Guidelines for Testing of IoT Security Products

Anti-Malware Testing Standards Organization

`

Notice and Disclaimer of Liability Concerning the Use of AMTSO Documents

This document is published with the understanding that AMTSO members are supplying this information for general educational purposes only. No professional engineering or any other professional services or advice is being offered hereby. Therefore, you must use your own skill and judgment when reviewing this document and not solely rely on the information provided herein.

AMTSO believes that the information in this document is accurate as of the date of publication although it has not verified its accuracy or determined if there are any errors. Further, such information is subject to change without notice and AMTSO is under no obligation to provide any updates or corrections.

You understand and agree that this document is provided to you exclusively on an as-is basis without any representations or warranties of any kind whether express, implied or statutory. Without limiting the foregoing, AMTSO expressly disclaims all warranties of merchantability, non-infringement, continuous operation, completeness, quality, accuracy and fitness for a particular purpose.

In no event shall AMTSO be liable for any damages or losses of any kind (including, without limitation, any lost profits, lost data or business interruption) arising directly or indirectly out of any use of this document including, without limitation, any direct, indirect, special, incidental, consequential, exemplary and punitive damages regardless of whether any person or entity was advised of the possibility of such damages.

This document is protected by AMTSO's intellectual property rights and may be additionally protected by the intellectual property rights of others.

`

AMTSO Guidelines for Testing of IoT Security Products

Terminology

- "Security product", "Product", "DUT", "Device Under Test" are interchangeable.
- "Tester"
- "LAN", "LAN protected by Product", "Perimeter"
- "RCE" - remote code execution
- "C2", "C&C" - command & control node, part of the attacker infrastructure
- MUD – Manufacturer Usage Descriptions
- ML – Machine Learning
- LAN – Local Area Network
- FN – False Negative, "missed" malicious sample or scenario
- FP – False Positive, legitimate software or behavior wrongly classified as malicious

Scope

The scope of this document is to define the guidelines for testing of IoT security devices. In this context, IoT security devices are products that are:

1. Plugged into an existing network as an additional network device
2. Acting as a replacement for, or adding features to, a network device such as a router/modem
3. A combination of the above

General Principles of testing of IoT security products

1. All tests and benchmarks should focus on validating the additional value delivered to the user, not the technical implementation of specific features that provide functionality. For example, a device that protects the network using MUD techniques should not score differently than a device that uses ML, assuming their performance and behavior are the same.

`

Different attack vectors

In this document we will distinguish between the following attack vectors, which may be used separately or in combination:

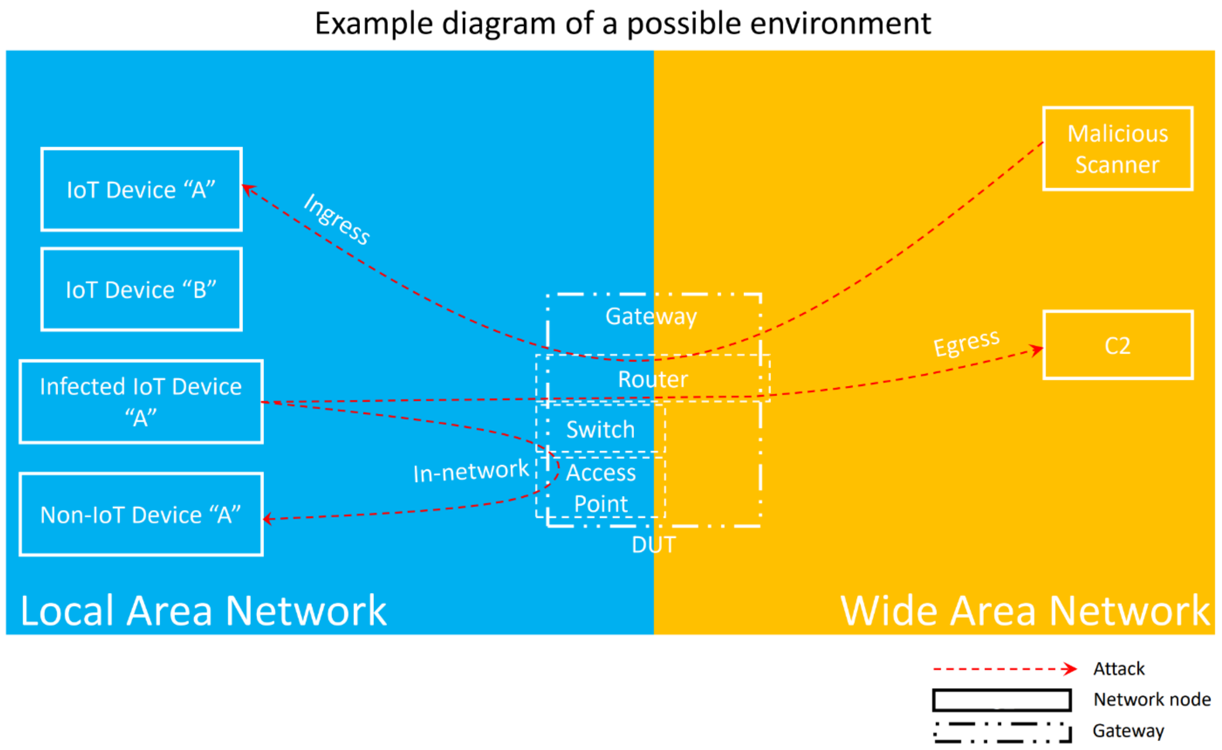
1. Ingress
    a. Ingress attacks are those that are initiated outside the LAN perimeter protected by the "DUT". These attacks could be:
        i. Active scanning performed by an attacker
        ii. Brute force attacks on the management or/and other services or interfaces accessible via the internet
        iii. Attacks that are actively performed by an attacker via the internet with the intent of gaining access to a device in the local network protected by "DUT".
    b. It is essential to understand that a product could protect the network using egress-focused shields despite the "ingress" nature of the attacks. The tester should account for such behavior.
        i. For example, in the IoT world, it is common for an attacker to perform a command injection, either directly or through other infected devices, which would classify such an attack as an "ingress". But because these commands are trying to download the actual payload from the internet, blocking the "egress" connection and not allowing the payload download could be a valid strategy for user protection.
    c. The tester must consider all attack stages and perform the scores based on the actual performance regarding the point above.
2. Egress
    a. Egress attacks are initiated by devices inside the LAN perimeter protected by the "DUT". These attacks could be originated from another type of attack, or a separate stage of a different kind of attack.
    b. Examples include "remote-code execution" attack prompts for downloading a script or a malicious binary onto the device under attack.
    c. Another common type of egress attack is scanning for other targets to attack in the future or using the infected device as part of a DDoS attack.
3. In-network
    a. In-network attacks are examples of "lateral movement" inside the perimeter protected by the security product, when an attack originates from one of the

`

devices inside the perimeter and the destination is another device in the same network.

b. It should be noted that in-network detection is extremely difficult and expensive for general SOHO/consumer applications.

Example diagram of a possible environment



Sample selection

Considering the volatility of the IoT Threat Landscape and the fact that the traditional approach of "agent-based" security is not possible, IoT Security Products have more constraints in terms of the available computing and storage resources. With that in mind it is important to remember that limited hardware resources present in the DUT should not be an excuse for FNs and/or FPs.

Tester must keep in mind that despite not blocking certain samples, there is a possibility that a product is not blocking those particular samples because blocking is no longer beneficial or necessary. For example, if a C2 is no longer available and no harm is done to

`

the network or device(s), this should not be considered an FN. On the other hand, if the sample is active and/or harmful it should be properly detected, and failure to do so by the DUT should be considered an FN.

Another critical point is to remember that IoT devices are usually not running Windows, so selecting malware explicitly targeting IoT Devices or generic Linux appliances is essential. Further filtering by CPU architecture that is commonly used by IoT devices like MIPS, MIPSEL, ARM is advised. On the other hand, some threats could target Windows-based IoT device or even some industrial ISC/DSC devices, so depending on the DUT, setup, and methodology, all of these variables should be taken into account. In an ideal but resource-intensive testing scenario, all samples would be categorized between industrial and non-industrial, with further separation into targeted operating systems and CPU architectures as well as severity scores.

Testers should keep in mind the scope of the test and select samples appropriately, taking into consideration the scope of protection of the product under test. So a product that is designed to protect against consumer IoT malware should not be primarily tested using Windows threats. Additional testing using a wider range of threat samples targeting a wider variety of environments may have value in some circumstances, but such testing should be considered supplemental to most tests.

Another suggestion is to separate the samples into different categories based on the target of the sample, that is:

a. Samples targeting the DUT/security product itself
b. Samples targeting the IoT devices
    i. Samples could be classified as active/inactive by execution with DUT security functionality turned off. Such classification can be performed before or after a test and the results should be interpreted accordingly. In a perfect world such classification would be performed both before and after the test, and any changes of state from (in)active to (in)active would be noted in the report.
c. Samples that are generic and are attacking devices of no specific category.
    i. Such samples are usually used for DoS as a part of a botnet and target multiple vulnerabilities and/or misconfigurations.

Finally, there is another level of complexity added by availability of C2/loaders in the context of sample selection and/or categorization.
Based on the C2 and/or loader server availability, samples could be considered:

a. Active/valid
    i. If C2 and/or loader is active

`

    ii. If C2 and/or loader is not active but some or all the malicious functionality is executed nevertheless. For example, if a Mirai sample has no active C2 but it still actively scans the internet for further spread and exploitation, the sample should be considered active.

b. Inactive
    i. If C2 and/or loader is not active and malicious functionality is not carried out.

Such classification can be performed before or after a test and the results should be interpreted accordingly. In a perfect world such classification would be performed both before and after the test, and any changes of state from (in)active to (in)active would be noted in the report.

It is important to keep in mind the restoration of all devices in the network to the desired state after each run and classification, and similarly important to note that some malware can modify a device's firmware.

As the tester is not bound by the knowledge of only one Security Provider, the best approach could be to gather and mix the samples from participating vendors and 3rd parties as well as available in-wild samples. If a mix of samples is used from participating vendors, equal distribution of samples from each vendor is preferable. In case this is not possible, disclosure of the distribution and correlation with the results is advisable. A suggested approach is to inform readers about sources of threats used in the test and the proportion of samples per each source.

In addition, separating or classifying samples based on severity scores could greatly improve the value of the test results.

Determination of "detection"

IoT devices can be difficult to monitor from the user/kernel level, especially without exploits and/or modifications to the Firmware or Hardware, it is important to point out what "detection" is.

Apparent and detectable definitions include prevention of the attack or notification of the user, or both. Some vendors do not provide "detection/prevention" notifications or provide an aggregated approximated value; it can be quite difficult to determine that actual detection of a specific attack happened.

One suggested approach is to use threats with admin consoles that can be controlled by the tester. As threat actors may not provide us with the necessary tools all the time, the next

`

best thing is to select IoT devices that will be "attacked" in such a way that it is possible to determine that an attack happened (or not happened) on the device under attack itself.

If that is not possible, network sniffing of the device "under attack" could be performed to determine the attack from the network point of view. The latter approach could be extended to all devices in the network protected by the DUT.

By essentially creating the attacker infrastructure in some cases, the tester body should avoid creating unnecessary risk to public infrastructure.

Preparation of the Environment

In a perfect world, all the tests and benchmarks would be executed in an ideal, controllable environment using real devices. In most circumstances some compromises must be made. If the tester decides against using real IoT devices in the testing suite, the tester should validate their approach by running their desired scenario(s) with the security functionality of the DUT disabled and checking the attack execution and success. An even better, but more demanding approach would be to repeat such validation steps before and after the benchmark/test itself, further validating the selected approach.

IoT Device Under Attack

When using real IoT devices in "under attack" simulation/benchmarking is not possible, the replacement should mimic the real device as closely as possible. A naive approach would be to use a cheap general-purpose computer, like Raspberry Pi, and mimic a real IoT device by installing relevant services (like RTSP server when mimicking an IoT IP camera) and changing the configuration (such as the MAC address, hostname, network configuration (MTU)). By doing so, fingerprints of network probing of the device (Nmap) and its running services should mimic the real device. Testers should reflect the usage of synthetic devices in their reports, and it is important to validate that samples will work on the synthetic device without a security solution present, as mentioned in the "Preparation of the Environment" section.

Advanced Threat Detection

Emulation of the actual attack can be challenging. When a vendor claims protection from advanced threats or even 0-days, a sufficiently skilled and resourced tester could potentially validate such claims.
By using publicly available source code of known IoT malware such as Mirai, combined with publicly available source code of IoT exploits, a tester could compile their own IoT botnet that has technically not been seen before. Following a common IoT botnet infrastructure design - server for payload hosting, server for C2, and a dropper/spreader - it should be possible to replicate such infrastructure in the testing environment.

`

It is important not to endanger public infrastructure, so a few precautions should be in place. If the botnet you are creating scans the internet for potential targets, it should only scan a predefined range sink-holed in the tester's infrastructure. Droppers/spreaders should also only use predefined ranges of IP addresses that are locally sink-holed or/and under control of the tester. The C2 and hosting should be hosted locally and/or not be accessible from the internet and/or protected in such a way that the tester's infrastructure is not abused.

Testing of specific security functionality

Disclaimer

As attacks on IoT devices usually have multiple stages, it is possible to test each stage individually instead of going through the whole attack at the same time. In whole-product testing, it is important to balance tests that are "stages" and complete end-to-end scenarios, as well as to reflect such choices in the methodology documentation.

Reconnaissance prevention

One of the first steps of almost any attack is recon, and in the IoT world, such attacks usually take the form of scanning for open ports and/or services. This could be prevented by many features in the security product, like advanced ingress block listing, proactive scanning, and warning the user, or even automatic reconfiguration of offending devices.

It is advisable to have a few approaches to testing recon prevention, both with in-lab and in-the-wild conditions:

- In-lab tests could consist of proactively scanning the device from the WAN side, or even of spoofing the IPs of a known in-the-wild attack and scanning the network from the WAN side using those IPs. If that is not possible, replaying of scanning activity could be used, but it is important to do this in timely manner, validating the approach and conditions separately before and after the test itself.

- In-the-wild conditions are difficult to harmonize but having multiple attack targets behind the DUT and opening the WAN interface to the internet should do the trick. The problem of how a particular service can be made publicly accessible from the internet is complex, but common vectors are default port forwarding by the internet gateway, UPnP, or devices that require users to create port forwarding rules manually.

All test scenarios should be done in consideration and context of the security solution under test.

`

Initial access prevention

Initial access prevention benchmarks should focus on “in-the-wild” prevalent attack vectors targeting IoT devices specifically, be it a command injection, a vulnerability in a particular service, or state-of-the-art packet corruption that leads to the RCE.

Execution prevention

Execution prevention is the set of measures designed to prevent the attack stages once the attacker gains initial access. Such attacks can include the download of other malware stages, C2 communication, outgoing DDoS attack, etc.

Platform-agnostic threats testing

With many threats in the past switching platforms and attack vectors, as well as generic threats that target multiple architectures, there are a lot of threats that can be used against IoT as well as non-IoT devices. As a part of complex test with good sample coverage it is important to include such threats, but testers are encouraged to separate their results based on the platforms target as well as the potential severity.

Benchmarking of performance

As IoT security products usually operate on the network level, it is important to consider their impact on user experience, mainly in terms of added latency or/and reduced throughput. Network benchmarking is a complex topic with a lot of information and methodology already available, and therefore it is outside of the scope of the IoT Security Testing guidelines. However, there is some general advice testers should bear in mind:

Protocol differentiation

When benchmarking the performance of a product, it is important to consider the customer use case; for example, an entry-level security solution will have a different throughput expectation than an SMB one. It is also essential to consider the added latency on a per-protocol basis; for example, HTTPS/HTTP traffic could be critical for real-time video delivery but added latency of a few seconds for IMAP protocol is not generally very critical.

Baseline

It can be informative to have different tiers of benchmarking performance with varying security features enabled and disabled. If the DUT is an add-on device, the baseline is the throughput and latency with no DUT present. If the DUT is a replacement for an existing gateway, the baseline should be counted against the throughput of the device that the DUT would normally replace.

`

Credits

Vladislav Iliushin, VI Labs, ex-Avast
Ilya Naumov, Kaspersky
Andrey Kiryukhin, Kaspersky
Evgeny Vovk, Kaspersky
Armin Wasicek, Avast
Stefan Dumitrascu, SE Labs
John Hawes, AMTSO
Scott Jeffreys, AMTSO

____

This document was adopted by AMTSO on 2022-06-01