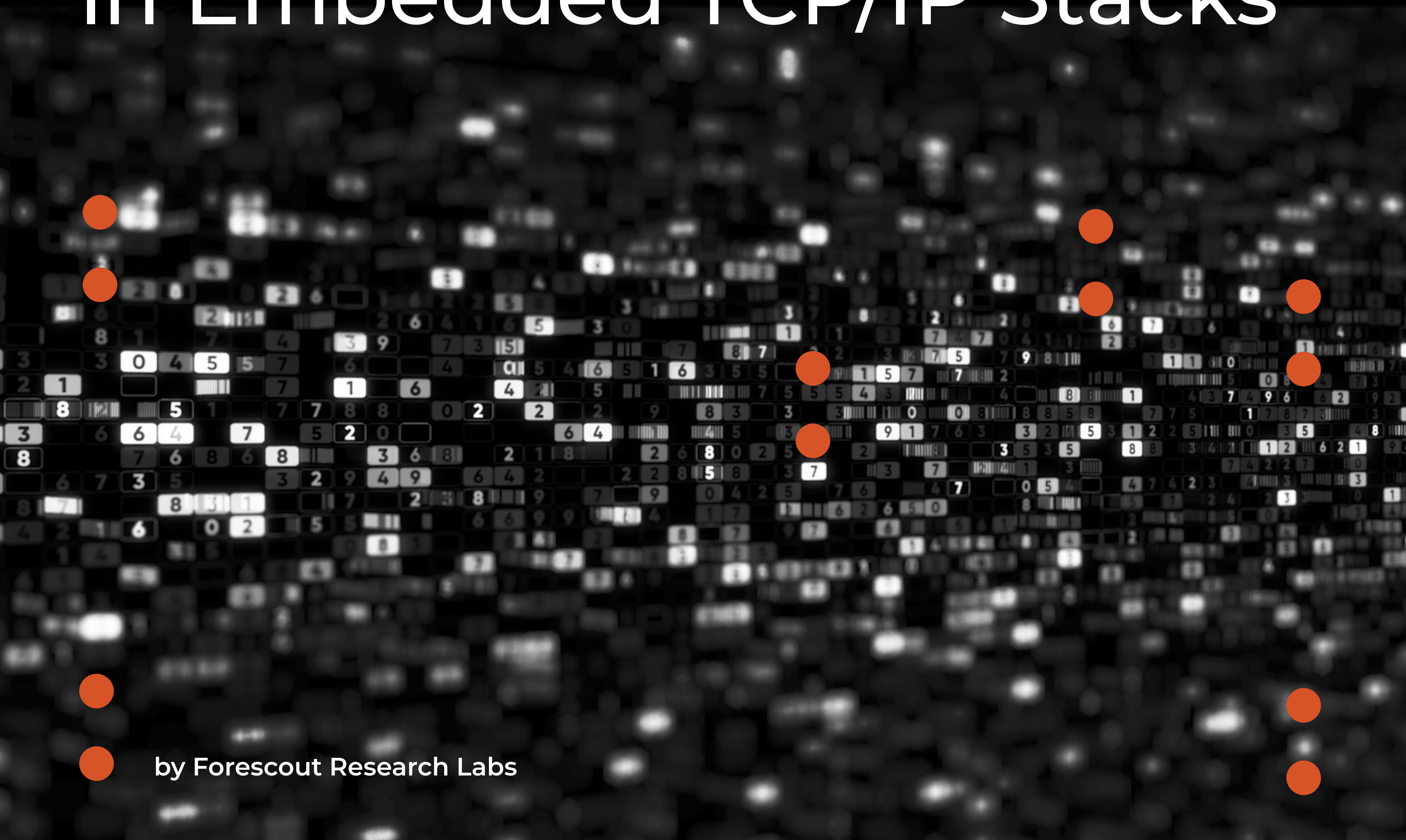
FORESCOUT
Active Defense for the Enterprise of Things™

NUMBER:JACK

Weak ISN Generation in Embedded TCP/IP Stacks



by Forescout Research Labs

Contents

Executive summary 3
1. Introduction 3
2. New vulnerabilities found 3
3. Impact 5
4. Recommendations for network operators 5
5. Technical dive-in 6
5.1. TCP connections and ISN generation 6
5.2. ISN generation vulnerability 7
5.3. Historical vulnerabilities 8
5.3. New vulnerabilities and non-vulnerable stacks 8
6. Conclusion 9
References 10

Executive summary

- In the second study of Project Memoria, Forescout Research Labs discloses NUMBER:JACK, a set of 9 new vulnerabilities affecting embedded TCP/IP stacks.
- The vulnerabilities are all related to the same problem: weak Initial Sequence Number (ISN) generation, which can be used to hijack or spoof TCP connections. Ultimately, attackers may be able to leverage those vulnerabilities to close ongoing connections, causing limited denials of service, to inject malicious data on a device or to bypass authentication.
- Although NUMBER:JACK vulnerabilities are not as critical as those of AMNESIA:33, they are even more prevalent, affecting 9 of 11 stacks analyzed. This study shows one more instance of historical vulnerabilities considered eradicated in the IT world affecting large numbers of IoT and OT devices.
- Recommended mitigations include using encrypted connections (for instance with IPsec), limiting the network exposure of critical vulnerable devices, and patching whenever device vendors release advisories.

1. Introduction

In 2020 Forescout Research Labs started *Project Memoria*, an initiative that seeks to provide the cybersecurity community with the largest study on the security of TCP/IP stacks. The first outcome of the project was AMNESIA:33 – a set of 33 vulnerabilities affecting 4 open source TCP/IP stacks.

The AMNESIA:33 technical report discussed how implementation flaws that have been well-known for decades have resurfaced as vulnerabilities that affect millions of IoT, OT and IT devices. While AMNESIA:33 was primarily focused on memory corruption bugs, NUMBER:JACK focuses on a fundamental aspect of TCP communication: Initial Sequence Number (ISN) generation.

ISNs ensure that every TCP connection between two devices is unique and that there are no collisions, preventing third parties from interfering with an ongoing connection. To guarantee these properties, ISNs must be randomly generated so that an attacker cannot guess an ISN and hijack an ongoing connection or spoof a new one. As discussed below, in many of the TCP/IP stacks that Forescout analyzed, ISNs are improperly generated, thereby leaving TCP connections of a device open to attacks.

2. New vulnerabilities found

In this second study of *Project Memoria*, Forescout researchers analyzed 11 TCP/IP stacks: 7 are the open source embedded TCP/IP stacks analyzed in AMNESIA:33 (uIP, FNET, picoTCP, Nut/Net, lwIP, cycloneTCP and uC/TCP-IP), while the remaining 4 are Microchip's MPLAB Net, Texas Instruments' NDKTCPIP, ARM's Nanostack and Siemens' Nucleus NET. Forescout found vulnerabilities in 9 of the 11 analyzed stacks. Table 1 details our findings.

Table 1 – The summaries of new vulnerabilities found in this research

CVE ID	CVSSv3 Score	TCP/IP Stack analyzed	Description	Fix
CVE-2020-27213	7.5	Nut/Net 5.1	ISN generator relies on a highly predictable source (system timer) and has constant increments.	Patch in progress.
CVE-2020-27630	7.5	uC/TCP-IP 3.6.0	ISN generator relies on LCG, which is reversible from observed output streams. The algorithm is seeded with publicly recoverable information (i.e., system timer count).	uC/TCP-IP is no longer supported. Patched in the latest version of Micrium OS (successor project).
CVE-2020-27631	7.5	CycloneTCP 1.9.6	ISN generator relies on LCG, which is reversible from observed output streams. The algorithm is initially seeded with a publicly observable CRC value.	Patched in version 2.0.0.
CVE-2020-27632	7.5	NDKTCPIP 2.25	ISN generator is initialized with a constant value and has constant increments.	Patched in version 7.02 of Processor SDK.
CVE-2020-27633	7.5	FNET 4.6.3	ISN generator is initialized with a constant value and has constant increments.	Documentation updated to warn users and recommend implement-ing their own PRNG.
CVE-2020-27634	7.5	uIP 1.0 Contiki-OS 3.0 Contiki-NG 4.5	ISN generator is initialized with a constant value and has constant increments.	No response from maintainers.
CVE-2020-27635	7.5	PicoTCP 1.7.0 PicoTCP-NG	ISN generator relies on LCG, which is reversible from observed output streams. The algorithm is seeded with publicly recoverable information (i.e., system timer count).	Version 2.1 removes the default (vulnerable) implementation and recommends users implement their own PRNG.
CVE-2020-27636	7.5	MPLAB Net 3.6.1	ISN generator relies on LCG, which is reversible from observed output streams. The algorithm is seeded with a static value.	Patched in version 3.6.4.
CVE-2020-28388	6.5	Nucleus NET 4.3	ISN generator relies on a combination of values that can be inferred from a network capture (MAC address of an endpoint and a value derived from the system clock).	Patched in Nucleus NET 5.2 and Nucleus ReadyStart v2012.12.

These vulnerabilities were discovered and disclosed to the vendors and maintainers of the affected TCP/IP stacks in October 2020. While most vendors have already issued patches and/or mitigation recommendations to their users (shown in the Fix column of Table 1), the developers of *Nut/Net* are still working on a solution, and Forescout has not received a response from the *uIP* developers.

The vulnerabilities found (except CVE-2020-28388) have a common CVSSv3 score and vector of 7.5 and AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N, respectively. Siemens has assigned to CVE-2020-28388 a score of 6.5 with the vector AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:L. However, the actual severity on a particular device and TCP connection may vary depending on, for example, the use of encrypted sessions and the sensitivity of data exchanged. The impact of these vulnerabilities is discussed in the next section, and comments about the stacks that were not found vulnerable (*lwIP and Nanostack*) are in Section 5.4.

3. Impact

There are two basic ways to exploit a weak TCP ISN.

1. By predicting the ISN of an existing TCP connection, attackers can close it, thus achieving a Denial-of-Service. Or, they can hijack it, thus injecting data into a session. Data can be injected on sensitive unencrypted traffic, such as Telnet sessions (to inject commands), FTP file downloads (to serve malware) or HTTP responses (to direct the victim to a malicious page).

2. By targeting new TCP connections, attackers may successfully complete a three-way handshake and spoof network packets intended for the victim endpoint or bypass address-based authentication and access control [1] [2] [3].

Although this type of vulnerability has been used historically to break into general-purpose computers (notoriously by Kevin Mitnick, which led it to be known as the "Mitnick attack"), the stacks Forescout researched are primarily used in *embedded* devices.

The popularity and some use cases of the vulnerable stacks *uIP, FNET, picoTCP* and *Nut/Net* were discussed in the AMNESIA:33 report. That report showed that these stacks are used by millions of devices of several different types, from IT file servers to IoT embedded components.

As for the new vulnerable stacks, Forescout believes that *CycloneTCP, uC/TCP-IP, NDKTCPIP, MPLAB Net* and *Nucleus NET* are no less popular than the previous ones. For example, according to the 2019 Embedded Markets Study [4], the Texas Instruments RTOS (which may be used with NDKTCPIP) is used by 6% of embedded projects, while Micrium's uC/OS-II or uC/OS-III (which may be used with uC/TCP-IP) are used by 7%. **The website of Nucleus RTOS mentions that it is deployed in more than 3 billion devices**. In all three cases, assuming a sizeable chunk of these projects utilizes TCP/IP connectivity, this certainly translates into other millions of devices running this software.

In this research, Forescout has not tried to identify affected devices or device manufacturers. Still, there are several notable public use cases of some of the stacks, such as medical devices, wind turbine monitoring systems, remote terminal units (RTUs) and IT storage systems.

4. Recommendations for network operators

As is always the case with vulnerabilities affecting TCP/IP stacks (such as AMNESIA:33 and Ripple20), identifying and patching devices running the vulnerable stacks is at the same time the best mitigation and the most challenging one. This process is challenging because it is often unknown which devices run a particular stack and because embedded devices are notoriously difficult to manage and update, often being part of mission-critical infrastructure.

For the reasons above, we recommend the following mitigation strategy.

- **Discover and inventory devices that run a vulnerable TCP/IP stack**. Although it is challenging to identify the TCP/IP stack running on a device, there are tools to help. Forescout Research Labs has released an open source script that uses active fingerprinting to detect which stack a target device is running. The script is updated constantly with new signatures. Besides that, Nmap allows the collection of ISN metrics and performs statistical analyses to understand whether a target device suffers from weak ISN generation.
- **Patch when possible**. Monitor progressive patches released by affected device vendors and devise a remediation plan for your vulnerable asset inventory balancing business risk and business continuity requirements.
- **Segment to mitigate risk**. For vulnerable IoT and OT devices, use segmentation to minimize their network exposure and the likelihood of compromise without impacting mission-critical functions or business operations. Segmentation and zoning also limit the blast radius and business impact if a vulnerable device becomes compromised.
- **Deploy IPsec**. End-to-end cryptographic solutions built on top of the Network layer (IPsec) do not require any modifications to a TCP/IP stack in use while allowing to defend against TCP spoofing and connection reset attacks [1]. Unfortunately, this comes at the cost of network bandwidth.

Monitoring the network to detect attempts to exploit weak ISN generation vulnerabilities is impractical. These vulnerabilities are related to cryptographic weaknesses, so there is no obviously malicious packet to exploit them (details of what an exploit looks like are shown in the next section). Therefore, Forescout recommends that network operators focus on the preventive measures described above.

5. Technical dive-in

5.1. TCP connections and ISN generation

TCP is a connection-oriented networking protocol that allows two endpoints to exchange data [5]. TCP is one of the protocols used in the TCP/IP Transport Layer, and it aims for ordered and error-checked delivery of data between network endpoints.

The protocol relies on sequence numbers that are transmitted with every packet over a TCP connection. Since every transmitted byte is sequenced, each of them can be accounted for during reception. A TCP connection is uniquely defined by a pair of *network sockets* – a combination of a network address and a network port. Since there is a finite number of ports in an endpoint, new connections may reuse the sockets utilized by already-terminated connections, which becomes problematic if connections are open and closed in quick succession. The *initial sequence number* (ISN) mechanism [5] was introduced to prevent connection collisions when sockets are reused.

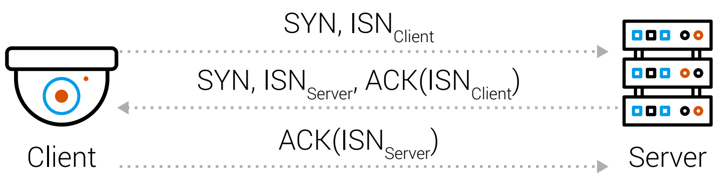


Figure 1 – TCP three-way handshake

An *ISN* generator is used at each endpoint to select a 32-bit sequence number that is supposed to be *unique* for every connection so that both endpoints can *synchronize* by exchanging their ISNs. The ISN synchronization occurs at the beginning of a connection with the *three-way handshake* mechanism [5] (illustrated in Figure 1): a *Client* sends a SYN packet with its unique $\text{ISN}_{\text{Client}}$, the *Server* acknowledges its reception and provides its own $\text{ISN}_{\text{Server}}$ within a SYN-ACK packet and the Client acknowledges the reception of $\text{ISN}_{\text{Server}}$ with an ACK packet. These three steps establish a TCP connection, allowing endpoints to exchange data.

5.2. ISN generation vulnerability

The original TCP design document [5] mentions that the 32-bit counter in an ISN generator should be incremented every 4 microseconds with the intention of being random, thus preventing accidental connection collisions.

However, this does not prevent attackers from deliberately forcing collisions to their advantage. As early as 1985, Morris [6] showed that ISNs can be easily guessed by attackers (because the ISN generator increments are constant), allowing them to disrupt or hijack TCP connections. TCP connection hijacking facilitates the execution of other kinds of attacks, such as unauthenticated access and Denial-of-Service, but this depends on the use of IP-based authentication protocols [1].

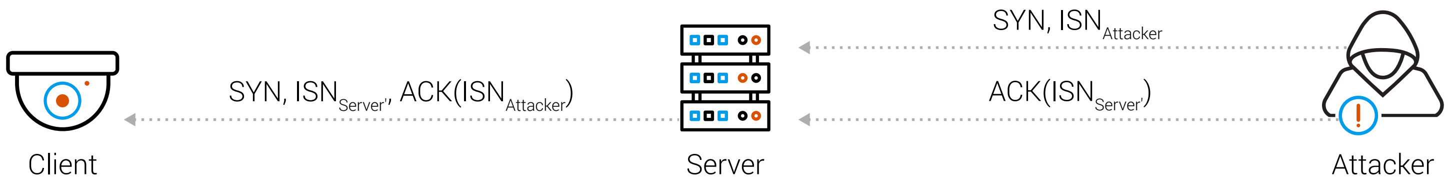


Figure 2 – ISN attack illustration

This attack [6] is illustrated in Figure 2. *Attacker* impersonates a legitimate *Client* (see [2]) and opens a connection to a *Server* (with a deterministic ISN generator). *Server*, thinking that the SYN packet is coming from *Client*, sends a SYN-ACK packet to *Client*, which includes $\text{ISN}_{\text{Server}}$. *Attacker* cannot see the SYN-ACK packet sent to *Client*. However, if *Attacker* can guess the value of $\text{ISN}_{\text{Server}}$, they can still send an ACK packet back to *Server* with the guessed $\text{ISN}_{\text{Server}}$ and complete the three-way handshake. In this way, the *Server* will think it has a legitimate connection with *Client*, when in fact it will be communicating with *Attacker*.

While the best possible solution to prevent this issue is strong cryptographic authentication [1, 2, 3], it may not be feasible for all networking scenarios. Therefore, RFC1948 [2] (updated by RFC6528 [3] after 16 years of its initial release) has proposed to compute the ISN as follows:

$$ISN = M + F\,(localip,\ localport,\ remoteip,\ remoteport,\ secretkey)$$

where *M* is the 4 microsecond timer, and *F()* is a pseudo-random function of the connection id (defined by the tuple *<localip, localport, remoteip, remoteport>*) and some secret data *secretkey*. RFC6528 [3] requires that neither *F()* nor *secretkey* are computable from the outside, otherwise attackers can still guess arbitrary ISNs by observing enough ISNs from other connections. **It is important to note that the aforementioned attack is possible because of a flaw in the original TCP design** (RFC0793 [5])**, and it is the responsibility of the developers who implement a TCP/IP stack for their device/platform to ensure that a proper protection mechanism is put in place.**

5.3. Historical vulnerabilities

The vulnerability described above was known as early as 1985 [6], but the first exploitation attempts were demonstrated only about 10 years later [7]. The first publicly known vulnerability reports (as found in the National Vulnerability Database) followed only from 1999 until 2001, affecting several individual operating systems (namely *Windows NT 4.0, FreeBSD, Cisco IOS, WindowsCE and Livingston ComOS*). It is difficult to distill the real root cause from the vulnerability descriptions, although the most likely root cause was non-compliance with RFC1948 [2].

Later in 2001, the CERT released a large vulnerability advisory [1] warning that even if RFC1948 [2] is followed, the pseudo-random number generator (PRNG) used in the ISN generator may rely on a known weak algorithm, and it would be practical for attackers to predict its outputs, as if constant increments to the ISN number are being used. The corresponding vulnerability (CVE-2001-0328) has been reported to affect many vendors and projects: *Cisco, Sun, IBM, HP, SGI, Fujitsu, FreeBSD, OpenBSD and Linux*. The root cause was that all the affected parties relied on the same weak PRNG algorithm.

These publications surmise that there are essentially **two possible root causes of weak ISN generation vulnerability**: (1) the RFC documents detailing the countermeasures are being ignored; or (2) a known weak or flawed PRNG is used when attempting to randomize the ISN generator output.

From 2001 until 2019, there were 15 more weak ISN generation vulnerabilities mostly reported for individual devices, such as network switches, gateways and smartphones. The vulnerability descriptions suggest that some of the affected vendors could be non-compliant with the RFCs (i.e., no PRNG has been used for ISN generation whatsoever), which is very surprising given that RFC1498 has been around for at least 16 years. While this is still only Forescout's hypothesis (based on vulnerability descriptions alone), next section shows that this still happens in 2021.

Out of these historical vulnerabilities, it is worth to single out CVE-2014-7284 found in the Linux Kernel. This vulnerability is more foundational than the rest. The PRNG seeding mechanism used by the ISN generator (and potentially other consumer APIs outside the Linux Kernel) was found to be flawed. In 2020, Klein [8] released a report arguing that the PRNG in the Linux Kernel is weak and can be used to facilitate various cross-layer attacks against its TCP/IP protocol suite, such as DNS cache poisoning and UDP port prediction. While the author does not mention the ISN flaw, it might be among possible attacks.

5.4. New vulnerabilities and non-vulnerable stacks

As shown in Table 1, the root causes for the newly found vulnerabilities are conceptually very similar to the root causes of the historical vulnerabilities discussed above. For example, **5 of the stacks** seem to ignore RFC1948 [2] /RFC6528 [3], as they **did not use any kind of PRNG to generate ISN values**. At best, *Nut/Net* generated ISNs based on the state of a system timer. Given that ISN is only 32 bits wide, and attackers may have the knowledge of the system uptime or be able to observe previous ISN values as well as other leakages of the system timer over the network, this value can be easily inferred. On the other hand, *uIP (also Contiki-OS and Contiki-NG), FNET* and *NDKTCPIP* initialized ISNs with a constant value, which attackers may find by examining the corresponding source or byte code, or by looking at a short network capture.

The remaining **4 issues** listed in Table 1 were due to the **usage of a weak PRNG algorithm**. Moreover, the PRNG **algorithm seeding was not implemented properly in all cases**, which would help the attackers to further reduce the search space (which is already quite limited) for predicting the output of the PRNG and guessing ISNs. *PicoTCP, PicoTCP-NG, CycloneTCP, uC/TCP-IP* and *MPLAB Net* all relied on a variation of a simple Linear Congruential Generator (LCG) algorithm for generating ISNs, which is known to be easily reversible from the observed output streams [9] [10] (e.g., network communications). Additionally, all these stacks used either a constant value or a dynamic value derived from a system clock for seeding the algorithm, which would defeat the purpose of using a PRNG.

Moreover, it is also challenging to account for all possible configurations in which an embedded TCP/IP stack will be used. Therefore, to ensure flexibility of a protocol suite, developers of TCP/IP stacks may implement naïve ISN generation solutions (e.g., using weak PRNG or no PRNG at all), expecting that system integrators and product developers will override this functionality according to their needs. The *lwIP* project handled this issue by introducing the possibility for system integrators and product developers to implement their own PRNG for ISN generation, as well as documented this feature. While Forescout cannot claim that *lwIP* is vulnerable to the weak ISN generation vulnerability per se, Forescout believes this kind of solution might still introduce vulnerabilities if the system integrators and product developers are not carefully reading the documentation. However, if a project/vendor provides no documentation that warns users about this, such implementations **must** be considered vulnerable, which was the case for the vulnerability Forescout reported to the PicoTCP-NG project maintainers.

Finally, the developers of *Nanostack* have implemented the ISN generation mechanism in accordance with FC6528 [3], which is the only stack (out of the 11 that Forescout analyzed) that implements this functionality properly.

6. Conclusion

Forescout's main conclusions from the NUMBER:JACK research are:

- As initially discussed in AMNESIA:33, misinterpretation or mis-implementation of RFCs is a major cause of vulnerabilities in TCP/IP stacks. In the case of ISN generation, stacks completely ignoring RFC recommendations are often seen.
- Weak ISN generation is one more instance of historical vulnerabilities discovered and fixed in the IT world decades ago that today affects large numbers of IoT and OT devices.
- This type of vulnerability is much easier for researchers to discover than the memory corruption issues discussed in AMNESIA:33. Although they do not allow for remote code execution directly, attackers can quickly and easily analyze several TCP/IP stacks and find similar issues that can be part of a larger attack campaign in a heterogeneous network with many IoT/OT devices.
- Unfortunately, this type of vulnerability is also difficult to fix permanently because of the resource constraints of many embedded devices, and what is considered a secure PRNG today may be considered insecure in the future. Some stack developers opt to rely on system integrators to implement their own ISN generation, which is a fair decision, but which means not all devices using a patched stack will be secure automatically.
- This research again highlights the security challenges of the IoT world and why it is fundamental for network operators to employ cybersecurity tools that ensure visibility and control of networked devices, including granular classification to detect vulnerable components, as well as the possibility of segmenting and enforcing policies on the network.

References

[1] CERT Division, "2001 CERT Advisories," 2001. [Online]. Available: https://resources.sei.cmu.edu/asset_files/WhitePaper/2001_019_001_496192.pdf.

[2] S. Bellovin, "RFC1948: Defending Against Sequence Number Attacks," 1996.

[3] F. Gont and S. Bellovin, "RFC6528: Defending Against Sequence Number Attacks," 2012.

[4] Embedded.com, "2019 Embedded Markets Study," [Online]. Available: https://www.embedded.com/wp-content/uploads/2019/11/EETimes_Embedded_2019_Embedded_Markets_Study.pdf.

[5] J. Postel, "RFC0793: Transmission Control Protocol," 1981.

[6] R. Morris, "A Weakness in the 4.2BSD UNIX TCP/IP Software," CSTR 117, 1985.

[7] T. Shimomura, "Technical Details of the Attack Described by Markoff in NYT," http://www.gont.com.ar/docs/post-shimomura-usenet.txt, 1995.

[8] A. Klein, "Cross Layer Attacks and How to Use Them (DNS Cache Poisoning, Device Tracking and More)," https://arxiv.org/abs/2012.07432, 2020.

[9] G. Argyros and A. Kyayias, "I Forgot Your Password: Randomness Attack Against {PHP} Applications," in USENIX Security, 2012.

[10] P. Bouchareine, "BSD Weak Initial Sequence Number Vulnerability," Bugtraq (SecurityFocus), 2010. [Online]. Available: https://www.securityfocus.com/bid/1766/discuss. [Accessed 11 January 2021].