TRUSTED FUTURE

WHEN INTEROPERABILITY MANDATES WEAKEN SECURITY – JIT HAPPENS

BY JIM KOHLENBERGER

→ Unfettered access to fundamental JIT features, or use of insecure JIT, creates significant vulnerabilities that hackers can easily exploit.

INTRODUCTION

Holy Cow! We looked at the very first interoperability request publicly posted as part of Europe's Digital Markets Act's (DMA) new interoperability rules and it's a digital doozy that would enable serious new security threats. A developer has requested that under the DMA's new interoperability rules, Apple should provide it with direct access to features of its Just-In-Time Compiler or JIT engine – which is a core capability built into all major browser engines. If Apple were forced to open up this feature to any developer, it could lead to unprecedented new security risks, give nation state threat actors a leg up, and put the integrity of core enterprise business operations at risk. The DMA's enforcers should reconsider these interoperability mandates and ensure that its requirements do not unintentionally expose users to higher security risks.

WHAT WE ARE HERE TO EXPLAIN.

In this paper we explain what a JIT engine is, how it achieves its many benefits, its well-known security risks, how those risks have been abused in the past ending in a major European scandal, how nation state actors are actively working to abuse or exploit these specific risks today, what broad access to a feature that enables apps to write and execute unvetted unsigned code directly in memory could mean, the high costs that granting such a simple request could mean to European businesses and critical infrastructure providers, and why this should be a red flag warning to any policymakers looking to replicate DMA-like policies.

A JIT engine is the software used by web browsers to compile code from a web-based language to the language the operating system of the smartphone understands. It is a crucial, necessary, and required function to make web browsers work fast and efficiently. It is also highly complex. As a result hackers may exploit security vulnerabilities in the JIT's software code itself, or abuse its critical function to hotline the delivery of an exploit to the heart of a device. In fact, roughly half of all exploits in major web browsers have involved the JIT. In some cases the exploits can be extreme. For example, the JIT was involved in gaining broad control of a phone in the original Pegasus mercenary spyware exploit as part of a sophisticated chain to mainline malware into the read/write/execute part of a smartphone. And the JIT compiler function can be such a critical process that it is often targeted by nation state actors as a useful pathway to execute and deliver sophisticated hacks. Given this fact, the JIT is a core feature that gets turned off when you activate Apple's "Lockdown Mode," Microsoft's "Super Duper Secure Mode," and Chrome's "Secure Mode."

In the first DMA interoperability request that was posted as part of the DMA's transparency reporting, a developer of a Linux application requested API access to the JIT and other functionality, so the developer can write and execute unverified, unsigned and possibly arbitrary code directly into read/write/execute memory. To security experts, hotlining unvetted code into sensitive areas is a major security vulnerability. Once in read/write/execute territory, even if sandboxed, a determined adversary can (as in Pegasus) escape the sandbox, elevate privileges, and assert control of functions and data.

But it's also an example of the kinds of major mobile security risks that DMA 'interoperability' proponents ignore, and the kind of security weakness that DMA regulators keep building into their smartphone rules that could have severe long-term security and financial implications.

Most importantly, this should be a **RED FLAG WARNING** to policymakers looking to replicate key aspects of the DMA without seriously contemplating its security risks.

INTEROPERABILITY IS AWESOME, UNTIL IT IS REQUIRED TO BE DONE INSECURELY.

Interoperability in the digital world can bring enormous benefits for consumers and businesses – enabling things never before possible. Smartphones have really become the poster child for what the broad benefits that an interoperable platform can enable as our modern mobile marvels have transformed into the Swiss Army Knife equivalent of interoperable tools unleashing broad third-party innovation. For example, the interoperability tools built into smartphones have enabled millions of third-party apps to work with an amazing array of cutting-edge on-board sensors, vast on-board computing power, a broad range of connectivity services and protocols, and innovative operating system functionality, which together have helped enable a broad ecosystem of third party connected apps and devices. Together they've enabled broad economic benefits that have occurred at an unprecedented scale in part because they have been enabled in thoughtful ways with safeguards built in to protect the safety, security and privacy of all users.

But mandated interoperability, as is required under the DMA, can be especially problematic when it is done in a way that directly harms or undermines these safeguards and puts consumers' fundamental privacy, safety, and security at risk. We've written before about the DMA's new interoperability mandates and some of the enormous security issues they create (here, here, and here.) Now that European regulators implemented a set of measures to insert transparency into the process developers use to request interoperability access, in this piece we look specifically at the very first interoperability request made public in this process, and some of the broad implications.

THE INTEROPERABILITY FEATURE BEING REQUESTED.

IN SIMPLE TERMS: In this first public request a developer is asking for access to an API to help make its Linux emulator app run an order of magnitude faster, more efficiently, and utilize less power which can extend battery life – the kind of key benefits that DMA regulators say their interoperability rules are intended to enable. And the developer is going through all the steps required by the DMA for interoperability access and making the same key argument that the DMA's rules were designed to address.

Since Apple has its own JIT engine, the developer is simply asking for equivalent access to Apple's core features necessary to enable its own JIT engine premised on the need to gain direct read/write/execute functionality in the device for its code. The applicant says its "needs are very basic." All it needs is a feature contained within all JIT engines which is the ability to write code directly to memory and execute it. In fact, this capability is already built into a JIT enabling API that Apple has already created to comply with another part of the DMA to allow trusted browser developers the ability to offer their own web browsers with their own JIT engines. And the DMA regulations broadly require "interoperability must be granted to the same feature under equal conditions." The developer says, the only obstacle "is Apple's policy limiting its use to browsers."

↗ Sounds pretty straightforward and simple enough right?

But it's wrong.

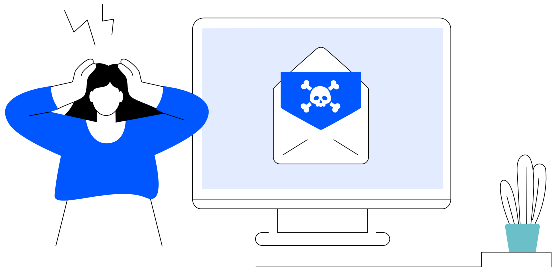
We have no reason to believe this small developer request wasn't in good faith, that it has any malicious intent whatsoever, nor do we doubt their commitment "to adopting best practices for writing secure software," or that such a change could improve their apps performance significantly. But we have significant concerns about what happens when security is weakened and direct access to a function which could be exploited or abused is made broadly available to developers whose apps could be vulnerable to hackers with a business plan, and to malicious state actors intent on gaining access to the valuable contents smartphones contain.

WHAT IS A JUST-IN-TIME COMPILER OR JIT?

A Just-In-Time compiler or JIT is a crucial and foundational technique used in modern browsers that enables complex interactive websites like games, Gmail, Instagram, TikTok, and millions of others to run faster, more efficiently, and more smoothly. The JIT function enables these complex JavaScript based websites to run significantly more efficiently by taking the text-based JavaScript code from the webpage and, through a series of complex pipelines, compiles or translates the code into a more efficient machine code that can run faster in an operating system. These JIT engines which are now used in every major browser can achieve quite impressive performance gains.

But these performance gains come with added security risks. One of the key features that helps enable these impressive gains is that a JIT engine is allowed to both write to memory and later execute that memory as code.

While this ability to dynamically generate executable machine code is essential to many high-performance browsers, when abused or exploited it's also incredibly useful for delivering malware into devices.



JIT ENGINES HAVE BECOME A PRIMARY TARGET FOR HACKERS, CRIMINAL GROUPS AND NATION STATE ACTORS.

1. **JIT ENGINES ARE EXTRAORDINARILY COMPLEX AND ACCOUNT FOR ROUGHLY HALF OF ALL BROWSER EXPLOITS.** Because a JIT engine is an extraordinarily complex piece of software, around 50% of security problems in a modern browser are related to its implementation of JIT across all major browsers. For example, an analysis by Mozilla shows that over half of the "in the wild" Chrome exploits took advantage of a JIT bug, with similar ratios as well for Edge and Firefox. Microsoft likewise looked at CVE (Common Vulnerabilities and Exposures) data after 2019 which shows that roughly 45% of CVEs issued for the Chrome V8 JavaScript engine (used by Microsoft's Edge browser) were related to the JIT engine as well. As security researchers explain it, "JIT compiler systems are typically large and complex, and often incur rapid code change — a combination that makes them fertile ground for bugs."

2. **JIT ENGINES ARE NOTORIOUSLY DIFFICULT TO DEBUG.** One reason they are often inherently more buggy, as security researchers point out, is that JIT engine bugs are infamously difficult to diagnose, debug and correct. That's because the JavaScript code that may exhibit incorrect behavior (the webpage application being optimized) is not the same program that may contain the bug (the JIT compiler itself.) In addition, the code that creates the incorrect behavior is transiently generated at runtime and not available for static inspection in a file or on a disk either before or after the fact.

3. **JIT ENGINES CAN LEAD TO ARBITRARY CODE EXECUTION.** But the biggest JIT engine weakness often is derived when its ability to write and execute unsigned unvetted code in memory is abused. Bad actors take advantage of the JIT engine's ability to execute unsigned and unvetted code to carry out a sequence of exploits that together can lead to something called arbitrary code execution. Arbitrary code execution refers to a security exploitation where an attacker can run any code of their choosing on a target system. Because a JIT browser is unique in that it is allowed to write unvetted unsigned code, an exploit can be used to inject malicious code directly into memory to be executed at the browser level -- a privileged process running on a system – allowing arbitrary code execution. Once embedded with read/write/execute privilege, even if in a sandboxed area, skilled adversaries can escape the sandbox, escalate privileges, and gain access to phone features and data to be manipulated at their will. The Pegasus example makes this as clear as day.

4. **THE EXPLOITS CREATED THROUGH JIT ENGINES ARE ESPECIALLY HARMFUL AND EXTREME.** These JIT weaknesses have been actively exploited by criminal groups and nationstate actors, as a part of sophisticated exploit chains targeting journalists, activists, and enterprise environments to break out of the sandbox, gain access to sensitive data, and effectively commandeer a victim's device. This means a sufficiently skilled attacker could set up a malicious webpage which contains code that exploits the JIT engine, use it to break out of the sandbox, run code at a high privilege, ultimately gaining unauthorized access to core underlying system resources and sensitive data. In some especially sophisticated circumstances, they can lead to the complete takeover of a phone and its contents. In short: what starts as a bug in the browser or a bug in the JIT itself can quickly escalate to become a full system compromise – and the Internet even has directions for it. Thus, it has become a powerful way attackers can hijack a system and make it do what they want, rather than what it's supposed to do.

5. **JIT BASED ATTACKS CAN BE INITIATED SIMPLY BY SENDING PEOPLE TO EXPLOITED WEBSITES.** Bad actors easily and commonly use phishing links, web ads, or targeted LinkedIn requests to direct users to a website containing a carefully crafted malicious JavaScript payload to be acted on by a JIT engine. This is often the first step in a sophisticated exploit chain involving JIT – a relatively simple and often invisible way to initiate an attack.

Because they are simple to target, and in sophisticated cases can lead to full system access, JIT abuse or exploits are often utilized by nation states actors and in intelligence grade spyware campaigns – and in these cases they use full system compromises to gain access to location, microphones, cameras, e-mails, texts, and login credentials. And because smartphones are in the pockets and purses of almost every CEO, critical infrastructure operator, national security professional, elected official, military commander, judge, and journalist – the exploits have high value.

BUT HOW SERIOUS CAN A JIT EXPLOIT REALLY BE?

We offer four examples:

- **FIRST**, Intelligence Grade Mercenary Spyware. The original Pegasus mercenary spyware took advantage of a JIT engine to compile malicious code and deliver in directly to the read/write/execute memory as part of a sophisticated chain of exploits to achieve unsigned arbitrary code execution to silently compromise smartphones and gain full remote access and surveil the phones of high-value targets. Pegasus, a sophisticated piece of mobile spyware, is used by nation state actors to silently gather a wide range of sensitive information. It can read your private messages and emails, listen to phone calls, capture screenshots, log keystrokes (passwords), track your location, and harvest information from apps (chat apps, social media, etc.) For organizations, a Pegasus-infected device in the hands of an employee or executive can lead to major data breaches, corporate espionage, and compromised enterprise networks.

  Pegasus also was notoriously at the root of a European scandal where it was used to target and silence journalists, human rights defenders, political opponents, and dissidents. Mercenary spyware has also been installed onto EU official's phones where it can extract messages, record calls, track location, and secretly activate microphones. The European Parliament Research Service investigators link Pegasus spyware to human rights harms including intimidation, harassment, detention, and even murder. While Europe has tried to stake a reputation as a global tech enforcer, it has long been criticized for its laissez-faire approach to spyware - which is exacerbated by regulatory efforts like the DMAs interoperability rules that loosen the technology safeguards that help protect Europeans from spyware.

- **SECOND,** Russian Hackers have also used JIT bugs to attack adversaries. Google's Threat Analysis Group (TAG) has also flagged that Russian hackers are also actively taking advantage of bugs in JIT software as part of a sophisticated exploit chain to attack adversaries, plant malware and steal information. The intrusion set has been attributed with moderate confidence to a Russian state-backed threat actor codenamed APT29 (aka Midnight Blizzard) which has also targeted French diplomatic entities. These campaigns deliver n-day exploits for which patches are available, but would still be effective against unpatched devices.

- **THIRD**, Nation State actors continue to actively exploit JIT engines as you read this. Security analysts report that a recent and rapid escalation in four separate high-severity active Chrome browser exploits over 7 months this year – each of which required an emergency patch – indicates that "nation-state hackers are systematically targeting browser infrastructure for espionage operations." Each of these exploits involved Chrome's V8 JavaScript JIT engine. The researchers say, "threat actors are becoming increasingly sophisticated at weaponising browser weaknesses" and conclude, this "reinforces that browser security has become a critical component of national infrastructure protection, requiring layered security strategies that assume systems will be compromised and focus on rapid detection and containment of threats."

- **FOURTH**, it's so serious that disabling the JIT Engine is a core feature of all high threat security features from major technology companies. Because of the severe nature of the security risks that JIT can create and the core nature of the technology, major browser engine makers have made the disabling of JIT a core part of their most critical security modes. For example, after major Pegasus attacks, Apple created an extreme security feature called "Lockdown Mode" which disables JIT and is recommended for "individuals who, because of who they are or what they do, might be personally targeted by some of the most sophisticated digital threats." Likewise, Microsoft created its "Super Duper Secure Mode" which also disables its JIT engine. And Chrome created a "Secure Mode" specifically to disable JIT.

WHAT DO THE DMA'S INTEROPERABILITY REQUIREMENTS HAVE TO DO WITH JIT?

DMA regulators say "The objective of Article 6(7) DMA is that devices, apps and products from third parties can be used on an iPhone as seamlessly as Apple's own products." The DMA regulations broadly require that "interoperability must be granted to the same feature under equal conditions." This has set up an expectation that any "features that are so far only available to Apple" should also be made interoperable and accessible through an API to third parties.

In this case, Apple does have a feature that was previously only available to Apple. And now because of the DMA's requirements that it allow alternative third-party web browser engines on its platform it has created a toolkit and an API to allow certain trusted browser developers to create their high-performance browsers by enabling them to create and use their own JIT engines to write and execute unsigned code in a secure and trusted environment. But now, because there is a JIT access API, and because Apple can write and execute code in memory itself, this developer believes the DMA allows, in fact requires, that it also be given access to the API so it can directly write and execute any code of its choosing.

A key problem in this case is that this is exactly the opposite approach of where the security establishment has been headed. Security professionals seek to ensure code has been securely developed, the supply chain protected, and code delivered to the end-user unaltered through cryptographic 'signing' of the state of the code, and signed patches as needed to fix vulnerabilities once discovered. Software signing has become a critical security best practice that is incorporated into every major operating system architecture, a practice recommended by the National Institute of Standards and Technology (NIST), and part of the European Union Agency for Cybersecurity (ENISA)'s larger push for stronger software supply chain security and product certification. These steps help prevent the execution of malicious code, build user trust, protect against supply chain manipulation, and enhance software security.

In Apple's case, it requires the code for all its apps on iOS to be vetted and signed, either as part of its App Store review process or through a notarization process for apps distributed from outside its App Store. However, unvetted DMA interoperability requests for access to the JIT could allow arbitrary unvetted and unsigned code to get around these safeguards. Unfettered access to the JIT, or use of insecure JIT could relieve hackers of the need to use sophisticated tricks to find and exploit the JIT or JIT bugs, they could just use this proposed DMA API access to directly write and execute code in memory.

WHAT COULD POSSIBLY GO WRONG?

While the DMA was intended to force "gatekeepers" to open-up their platforms to any interested competitor providing the same service as the gatekeeper, the first applicant disclosed does not even seem to be an Apple competitor. Apple does not offer a Linux emulator, the service subject to the request.

And it comes at a time when eight in ten (79%) of Europeans want stronger cybersecurity according to Eurobarometer. But DMA regulators have ignored its security experts, public sentiment, and even the law. In fact some EU leaders even insist that claims that the DMA undermines security is "complete nonsense." We tend to disagree.

WHO LOSES WHEN INTEROPERABILITY MANDATES WEAKEN SECURITY?

Insecure interoperability can create significant new security risks for businesses, critical infrastructure providers, national security personnel and journalists. While the requestor in this DMA interoperability request argues access could inure to its benefit, looking at the ecosystem, and combined with other security weaknesses DMA regulators have mandated, it would likely could cause significant harm to the broad number of businesses and others who rely on the integrity of their technologies to protect them.

As companies throughout Europe turn to mobile devices to boost productivity, improve customer services, and improve outcomes, trust in mobile technology has become even more essential. But trust in the integrity of the EU's mobile ecosystem is now increasingly at risk – as is enterprise security.

ENTERPRISES ARE ESPECIALLY PUT AT RISK.

Weakening Apple's critical security safeguards by allowing any developer to broadly use an API that allows JIT direct code writing and execution, combined with the other key security weaknesses Europe is enabling, would predictably expand potentially significant cyber risks to the 82% of European organizations leveraging mobile platforms in the workplace whose cybersecurity posture in the mobile environment would be weakened. Europe's service sector, industrial sectors, agriculture, finance, SMEs and others are heavily dependent on being able to secure their technology for essential operations.

As Lookout reports, "Thanks to Apple's walled garden approach, iOS is the mobile operating system of choice for most enterprise organizations." Because Apple has driven security into the heart of its mobile architecture, three-quarters of IT professionals report Apple is more secure than other types of devices. In fact, in a survey taken before the DMA came into effect, 76% of IT professionals report that Windows or Android devices are more likely to be targeted in a cyberattack than Apple devices, and 77% agree Apple devices are easier to secure than Window/Android, according to a Kandji survey of IT professionals. Kandji also found companies with a higher percentage of Apple devices are significantly more likely to report security benefits.

These are important benefits from a securely developed smartphone ecosystem. Europe's enterprises make enormous investments to secure their businesses, protect their data, and secure their networks – investing an estimated USD 63.12 billion in cybersecurity 2025. These investments rely upon the use of secure mobile devices, software and applications. By contrast, according to Omdia, weakening mobile device security can have devastating impacts on enterprises including on financial institutions, multinational corporations, major retailers, telecommunications providers -- impacting almost every sector of the economy.

JIT exploits or abuse, or mandates that make it easier for an attacker to write and execute unsigned unverified code, can be involved in sophisticated attack chains that start with mere phishing attacks and unleash new potential vectors for ransomware attacks, data theft, and enabling other forms of malicious software to gain access to enterprise networks, putting the security of the entire enterprise at risk.

A SINGLE MOBILE BROWSER EXPLOIT COULD COST EUROPEAN BUSINESSES HUNDREDS OF MILLIONS.

- **A SINGLE MOBILE PHISHING ATTACK** could cost a 5,000-employee organization almost $4 million according to Lookout. But unmitigated mobile phishing threats could cost organizations with 10,000 mobile devices as much as $35 million per incident and up to $150 million per incident for organizations with 50,000 mobile devices.

- **FOR SMALL BUSINESSES**, the average cost of a cyber-attack ranges between $120,000 and $150,000, with some breaches potentially costing much more depending on the severity, company size and industry.

The security weaknesses that the DMA's mandates enable could force these companies to increase their costs for effective deterrence, and they may also have to pay the substantial costs from the increased number of significant cyber events – putting European companies at a competitive disadvantage in the global market.

As mobile devices have become such a crucial platform for almost every business, consumer and almost every sector of the economy, this is not the time to be requiring security safeguards to be lowered or weakened. Nor is it time to allow third parties to circumvent core platform protections, or to gain direct interoperable access to critical functionality that could inadvertently put security at risk. We've seen how this can play out before.

Enabling weaknesses in an area where bugs are hardest to detect, where exploits are of the highest severity, in systems that are actively being exploited by nation state actors, and exploits that enable broad enterprise security risks for companies throughout Europe just doesn't make sense.

This should be a RED FLAG WARNING to policymakers looking to replicate key aspects of the DMA without seriously contemplating its security impacts.

TRUSTED FUTURE

trustedfuture.org