=a0f64d3ecd7ca0abf75d2e77fca2c917#p51486](#))

Thu Nov 07, 2019 9:34 am

Even if the whole wireless functionality would be left out we'd still have a dual core 240mhz 32bit cpu that can run complex applications for a dollar (soc). I couldn't find any other cpu in the same performance class that comes even close in terms of price performance.

How is this pricing possible?

The short answer is that it's cheap to manufacture. In particular the RF engineers have done a bunch of very clever things on the Wi-Fi side. You will also notice that in a lot of ways ESP32's design is not like other common microcontrollers. This is generally not by accident, it's to keep the overall cost down.

“ [greengnu \(./memberlist.php?mode=viewprofile&u=10548&](#)

[sid=a0f64d3ecd7ca0abf75d2e77fca2c917](#)).wrote: ↑(./viewtopic.php?p=51486&

[sid=a0f64d3ecd7ca0abf75d2e77fca2c917#p51486](#))

Thu Nov 07, 2019 9:34 am

Just saying. If I were the chinese government and wanted to spy on the world through an armada of small IoT devices linked into every private wifi around the globe....

This conspiracy theory comes up every now and again, and as a core Espressif engineer I've tried to figure how it would technically work. And I have no idea. Would all ESP32s ping a server once a day to check if they should switch into "backdoor mode"? And what would "backdoor mode" do exactly, given every chip has different peripherals attached to it in different ways? And noone would notice this additional unauthorised behaviour on their WiFi, despite millions of chips in daily use?

Keep in mind that all network code in ESP32 above the Wi-Fi MAC layer is open source, the WPA2 integration is open source, and the build system and toolchain are all open source. So the binary Wi-Fi libraries somehow have to implement an entire "shadow" firmware implementation that could take over the main firmware, no matter what that main firmware happens to do, and without any ability to analyze the real firmware source. Then they'd all have to phone home with no interruption to normal device behaviour and nothing unusual that shows up on a TCP packet dump.

Engineering a chip that is cheap to manufacture seems positively simple by comparison.

## **Re: How is it possible that the ESP32 is so cheap? (#p51490)**

by **greengnu** » Thu Nov 07, 2019 11:02 am

greengnu

Posts: 24

Joined: Wed May 08, 2019 8:45 pm

“

as a core Espressif engineer I've tried to figure how it would technically work. And I have no idea. Would all ESP32s ping a server once a day to check if they should

switch into "backdoor mode"? And what would "backdoor mode" do exactly, given every chip has different peripherals attached to it in different ways?

Well, this <https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies> chip was able to do it somehow too, and it's a fraction of the size of an esp32 and was embedded into a much more expensive, much lower volume product. The penetration of the esp32 is likely going to be many orders of magnitude higher than this and basically all of the devices that contain it will already be network connected.

Image

Don't get me wrong - I'm absolutely amazed by the ESP32. I love this little device a lot! I'm just getting suspicious when something seems too good to be true...

Last edited by greengnu ([./memberlist.php?mode=viewprofile&u=10548&sid=a0f64d3ecd7ca0abf75d2e77fca2c917](https://www.esp32.com/memberlist.php?mode=viewprofile&u=10548&sid=a0f64d3ecd7ca0abf75d2e77fca2c917)) on Thu Nov 07, 2019 11:07 am, edited 1 time in total.

### **Re: How is it possible that the ESP32 is so cheap? (#p51492)**

🔍 by **WiFive** » Thu Nov 07, 2019 11:07 am

WiFive

Posts: 2956

Joined: Tue Dec 01, 2015 7:35 am

If a 200mm processed silicon wafer costs \$1000 and esp32 has a die area of 8mm<sup>2</sup> then the cost per chip is somewhere around 30c so it is possible. You might have to sell 5MM to break even on NRE but espressif has sold 100MM chips (all products). Another reason why it is so cheap for everyone is because there are no traditional (Western) pricing tiers.

Also I think that Bloomberg article has been debunked many times over but it gets clicks which is all that matters today.

Last edited by WiFive ([./memberlist.php?mode=viewprofile&u=239&sid=a0f64d3ecd7ca0abf75d2e77fca2c917](https://www.esp32.com/memberlist.php?mode=viewprofile&u=239&sid=a0f64d3ecd7ca0abf75d2e77fca2c917)) on Thu Nov 07, 2019 11:19 am, edited 1 time in total.

### **Re: How is it possible that the ESP32 is so cheap? (#p51493)**

🔍 by **greengnu** » Thu Nov 07, 2019 11:17 am

greengnu

Posts: 24

Joined: Wed May 08, 2019 8:45 pm

“

If a 200mm processed silicon wafer costs \$1000 and esp32 has a die area of 8mm<sup>2</sup> then the cost per chip is somewhere around 30c so it is possible.

By that logic a 28-core intel xeon platinum should cost 20\$ instead of 13'000\$.

How do you pay for all the engineering, marketing, legal cost etc. etc. that are required to get something like the esp32 out?

### **Re: How is it possible that the ESP32 is so cheap? (#p51495)**

 by **WiFive** » Thu Nov 07, 2019 11:33 am


WiFive

Posts: 2956

Joined: Tue Dec 01, 2015  
7:35 am

Well Xeon Platinum is a much more expensive silicon process with lower yield so it probably does cost over \$500 in just silicon.

### **Re: How is it possible that the ESP32 is so cheap? (#p51501)**

 by **ESP\_Angus** » Thu Nov 07, 2019 12:55 pm

ESP\_Angus

Posts: 2181

Joined: Sun May 08, 2016  
4:11 am

“ greengnu (./memberlist.php?mode=viewprofile&u=10548&sid=a0f64d3ecd7ca0abf75d2e77fca2c917) wrote: ↑ (./viewtopic.php?p=51493&sid=a0f64d3ecd7ca0abf75d2e77fca2c917#p51493)

Thu Nov 07, 2019 11:17 am

How do you pay for all the engineering, marketing, legal cost etc. etc. that are required to get something like the esp32 out?

Investors.

I'm locking this thread because speculative conspiracy theories are off topic for this forum.

Display posts from previous:

All posts

Sort by

Post time

Ascending


Go

 Locked

 | 

7 posts • Page 1 of 1

 Return to “General Discussion”

Jump to | 

 WHO IS ONLINE



Users browsing this forum: No registered users and 34 guests

## About Us

Espressif Systems is a fabless semiconductor company providing cutting-edge low power WiFi SoCs and wireless solutions for wireless communications and Internet of Things applications. ESP8266EX and ESP32 are some of our products.

## Extra

- ➔ [Espressif Homepage](#)
- ➔ [ESP8266EX Official Forum](#)
- ➔ [ESP8266 Community Forum](#)

## Information

- ➔ [Terms of use](#)
- ➔ [Privacy policy](#)
- ➔ [FAQ](#)
- ➔ [The team](#)
- ➔ [Contact us](#)

---

Espressif ESP32 ... Available now!



## Appendix B

**The following response was provided by Espressif CEO Swee-Ann Teo, and has been published “As-Is.”**

### Espressif’s Response to ICIT’s “The Perfect Weapon, Hidden in Plain Sight” Report.

#### Introduction

Hardware supply chain issues and ensuring the integrity of connected devices is a complex topic, and one that we take seriously. As the ICIT report notes, it is technically challenging to verify all aspects of any complex electronic product.

Security is of great concern to Espressif and we endeavor to be as open and auditable as possible given commercial constraints. Espressif’s SDKs and other resources for OEMs are published openly, not hidden behind NDAs. Internal processes are followed to ensure system integrity, and external analysis by OEMs and interested third parties is encouraged.

The ICIT report includes some specific claims about the potential for a widespread compromise of Espressif devices; I shall discuss the technical barriers to such a compromise. The report also draws some conclusions regarding Wi-Fi functionality included in Espressif SDKs. As I shall explain, these conclusions about Espressif’s SDK are not logically supported by the described functionality.

#### Understanding Espressif’s Technical Role

It is important to understand the specific technical environment of an OEM device incorporating an Espressif SoC, in order to analyze claims made in the report.

Espressif produces SoCs (system-on-chips) and hardware modules. Espressif publishes technical documentation and SDKs (Software Development Kits) for these SoCs. The SDKs are published publicly on GitHub -- primarily as source code, with precompiled object code libraries for the low-level device-specific Wi-Fi and/or Bluetooth radios.

Espressif’s customers are tens of thousands of individual Device OEMs. Most Device OEMs download an SDK from GitHub and create their own firmware application, which is compiled along with the Espressif SDK. The final product design and firmware is typically completed and tested by the OEM’s engineers and the product is produced and tested in the OEM’s factory. The final device firmware application is a single unique unit of compiled binary firmware: built from the needed SDK functionality, any RF functionality, and the OEM’s application source

code. Because of resource constraints, any functionality that is not used by a specific OEM device is not compiled into the firmware.

Device-specific functionality including deployment, cloud services, remote updates, etc. is implemented by each Device OEM independently. They may host their own servers, use a cloud provider's IoT service, have the device communicate locally via Bluetooth, etc.

Ultimately, each OEM has full control over their individual product firmware.

### Cost of Espressif SoCs

The report asks, "why are Espressif's competitors unable to market chipsets at comparable prices?" No semiconductor company typically discuss their costs or their competitors' costs. In fact, many of Espressif's competitors will only provide prices via confidential direct quote.

However, I can share some domain knowledge from the semiconductor industry. The cost to produce an individual chip depends primarily on two things: the semiconductor process node, and the silicon area of the chip (die size). By working with a low-cost process node, packing as much functions as we can in the chip and minimizing the silicon area of each chip, then it is possible to proportionally reduce the per-unit cost.

These factors can be independently verified by de-encapsulating ("decapping") any chip and viewing it under high magnification. Third party companies can provide this service, including an estimate of per-unit cost. Anyone who wishes to independently verify that Espressif SoCs cost significantly less to produce than a particular competing chip, in some cases a small fraction of that cost, can easily do so. (More about this later.)

Producing a secure, standards-compliant, and fully functional SoC within a constrained resource footprint requires significant innovation. We invested within the limits of what we have in research and development and continues to do so. For example, Espressif has been granted 66 patents related to their designs.

In 2011, Espressif was the first company in the industry to fully integrate all the RF components of a Wi-Fi chip (balun, antenna switch, power amplifier, and low noise amplifier) into a deep sub-micron CMOS process, thereby significantly reducing the cost of manufacturing, and which we patented. This lead in design enabled cost savings that led to ESP8266 capturing a large share of the market.

We later introduced ESP32, SMP (symmetrical multiprocessing) dual core MCU with WiFi and Bluetooth, chip that was probably the first of its kind in the industry to combine a SMP dual core architecture with WiFi and Bluetooth.

We have also recently introduced a RISC V WiFi 4 Bluetooth 5.0 combo with advanced encryption features, which will be detailed later. The latest chip is also one which we believe

has cost advantages because we have chosen the correct minimal set of specifications and features, in line with the philosophy of frugal innovation.

### Market Strategy

According to ICIT, Espressif has “questionable financials” and there are “inconsistencies in market position and behavior that we found troubling”. The questions raised were whether Espressif is sacrificing profits in order to grab market share and the amount of R&D that Espressif is spending. However, these characteristics described in the report are normal for any innovative technology company entering an existing market.

As acknowledged in the ICIT report, Espressif is profitable. But not only is Espressif profitable, but we maintain a reasonable margin of above 40% and Espressif has also never engaged in non-ethical or anti-competitive behavior in order to increase market share. Espressif’s operating income has increased each year between 2016 and 2019 and most likely this year as well (2020).

Based on interaction with our customers, we are also not the lowest priced chip in the market. We use TSMC process which commands a higher premium than other competing fabs and we build chips with more memory so that our customers can create more complex applications, thereby create more value; we like to think that we enable our customers to create better applications.

For instance our latest ESP32-C3 chip has 400 kB memory while most competing products have around 256 kB. (Memory takes up a significant portion of the cost of a chip.) It is not the strategy of Espressif to simply go for large volume with very low cost chips or to sacrifice profit margins solely for market share.

Our margins have been above 42% over the few years, which is higher than most of the companies in this domain. Profit margin is of the highest priority to Espressif--profit margins are the best measure of the quality of our design efforts.

### Change Management

Espressif has internal change management and service reliability teams and policies to ensure the integrity of resources it publishes for OEMs.

Internal processes include peer code review of software changes and a multi-stage process for verifying changes before making official releases. Critical functionality is developed on standalone isolated networks. We have also contracted external security consultants to perform reviews of our code. When problems are found, they are documented and fixed immediately.

By using the “git” version control system exclusively for software development and SDK releases, the specific contents of the SDK can be very easily compared between individual copies – including by third parties. It is not possible for a file to have been changed outside of the official development process without introducing a different “git hash” compared to legitimate copies.

## Analysis

We welcome analysis of our devices, as indicated by the security bug bounty program. Where security issues have been reported Espressif has patched the issue rapidly, worked with the reporters to verify the patch, and supported a responsible public disclosure process. Bug bounty programs and coordinated disclosure are commonplace in the software industry, but not yet widely adopted by semiconductor vendors.

Major device OEMs have also engaged third party contractors to perform security code review of internal source code, including source code for RF libraries.

Third party analysis of the software that runs on a connected device is often considered prohibitively complex or impossible. However, in the case of an Espressif device’s OEM firmware the final binary is usually smaller than 1 MB total size. Other connected devices may have filesystems with hundreds of megabytes of binary libraries and executables. As each OEM compiles their own complete firmware, they have access to corresponding source code and binary object code to perform specific analysis. Even the low-level RF libraries which are provided as binary object code have all symbols included (“unstripped”).

As such, Espressif’s firmwares have also been extensively reversed engineered by many in the community, with some even building their own firmwares:

- <https://boredpentester.com/reversing-esp8266-firmware-part-1/>
- <https://github.com/jsandin/esp-bin2elf>
- <https://github.com/pfalcon/esp-open-sdk>

Overall, this is more open and auditable than many vendors. Often device SDKs are only provided to OEMs individually under NDA, preventing wider scrutiny and aiming instead for “security by obscurity”. Elsewhere, functionality is encompassed in a “binary blob” that runs independently to the main application firmware, so third parties cannot gain access to symbols or other metadata about its contents and functions.

As stated in the introduction, we acknowledge that hardware verification is an issue across the industry and there is no perfect solution yet. However, Espressif’s practices exceed the openness and auditability of similar vendors.

## Malicious functionality

The report states “Espressif ICs [...] could be infected with a malicious backdoor prior to distribution or remotely replaced with malicious code. A single malicious insider could infect a firmware update and compromise tens of millions of devices”. Due to the relationship between Espressif and device OEMs, and the technical architecture of Espressif SoCs, this process is much more challenging for a would-be “malicious insider” than what is suggested in the report. There are multiple points where such a hypothetical infection would be prevented or discovered.

We will address the two claims of a “backdoor prior to distribution” and a device “remotely replaced with malicious code” independently.

## Pre-existing backdoor

The report claims a backdoor could be introduced prior to distribution by Espressif. As described in the “Change Management” section, development processes are in place which would prevent such a malicious change from being introduced internally without detection. Even if internal processes were bypassed or failed, as I will explain, there are still significant opportunities for the behavior of a hypothetical function like this to be detected.

## Network Protocol Obfuscation

The report notes that an innocent seeming network protocol can encapsulate additional data, used as a covert channel. A hypothetical situation is described where “nothing is stopping hidden code in ESP8266 or ESP32 from randomly checking the time from an NTP server in Beijing”.

There has never been any spurious network connection of this kind, malicious or otherwise. Moreover, there are no third-party reports of anything resembling such a connection.

Espressif SoCs do not make any network connections that are not initiated by the OEM firmware itself. Although the SDKs do offer functionality to enable functionality such as NTP, this functionality is optional and not always enabled. The code that determines the timing of updates and the server hostname or IP to request is all configured and compiled into the firmware by the Device OEM not by Espressif. Therefore, the network functionality varies enormously between different OEM devices. A random connection to an unprecedented IP address would be a marked and noticeable departure from the normal behavior of most OEM devices.

This is different from more complex connected devices where the silicon vendor may provide their own management executables to perform “housekeeping” functions on all devices. These may include regular network connections for purposes like NTP, telemetry, a “management



engine”, or updating parts of the device software independently. Due to both design decisions and resource constraints, Espressif devices do not include this type of vendor functionality.

Device OEMs conduct their own testing of each firmware, and packet capture and analysis is a common tool to check correct network behavior. For low power applications, minimizing the number of radio interactions is key to maximizing battery life - so every packet is closely analyzed to see if it is necessary.

Even if some device OEMs did not notice such hypothetical behavior, any behavior included in all Espressif devices would need to remain unnoticed by all device OEMs, all end users and their organizations, as well as the tens of thousands of hobbyists and hackers who experiment with Espressif SDKs every day.

### Remote Updates

The second claim is that Espressif devices may be “remotely replaced with malicious code”. As explained, Espressif has no update channel to control a device OEM’s firmware. So, there is no functionality that would allow Espressif to deliver updates to OEM devices, without going via each individual OEM’s infrastructure.

This does not prove non-existence of a covert update channel that can patch OEM’s application firmware without their knowledge. However, this hypothetical update channel would be a pre-existing backdoor - so it depends logically on the first claim.

### Wi-Fi Protocol Functions

The report discusses Espressif SDK Wi-Fi functionality and concludes that “these devices appear designed to be exploited”. It is not clear that this conclusion follows from the characteristics described.

The key issue is whether an SDK function to send custom (“raw”) 802.11 Wi-Fi frames in the SDK is a “vulnerability” as described. There are legitimate uses for sending arbitrary 802.11 Wi-Fi frames, including custom mesh networking protocols or peer-to-peer local Wi-Fi protocols targeting high throughput or low latency.

ICIT writes “almost all manufacturers of embedded Wi-Fi devices do not expose the ability to create and send raw frames”. However, this is a common Wi-Fi driver function. Consider the standard Linux mac80211 driver, which supports this function for nearly all underlying chipsets. This driver is used by most modern Linux Wi-Fi devices, including hundreds of millions of Linux-based IoT devices. Open-source networking libraries such as “scapy” simplify 802.11 packet injection on any Linux system by using this driver interface. This is identical to the functionality

described in the Espressif SDK. All these tools can be used by programmers for either productive or malicious purposes.

All the same, we are concerned that users of Espressif-based devices may program their own devices for malicious purposes. For this reason, the original “wifi\_send\_pkt\_freedom” function for sending 802.11 frames was removed. However, after receiving requests from multiple OEMs with legitimate use cases for sending custom 802.11 frames - and after review of similar drivers revealed the same functionality is common - we decided to introduce this support again.

Analogously, any device that can establish TCP/IP connections can be programmed to perform a DoS or a DDoS attack. This does not mean that a vendor who provides customers with TCP/IP networking capabilities in an SDK is consequently negligent or malicious.

### Securing the Supply Chain

When requested, we also work with customers to secure the manufacturing of our Wi-Fi parts through the use of specialized hardware and set of best practices. For instance, on the hardware front, the latest ESP32-C3 chips has implemented several such features to ensure the security of customer firmware:

- **\*\*Secure Boot\*\***: ESP32-C3 implements the standard RSA-3072-based authentication scheme to ensure that only trusted applications can be used on the platform. This feature protects from executing a malicious application programmed in the flash. Espressif also understands that secure boot needs to be efficient, so that instant-on devices (such as light bulbs) can take advantage of this feature. ESP32-C3's secure boot implementation adds less than 100 ms overhead in the boot process.
- **\*\*Flash Encryption\*\***: ESP32-C3 uses the AES-128-XTS-based flash encryption scheme, whereby the application as well as the configuration data can remain encrypted in the flash. The flash controller supports the execution of encrypted application firmware. Not only does this provide the necessary protection for sensitive data stored in the flash, but it also protects from runtime firmware changes that constitute time-of-check-time-of-use attacks.
- **\*\*Digital Signature and HMAC Peripheral\*\***: ESP32-C3 has a digital signature peripheral that can generate digital signatures, using a private-key that is protected from firmware access. Similarly, the HMAC peripheral can generate a cryptographic digest with a secret that is protected from firmware access. Most of the IoT cloud services use the X.509-certificate-based authentication, and the digital signature peripheral protects the device's private key that defines the device's identity. This provides a strong protection for the device's identity even in case of software vulnerability exploits.
- **\*\*World Controller\*\***: ESP32-C3 has a new peripheral called world controller. This provides two execution environments fully isolated from each other. Depending on the configuration,

this can be used to implement a Trusted Execution Environment (TEE) or a privilege separation scheme. If the application firmware has a task that deals with sensitive security data (such as the DRM service), it can take advantage of the world controller and isolate the execution.

Besides the hardware features, Espressif have also provided **\*\*ESP-JUMPSTART\*\*** (<https://docs.espressif.com/projects/esp-jumpstart/en/latest/introduction.html>), which is a framework for designing secure IOT device and addresses some of the concerns of the supply chain and securing the manufacturing.

Security is of first priority in Espressif and we have continually made improvements in our design of our chips, at the expense of additional cost and R&D investment, in order to better serve the needs of our customers and the industry.

As noted in ICIT's report, the security of the entire supply chain is as important and this includes the manufacturing portion. Espressif works directly with our customers, including many tier one companies, to ensure the security of their data and integrity of the manufactured devices.

## Conclusion

Connected device integrity and supply chain integrity are crucial factors in information technology security, and factors that Espressif takes very seriously. Espressif welcomes analysis and scrutiny as evidenced by a publicly available SDK and successful Bug Bounty program. Despite the lack of perfect solutions anywhere in the industry, Espressif's efforts meet or exceed the standards of similar vendors.

Part of the ICIT report highlights many general concerns that customers of connected devices should indeed consider. However, the specific conclusions about Espressif are not supported by the evidence. Doubt is cast on Espressif's comparatively low product pricing, but it is possible for a third party to verify the equivalent low manufacturing cost. Espressif's market strategy is criticized without recognizing that it is a common strategy for a small and innovative technology company. Descriptions of covert network data channels are purely speculative, and descriptions of "malicious remote updates" do not consider the technical relationship between Espressif and individual device OEMs. Characterizing a common Wi-Fi function as a vendor-specific "vulnerability" does not take into consideration the legitimate uses of this function, or its existence on many other platforms. Espressif encourages continued engagement and technical analysis of our products, and wholeheartedly supports ICIT's recommendation that "stakeholders [...] form their own conclusions, pen test their devices containing ESP8266 or ESP32 components, or otherwise test the assertions in this case study".

## Final Notes

Espressif's employees come from more than different 20 different countries, including China, India, Czech Republic, Slovakia, Nigeria, Bulgaria, Dominican Republic, Russia, Australia, Philippines, Singapore, Brazil, UK, Greece, Sweden, Germany, France, Norway, etc. Many of our employees work in their home country; we are highly collaborative and transparent. And I am one of 6 Singaporeans in Espressif. We believe that we can make a difference by creating interesting and useful technology for IOT applications.

Espressif also provides support to Makers because we believe that with the democratization of technology, Makers can make a difference to create interesting and important applications that would otherwise not be addressed by companies with purely commercial interests.

We are also in support of ICIT's effort to secure the supply chain and raise awareness of the need for better security. Many of the concerns about supply chains are valid and deserve to be looked into. We thank the authors of the ICIT report for raising these issues which have also been an important part of our ecosystem. We hope that with raised awareness, we could improve upon the workflow and thereby reduce any security risks to a minimal level.