Time as an Attack Surface

Andy Davis, Global Research Director

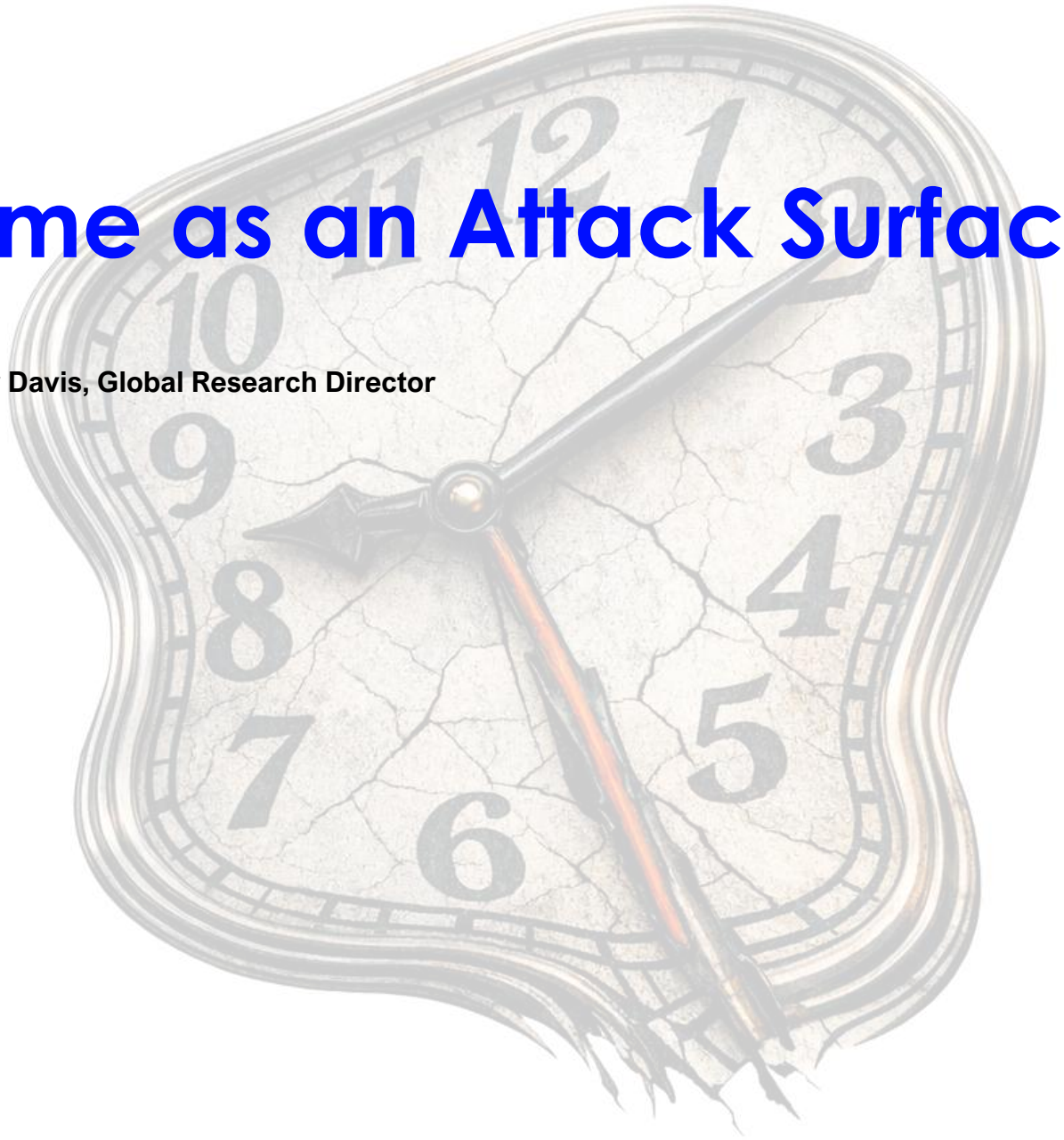

Table of Contents

1. Introduction

What is the time? Time is one of the most fundamental assumptions embedded in modern computing systems. From cryptographic protocols and authentication mechanisms to distributed systems and industrial control processes, accurate and consistent timing is treated as a given. Clocks are assumed to run forward, to drift only within narrow tolerances and to reflect a shared reality across devices and networks. As a result, time is rarely treated as an input that must be validated or defended, but rather as trusted infrastructure that quietly underpins security, safety and reliability.

This assumption is increasingly fragile. Modern systems depend on complex layers of hardware oscillators, software clocks, virtualised environments and network-based synchronisation protocols such as NTP (Network Time Protocol), PTP (Precision Time Protocol), and GNSS (Global Navigation Satellite System)-derived time. These mechanisms were largely designed to maximise accuracy and availability, not to withstand an active and adaptive adversary. In many cases, a subtle manipulation of time can be enough to undermine cryptographic guarantees[1], bypass authentication controls[2], destabilise control loops[3], or invalidate forensic records[4] without triggering obvious alarms.

The risk is amplified by broader technological trends. Cloud computing, containerisation and virtual machines abstract time away from physical hardware, placing it under the control of hypervisors and orchestration layers. At the same time, Industrial Control Systems, IoT devices, and safety-critical platforms increasingly rely on low-cost oscillators and commodity components that are vulnerable to environmental influence[5] and physical manipulation. Systems that once relied on isolated, deterministic timing sources are now interconnected, synchronised, and exposed.

This white paper argues that time should be treated as a first-class attack surface. It explores how clock drift, time synchronisation failures, and deliberate oscillator manipulation can be exploited to undermine cryptography, authentication, industrial automation, and safety systems. By examining both technical mechanisms and real-world impacts, it aims to reframe time from a background assumption into an explicit security and resilience concern - one that demands the same rigor as memory safety, input validation, and firmware integrity.

2. A Primer on Clocks, Oscillators, and Synchronisation

At the lowest level, all digital systems derive their notion of time from a physical oscillator. These oscillators, commonly quartz crystal or MEMS (Micro-Electro-Mechanical Systems)-based devices, produce a periodic electrical signal that is counted to measure the passage of time. No oscillator is perfect: all are subject to drift caused by temperature variation, voltage instability, mechanical stress, and long-term aging. Higher-quality oscillators such as temperature-compensated (TCXO) or oven-controlled (OCXO) devices reduce (but do not eliminate) these effects. In cost-constrained devices, including many IoT and embedded platforms, relatively inexpensive oscillators are used, making time inherently imprecise and environmentally sensitive.

On top of this hardware foundation sit software clocks, which translate raw oscillator ticks into system time. Operating systems typically maintain multiple notions of time, including a wall-clock time intended to reflect real-world chronology and a monotonic clock intended for measuring intervals. These clocks may be adjusted, stepped forwards or backwards, or slewed gradually to correct drift. In virtualised and containerised environments, clocks are often abstracted further, with guest systems relying on a hypervisor or host-provided time source. This layering introduces additional assumptions and failure modes, particularly where guests implicitly trust time values that are ultimately controlled by another software component.

Because individual clocks inevitably drift, most systems rely on time synchronisation protocols to maintain a shared notion of time across devices. NTP is widely used in enterprise and internet-connected environments and is designed to balance accuracy, scalability, and resilience to benign faults. PTP, defined by IEEE 1588, provides much tighter synchronisation and is commonly used in industrial and telecommunications systems. Many deployments also treat satellite-derived time, such as GPS (Global Positioning System) or other GNSS sources, as a root of trust. While these mechanisms are highly effective at maintaining accuracy, they often assume that time sources and communication paths are honest or at least non-adversarial.

Crucially, most synchronisation technologies prioritise correctness and availability over security[6]. Authentication, integrity protection, and plausibility checking are frequently optional or inconsistently deployed. Even where

cryptographic protections exist, systems may tolerate significant time offsets to maintain availability, creating exploitable margins. Subtle manipulation, introducing small but persistent skew rather than abrupt changes, can remain within expected operational bounds while still causing dependent systems to misbehave in security-relevant ways.

The result is that time in modern systems is not a single, stable value but an emergent property of interacting hardware components, software layers, and network services. Errors or manipulation at any point in this chain propagate upward, often invisibly, into application logic, cryptographic protocols, control systems, and safety functions. Understanding this layered construction is essential to understanding why time should not be treated as an unquestioned constant, but as a dependency whose integrity must be explicitly protected.

3. Threat Model: Time as an Adversarial Control Plane

Traditional security threat models focus on data, identity, and execution flow, but they rarely treat time itself as something an attacker might deliberately influence. In most systems, clocks are assumed to be correct within a reasonable tolerance, and any deviation is treated as a fault rather than a hostile act. This creates a blind spot: an attacker who can manipulate a system's perception of time is often able to bypass controls indirectly, without breaching cryptographic algorithms or application logic. Time becomes an invisible control plane through which higher-level security assumptions can be subverted.

Attackers need not gain full control over a system's clock to cause harm[7]. In many cases, partial influence is sufficient. Network-positioned adversaries may be able to delay, replay, or spoof time synchronisation traffic, subtly skewing clocks while remaining within accepted thresholds. Local attackers, or those with limited physical proximity, may influence oscillator behaviour through environmental means such as temperature, voltage fluctuation, or electromagnetic interference[8]. More capable attackers may exploit software interfaces, privileges, or hypervisor controls to directly adjust system time in virtualised or cloud environments.

Time-based attacks can be categorised by their effect rather than their mechanism. In some cases, clocks are shifted abruptly forwards or backwards, potentially invalidating certificates, reviving expired tokens, or causing safety systems to misinterpret system state. In other cases, attackers introduce gradual skew (small changes applied continuously) allowing systems to remain nominally "synchronised" while slowly diverging from reality. This form of manipulation is particularly dangerous because it is harder to detect and often remains within operational tolerances designed to accommodate benign drift.

Another class of attacks involves desynchronisation[9] rather than absolute error. Different components of a distributed system may be forced to disagree about the current time, even if none of them appear obviously incorrect in isolation. This cross-domain skew can break assumptions about ordering, consensus, timeout behaviour, and fault detection. In industrial and safety-critical systems, such inconsistencies can destabilise control loops or undermine redundancy mechanisms that rely on coordinated timing.

Finally, attackers may target availability by freezing or destabilising time altogether. Watchdog timers, safety interlocks, and heartbeat mechanisms frequently depend on the monotonic progression of time. Preventing clocks from advancing, or causing them to behave erratically, can trigger fail-open conditions, suppress fault detection, or push systems into degraded modes. Because these behaviours are often designed to handle accidental faults, they may lack robust protections against deliberate abuse.

Viewed through this lens, time is not merely an environmental dependency but a shared resource that can be contested, manipulated, and weaponised. Treating time as an adversarial input, subject to deception, distortion, and partial control, provides a more realistic foundation for assessing risk in modern IT, OT, and cyber-physical systems.

4. Undermining Cryptography and Secure Protocols

Modern cryptographic systems rely heavily on time to establish freshness, prevent replay, and enforce the validity of keys and credentials. While the strength of cryptographic algorithms is usually analysed in isolation, their real-world security often depends on assumptions about clocks behaving correctly. When these assumptions fail, either through

accident or adversarial manipulation, the guarantees provided by otherwise sound cryptography can be quietly undermined.

One of the most common uses of time in cryptographic protocols is replay protection. Timestamps, expiry windows, and time-bound nonces are widely used in protocols such as TLS, Kerberos, OAuth, and SAML (Security Assertion Markup Language) to ensure messages are recent and cannot be reused by an attacker. If a system's clock is forced backwards, previously expired credentials or messages may suddenly become valid again. Conversely, pushing time forwards can prematurely invalidate sessions or keys, triggering denial-of-service conditions or forcing insecure fallback behaviour. In many deployments, tolerance windows are deliberately wide to handle benign clock drift[10], creating exploitable margins for an attacker who can introduce controlled skew.

Time also plays a critical role in key management and certificate validation. Digital certificates depend on validity periods defined by "not before" and "not after" timestamps. If an attacker can influence a relying party's perception of time, they may be able to bypass certificate checks by making expired certificates appear valid, or by accepting certificates that are not yet valid. Key rotation mechanisms, which are essential for limiting the impact of compromise, similarly depend on accurate timekeeping. Manipulating time can delay rotation indefinitely or cause systems to fall back to older, weaker keys.

Trusted execution environments and secure elements introduce further complexity. These platforms often rely on monotonic counters or trusted time sources to prevent rollback attacks and to enforce security policies over time. In practice, trusted time is frequently bridged to untrusted external clocks, particularly in virtualised or cloud environments. If this linkage is poorly defined or insufficiently protected, attackers may be able to roll back system state, re-enable deprecated secrets, or bypass usage limits without directly breaking hardware protections.

Importantly, many cryptographic failures caused by time manipulation are silent. Systems continue to operate normally, keys verify successfully, and protocol handshakes complete as expected. From the perspective of application logic, nothing appears wrong. This makes time-based cryptographic attacks particularly dangerous: they do not require weaknesses in algorithms or implementations, only the ability to influence a shared assumption about time.

These issues illustrate a broader problem in cryptographic system design. Time is often treated as a benign parameter rather than an adversarial input. As long as clocks are assumed to be "approximately correct," cryptographic logic built on top of them remains fragile. Treating time as untrusted and designing protocols that minimise or explicitly constrain its influence is therefore essential to preserving the real-world security of cryptographic systems.

5. Authentication, Authorisation, and Identity Systems

Authentication and identity systems are among the most time-dependent security components in modern IT estates. They routinely use time as a proxy for freshness, user presence, and policy enforcement: tokens expire, sessions time out, one-time passwords roll over and access decisions are bounded by validity windows. This works well when clocks are accurate and consistent, but it also creates a predictable dependency that attackers can target. If an adversary can influence the local clock on a client, a server, or a key identity component, or can create disagreement between them, then controls intended to limit replay, constrain session lifetime, or enforce step-up authentication can be weakened or bypassed.

TOTP (Time-based One-Time Passwords) illustrate the problem clearly. TOTP assumes that the prover and verifier share a sufficiently similar notion of current time, and most implementations allow a drift window to reduce user friction. That tolerance is operationally necessary, but it can also be exploited: an attacker who can push a device clock forward or backward can increase the effective window in which captured codes remain acceptable or can trigger repeated resynchronisation and recovery flows that degrade security. Even when the cryptography is correct, the system's security is bounded by how strictly it validates time and how it handles exceptions, particularly in fallback paths, account recovery, and "I can't access my authenticator" processes.

Session management and token lifetimes create similar opportunities. Web sessions, API access tokens, refresh tokens and federated identity assertions typically carry expiry times that are evaluated against system clocks. If a relying party's clock is manipulated backward, expired tokens may be accepted again, prolonging attacker access and undermining incident response measures such as forced logouts. If time is pushed forward, legitimate sessions may be invalidated early, leading to denial-of-service conditions or prompting users and systems to adopt insecure

workarounds. In distributed environments, inconsistent clocks between front ends, identity providers, gateways, and downstream services can produce subtle “split-brain” authorisation behaviour, where one component accepts a token that another rejects, creating openings for privilege escalation, confused-deputy problems, or bypasses through alternate request paths.

Federated identity and single sign-on amplify these risks because they depend on time to bind trust across organisational boundaries. Assertions in SAML and tokens in OAuth/OIDC include time claims (such as “issued at,” “not before,” and “expires at”) that help constrain replay and prevent indefinite reuse. If an attacker can induce clock skew in the relying party, they may widen the effective acceptance window for assertions or cause the relying party to accept tokens outside the intended policy. Conversely, manipulating time can reliably trigger authentication failures that look like routine operational issues, masking an attack in a sea of “clock drift” errors, user complaints, and intermittent login problems.

Time also influences authorisation decisions beyond authentication tokens. Many organisations enforce time-of-day rules, conditional access, step-up authentication windows, “recently authenticated” requirements for privileged actions, and timed elevation models (for example, privileged roles granted for a limited duration). These controls assume the clock is correct on whichever system evaluates policy. If the policy engine relies on client-supplied time, a compromised endpoint can trivially lie. Even when policy is evaluated server-side, attacks that skew server or intermediary clocks can shorten or extend privileged windows in ways that defeat the intent of controls. In operational terms, this means attackers may be able to make privileges “stick” longer than they should, or to force repeated reauthentication that fatigues users and increases the chance of social engineering success.

The key lesson for identity systems is that time should be treated as an adversarial input and a fragile dependency, not a background constant. Robust designs reduce reliance on wall-clock time where possible, use monotonic clocks for measuring durations, enforce bounded rate-of-change checks on time adjustments, and avoid fail-open behaviour when time becomes uncertain. Perhaps most importantly, identity architects should scrutinise exception handling and recovery workflows: many real-world compromises[11] occur not through the primary authentication path, but through the “helpful” paths that activate when time drift, token expiry, or synchronisation errors occur. In a world where attackers can contest time, the resilience of identity systems depends on how safely they fail when time cannot be trusted.

6. Industrial Control Systems and Cyber-Physical Safety

Industrial control systems (ICS) and other cyber physical systems place especially strong trust in time. Control logic assumes that inputs are sampled at predictable intervals, control outputs are applied in the correct sequence, and safety mechanisms activate within known time bounds. These assumptions are deeply embedded in PLCs (Programmable Logic Controllers), DCS (Distributed Control Systems) and the engineering standards that govern them. Unlike many IT systems, ICS environments often prioritise determinism and availability over flexibility, making them particularly sensitive to disruptions or manipulation of timing.

Accurate timing is central to the stability of control loops. Sensors are polled at fixed rates, control algorithms assume consistent cycle times, and actuators respond based on the relative ordering of events. If clocks drift or are deliberately skewed, even slightly, these assumptions can break down. Errors introduced by timing inaccuracies can degrade control performance or push systems toward unstable operating regimes. In many cases, this degradation is gradual and silent, manifesting as reduced efficiency, increased wear or intermittent faults rather than immediate failure. These are conditions that are difficult to diagnose and easy to misattribute to mechanical or process issues.

Safety systems depend on time in different but equally critical ways. Watchdog timers, heartbeat signals, and time-based interlocks are designed to detect failures and bring systems into a safe state when something goes wrong. These mechanisms assume that time progresses monotonically and within known bounds. If an attacker can freeze a clock, slow its progression, or cause different components to disagree about elapsed time, safety logic may fail to trigger when it should, or may trigger unexpectedly. A failure-to-trip scenario can allow unsafe conditions to persist, while nuisance trips can cause operational disruption, mask real hazards, or encourage operators to bypass safety mechanisms altogether.

Time synchronisation also plays a growing role in large-scale industrial environments. Power grids, substations, and energy management systems use precise time alignment to maintain stability. Manufacturing systems increasingly coordinate actions across multiple controllers and sites, relying on shared time to ensure correct sequencing. In such

environments, it is not enough for each device to have a plausible local clock; the system as a whole depends on consistent time across domains. Attacks that induce desynchronisation, rather than grossly incorrect time, can therefore result in undesirable effects, undermining redundancy, fault detection, and coordinated response.

The convergence of IT and OT further expands the timing attack surface. Industrial networks are increasingly connected to enterprise systems, remote maintenance platforms, and cloud-based monitoring services, pulling in time synchronisation mechanisms originally designed for office or internet environments. Virtualisation, software-defined networking and remote engineering workstations introduce additional layers where time may be abstracted, adjusted, or indirectly controlled. As a result, assumptions that once held in isolated, air-gapped environments are no longer universally valid and timing faults can potentially propagate from corporate IT into operational and safety domains.

Crucially, many industrial standards[12] and safety certifications treat timing errors as accidental faults rather than adversarial acts. Hazard analyses typically consider random clock failure, loss of synchronisation, or component drift, but not deliberate and adaptive manipulation. This distinction matters, as systems designed to tolerate benign faults may fail in dangerous ways when timing is contested by an attacker who understands tolerance thresholds, fallback behaviours, and operator responses. Addressing time as an attack surface therefore requires not only technical mitigations, but also a shift in how risk is modelled, tested, and assured in cyber-physical and safety-critical systems.

7. Oscillator Manipulation and Physical-Layer Attacks

At the physical layer, time ultimately depends on the behaviour of an oscillator, and oscillators are governed by the laws of physics rather than software policy. Crystal and MEMS oscillators are sensitive to their operating environment, and their frequency can be influenced by temperature, supply voltage, mechanical stress, and aging. While these effects are well understood and accounted for in benign fault models, they also present an opportunity for adversarial manipulation. An attacker who can influence the physical conditions around a device may be able to induce predictable clock drift or instability without modifying firmware or accessing logical interfaces.

Environmental manipulation represents one of the simplest classes of physical-layer timing attacks. Changes in temperature can shift oscillator frequency, sometimes significantly, particularly in cost-optimised designs. Similarly, voltage fluctuations, whether introduced intentionally through power-supply manipulation or indirectly through load effects, can cause oscillators to speed up or slow down. In many embedded and industrial systems, such variations remain within documented tolerances and therefore do not trigger fault alarms. From the system's point of view, the clock is "behaving normally," even as timing-dependent security and safety assumptions are being eroded.

Beyond coarse environmental effects, more targeted techniques can be used to influence oscillator behaviour. Electromagnetic interference can inject noise or bias into clock circuits, altering frequency or introducing jitter. Acoustic attacks, exploiting mechanical resonance[13] in certain crystal packages, have been shown to perturb oscillators in ways that meaningfully affect system timing. While these methods may sound exotic, they can be practical in real-world settings where devices are deployed in uncontrolled environments, shared facilities, or exposed field installations. Importantly, such attacks do not require persistent access or modification - they can be subtle, and difficult to attribute.

The feasibility of oscillator manipulation is strongly influenced by economic and design pressures. Many modern systems, including IoT devices and safety-adjacent embedded platforms, prioritise cost, size, and power consumption over timing robustness. This often leads to the use of inexpensive oscillators with limited compensation and little redundancy. In high-volume deployments, even small, systematic timing errors can have large aggregate effects, particularly when devices rely on synchronisation protocols that assume roughly similar oscillator behaviour across the environment.

Physical-layer timing attacks blur traditional security boundaries. They may be invisible to malware detection, network monitoring, and integrity checks, yet still produce security-relevant effects at higher layers. A cryptographic protocol that fails due to clock drift, or a safety mechanism that misfires because a watchdog timer behaves unexpectedly, may appear to be a software or configuration issue when its root cause lies in the physical manipulation of time itself. This complicates incident response and can significantly delay correct diagnosis.

From a defensive perspective, oscillator manipulation highlights the limits of assuming a trustworthy hardware substrate. Hardening against these attacks requires both technical and architectural responses: higher-quality

oscillators or compensated designs for critical functions; redundancy and diversity in time sources; environmental sensing to detect anomalous conditions and system-level checks that constrain how rapidly or how far clocks are allowed to drift.

8. Distributed Systems, Cloud, and Virtualised Time

Distributed systems depend on some notion of shared time to coordinate work across components that do not share memory or execution context. Time is used to trigger retries, enforce leases, expire state, and impose order on events that occur concurrently. While many systems are theoretically designed to avoid reliance on perfectly synchronised clocks, in practice wall-clock time is widely used as a simplifying assumption. This creates a fragile dependency: when clocks diverge, stall, or behave inconsistently, the system may continue operating while making unsound decisions about coordination, freshness, and trust.

Cloud computing and virtualised environments complicate this picture by abstracting time away from physical hardware. Virtual machines and containers typically obtain time from a host or hypervisor, which may itself be synchronised via network time services. As a result, guest systems often assume they have direct access to a reliable clock, even though they are several layers removed from the underlying oscillator. Operations such as live migration, snapshotting, pausing, or restoring workloads can legitimately cause time to jump forwards or backwards from the guest's perspective. These behaviours are usually acceptable for availability, but they can violate the assumptions made by security-sensitive software.

In cloud environments, time is effectively part of the control plane. Providers may adjust host clocks to maintain estate-wide synchronisation, respond to detected drift, or recover from infrastructure faults. While these adjustments are generally benign, they highlight a change in trust boundaries: time is no longer exclusively owned or controlled by the workload operator. A compromise, misconfiguration, or failure in the underlying platform can therefore propagate timing anomalies into large numbers of otherwise isolated systems simultaneously. This raises the stakes of time manipulation, turning what might once have been a local fault into a systemic issue.

Distributed coordination mechanisms are particularly sensitive to these effects. Leader election, leases, distributed locks, and failure detection often rely on timeouts that assume clocks advance reasonably consistently across nodes. If time moves too slowly, too quickly, or inconsistently, systems may experience cascading retries, spurious failovers, or prolonged unavailability. In adversarial scenarios, selectively skewing time on a subset of nodes can be enough to destabilise coordination without directly violating safety properties, creating a low-noise denial-of-service vector that is difficult to diagnose.

The core challenge in distributed and virtualised environments is that time becomes an emergent property rather than a stable resource. It is shaped by hardware, software layers, orchestration behaviour, and network services, each with distinct failure modes and trust assumptions. Systems that treat time as a simple, reliable scalar are therefore brittle by design. Robust designs assume that time may jump, stall, or disagree between components, and explicitly account for these possibilities through measurements, and redundancy. In a world of cloud scale and pervasive virtualisation, the security of distributed systems increasingly depends not on how precisely time is synchronised, but on how well systems behave when that synchronisation cannot be trusted.

9. Ordering, Consensus, and Logging

Distributed systems rely on time to impose order on events that occur across multiple nodes, networks, and failure domains. Whether coordinating database writes, enforcing timeouts, or determining which message arrived "first," many systems use timestamps as a convenient way to reason about causality. While alternative models such as logical clocks exist, wall-clock time is still widely used in practice because it is intuitive, cheap, and readily available. This reliance creates a hidden fragility: if the system's perception of time is inaccurate or inconsistent, the ordering guarantees that higher-level logic depends on can silently fail.

Consensus algorithms are particularly sensitive to timing assumptions. Protocols used in distributed databases, coordination services, and control planes rely on carefully tuned timeouts and assumptions about relative clock behaviour. While these algorithms are often designed to tolerate some degree of clock skew, they typically assume

that clocks progress monotonically and do not diverge arbitrarily. An attacker who can slow, pause, or skew clocks on selected nodes may be able to destabilise leader election, trigger unnecessary reconfiguration, or fragment the cluster's view of the world. Even when safety properties are preserved, liveness can be degraded in ways that resemble intermittent faults, making attacks difficult to distinguish from benign network or performance issues.

Logging, monitoring, and forensic analysis are another critical area where time assumptions matter. Security investigations depend heavily on timestamps to reconstruct sequences of events across systems and domains. If logs are generated using clocks that are skewed, manipulated, or inconsistently synchronised, correlating events becomes unreliable. An attacker who can influence time may be able to obscure the true order of actions, create misleading narratives, or cast doubt on otherwise definitive evidence. In some cases, manipulation need only be subtle: a few seconds of skew between systems may be enough to frustrate correlation at scale.

The problem is compounded in environments that span on-premise infrastructure, cloud platforms, and virtualised workloads. In these settings, time may be sourced from different layers: hardware clocks, hypervisors, host operating systems, or external synchronisation services, each with its own trust assumptions. Migration, snapshotting, and restore operations can legitimately cause clocks to jump or rewind, creating edge cases that attackers can exploit or hide within. When time behaves unexpectedly, operators often focus on availability and performance impacts, overlooking the security implications of ordering and audit failure.

The overarching risk is that time is frequently used as a shortcut for reasoning about causality, trust, and history. When that shortcut is compromised, systems may still appear to function correctly while making fundamentally unsound decisions about ordering, consensus, and accountability. Defending against these risks requires explicit recognition that wall-clock time is an unreliable narrator. Systems should minimise reliance on absolute time for security-critical decisions, incorporate logical or monotonic ordering where possible, and ensure that logging and audit mechanisms record enough contextual information to remain meaningful even when time is uncertain or contested.

10. Detection: How Do You Know Time Is Under Attack?

Detecting time-based attacks is challenging precisely because time is treated as background infrastructure rather than as a monitored security signal. Most systems are designed to cope with benign clock drift, transient synchronisation loss, or occasional adjustment without raising alarms. As a result, many indicators of adversarial time manipulation are filtered out as routine operational noise. When time is under attack, systems often continue to function (cryptographic checks pass, control logic executes, and services remain available) while security guarantees can quietly degrade.

One of the earliest signs of a timing problem is anomalous behaviour in time-dependent controls. Examples include authentication tokens being accepted or rejected unexpectedly, certificates appearing valid when they should not be, or multi-factor authentication failures clustering around specific time windows. In industrial or cyber-physical systems, operators may observe instability in control loops, unexplained watchdog resets, or safety interlocks triggering inconsistently. These symptoms are rarely flagged as security issues and are more commonly attributed to configuration drift, software bugs, or hardware aging.

In distributed systems, disagreement about time between components is often more revealing than absolute clock error. Logs from different systems may show events occurring "out of order," negative time deltas, or implausible sequences that cannot be explained by network latency alone. Consensus systems may demonstrate repeated leader elections, unexplained failovers, or erratic timeout behaviour. Importantly, none of these effects require clocks to be wildly incorrect; differences that remain within documented tolerance windows can still be enough to break assumptions about ordering, freshness, or liveness.

Direct monitoring of clock behaviour can provide valuable signals, but it is rarely implemented with adversarial use cases in mind. Sudden time steps, repeated adjustments, or oscillation between time sources may indicate interference or manipulation, particularly if they correlate with security-relevant events. Gradual skew is harder to detect, but rate-of-change analysis (tracking how quickly a clock diverges rather than where it ends up) can expose behaviour that is inconsistent with natural drift. Similarly, discrepancies between wall-clock time and monotonic clocks can reveal rollback or freezing attempts that might otherwise go unnoticed.

Cross-checking time against multiple independent sources is one of the most effective detection strategies, but only if discrepancies are actively analysed rather than silently corrected. Many systems are configured to "always trust" a preferred time source, masking faults or attacks elsewhere in the chain. Alerting on loss of synchronisation, asymmetric path delays, or repeated selection of fallback time sources can provide early warning that timing assumptions are being stressed or exploited. In safety-critical environments, correlating environmental sensor data, such as temperature or voltage anomalies, with clock drift can help distinguish adversarial manipulation from ordinary operating conditions.

For logging and forensics, detection depends as much on handling uncertainty as on spotting errors. Systems that assume timestamps are authoritative may produce logs that are misleading or contradictory under attack. Recording additional contextual information, such as sequence numbers, monotonic counters and synchronisation state, can preserve investigative value even when absolute time is unreliable. From a defensive standpoint, the goal is not to eliminate all clock error, but to make timing anomalies visible and actionable rather than silent and ambiguous.

Ultimately, knowing when time is under attack requires a shift in perspective. Instead of asking whether clocks are accurate, defenders must ask whether clocks are behaving plausibly under adversarial conditions. Treating time as a monitored security signal, subject to validation, correlation, and anomaly detection, allows organisations to detect attacks that would otherwise blend seamlessly into the background noise of modern systems.

11. Mitigations and Defensive Design

Mitigating time-based attacks requires a shift in mindset as much as a change in technology. The most important design principle is to treat time as an untrusted input rather than as an inherent property of the system. Just as modern systems validate external data, constrain memory access, and authenticate identities, they must also validate, constrain, and reason explicitly about time. This does not mean eliminating all dependence on clocks, but rather acknowledging that clocks can be wrong, inconsistent, or adversarially influenced and designing systems that fail safely when those assumptions are violated.

At an architectural level, one of the most effective strategies is to reduce reliance on absolute wall-clock time for security-critical decisions. Where possible, systems should use monotonic clocks to measure durations and enforce timeouts, rather than comparing absolute timestamps. Logical counters, sequence numbers, challenge-response protocols, and state-based freshness checks can often replace or supplement time-based validity windows. When absolute time is unavoidable, its influence should be tightly bounded: acceptance windows should be explicit, minimal, and justified, rather than inherited implicitly from libraries or defaults.

Time synchronisation itself must be treated as part of the attack surface. Defensive designs use multiple, independent time sources and compare them for plausibility rather than blindly trusting a single "authoritative" source. Sudden jumps, rapid drift, or asymmetrical disagreement between sources should be treated as security-relevant events, not silently corrected. In high-risk environments, rate-of-change limits can prevent clocks from being adjusted faster than physically plausible, reducing the effectiveness of both abrupt and gradual manipulation. Importantly, systems should define clear behaviour for loss or degradation of synchronisation that prioritises safety and security over convenience.

Cryptographic and identity systems benefit from explicit defensive patterns around time. Tokens, certificates, and assertions should be validated using both time-based and non-time-based checks, so that failure of one signal does not automatically imply acceptance. Replay protection mechanisms that combine timestamps with unique identifiers or counters are more resilient than those that rely on time alone. Just as critically, fallback and recovery paths should be designed to fail closed when time is uncertain, rather than degrading silently into insecure modes.

In industrial and safety-critical systems, mitigation extends beyond software into hardware and physical design. Redundancy and diversity in oscillators and time sources can prevent single-point failures or manipulations from propagating unchecked. Environmental monitoring, such as temperature and voltage sensing, can provide early warning that clock behaviour may be unreliable. Where timing is integral to safety functions, explicit validation of timing assumptions should be incorporated into hazard analyses and testing regimes, including adversarial fault injection rather than purely random failure models.

Detection and response capabilities are a necessary complement to preventative measures. Systems should log not only timestamps, but also synchronisation state, clock adjustments, and monotonic counters, enabling operators and investigators to reason about event ordering even when absolute time is disputed. Alerts should be generated for anomalous clock behaviour, not just loss of synchronisation. Critically, organisations need operational playbooks that recognise timing anomalies as potential security incidents rather than purely technical faults, ensuring that unusual behaviour triggers appropriate scrutiny rather than routine troubleshooting.

Ultimately, defending against time-based attacks is about making systems resilient to uncertainty. Absolute precision is rarely achievable, and perfect synchronisation is not required for security. What matters is that systems behave predictably and defensibly when time becomes unreliable. By constraining the influence of time, cross-checking its sources, and designing for failure rather than perfection, engineers can significantly reduce the impact of an attacker who seeks to exploit one of the most fundamental and most overlooked assumptions in modern computing.

12. Strategic Implications

Treating time as an attack surface has implications that extend well beyond individual vulnerabilities or protocol misconfigurations. Strategically, it challenges a common organisational assumption: that security failures are primarily driven by flaws in applications, credentials, or network exposure. Time-based attacks sit beneath those layers, exploiting shared infrastructure and implicit dependencies that span teams, vendors, and environments. This means the blast radius is often cross-functional: a timing issue can manifest simultaneously as an IAM outage, a cryptographic validation anomaly, a distributed-systems instability, and an OT process upset, each investigated separately unless organisations deliberately connect them under a unified risk narrative.

Time attacks also blur the line between cyber security and physical security in a way that many governance models are not equipped to handle. Oscillator manipulation, GNSS interference, power instability, and environmental influence do not fit neatly into “IT controls” or “software hardening” programmes, yet they can have direct security and safety consequences. In operational environments, especially in industrial and safety-critical systems, timing manipulation can become a pathway to real-world impact without the attacker needing to compromise firmware or deploy malware. This complicates assurance: traditional security testing and compliance checks, focused on patch levels, configurations, and network controls, may declare systems “well secured” while leaving timing dependencies unexamined.

From an enterprise risk perspective, time manipulation is particularly disruptive because it undermines confidence in evidence. Audit trails, security logs, and telemetry are foundational to detection, incident response, and legal defensibility, but they are only as reliable as the clocks that timestamp them. If time can be contested, then investigations become slower and less conclusive, and attribution becomes harder. In regulated industries, this can translate into higher operational risk and liability: it is difficult to prove due diligence or reconstruct events when the chronology itself is questionable. Strategically, organisations need to recognise that resilience is not just about preventing compromise, but about preserving the integrity of the narratives used to understand and respond to compromise.

There is also a strategic capability gap in the market. Many security tools and managed services are not designed to reason about time integrity as a primary signal. They may alert on loss of synchronisation, but they rarely model adversarial clock skew, cross-domain time disagreement, or slow drift within tolerance. Similarly, many engineering organisations lack clear ownership for time: it falls between infrastructure teams, platform teams, security architecture and OT engineering. This “ownership gap” leads to predictable failures where systems degrade into permissive modes to maintain availability. Addressing time as an attack surface therefore requires new coordination approaches: joint requirements between security and reliability engineering, explicit time-integrity controls, and testing regimes that simulate adversarial timing faults rather than only random drift.

Finally, recognising time as an attack surface has forward-looking implications for technology strategy. As organisations adopt more distributed architectures, more cloud-native control planes and more autonomous systems, the dependency on synchronised, trustworthy time increases rather than decreases. At the same time, cost and complexity pressures push designs toward commodity oscillators, virtualised time sources, and network-mediated synchronisation. The strategic response is to shift from an accuracy mindset to an integrity mindset: not “how precise is our time,” but “how robust are our systems when time is wrong, inconsistent, or manipulated.” Organisations that

make this shift will be better positioned to maintain security and safety in environments where the most damaging attacks exploit assumptions that were never made explicit.

13. Conclusion

Time is one of the most deeply embedded assumptions in modern computing and cyber-physical systems. It underpins cryptography, authentication, distributed coordination, industrial control, and safety mechanisms, yet it is rarely treated with the same care as other foundational security primitives. Throughout this paper, we have shown that clock drift, synchronisation failures, and deliberate manipulation of oscillators can quietly undermine systems that are otherwise well designed, well implemented, and cryptographically sound. In many cases, these failures do not announce themselves as attacks, but instead masquerade as benign faults, operational quirks, or unavoidable complexity.

The central lesson is that time should not be treated as a background constant, but as a dependency that can be wrong, inconsistent, or adversarially influenced. When systems assume that clocks are approximately correct, attackers gain an opportunity to bypass controls without breaking software, stealing credentials, or exploiting vulnerabilities in the conventional sense. Expired tokens may become valid again, safety interlocks may fail to trip, logs may lose their evidentiary value, and distributed systems may lose their ability to reason reliably about order and causality. These are not edge cases, they are systemic risks created by implicit trust in time.

Addressing this challenge does not require perfect clocks or absolute precision. Instead, it requires a change in perspective. Designers and defenders must ask how systems behave when time is uncertain, contested, or actively manipulated. Security-critical logic should constrain the influence of time, cross-check it where possible, and avoid fail-open behaviour when timing assumptions are violated. Detection mechanisms must treat anomalous clock behaviour as a potential security signal, not merely an infrastructure concern. In industrial and safety-critical environments, timing integrity must be considered explicitly in hazard analysis and assurance, alongside more familiar cyber and physical threats.

As systems become more distributed, more virtualised, and more tightly coupled to the physical world, the strategic importance of time will only increase. Low-level assumptions that once held in isolated environments no longer do, and attackers are increasingly adept at operating below the application layer where these assumptions reside. Treating time as a first-class security primitive, designed, monitored, tested, and defended with intent, is therefore no longer optional. It is a prerequisite for building resilient systems when the clock itself can no longer be trusted.

14. References

1. https://www.usenix.org/system/files/sec23fall-prepub-520-kwon.pdf
2. https://clickcontrol.com/cyber-attack/silent-mfa-killer-microsofts-authquake-bug-let-hackers-bypass-security-undetected/
3. https://www.faa.gov/about/office_org/headquarters_offices/avs/offices/afx/afs/afs400/afs410/GNSS/GPS_GNSS_Interference_Resource_Guide.pdf
4. https://dfrws.org/wp-content/uploads/2024/07/Was-the-clock-correct-Exploring-timestamp-interp_2024_Forensic-Science-Inte.pdf
5. https://ww1.microchip.com/downloads/en/Appnotes/Atmel-42251-RTC-Calibration-and-Compensation_AP-Note_AT03155.pdf
6. https://www.ntp.org/reflib/enhancements/suggestions-protocol-and-security-enhancements-ntp/
7. https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/security-policy-settings/maximum-tolerance-for-computer-clock-synchronization
8. https://www.usenix.org/system/files/usenixsecurity25-liu-jianshuo.pdf
9. https://www.ndss-symposium.org/wp-content/uploads/2017/09/attacking-network-time-protocol.pdf
10. https://runebook.dev/en/docs/apache_http_server/mod/mod_ssl/sslocspresponsetimeskew
11. https://hackread.com/authquake-flaw-mfa-bypass-azure-office-365-accounts/
12. https://en.wikipedia.org/wiki/IEC_61508
13. https://tf.nist.gov/general/pdf/92.pdf

15. About Us

Andy Davis is Global Research Director at NCC Group. With 30+ years' experience across UK Government, consultancy and professional services, he leads the Group's global research programme, setting strategy, ensuring technical excellence and governing research output. Previously Chief Research Officer at IRM and Head of Security Research at KPMG, Andy joined NCC Group in 2010 and later built its global Transport Practice. He is known for turning complex cyber risk into clear, actionable guidance for engineers and boards.

NCC Group

NCC Group is a people-powered, tech-enabled global cyber resilience and software escrow business.

Driven by a collective purpose to create a more secure digital future, over 2,000 colleagues across Europe, North America, and Asia Pacific harness their collective insight, intelligence, and innovation to deliver cyber resilience to clients across the public and private sector.

With decades of experience and a rich heritage, NCC Group is committed to developing sustainable solutions that continue to meet clients' current and future cyber security challenges.

Follow NCC Group on LinkedIn and at https://www.nccgroup.com/