NORTH ATLANTIC TREATY
ORGANIZATION

SCIENCE AND TECHNOLOGY
ORGANIZATION

AC/323(IST-151)TP/1030

www.sto.nato.int

STO TECHNICAL REPORT **TR-IST-151**

Cyber Security Risk Assessment Process for Military Systems

(Processus d'évaluation des risques de cybersécurité
pour les systèmes militaires)

This document presents a cyber security risk assessment
process tailored to military systems.

Published October 2021

Distribution and Availability on Back Cover

NORTH ATLANTIC TREATY
ORGANIZATION

SCIENCE AND TECHNOLOGY
ORGANIZATION

AC/323(IST-151)TP/1030

www.sto.nato.int

STO TECHNICAL REPORT **TR-IST-151**

Cyber Security Risk Assessment Process for Military Systems

(Processus d'évaluation des risques de cybersécurité
pour les systèmes militaires)

This document presents a cyber security risk assessment
process tailored to military systems.

The NATO Science and Technology Organization

The content of this publication has been reproduced directly from material supplied by STO or the authors.

Published October 2021

Copyright © STO/NATO 2021
All Rights Reserved

ISBN 978-92-837-2350-9

Table of Contents

List of Figures

List of Tables

List of Acronyms

AA	Application Assessment
BIOS	Basic Input/Output System
CANBUS	Controller Area Network Bus
CIS	Communication Information Systems
CONOPS	CONcept of OPerationS (see Glossary)
CPU	Central Processing Unit
DMO	Direct Mode Operation
DQPSK	Differential Quadrature Phase Shift Keying
DRM	Digital Rights Management
DSP	Digital Signal Processor
ECU	Engine Control Units
EPROM	Erasable Programmable Read-Only Memory
ET	Exploratory Team
F	Fuzzing
FACT	Firmware Analysis and Compare Tool
FIPS	Federal Information Processing Standard
FPV	First Person View
GHz	GigaHertz
GIS	Geographic Information System
GPS	Global Positioning System
IDA	Interactive DisAssembler
IEC	International Electrotechnical Commission
IED	Improvised Explosive Device
IEEE	Institute of Electrical and Electronic Engineers
IEIA	Internal/External Interface Assessment
IMU	Inertia Measurement Unit
IO	Input/Output
ISO	International Organization for Standardization
IST	Information Systems Technology
IT	Information Technology
JTAG	Joint Test Action Group (developer of IEEE Standard 1149.1-1990)
LCD	Liquid Crystal Display
LNODCT	Low Noise Offset Direct Conversion Transmitter
MCU	Micro Control Unit
MHz	MegaHertz
NATO	North Atlantic Treaty Organization
NIST	National Institute of Standards and Technology

ODCT	Offset Direct Conversion Transmitter
OT	Operational Technology
OWASP	Open Web Application Security Project
PA	Power Amplifier
PC	Personal Computer
PID	Proportional Integral Derivative
PT	Platform Technology
PTT	Push To Talk
RC	Remote Control
RCMAP	Risk-Based Cyber Mission Assurance Process
RE	Reverse Engineering
ROM	Read-Only Memory
RPV	Remote Person View
RTG	Research Task Group
SA	System Analysis
SAID	System Architecture Identification
SCR	Source Code Review
SMB	Server Message Block
SP	Special Publication
SPA	Software Platform Assessment (SPA)
SWA	Software Assessment
TDA	Technical Documentation Analysis
TDMA	Time Division Multiple Access
TETRA	TErrestrial TRunked RAdio
TMO	Trunked Mode Operation
UART	Universal Asynchronous Receiver-Transmitter
UAV	Unmanned Aerial Vehicle
USB	Universal Serial Bus
WPIC	World Phone IC

Glossary

Architecture	Fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution [1]. [ISO/IEC/IEEE 2015, Section 4.5]
Asset	Something valuable belonging to a person or organization which is to be protected. (Adapted from Cambridge English dictionary [2]).
Attack	Any kind of malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself [3].
Black-Box Testing	A penetration test in which the tester has no prior information about the network being tested [4].
Capability	A military equipment that gives the ability or power to do something. (Adapted from Cambridge English dictionary [2]).
CONOPS	Abbreviation of Concept of Operations. The concept behind how something is operated.
Custom Software	Also known as bespoke software or tailor-made software, is the software specially developed for specific customer needs.
Cyber Security	Practices to protect networks and devices from attacks.
Function	The natural purpose of a system. A system can have multiple functions. (Adapted from Cambridge English dictionary [2]).
Fuzzing	A software testing technique that involves providing invalid or unexpected data to monitor the software’s reaction.
Grey-Box Testing	A combination of black-box testing and white-box testing [4].
Impact	The magnitude of harm from a threat event that can be expected to result from the consequences of unauthorized disclosure of information, unauthorized modification of information, unauthorized destruction of information, or loss of information or information system availability [3].
Likelihood	A weighted factor based on a subjective analysis of the probability that a given threat is capable of exploiting a given vulnerability or a set of vulnerabilities [3].
Mission	The coordinated military actions of a state, or a non-state, actor, in response to a developing situation. (Adapted from Cambridge English dictionary [2]).
Operations	A military activity within a mission that is planned to achieve something. (Adapted from Cambridge English dictionary [2]).

Operational Technology (OT)	Hardware and software that detects or causes a change through the direct monitoring and/or control of physical devices, processes and events [5]. [Gartner] In the military domain, operational technology generally refers to facility monitoring and control systems and their devices, such as power, fuel, water, safety and physical security control systems.
Platform Technology (PT)	Hardware and software included in all military platforms and weapon systems such as flight, mission, air traffic management, air defence, command and control and surveillance and navigation systems.
Risk	A measure of the extent to which an entity is threatened by a potential circumstance or event, and typically a function of: i) The adverse impacts that would arise if the circumstance or event occurs; and ii) The likelihood of occurrence [3].
Risk Assessment	The process of identifying, estimating, and prioritizing risks to organizational operations (including mission, functions, image, reputation), organizational assets, individuals, other organizations, and the Nation, resulting from the operation of an information system. Part of risk management, incorporates threat and vulnerability analyses, and considers mitigations provided by security controls planned or in place [3].
Scope	The range, the extent of the risk assessment to consider. (Adapted from Cambridge English dictionary [2]).
Software Platform	The whole set of software components which constitutes the runtime environment for the custom software (e.g., Operating Systems, Data Base Management Systems, Application Servers, Web Server, etc.).
System	A discrete set of information resources organized for the collection, processing, maintenance, use, sharing, dissemination, or disposition of information [3].
Threat	Any circumstance or event with the potential to adversely impact organizational operations and assets, individuals, other organizations, or the Nation through an information system via unauthorized access, destruction, disclosure, or modification of information, and/or denial-of-service [3].
Threat Event	Determined by organizations during risk assessments with a varying level of detail. Descriptions of threat events can be expressed in highly general terms (e.g., phishing, distributed denial-of-service), in more descriptive terms using tactics, techniques, and procedures, or in highly specific terms (e.g., the names of specific information systems, technologies, organizations, roles, or locations) [3].
Threat Scenario	A set of discrete threat events, attributed to a specific threat source or multiple threat sources, ordered in time, that results in adverse effects [3].]
Threat source	Characterized as: i) The intent and method targeted at the exploitation of a vulnerability; or ii) A situation and method that may accidentally exploit a vulnerability.

In general, types of threat sources include:

i) Hostile cyber or physical attacks;

ii) Human errors of omission or commission;

iii) Structural failures of organization-controlled resources (e.g., hardware, software, environmental controls); and

iv) Natural and man-made disasters, accidents, and failures beyond the control of the organization [3].

Vulnerability	A weakness in an information system, system security procedures, internal controls, or implementation that could be exploited by a threat source [3].
Vulnerability Assessment	Systematic examination of an information system or product to determine the adequacy of security measures, identify security deficiencies, provide data from which to predict the effectiveness of proposed security measures, and confirm the adequacy of such measures after implementation [3].
Vulnerability Discovery	The process of searching for and finding vulnerabilities.
Vulnerability Exploitation	Attack that takes advantage of vulnerabilities.
White-Box Testing	Test in which the tester has prior information on the component(s) being tested [4].

Acknowledgements

The chair of this NATO IST RTG activity would like to acknowledge the important contribution of his fellow team members in the creation of this cyber security risk assessment process. It was possible only because of your hard work.

The entire team of this NATO IST RTG would like to thank NATO CSO, the NATO IST Panel and each of their respective organizations for making this work possible. The team also wants to thank the reviewer of this document for the valuable input provided.

IST-151 Membership List

CHAIR

Mr. Frédéric PAINCHAUD
Defence R&D Canada – Valcartier
CANADA
Email: Frederic.Painchaud@drdc-rddc.gc.ca

MEMBERS

Mr. Feyyaz DOMBAYCI
ROKETSAN Inc.
TURKEY
Email: feyyaz.dombayci@roketsan.com.tr

Mr. Raphael ERNST
Fraunhofer FKIE
GERMANY
Email: raphael.ernst@fkie.fraunhofer.de

Mrs. Aylin HATİP İPEK
ROKETSAN A.S.
TURKEY
Email: aylin.hatip@gmail.com

Mr. Paolo MANFRE
Leonardo S.P.A.
ITALY
Email: paolo.manfre@leonardocompany.com

Ir. Judith VAN BRUGGEN – VAN PUTTEN
National Aerospace Laboratory (NLR)
NETHERLANDS
Email: putten@nlr.nl

Mr. Lorenzo ZAMBURRU
SELEX Galileo
ITALY
Email: lorenzo.zamburru@selex-es.com

PANEL/GROUP MENTOR

Ms. Francine DESHARNAIS
Defence R&D Canada – Valcartier
CANADA
Email: Francine.Desharnais@forces.gc.ca

Cyber Security Risk Assessment Process for Military Systems

(STO-TR-IST-151)

Executive Summary

Military platforms are more computerized, networked and processor-driven than ever. The consequence is an increased exposure to cyber attacks and thus, an amplified risk. However, the continuous and stable operation of these platforms is critical to the success of military missions and public safety.

Perfect cyber security does not exist. Cyber security must be continuously managed through iterative risk assessments. Many cyber security risk management frameworks and processes exist for traditional IT systems. However, this is far from being the case when it comes to military platforms and systems. This document presents a cyber security risk assessment process tailored to military systems. This process was developed by the team members of the NATO IST-151 Research Task Group (RTG) activity entitled "Cyber Security of Military Systems". The process can be applied to both traditional IT and firmware-based embedded systems, which are everywhere in military platforms and systems.

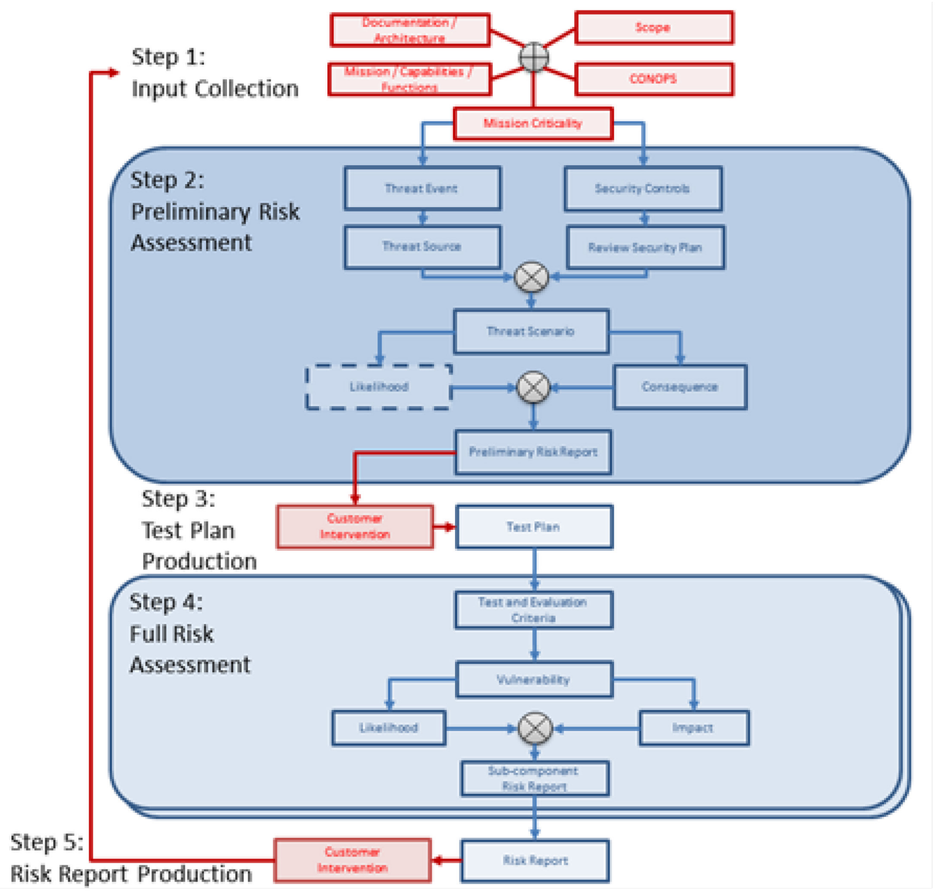


Figure i: The Five Main Steps of the Assessment Process.

Processus d’évaluation des risques de cybersécurité pour les systèmes militaires

(STO-TR-IST-151)

Synthèse

Jamais les plateformes militaires n’ont été plus informatisées, mises en réseau et alimentées par des processeurs. Il en résulte une exposition accrue aux cyberattaques, et ainsi, une amplification du risque. Or la continuité et la stabilité de fonctionnement de ces plateformes revêtent une importance critique pour le succès des missions militaires et pour la sécurité publique.

La cybersécurité n’est jamais parfaite : elle doit être gérée en permanence au moyen d’évaluations itératives du risque. Nombreux sont les cadres et processus de gestion des risques de cybersécurité régissant les systèmes TI traditionnels. Tel n’est pas le cas, en revanche, pour les plateformes et systèmes militaires. Ce document présente justement un processus d’évaluation des risques de cybersécurité adapté aux systèmes militaires. Le processus concerné a été élaboré par les membres de l’équipe de l’activité « Cybersécurité des systèmes militaires » du Groupe de recherche (RTG) IST-151 de l’OTAN. Il s’applique à la fois aux systèmes intégrés TI traditionnels et à ceux équipés d’un micrologiciel, qui sont omniprésents dans les plateformes et systèmes militaires.

CYBER SECURITY RISK ASSESSMENT PROCESS FOR MILITARY SYSTEMS

1.0 INTRODUCTION

This document is the final report of the NATO IST-151 Research Task Group (RTG) activity entitled "Cyber Security of Military Systems". This RTG focused on the study of cyber security risk assessment methodologies for military systems and platforms. The objectives of the RTG were the following:

- Collaboratively assess the cyber security of military systems with shared access among the RTG's member NATO nations;
- Share risk assessment methodologies and results among the RTG's member NATO nations; and
- Integrate assessment methodologies used by the RTG's member NATO nations into a coherent cyber security risk assessment methodology for NATO nations to benefit.

Military platforms are more computerized, networked and processor-driven than ever. They heavily use data buses like MIL-STD-1553A/B, CAN/MilCAN, RS-422/RS-485, AFDX and even plain Ethernet, and old standards for tactical communications like MIL-STD-188C and Link 16, to cite only a few. Moreover, captors, sensors, actuators and many embedded systems are additional unguarded potential inputs for aggression that extend the attack surface. The consequence is an increased exposure to cyber attacks and thus, an amplified risk. However, the continuous and stable operation of these platforms is critical to the success of military missions and public safety.

Military systems and platforms are targets of choice for cyber attacks not because of their prevalence, like consumer electronics, but because of their potential strategic impact. Once compromised, all sorts of short-term and long-term effects can be achieved, ranging from denying a capability to covertly reducing its effectiveness or efficiency, on demand. Therefore, military forces must address cyber security at all levels: strategic, while acquiring platforms and systems, operational, while planning military missions and tactical, while in operations.

NATO nations own a considerable number of military platforms and systems that may face cyber attacks. Therefore, NATO would benefit from leveraging current processes and methodologies to design more secure systems and assess current systems' cyber security.

This report presents a cyber security assessment methodology tailored to military systems and platforms, developed from the collaboration of the RTG team members and built on top of their experience and expertise. Processes already used by the team members were shared, analyzed, integrated and augmented to produce the process depicted in this report. The intended audience for this report is decision makers who are willing to assess and mitigate the cybersecurity risks of their military systems.

Section 2.0 presents the methodology used by the RTG team over its three years of existence to develop the process. Section 3.0 lays down some characteristics of the systems on which the process can be applied. Finally, Section 4.0 depicts the process itself while Chapter 5 concludes this report.

2.0 METHODOLOGY

The NATO RTG IST-151 "Cyber Security of Military Systems" stood up between Fall 2016 and Summer 2019. The group had either monthly or bimonthly teleconferences, depending on the needs. The group also had one to two face-to-face meetings per year, to make more concentrated progress when necessary.

Prior to the creation of the RTG, in 2014 – 2015, the NATO Exploratory Team (ET) IST-ET-078 prepared the ground for this RTG by defining a scope, exchanging some cyber security assessment methodologies and best practices and selecting candidate systems to assess. The ET was also great for bonding the team members.

Right from the beginning, the goal of the RTG was to produce a cyber security risk assessment process from the combination of the team's experience and expertise, that is, from how the team is already conducting risk assessments. To achieve this, a device needed to be selected so that everyone could perform an assessment its own way and share the results with the rest of the team. That way, similarities and differences in assessment methodologies could be observed, and a common process could be developed, combining strengths from all methodologies. After studying each other's methodologies, one methodology, called Risk-based Cyber Mission Assurance Process (RCMAP), developed by one of the team members, was agreed by all parties to be used as the basis of the process, as it was judged by all the most appropriate foundation. Elements of everyone's methodologies were then added. It is important to note that the entirety of RCMAP was not used. Rather, only its high level activities were taken as a starting point so elements of other methodologies could be added. On top of team members' methodologies, well-known existing methodologies, such as the ISO 27000 series, were also considered and elements were integrated.

After brainstorming, the team decided to select a drone as the common device to assess for the following reasons:

- It is easy to procure for everyone and is not too expensive;
- Drones are now used by military forces around the world and the particular drone selected has been used in some theaters for surveillance purposes so it is a military-relevant target;
- Drones are good examples of embedded systems that include Operational Technologies (OT), for which this process is tailored; and
- Working on a non-military drone makes the assessment work less sensitive and the sharing of results easier.

Thus, each organization within the team bought the drone and assessed it for cyber security. After sharing the methodologies, we realized they were varied: documentation review, field tests, reverse engineering, and automated firmware analysis were all used but not by all parties, depending on available resources and expertise. In other words, the team felt that the assessments were complementary. Similarly, results were also complementary: what you find during documentation review is totally different than what you find during reverse engineering, for instance. Thus, by having a more complete methodology, you also get more complete results.

Once the methodologies were shared, the group started to integrate them, using RCMAP's high level activities as a starting point. The result of this integration work is the process presented in this report.

3.0 APPLICABLE SYSTEMS

Nowadays, operational military systems are generally composed of several sub-systems and have multiple communication interfaces. Each of these sub-systems has its own hardware, firmware, software and external communication interfaces for interaction with other sub-systems.

The process detailed in this report can be applied to any military electronic systems and their sub-systems, thus any military systems embedded in military platforms. Examples of such military electronic systems are GPS equipment, radio terminals, cryptographic equipment and Unmanned Aerial Vehicles (UAV), to name only a few. Thus, both Platform and Operational Technologies (PT and OT) can be assessed with this process.

In addition, this process can be applied to traditional military computer applications in, for instance, command and control systems. Thus, traditional Information Technology (IT) systems can also be assessed.

Since there are no standardized sets of systems, this process does not provide standardized sets of checklists to comply with, depending on the system under study. This process rather helps one discover cyber security risks to mitigate in a given system or set of systems.

Two examples of applicable systems, a drone and a radio terminal, are detailed in Annex A and Annex B, respectively. These systems have several communication interfaces, chips with embedded software, complex hardware, are used by military forces, etc., and are thus perfect targets for cyber security assessments using the process detailed in this report.

4.0 ANALYSIS PROCESS

The main purpose of the process depicted in this section is to determine the cyber security risks of military systems. Those identified risks can then be mitigated using appropriate security controls and the overall security posture of the analyzed systems improved. Note that the process needs to be applied *repetitively* as the systems get updated or replaced and as the threats evolve. One can never assume that the job is done forever.

The process is generic enough to be applied to systems outside of the military world, but it has been specifically tailored to military systems, using inputs such as CONOPS and missions, capabilities and functions, which are more common in the military world than the civilian world.

Figure 1 shows that the process is composed of five main steps: Input Collection, Preliminary Risk Assessment, Test Plan Production, Full Risk Assessment and Risk Report Production. Each of these steps is detailed in the following subsections.

4.1 Step 1: Input Collection

Before analyzing the cyber security of a military system, it is key to gather all available documentation on the system first. That documentation will kickstart the understanding of the system, which drives the entire analysis process afterwards.

While having access to more documentation is usually better, not finding all documentation referenced below is not a showstopper. It would only mean that the risk assessments would be more time-consuming.

All types of documentation can help but the following is suggested.

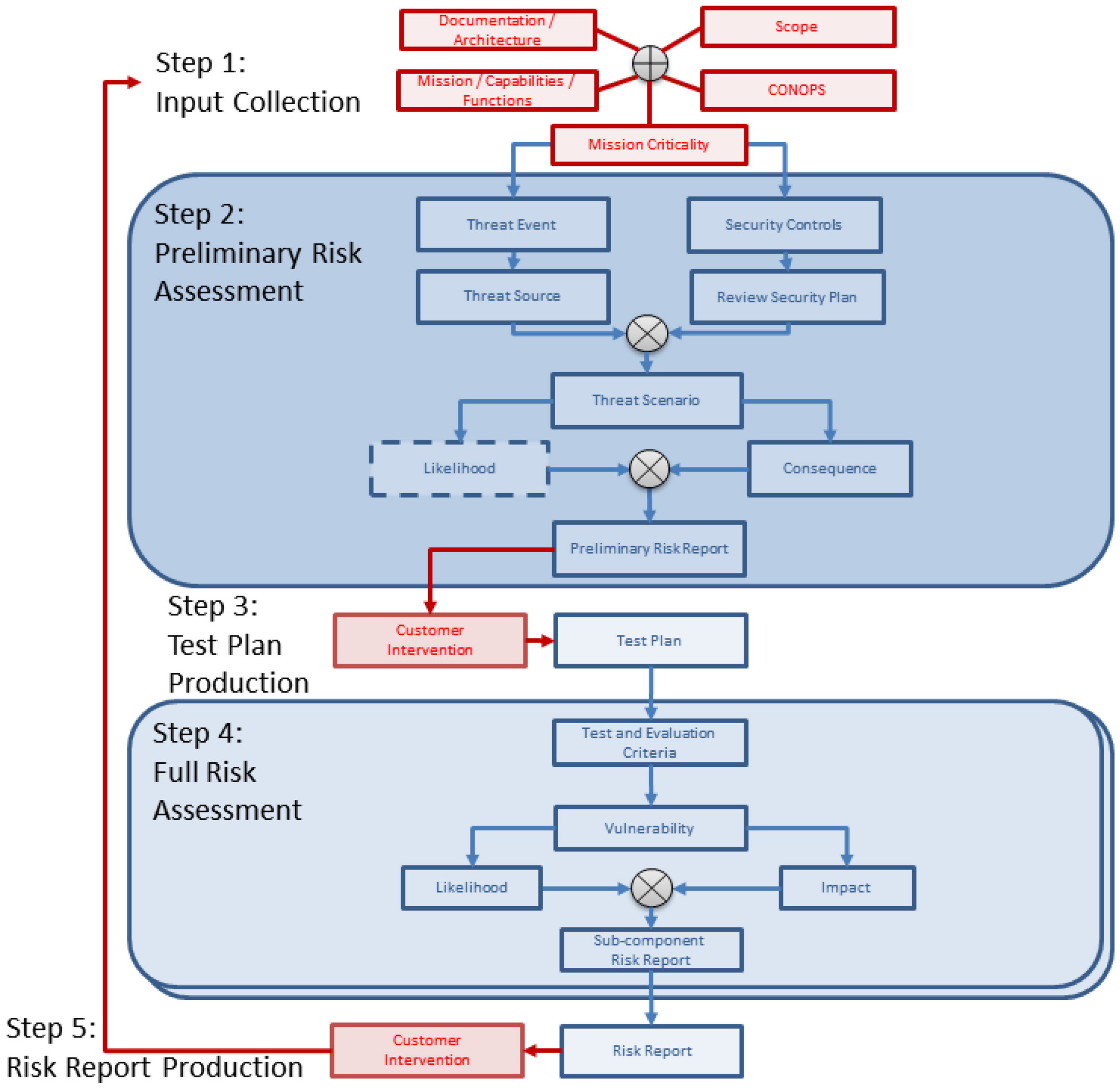


Figure 1: The Five Main Steps of the Assessment Process.

4.1.1 Missions, Capabilities and Functions

It is usually impossible resource-wise to assess the cyber security of a complex military system as a whole on the first go. Thus, the analysis of the sub-components of such a system must be prioritized. Such a prioritization must be based on criteria. One of the main criteria is the importance of each sub-component for the accomplishment of the missions that the system supports. In other words, the sub-components that are critical to assure the success of the missions are the ones that are going to be assessed first.

As shown in Figure 2, to evaluate the importance to the missions, it is first necessary to determine which missions are supported by the system and to describe them. The people most likely to know about these missions are the military operators of the system and their commanders. They must thus be interviewed at this point in the process.

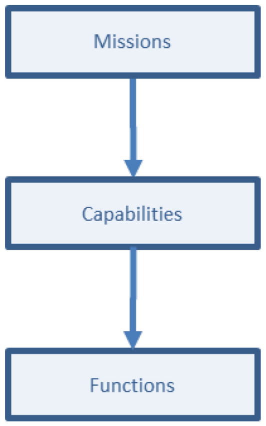


Figure 2: Relationships between Missions, Capabilities and Functions.

Oftentimes, specific missions fall under mission classes or categories. Those classes along with short descriptions can be included in this mission portrayal. The goal of describing the missions is mainly to provide a context later on to risk assessors. If they are less familiar with the military world, which is common, it will make them understand the context in which the system is used, which is key to assessing its cyber security. Table 1 gives examples of missions.

Table 1: Examples of Missions.

Support to help monitor fishing vessels and enforce fishing regulations in the Atlantic Ocean
Respond to forest fires, floods and natural disasters
Enforce maritime security and support counter-terrorism operations
Reinforce NATO’s collective defence

After describing the missions that the system supports, the military capabilities that support these missions should also be described. Capabilities are thus one level of granularity down under the missions. Operators using the system under study are most likely to know about the relevant capabilities. Capabilities are usually standard across the entire Forces of a country. They are also most likely similar across Forces of different countries. Refer to Table 2 for examples of military capabilities.

The third and last level to describe is the functions that the military system supports. This information becomes a bit more technical and is thus most likely known by engineers, either the ones who have designed or built the system, if accessible, or who are knowledgeable in the domain of the system. Table 3 shows examples of functions.

Table 2: Examples of Military Capabilities.

Establish and maintain deployable environment
Establish spaced based communications
Exploit multi-domain sense capabilities
Conduct ground surveillance
Conduct counter-IED operations

Table 3: Examples of Functions.

Communicate with the external world
Navigate
Fire weapon
Sense the environment

4.1.2 System Documentation

All technical information on the system is, of course, relevant to collect at the beginning of this process. For instance, system architecture, interface documents, design documents, deployment plans, testing plans, security plans, instruction manuals, user guides, data sheets, patents, white papers, source code (if accessible), etc., are all very relevant. Cyber security risk assessors will most likely benefit from reading those documents before and while doing their work.

4.1.3 CONOPS

In the military world, a Concept of Operations document, usually named the CONOPS, is generally produced before procuring any system. That document basically describes in what context the system is going to be used and how it is going to be used and deployed. That information can be key to risk assessors to make them think about cyber security threats to the system. It also gives them the appropriate terminology to use to produce reports that will talk to military stakeholders.

4.1.4 Mission Criticality Analysis

Once one has a list of the missions, capabilities and functions a system is taking part of, and has potentially also access to some documentation on the system, one can assess the criticality of the system's sub-components to the missions the system supports.

Sometimes, one is so experienced with the system and the missions it supports that it is possible to directly rate each of the sub-components with respect to its criticality to each mission. It is usual to rate such criticality on the scale presented in Table 4.

Table 4: Mission Criticality Scale.

Catastrophic	Loss of the sub-component causes a total failure of the mission. Objectives of the mission are not accomplished.
Critical	Loss of the sub-component causes a significant degradation of the mission. Only few mission objectives are met and with a significantly reduced effectiveness.
Marginal	Loss of the sub-component causes a limited degradation of the mission. Mission objectives are met, but with reduced effectiveness.
Negligible	Loss of the sub-component has little or no adverse impact on mission objectives.

However, and oftentimes, one cannot perform that criticality analysis from experience alone. Then, it helps to take each sub-component one by one and answer the following questions:

- What would be the impact on the missions if the confidentiality of the information processed by that sub-component be compromised?
- What would be the impact on the missions if the integrity of the information processed by that sub-component be compromised?
- What would be the impact on the missions if the availability of the information processed by that sub-component be compromised?

Each of these questions can be answered using the scale presented in Table 4 and then the maximum criticality of the three can be used as the overall criticality of the sub-component. At the end, the more critical sub-components are to be assessed before the less critical ones. Of course, the number of most critical sub-components to assess depends on the available resources.

4.1.4.1 Scope

It is very important to determine a preliminary scope for the cyber security risk assessment. At this stage, and especially when it is the very first security assessment for the system, the cyber security risks of the system's sub-components are not known, so the scope cannot be established with respect to that prioritization criteria. However, scope can be established based on the mission criticality analysis. Also, scope can be based on the stakeholders' interests (the ones asking for a cyber security risk assessment). For instance, the stakeholders can know that certain sub-components are more important to them, or they can have threat intelligence informing them of some adversaries' capability for which they would like to understand the impact on their system. The CONOPS can also be used to set a preliminary scope.

Sometimes, the entire system can be determined to be in scope, for instance when many or all sub-components are mission-critical, and then, preliminary risk assessment will narrow that scope based on preliminary risks prioritization and resource availability.

4.2 Step 2: Preliminary Risk Assessment

Mission criticality analysis along with stakeholders, cyber intelligence and system CONOPS all influence the scope of the cyber security risk assessment. The preliminary risk assessment uses this scope, and information on the system itself, to derive threat scenarios that may harm the system. A threat scenario is a set of threat events, associated with one or more threat sources, partially ordered in time.

4.2.1 Objectives

The objectives of a preliminary risk assessment are to identify the most critical sub-components inside the system that need further risk assessment (performed during step 4, Full Risk Assessment) and to guide that risk assessment with preliminary threat scenarios to investigate.

The process defined in this section is the same for each sub-component, and can be performed for each sub-component separately or in parallel, depending on resources availability.

The following items are considered:

- High level architecture of the system;
- Derivation of threat events and threat sources;
- Determination of existing security controls;
- Review the security plan (planned future security controls);
- Designing threat scenarios;
- Likelihood estimation for each threat scenario;
- Consequences for each threat scenario; and
- Definition of preliminary risk report.

4.2.2 High Level Architecture of the System

First, it is important to have a common view of the system to be assessed. This common view should functionally describe how the system operates, which sub-components (or main functions) are used by the system, how these sub-components cooperate, and what information is exchanged between them. The high level architecture is used to locate the most probable entry points for cyber attacks.

4.2.3 Threat Events and Threat Sources

A threat event is an event that reduces a system functionality or effectiveness and may occur as a result of the exploitation of one or more vulnerabilities. These threat events influence system operations and may result in operational and safety critical situations.

Threat events derive from two different groups:

- Unintentional threats such as floods, quakes, tornadoes, but also technical failures and human or organizational errors. Those threats fall outside the scope of this report.
- Intentional threats which are targeted and untargeted attacks from a variety of sources, such as criminal groups, hackers, disgruntled employees, foreign nations and terrorists.

For intentional threats, a threat source is required, who has the capacity and intention to initiate a threat event. Note that this threat source can use equipment used by a well-intentioned human in its threat scenario, such as the maintenance computer of a mechanic or the electronic flight bag of a pilot. Thus, while unintentional threats are out of scope, trusted people and their equipment are in scope.

When identifying threat sources of concern, it is important to determine the characteristics associated with those threat sources.

The mission context may already give some information of the most relevant threats, and are, as such, input to the preliminary risk assessment process. Together with the experience of operators, this will result in combinations of threat events and threat sources that may seriously impact the mission's success.

4.2.4 Determination of Existing Security Controls

It is important to take the time to study existing cyber security controls that may be present in the sub-component since those security controls could already mitigate some risks. These security controls can be technical measures such as the introduction of redundancy for specific functions but can also be procedural checks to be performed by the system operator, or verification tests that need to be part of the maintenance procedures. When deemed necessary, the testing of an existing security control could be mandated in the test plan for full risk assessment, to make sure that its mitigation effect is effective.

4.2.5 Review the Security Plan (Planned Future Security Controls)

It is also important to take the time to study any existing security plan since upcoming security controls could be planned to be added into or around the sub-component in the near future. These security controls would also mitigate some risks and would thus influence the preliminary risk assessment.

4.2.6 Designing Threat Scenarios

A threat event occurs when a threat source uses a specific threat to exploit a vulnerability in a sub-component. Combinations of threat events and threat sources that lead to serious consequences for the mission are called threat scenarios. Some examples of threat scenarios and events that may happen to a drone are given in Figure 3. Within these examples, threat events exploit vulnerabilities of the Wi-Fi interface to force a drone to divert from its intended mission. In the first example, a drone flies to a fake home point after Wi-Fi communication with its remote controller station is lost. In the second example, malware is uploaded to the drone flight controller during a software update which opens a TCP/IP port on the Wi-Fi interface of the drone. During flight, this port is used to take over control of the flight.

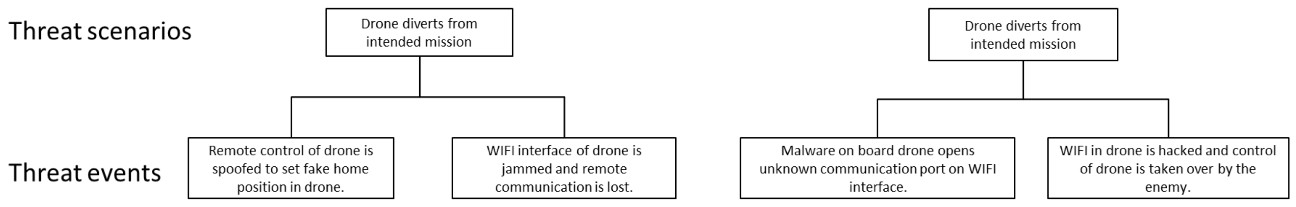

Threat scenarios
Threat events
Drone diverts from intended mission
Remote control of drone is spoofed to set fake home position in drone.
WIFI interface of drone is jammed and remote communication is lost.
Drone diverts from intended mission
Malware on board drone opens unknown communication port on WIFI interface.
WIFI in drone is hacked and control of drone is taken over by the enemy.

Figure 3: Threat Scenario and Threat Event Examples.

With the help of multiple experts in different disciplines (operations, maintenance, cyber security, etc.), a first set of threat scenarios can be derived. Assumptions are made – based on the expertise of the team – on the possibility of threat events on sub-components. Detailed analysis of sub-components will need to confirm these assumptions during the Full Risk Assessment step. Also, analysis of sub-components during the Full Risk Assessment step may give new insights in possible threat events. In this case, threat scenarios may need to be updated.

4.2.7 Likelihood Estimation for Each Threat Scenario

Likelihood is the probability of occurrence of consequences that impact the mission's success.

The likelihood is assessed for the occurrence of a threat event. The likelihood of a threat scenario is derived from the relationships between threat events as described in the threat scenarios.

Likelihood determination uses assessment scales for assessing the likelihood of threat event: Very High, High, Moderate, Low, Very Low. An assessment scale can be based on the likelihood of threat event occurrence, or on the likelihood of threat event initiation.

The likelihood of threat event occurrence can be estimated using historical evidence, focused for example on the number of occurrences in a year.

The likelihood of threat event initiation can be estimated based on the difference between the capabilities of the threat source and the required capabilities to initiate a threat event.

Sometimes, likelihood estimation of threat scenarios can be difficult for non-cyber security experts. Even for cyber security experts, in fact, it can also be difficult. If likelihood estimation is deemed too hard at this stage of the process, it can be postponed to step 4, Full Risk Assessment. It is indeed easier to estimate likelihood when one has actually tested the threat scenario and thus, has more information available.

4.2.8 Consequences for Each Threat Scenario

Consequences of threat scenarios are assessed from an operator point of view. The consequences can be categorized as technical such as loss of communication or inaccurate navigation position, safety critical such as loss of life, or financial. It is up to the team to decide on the most relevant areas.

For each category, assessment scales for the consequences are used and examples are given in Table 5.

Table 5: Assessment Scales for Threat Scenarios' Consequences.

	Areas of Consequences							
Assessment Scale	**Flight Safety**	**Communication**	**Navigation**	**Surveillance**	**Task Execution**	**Damage Human**	**Damage Material [$ or €]**	**Damage Environment**
5 – Very High	High	High	High	High	Complete failure	Fatalities	> 10.000.000	Catastrophic impact
4 – High	Some but not all of the HIGH consequences above	Medium-High	Medium-High	Medium-High	Almost complete failure	Multiple severe injuries	> 1.000.000, < 10.000.000	Long-term impact
3 – Moderate	Medium	Medium	Medium	Medium	Partial failure	Severe injuries	1.000.000	Noticeable impact
2 – Low	Some but not all MEDIUM consequences above	No-Medium	No-Medium	No-Medium	No failure, but additional effort	Minor injuries	> 5000, < 1.000.000	Short-term impact
1 – Very Low	No	No	No	No	No failure	No injuries	< 5000	Insignificant

4.2.9 Definition of Preliminary Risk Report

The risk of a threat scenario is derived from the likelihood and consequence. This process uses Table 6 to determine a risk level to each threat scenario, which is the same table used in NIST SP 800-30 R1 on Information Security, Table I.6 [3].

Table 6: Risk Determination from Likelihood and Consequence.

		Scale of Consequences				
		Very Low	**Low**	**Moderate**	**High**	**Very High**
Scale of Likelihood	Very High	Very Low	Low	Moderate	High	Very High
	High	Very Low	Low	Moderate	High	Very High
	Moderate	Very Low	Low	Moderate	Moderate	High
	Low	Very Low	Low	Low	Low	Moderate
	Very Low	Very Low	Very Low	Very Low	Low	Low

At the end of step 2, the preliminary risk report:

- Gives a list of threat events and threat sources;
- Defines threat scenarios together with a likelihood estimation and the consequences;
- Derives the risks of threat scenarios; and
- Identifies the most critical sub-components to be further analyzed (prioritization).

4.3 Step 3: Test Plan Production

In step 3, the client for whom this process is performed gets the preliminary risk report and determines which sub-components will be further assessed. This gets documented in a test plan. The test plan definition process is really up to the client as each one will have particular preferences. However, the following suggestions for the test plan contents are given:

- Selected threat scenarios from the preliminary risk report.
- Selected sub-components to be further tested (out of the selected threat scenarios). For each selected sub-component:
  - Test objectives/goals.
  - Test guidelines, procedures, tools related to test management.
  - Test output format.
  - Test and evaluation criteria/requirements.
  - Test configuration.
  - Roles/expertise needed in the test team.

  - Time needed estimation.
- Overall planning and time needed to perform the test activities for all sub-components.

The test plan can be developed with the help of the risk assessment team.

4.4 Step 4: Full Risk Assessment

In the previous steps, it has been possible to obtain an overview of the system under assessment and to identify the main sub-components which will be subjected to the Full Risk Assessment step.

Step 3 has identified a test plan, according to results of the preliminary risk assessment. Full Risk Assessment is composed of the following set of steps, which are to be repeated for each sub-components selected in the test plan:

- Test and evaluation criteria definition;
- Discovery of vulnerabilities affecting sub-components;
- Estimation of impact that may occur if a threat can exploit vulnerabilities;
- Estimation of the likelihood of a successful exploitation of vulnerabilities; and
- Preparation of a sub-component risk report, relating all the analyzed aspects.

The end result is a determination of cyber security risk for each selected sub-components. These risks can either confirm or infirm preliminary risks that were identified during the preliminary risk assessment or add new risks that were not previously identified. Also, after the full risk assessment of a sub-component, it becomes easier, from direct experience, to estimate the likelihood of a given threat scenario.

In this section, all time estimations are based on:

1) Assessments performed by the team on the example systems; and
2) The experience of the team on past assessments.

4.4.1 Test and Evaluation Criteria Definition

For each sub-component under evaluation included in the test plan, the first step is the definition of the test and evaluation criteria.

For each sub-component it is required to:

- Identify the appropriate test method and tool. Possible test methods and tools are provided in Annex C.
- Identify the standard or the benchmark to apply during test. This selection is done in accordance with the characteristics of the sub-component to be assessed. For instance, it could be possible to conduct an evaluation of the main security functions against Common Criteria, to assess the maturity level against a subset of NIST SP 800-53 Security Controls, to validate some cryptographic module according to FIPS criteria, or to validate a web application against OWASP Top Ten. If no appropriate standard exists, one could be defined so the work becomes more easily repeatable afterwards.
- Consider the following security principles [6] as evaluation criteria:
  - Minimality – only the functions, protocols, and services required to carry out the operational mission shall be installed and used.

- Least Privilege – entities using the system shall only be given privileges and authorizations they require to perform their tasks and duties.
- Self-protecting – the system shall treat other external system as untrusted and implement protection measures to control the exchange of information.
- Defence-in-Depth – protection measures shall be designed using an architectural approach and implemented in system sub-components, in security controls and in data to the extent possible, so that there are multiple lines of defence.
- Up-to-date Security Posture – system secure configuration shall evolve to maintain the required level of security while addressing changes in the threat environment.

4.4.2 Discovery of Vulnerabilities Affecting Sub-Components

Identification of vulnerabilities in a military system requires structured and in-depth work and can include the use of automatic tools but cannot be reduced only to this aspect. Vulnerabilities can be found in system architecture, system documentation, software configuration, security mechanism implementation and network interfaces.

The process of Vulnerability Discovery described hereinafter is shown in Figure 4.

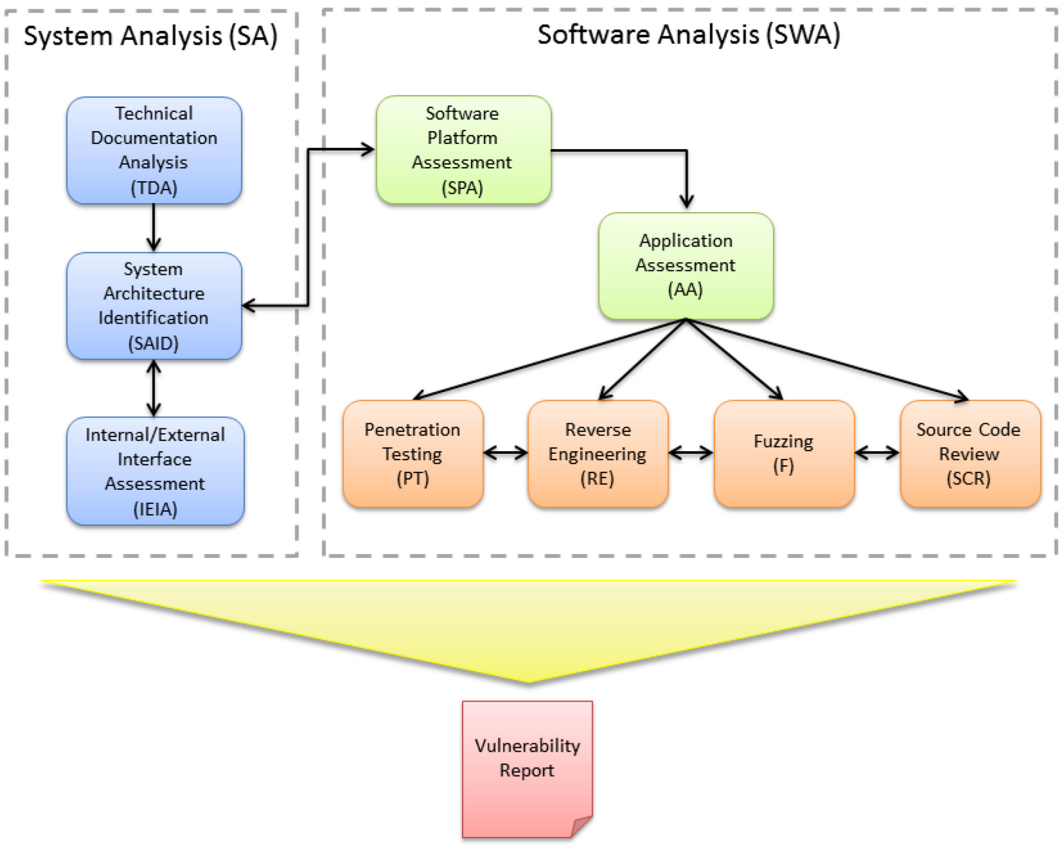

System Analysis (SA)
Technical Documentation Analysis (TDA)
System Architecture Identification (SAID)
Internal/External Interface Assessment (IEIA)
Software Analysis (SWA)
Software Platform Assessment (SPA)
Application Assessment (AA)
Penetration Testing (PT)
Reverse Engineering (RE)
Fuzzing (F)
Source Code Review (SCR)
Vulnerability Report

Figure 4: Process of Vulnerability Discovery.

The Vulnerability Discovery process is organized on two levels:

- System-level analysis (TDA, SAID, IEIA); and
- Software-level analysis (SPA, AA, RE, F, SCR).

As presented in Section 3 – Applicable Systems, the systems under assessment can be very different from one to another. At the two extremes, one could be an integrated device with embedded software (operational technology) and the other a traditional information technology system. The types of sub-components found in those systems are also very different.

Depending on the nature of the sub-component under assessment, it is possible to execute only some of all the activities part of this step of the process. The process can indeed be tailored according to the sub-component under assessment and to the available resources. The different steps lead the analyst in digging more and more into the sub-component, thus increasing the probability to find vulnerability, reducing the residual risk in the use of the system but also increasing the required time to complete the assessment.

The process of Vulnerability Discovery provides fundamental output for the assessment of the cyber security risks of the sub-component under evaluation. It has the following characteristics:

- Repeatable: other people (testers, developers) can reproduce the results.
- Documented: discovered vulnerabilities are explained in an understandable way.
- Widely applicable: the methodology is independent from the technology of the assessed sub-component.
- The output of the Vulnerability Discovery process is a report, providing information about:
  - The security posture of the sub-component; and
  - A complete list of the identified vulnerabilities.

The following sub-sections provide a description of each phase, explaining activities, required skills and estimated duration. Estimates of the duration of an activity are based on past experience and are applicable under the assumption that a team with all the required skills (Mission/Domain Experts, Security Analysts, Experts in fuzzing, Experts in reverse engineering, etc.) and with a number of people proportional to system dimension is available.

4.4.2.1 Technical Documentation Analysis (TDA)

This is the starting point to identify and scope the target of the Vulnerability Discovery activity. Documentation analysis consists in the study of:

- User manuals;
- Administration manuals;
- Architecture documentation and diagrams; and
- Any other documentation such as documentation about communication protocols, cryptographic approach, etc.

In this phase, it is useful to gather public information about problems already discovered on similar systems or vulnerabilities related to the technology in use.

This very first step allows the analyst to:

- Gather basic information about the sub-component (manuals, specs, etc.); and
- Gather information about possible vulnerabilities arising from public documentation.

At the end of this phase, a list of findings related to the documentation is available. Findings will be included into the final report.

Estimation of time needed for this phase: 4 to 8 weeks.

Needed skills: System/Software engineers with expertise in the domain of the sub-component and cyber security experts.

4.4.2.2 System Architecture Identification (SAID)

The objective of the System Architecture Identification phase is the determination of the technical characteristics of the sub-component's architecture, such as:

- Hardware components;
- Presence of firmware;
- Communication interfaces (Wi-Fi, Bluetooth, CANBUS, etc.);
- Software components (Operating systems, COTS, custom software, etc.); and
- Relationships between components of the sub-components.

This is different from the high level architecture of the system that was developed during preliminary risk assessment. Here, it is the architecture of a specific sub-component and the level of details can be greater.

Each type of sub-components can require different approaches and tools to accomplish the objectives of this phase. For instance, very small and integrated sub-components can require microscopes. Sub-components based on IP networks can benefit from the use of automated tools, like automatic network vulnerability scanners.

Estimation of time needed for this phase: 3 to 4 weeks.

Needed skills: System/Software engineers with expertise in the domain of the sub-component and cyber security experts.

Note: Since it is possible and desirable to repeat this phase using knowledge obtained from previous iterations of the process, it is better to perform multiple, shorter iterations of this phase.

4.4.2.3 Internal/External Interface Assessment (IEIA)

Communication interfaces are crucial since they can provide entry points for a threat to exploit a vulnerability.

The objective of this phase is to find vulnerabilities at the interface level. Different types of interfaces can be analyzed: communication (both wired and wireless), USB, Bluetooth, power, satellite navigation, CAN bus, dedicated connections, etc.

It is important to keep in mind that the interfaces are not necessarily limited to their intended use. Attackers might use sensors/actuators to influence the system or leak data.

IEIA is led by the following security principles:

- Minimality – only the functions, protocols, and services required to carry out the operational mission shall be reachable by the operator; and
- Defence-in-Depth – protection measures shall be designed using an architectural approach and implemented in security controls in layers, so that there are multiple lines of defence.

This step starts with the setup of the test environment, then tests are performed and results are analyzed. Tests are primarily penetration testing and verification of implemented security measures.

Estimation of time needed for this phase: minimum of 1 to 2 weeks per interface for standard/well-known interfaces and minimum of 4 weeks per interface for unknown/proprietary interfaces. For example, a sub-component with 3 standard interfaces and 2 proprietary interfaces could take 3 weeks for the standard interfaces and 8 weeks for the proprietary interfaces, for a total duration of 11 weeks for the IEIA phase. This estimated duration could be rounded up to 3 months (12 weeks).

Needed skills: System/Software engineers with expertise in the domain of the sub-component and cyber security experts.

4.4.2.4 Software Platform Assessment (SPA)

The objectives of Software Platform Assessment are first, the determination of the software architecture and second, the research of known vulnerabilities in the software platform. Vulnerabilities include:

- Patch level;
- Outdated software versions;
- Misconfigurations;
- Exposition to public vulnerabilities; and
- Compliance with or deviations from organization’s security policies.

A software platform is made up of:

- Operating systems;
- Database Management Systems, Web Servers, Application Servers, GIS Servers, etc.;
- Runtime Environment (e.g., .NET Framework, Java runtime, etc.);
- Middleware Platform (e.g., Hibernate, Oracle Service Bus, etc.); and
- Firmware.

The Software Platform Assessment is very different depending on the nature of the sub-component: firmware-based embedded or traditional IT sub-component.

With traditional IT sub-components, it is possible to use vulnerability scanners (e.g., Nessus, openVAS). Vulnerability scanners are very effective in finding known vulnerabilities, misconfigurations and non-compliance to security baselines. Vulnerability scanners provide a severity indication for each identified vulnerability and the remediation action to mitigate each vulnerability.

However, vulnerability scanners are also prone to false positives, so the tester must double-check before including the scan results into the results report. Since vulnerability scanners are also prone to false negatives, a common best practice is to use multiple vulnerability scanners from different vendors, to decrease the number of undetected vulnerabilities.

With firmware-based embedded sub-components, vulnerability scanners do not exist. Thus, finding known vulnerabilities in those sub-components usually means to search the Internet. For older, rare or proprietary firmware-based embedded sub-components, however, known vulnerabilities are rarely available. More advanced, dedicated techniques and tools must be used to find yet unknown vulnerabilities which is done in the next phase.

This phase provides an evaluation of the security posture of the software platform and that posture is documented in the report.

Estimation of time needed for this phase: For traditional IT sub-components, 1 week. For firmware-based embedded sub-components, 1 or 2 days are enough to search the Internet.

Skills needed: Software engineers with expertise in the domain of the sub-component and cyber security experts.

4.4.2.5 Application Assessment (AA)

The objective of this phase is to find *unknown* vulnerabilities in the software.

Thanks to the information gathered in the previous phases, it is now possible to:

- Identify the technological stack in use;
- Identify the application's entry points;
- Identify the input areas; and
- Understand the general application functions and data flow.

The specific aspects which are assessed in this phase include:

- Authentication and Authorization;
- Session Management;
- Input validation;
- Business Logic; and
- Data encryption.

As the search for unknown, new vulnerabilities demands deep understanding of the software inner workings, low-level analysis techniques are necessary. These include:

- Black-box and grey-box penetration testing;
- Fuzzing;
- Reverse Engineering; and
- Source code review.

These techniques can all be used on either traditional IT or firmware-based embedded sub-components, with the exceptions of fuzzing and source code review, which are usually much more difficult to perform on firmware-based embedded sub-components.

It is always the easiest to begin with the analysis of functions available to non-privileged users, to find vulnerabilities potentially exploitable by any legitimate user. Then, looking at all functionalities is the main activity behind penetration testing. Penetration testing is explained in Annex C.

Fuzzing can be used on traditional IT sub-components to more or less systematically find unknown vulnerabilities in software. Fuzzing is explained in Annex C.

If source code is not available, like it is almost always the case with firmware-based embedded sub-components and also oftentimes with traditional IT sub-components, it could be a good choice to plan for reverse engineering. Reverse engineering is explained in Annex C.

When source code is available, it is fundamental to plan for source code review. This activity is usually quite expensive when done manually, but source code analyzers are available to help automate the process to some extent.

Estimation of time needed for this phase: For black-box and grey-box penetration testing, 1 to 2 months. For fuzzing, 1 month. For reverse engineering, 2 to 6 months. For source code review, 1 to 4 months.

Skills needed: Software engineers with expertise in the domain of the sub-component, cyber security experts and experts in penetration testing, fuzzing, reverse engineering and/or source code review.

Overall estimation of the duration of this phase is thus from 2 to 12 months, depending on which activities are performed.

4.4.2.6 Estimation of Time Required for the Software Analysis (SWA) Phase

It is useful to have an estimation of the minimum time required for completing the SWA phase. Since the activities executed during SWA depend on the type of sub-component under assessment, the estimation will be focused on the two types of sub-components at the extremes in the range, which are traditional IT-sub-components (IT) and Firmware-based embedded sub-components (FW).

Table 7 provides the estimation, for each type of sub-component, of the duration of each activity. The execution of some activities has no or little sense depending on the features of the sub-component. If, for instance, the source code is available then source code review is possible. Otherwise, reverse engineering can be done. These two cases have been differentiated with letters SCR, meaning Source Code Review, and RE, meaning Reverse Engineering, in subscript.

Taking into account the different cases summarized by Table 7, it could be considered that 5 months is the average value for the minimum time required to complete the SWA phase.

This time estimation shows how important prioritization is in risk management. When considering a few dozen different military platforms, one can never perform SWA on everything. For each platform, only the few most critical sub-components must be selected and then, the SWA phase can be properly scoped for each sub-component. For each platform, having to assess between 5 and 10 different sub-components is typical. Thus, if only one person can do the job, it means it will take approximately 25 to 50 months to complete the

SWA work, which is, in the majority of cases, way too much time. When assessing an entire platform, having at least 3 people able to perform the SWA phase is thus required.

Table 7: Estimation of Time Required for SWA.

	SPA	PT	SCR / RE	F	TOTAL (months)
IT_{SCR}	0.25	1	2	1	**4.25**
IT_{RE}	0.25	1	3	1	**5.25**
FW_{SCR}	0.07	2	2	1	**5.07**
FW_{RE}	0.07	2	3	1	**6.07**

4.4.2.7 Vulnerability Report

The results of all activities performed under Vulnerability Discovery are documented in a structured deliverable, the Vulnerability Report. The exact structure depends on the preferences of the assessors, but it must provide information about:

- Activities performed.
- Tools used.
- Discovered vulnerabilities, with details on:
  - Technical information on how these vulnerabilities can be exploited.
  - Information on the impact (expressed in terms of the harm caused to the mission) of the exploitation of these vulnerabilities.

All this information is summarized in two values, to be provided for each vulnerability:

- Likelihood of a successful vulnerability exploitation.
- Impact of a successful vulnerability exploitation.

These values are very important because, at this stage, they are no longer the estimates hypothesized during the preliminary risk assessment, but they are supported by the technical activities which have been executed.

The likelihood of the discovered vulnerabilities' exploitation can be gauged during Vulnerability Discovery, using assessors' expertise and experience. This likelihood is generally called the exploitability of the vulnerability. Sometimes, assessors can even try to exploit the vulnerabilities themselves in order to better evaluate its exploitability. Exploitability is evaluated as Very High, High, Moderate, Low or Very Low.

The impact of successful vulnerability exploitation should always be evaluated with respect to the mission that the sub-component supports. The mapping from missions to capabilities to functions was done in step 1. Thus, by determining which function the sub-component's system supports, the potentially impacted mission can be determined. The level of impact should correlate with the level that was determined for the consequences of the threat scenario in which the vulnerability takes part. For instance, if the consequences of the corresponding threat scenario are high, the impact of the successful vulnerability exploitation should also be high.

In conclusion, all the collected data, activities performed, and results gained are summed up in the executive summary of the Vulnerability Report, which lists the discovered vulnerabilities and provides both the Likelihood and Impact of successful vulnerability exploitation.

This type of report provides to the stakeholders all the required input to determine the risk and if it can be accepted.

4.4.2.8 Estimation of Time Required for Vulnerability Discovery

It is possible to provide an estimation of the minimum time required to complete the Vulnerability Discovery phase.

Figure 5 provides a graphical representation of the minimum duration of each activity under Vulnerability Discovery. Vulnerability Discovery cannot be done without doing TDA and SAID. It is why they are also included. For IEIA, the needed time depends on the number of interfaces, and if they are known or unknown. However, 3 months is typically the minimum amount of time needed to analyze interfaces.

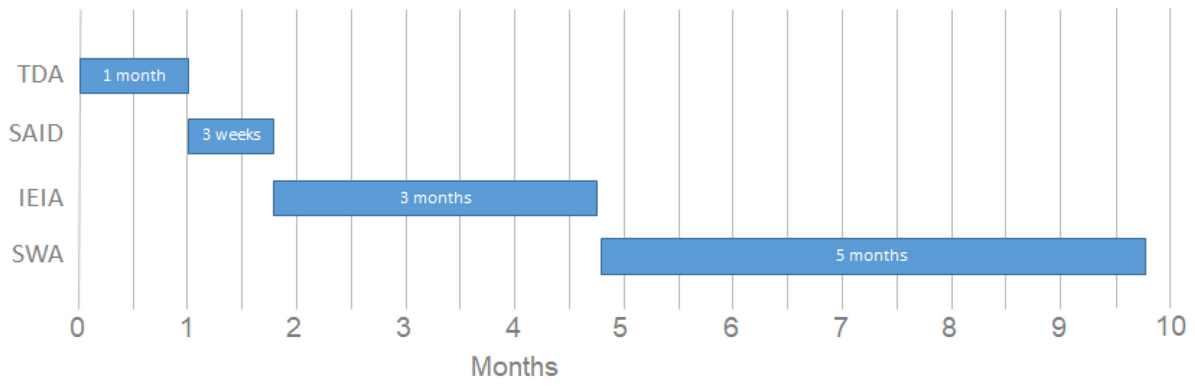


Figure 5: Minimum Time Required for Discovery of Vulnerability.

The minimum required time for Vulnerability Discovery is thus approximately 10 months.

This time estimation shows how important prioritization is in risk management. When considering a few dozen different military platforms, one can never perform Vulnerability Discovery on everything. For each platform, only the few most critical sub-components must be selected and then, the Vulnerability Discovery phase can be properly scoped for each sub-component. For instance, sub-components for which automatic tools can be used to help should be privileged and assessed first. Also, sub-components for which the Vulnerability Discovery team has relevant experience can be privileged. Here again, one person cannot do the necessary job for an entire platform. Having at least 5 people able to perform the Vulnerability Discovery phase is usually required, which includes the 3 people working on the SWA phase.

4.4.3 Sub-Component Full Risk Report

The sub-component full risk report uses the Vulnerability Report along with the determined likelihood and impact of each vulnerability exploitation to document the risk of successful vulnerability exploitation. It is common to present those risks in a prioritized list such that higher risks are at the beginning of the list.

As in a preliminary risk assessment, risk here is a function of likelihood and impact. Thus, the risk of successful vulnerability exploitation can be determined using Table 8.

Table 8: Risk Determination from Likelihood and Impact.

		Scale of Impact				
		Very Low	**Low**	**Moderate**	**High**	**Very High**
Scale of Likelihood	Very High	Very Low	Low	Moderate	High	Very High
	High	Very Low	Low	Moderate	High	Very High
	Moderate	Very Low	Low	Moderate	Moderate	High
	Low	Very Low	Low	Low	Low	Moderate
	Very Low	Very Low	Very Low	Very Low	Low	Low

The sub-component full risk report presents the prioritized list of risks for the sub-component and can refer to the Vulnerability Report when necessary. However, since the Vulnerability Report is usually a very technical report, the sub-component full risk report should make sure that it can still be understood by all stakeholders.

4.4.4 Estimation of Time Required for Full Risk Assessment

Full Risk Assessment includes many different activities which are not always all performed depending on the type of sub-component under assessment.

Table 9 provides a summary of the activities included in Full Risk Assessment with an estimation of the minimum time needed for each.

Table 9: Estimation of Time Required for Full Risk Assessment.

Activities	Time Needed
Test and Evaluation criteria	2 weeks
Vulnerability Discovery	2 or 10 months
Sub-component Full Risk Report	2 weeks
Minimum time for Full Risk Assessment	**3 or 11 months**

The minimum time needed for Full Risk Assessment has 2 possible values:

- 3 months, if Vulnerability Discovery is executed in its minimal extension (only TDA and SAID activities).
- 11 months, if Vulnerability Discovery is executed in its complete extension (TDA, SAID, IEIA and SWA).

4.5 Step 5: Risk Report Production

The final step of the process is rather simple. It is about producing the final risk report summarizing the results of the entire process. The exact format and contents are up to the assessors, but the report must at least include:

- An overview of the system architecture and the documentation used to understand the system.
- A description of the supported missions, capabilities and functions, with the relationships between them.
- The scope of the current assessment.
- The results of the preliminary risk assessment.
- The test plan.
- The results of the Full Risk Assessment.
- The prioritized list of risks that were identified.

4.6 Application of the Risk Assessment Outcomes

Once the final risk report is produced, it is given to the customer of the assessment. The customer can then decide which risks, from the prioritized list, will need to be mitigated and to what extent. The assessment team can help the customer make that selection.

Once the selection is made, a mitigation plan must be produced. This mitigation plan was out of the scope of this process. However, once again, the assessment team, especially the Full Risk Assessment team, can support the customer in the creation of that plan. On top of a mitigation strategy, like a security measure, for each selected risk, a mitigation plan can also include a security architecture, that is, a plan for how all defined security measures will be integrated into the architecture of the entire system.

4.7 Maintenance of the Risk Assessment Outcomes

It has been stated a few times in this document, cyber security risk assessment must be conducted in an iterative fashion. Thus, to more easily maintain the knowledge acquired from one risk assessment to the other, it is advisable to develop a system, a tool, to support the process depicted in this document. This tool would need to be backed by a database to keep track of the results of each assessment and, of course, to let future assessors reuse the work of their predecessors. Over time, this tool would contain a mine of very valuable information on the security posture of the assessed systems but also on their architecture, use cases, missions they support, etc.

4.8 Estimation of Time Required for the Overall Process

Based on past experience in system assessment, it is possible to provide an estimation of the minimum time required for executing a complete iteration of the process presented in this document.

The estimations presented assume that the necessary skillsets are readily available to perform the process. It has to be noted also that the time needed to perform the process does not scale linearly with the number of people in the team. So, if one triples the team size, the time needed is not necessarily three times less.

Table 10 provides a summary of the steps of the process and the minimum time needed for each step.

Table 10: Estimation of Time Required for the Overall Process.

Step	Phase	Time Needed
1	Input Collection	2 weeks
2	Preliminary Risk Assessment	2 months
3	Test Plan Production	2 weeks
4	Full Risk Assessment	3 or 11 months
5	Risk Report Production	2 weeks
	Minimum time for the overall process	**6.5 or 14.5 months**

Figure 6 shows the sequence and the duration of each step of the complete process. Blue color indicates a process execution with the Full Risk Assessment in its minimal extension, whereas orange color indicates that Full Risk Assessment has been executed in its complete extension.

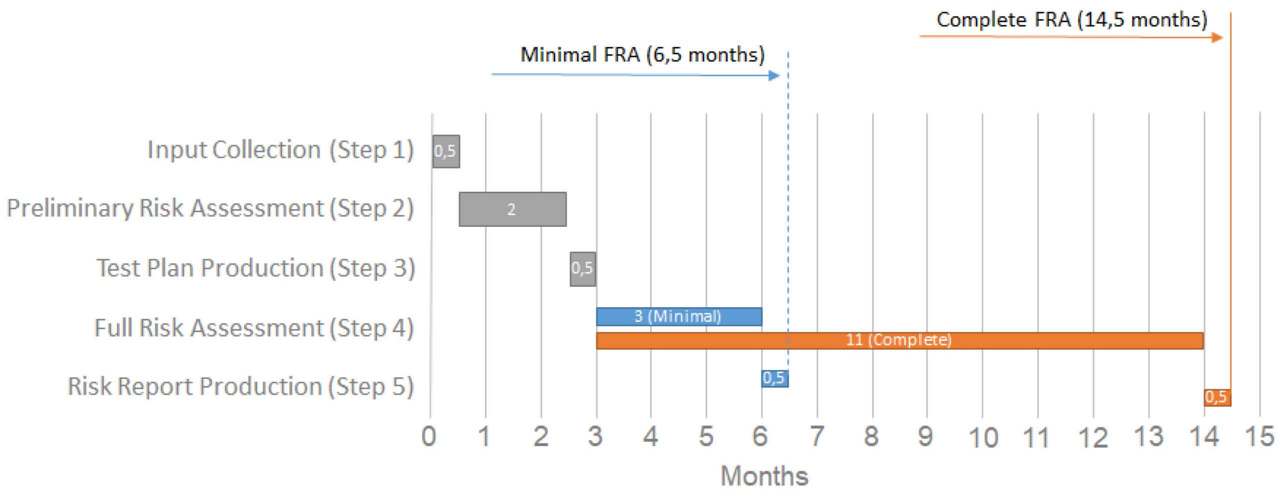


Figure 6: Minimum Duration of the Complete Process.

The minimum durations of a complete iteration of the process are:

- 6.5 months, if Full Risk Assessment is executed in its minimal extension (only TDA and SAID activities).
- 14.5 months, if Full Risk Assessment is executed in its complete extension (TDA, SAID, IEIA and SWA).

The process should be repeated periodically, to update threat scenarios to new threats, to analyze system changes or evolution and to assess the mitigations introduced to reduce the risks associated to discovered vulnerabilities. These duration estimations are mostly applicable to the first process iteration. Successive iterations should have shorter durations if the same team is used since team members will now have gained a lot of knowledge on the system.

4.8.1 General Criteria for Implementing Full Risk Assessment

According to the data showed in the previous section, it is important to define the Full Risk Assessment (FRA) phase and its related activities, since FRA can change the overall process duration by more than double (from 6.5 to 14.5 months).

The variable part in the FRA process is the Vulnerability Discovery phase (Figure 7). This phase may vary depending on different factors, for instance, the complexity of the system under analysis, the availability of information and the amount of available budget and time.

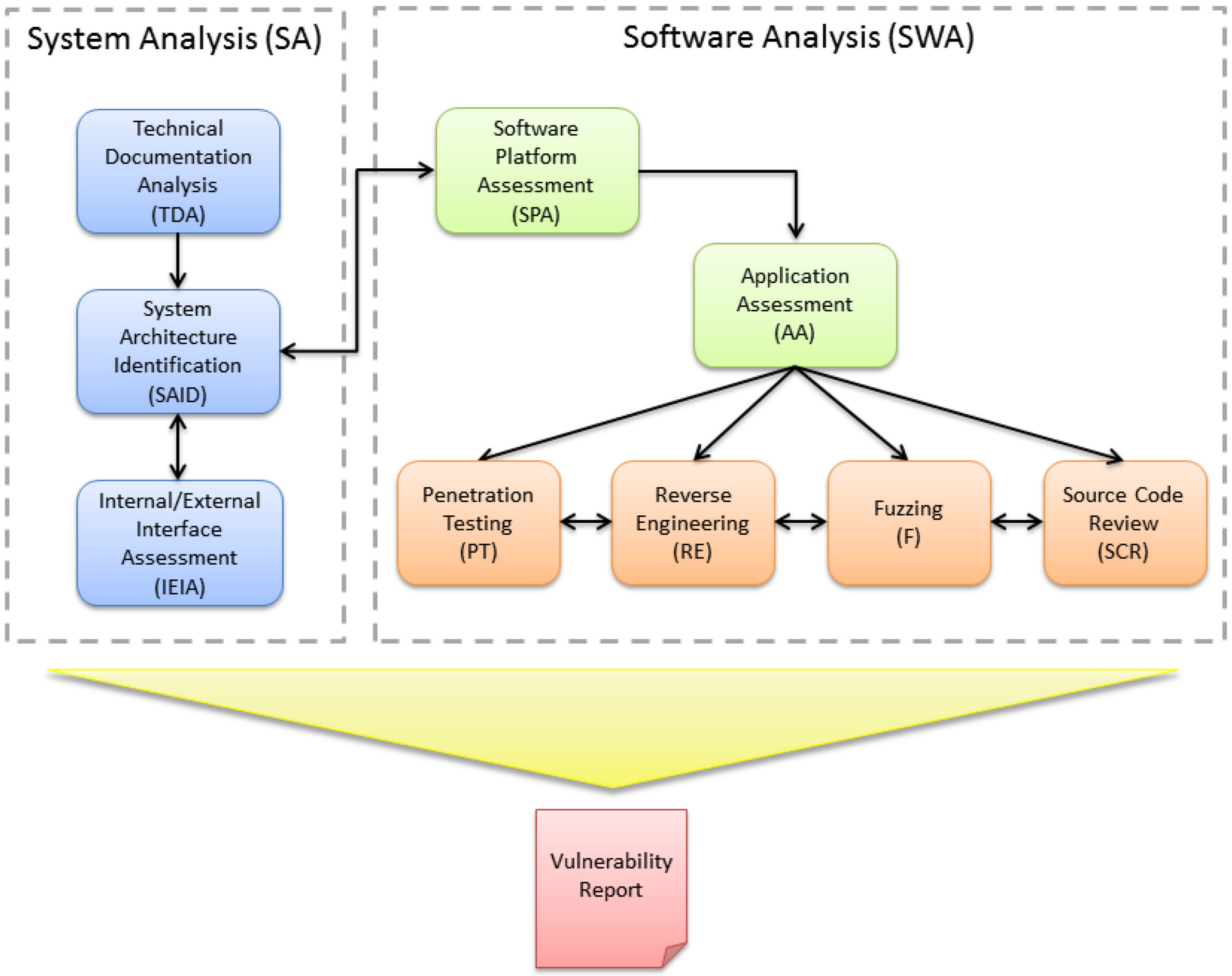

System Analysis (SA)
Software Analysis (SWA)
Technical Documentation Analysis (TDA)
System Architecture Identification (SAID)
Internal/External Interface Assessment (IEIA)
Software Platform Assessment (SPA)
Application Assessment (AA)
Penetration Testing (PT)
Reverse Engineering (RE)
Fuzzing (F)
Source Code Review (SCR)
Vulnerability Report

Figure 7: Process of Vulnerability Discovery.

It is practically useful to identify the various cases in which to use the different activities under Vulnerability Discovery:

- A minimal sub-component analysis should include TDA and SAID.
- If the sub-component only includes well-known interfaces then IEIA is certainly advised as the level of effort would be limited.
- The number of unknown or proprietary interfaces determines the effort needed to perform IEIA. A selection among those unknown interfaces can then be made.
- IEIA usually has priority over SWA.
- The first step of SWA should always be SPA.
- When doing SWA:
    - If budget and/or time are limited, start with F.
    - If source code is available, combine F with SCR.
    - If source code is not available, combine F with RE.

5.0 CONCLUSION

This document presented a cyber security risk assessment process for military systems. Risk assessment processes for the cyber security of traditional IT systems abound in the literature but very few exist, if none, for military platforms and systems which contain hundreds of firmware-based, communicating embedded systems. By using this process, NATO countries can better understand the current cyber security posture of their military systems and proceed to improve it.

In the next few years, the process would need to be applied so it can be tweaked where necessary. This is something the team members of this NATO IST RTG are planning to do in a follow-up activity. It would not only improve the process but also augment its visibility and awareness among NATO countries, so more can benefit from it.

6.0 REFERENCES

[1] "IEEE/ISO/IEC 42010-2011 – ISO/IEC/IEEE Systems and Software Engineering – Architecture Description", C/S2ESC – Software & Systems Engineering Standards Committee, December 2011.

[2] Cambridge Advanced Learner's Dictionary, 4th Edition, Cambridge University Press, June 2013.

[3] NIST SP 800-30 NIST Special Publication 800-30 Revision 1, "Guide for Conducting Risk Assessments", September 2012.

[4] Federal Office for Information Security, BSI Durchführungskonzept für Penetrationstest, "Study – A Penetration Testing Model", Federal Office for Information Security, Bonn, Germany. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration_pdf., April 2008.

[5] Gartner Information Technology Glossary, https://www.gartner.com/en/information-technology/glossary, July 2021.

[6] NATO: AC/35-D/2004-REV3, “Primary Directive on CIS Security”, NATO UNCLASSIFIED, November 2013.

Annex A – DRONE

The drone or quadcopter is a four-rotor helicopter. The rotors point upwards and are placed in a square with an equal distance from the center of mass of the aircraft. The quadcopter is controlled by adjusting the rotational speed of the rotors rotated by the electric motors. Due to its structure, it is typical for small vehicles without air pilot [1].

There are many electronic components on a drone [2]. The data collected via the GPS receiver, compass, accelerometer and barometer on the drone is transferred to the flight control system, which allow the drone to fly in a balanced manner. The user performs the take-off and controls the flight with the remote control (Figure A-1). The image taken by the camera of the drone is continuously transferred to a smartphone or LCD screen that the user attaches to the structure on the remote control. The user can control the flight with this display. In addition to the image of the drone camera on the user interface, information about charge status and flight can be seen. In addition, the point where the take-off takes place can also be marked, and when an unexpected signal interruption or the end of the charge happens, the drone can return to the point at which it took off.

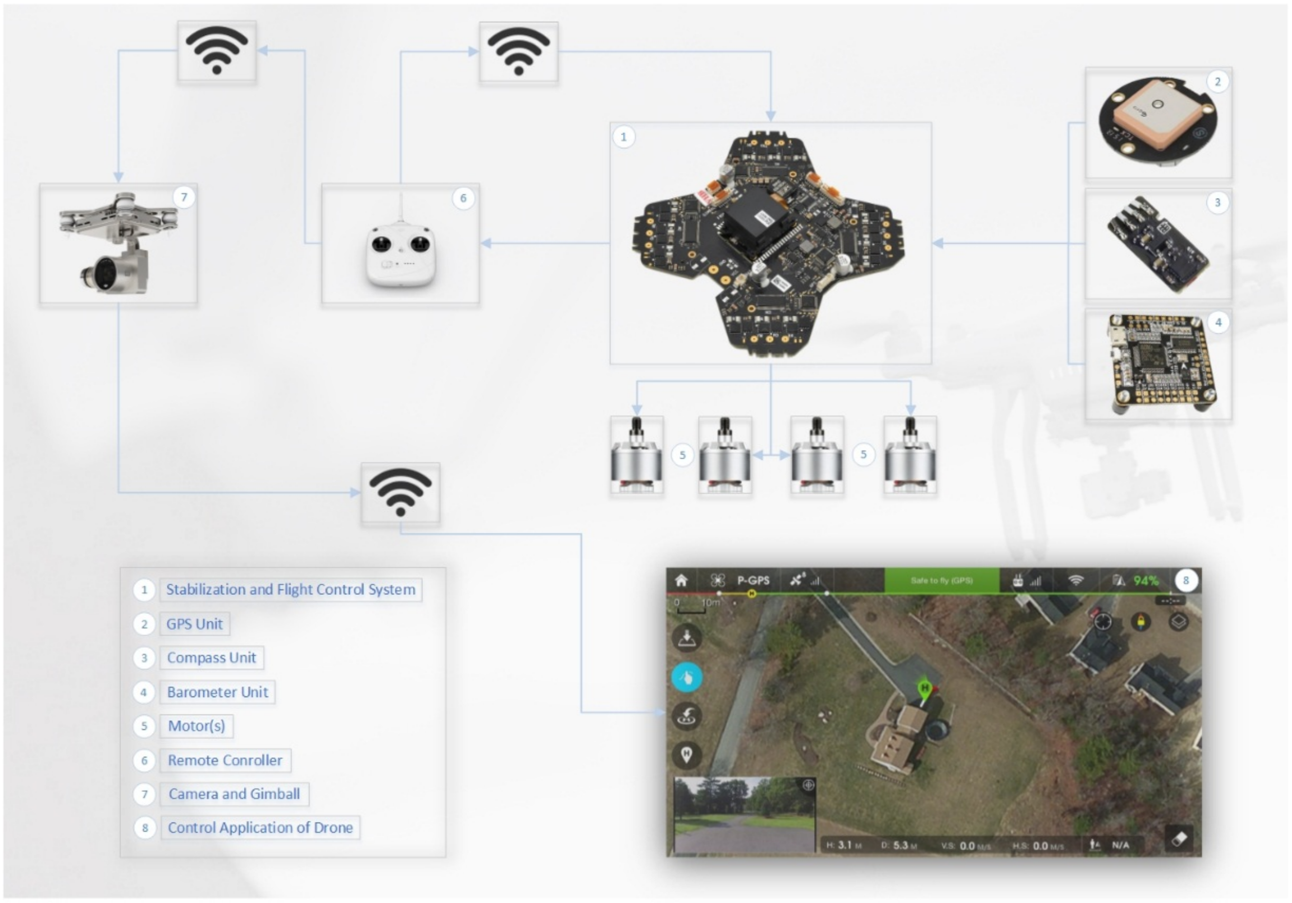


Figure A-1: Flight Management System of Drone.

A.1 STABILIZATION CONTROL SYSTEM

A stabilization control system is used to obtain stable flight. This system is needed to overcome real-world disturbances and take account of unknown offsets. A series of Proportional Integral Derivative (PID) controllers are applied to guide the quadcopter orientation to the desired values.

A.2 FLIGHT CONTROLLER SYSTEM

The flight controller consists of an MCU (Micro Control Unit), an IMU (Inertia Measurement Unit), a radio and an external IO [3]. The flight controller is responsible for the balance of the quadcopter and the execution of the user control. In order to simplify the software, a microprocessor with sufficient processing power is needed for multiple floating point instructions.

To keep the control schedule and the IMU collection stable, the software is implemented in three separate threads. These threads primarily manage IMU filter updates, control system updates, and radio link updates. This quadcopter software application is based on a consistent deduction of IMU data.

A.3 GPS UNIT

The global positioning system is a satellite navigation system that uses a radio receiver to collect signals from orbiting satellites to determine position, speed and time. This navigation system is more accurate than other forms of navigation and provides position knowledge to within a few meters. Advanced GPS systems can provide even better accuracies to within a few centimeters. Thanks to this accuracy of GPS systems manufacturers added features like “Return to Home” to products.

A.4 COMPASS UNIT

The compass sensor, or magnetometer, measures the magnetic force, just like a compass. This sensor is important for multi rotor drones because the accelerometer and gyroscope sensors are not enough to let the flight controller know which direction the drone is facing.

Compass sensors are very sensitive to magnetic interference. Things such as wires, motors and ESCs can all cause magnetic interference. That is why an additional compass sensor is mounted on the GPS module as the GPS module is mounted far away from all other equipment.

A.5 BAROMETER UNIT

Designed to allow drones to know at what height above the ground they are by measuring the air pressure. Considering that the air pressure changes with altitude, the drone can determine its own height using the barometer.

A.6 MOTOR

Part of the drone that rotates the propeller. Weight is an important factor when selecting motor type and capacity.

A.7 REMOTE CONTROLLER SYSTEM

FPV (First Person View), also known as RPV (Remote Person View) or video pilot, is a method used to control an RC Vehicle, from a driver's or pilot's point of view [4]. The sudden rise in the use of Drone or Unmanned Aerial Vehicles over the past few years has also increased the use of the most commonly used FPVs to steer these objects.

The Quadcopter is driven or controlled by a remote control from the First Person perspective transmitted via wireless technology to the pilot's goggles, monitor or smartphone which have the necessary application. More complex versions of the aircraft include a pan and tilt camera controlled by a gyroscope sensor with two built-in cameras providing 3D images in the pilot's glasses.

The 2.4 GHz and 5.8 GHz are the two most common frequencies when dealing with a FPV Quadcopter. 2.4 GHz is the common RF used by the Quadcopters to connect the Ground Transmitters to the drone. 2.4 GHz is the frequency at which wireless computer networks operate. As expected, there are a few incidents of loss of control over flying objects in dense residential areas with too many wireless signals.

A.8 CAMERA

Cameras are usually GoPro or other compact high-definition video units with built-in storage. Real time flow is possible on the latest quadcopters [5]. Because of the position of the drone, it is very nice to take photographs or videos with a drone, but it is not possible to take the right photo or film with the wrong camera on the drone. Many drone manufacturers produce gimbals compatible with the GoPro Hero camera series. The newest planes all come together with a gimbal and camera. These cameras and lenses are especially designed for air shots and photos. Images captured by the camera are processed by a video processor and adapted to wireless transfer. In this way, images are transmitted to the user.

A.9 GIMBAL

Consists of a case to store the microcontroller unit as well as the motor driver circuit and a mechanical construction to hold up each motor and the camera. The wiring from each motor driver runs through the handle into each separate motor. The placement of the sensor was mounted under the camera parallel with the lens to get the best result of the camera angle [6].

A.10 SECURITY OF DRONES

The cyber security of drones generally consists in the ability to keep their control. A drone owner does not want an attacker to be able to overtake control of the drone. Thus, when evaluating the cyber security of a drone, someone should identify the protocol used between the drone and its remote controller, along with its encryption and authentication and its implementation in both the drone and remote controller's firmware, as primary targets for the evaluation. This evaluation can be done by following the guidance in this report.

A.11 REFERENCES

[1] Vergouw, B., Nagel, H., Bondt, G., and Custers, B. “Drone Technology: Types, Payloads, Applications, Frequency Spectrum Issues and Future Developments”, The Future of Drone Use, Information Technology and Law Series 27, 2016.

[2] Gamulescu, O.-M., Risteiu, M.-N., and Leba, M., “The Hardware Used in the Structure of Drones”, University of Petrosani, Romania, April 2016.

[3] Zimmerman, N.M., “Flight Control and Hardware Design of Multi-Rotor Systems”, Marquette University, WI, August 2016.

[4] Pastor, E., Lopez, J., and Royo P. “UAV Payload and Mission Control Hardware/Software Architecture”, Technical University of Catalonia, IEEE A&E Systems Magazine, June 2007.

[5] Department 13, Mesmer Counter Drone, White Paper: Anatomy of DJI’s Drone Identification Implementation, November 2017.

[6] Algoz, A., and Bakhtiyar, A. H., “A Control System for a 3-Axis Camera Stabilizer”, Teknisk-naturvetenskapliga fakulteten UTH-enheten, Uppsala Universitet, June 2018.

Annex B – MOTOROLA TETRA PORTABLE RADIO (MTH500)

Portable radios send voice communications over a digital bit stream. This flow is converted into a Radio Frequency (RF) signal transmitted over the air to another radio. This process is called digital modulation.

B.1 DIGITAL MODULATION

The MTH500 is a portable radio that has two models operating in two different frequency bands: R1 operates within 380 to 400 MHz and R2 within 410 to 430 MHz [1]. There are two modes of operation. Trunked Mode Operation (TMO) is when a MTH500 mobile radio starts a group call when the user presses the “Push To Talk” (PTT) button. Audio is then transmitted to the selected TETRA base station. The other radios that have the same talk group selected and are being served by a different TETRA base station receive the audio via their base station. In TMO mode the network state is used to assign a channel and transport the audio from sender to receiver. The audio is transported by the TETRA network or infrastructure. The other mode of operation, the Direct Mode Operation (DMO), is the term used by the TETRA industry to describe the ability of TETRA radio terminals to communicate directly with each other (like ‘Walkie-Talkies’) independent of the Trunked Mode Operation (TMO) network.

The MTH500 uses two digital technologies: Differential Quadrature Phase Shift Keying (DQPSK) and Time Division Multiple Access (TDMA). DQPSK is a modulation technique that transmits information by changing the phase of the Radio Frequency (RF) signal. The data is converted into complex symbols that change the RF signal and transmit the information. When the signal is received, the phase change is converted back to the symbols and then to the original data. The system can accommodate up to 4 audio channels in a standard 25 kHz channel, as used in a two-way radio. Time Division Multiple Access (TDMA) is used to allocate portions of the RF signal into four slots, one for each unit. The time allocation allows each unit to transmit voice data without interference from other transmitting units. Transmission from a unit or base station is placed in slot lengths of 15 milliseconds and frame lengths of 60 milliseconds. The TDMA technique requires advanced algorithms and a Digital Signal Processor (DSP) to perform audio compression / decompression and RF modulation / demodulation.

B.2 TRANSCEIVER

All radio circuits are available in the digital / RF board and the keypad panel. The digital / RF board is divided into the following sections: digital, transmitter and receiver (Figure B-1) [2].

B.2.1 Digital Section Description

The digital section contains the Redcap 2 chip, which consists of a Mcore RISC machine and a Digital Signal Processor (DSP). Mcore is the controller of the digital / RF board. It controls the operation of the transmitter, receiver, sound and synthesizer integrated circuits in the RF section. It also communicates with the keypad and display. The Digital Signal Processor (DSP) performs modulation and de-modulation functions for the radio. It also performs Forward Error Correction and other correction algorithms to overcome channel errors. It carries out linear 16-bit analog to digital conversions, audio filtering, and level amplification for the microphone audio input and the received audio output. The power and sound section is based on the GCAP III chip and includes power supplies, 13 bit CODEC, audio routing, microphone and earpiece amplifiers. An audio power amplifier is used for the speaker.

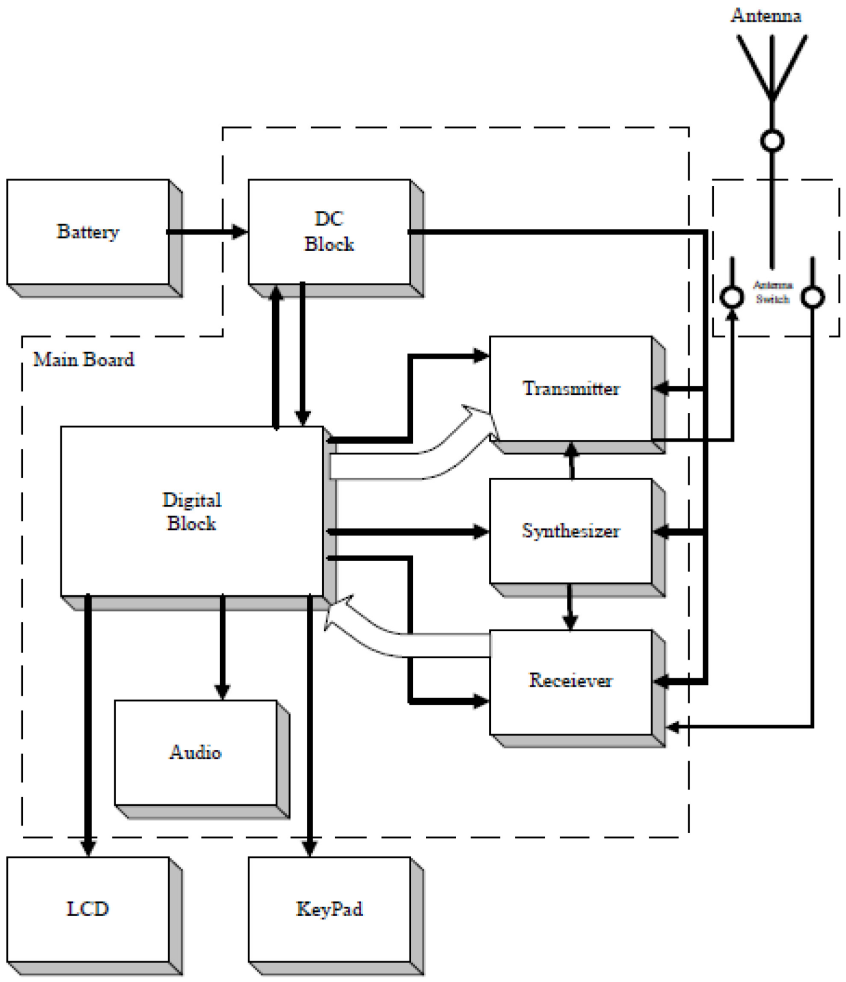


Figure B-1: General Block Diagram of MTH500.

B.2.2 Transmitter Section Description

The transmitter circuitry includes a linear class AB Power Amplifier (PA) for the linear modulation of the MTH500. It also includes a cartesian feedback loop to enhance its transmitter linearity and reduced splattering power into adjacent channels. The transmitter path consists of a cartesian feedback loop that contains the forward and loop feedback paths. The forward path includes the low noise Offset Direct Conversion Transmitter (ODCT), Balun, Attenuator, and Power Amplifier. The loop feedback path includes the directional coupler, attenuator, and Low Noise Offset Direct Conversion Transmitter (LNODCT) ASIC. The cartesian feedback output power passes to the antenna through the Isolator, Antenna Switch, and Harmonic Filter.

B.2.3 Receiver Section Description

The received signal from the antenna is directed by the Antenna Switch to the front filter. This block-type filter, which defines the receive frequency range, blocks the half IF and image frequency entry, and reduces the RF oscillator leakage. The signal is mixed with the local oscillator to create the first IF at 109.65 MHz. The signal is filtered by the crystal filter and sent to the World Phone IC (WPIC) ASIC.

B.3 SECURITY OF THE MTH500

The cyber security of the MTH500 essentially relies on the firmware that the digital block executes. Thus, if someone would want to evaluate the cyber security of such a radio, the primary target would be to evaluate that firmware, using the process described in this report.

B.4 REFERENCES

[1] MTH500 TETRA Portable Radio R1:380-400 MHz (PT811F) R2:410-430 MHz (PT511F), "Basic Service Manual", Motorola Inc., 2001.

[2] MTH500 TETRA Portable Radio R1:380-400 MHz (PT811F), "Detailed Service Manual", Motorola Inc., 2001.

Annex C – STATIC AND DYNAMIC PROGRAM ANALYSIS

C.1 INTRODUCTION

Software analysis can be divided into two classes: Static and Dynamic Program Analysis.

Static program analysis is the analysis that is performed without executing the program. It can thus be performed at early stages of the development process. It can be performed on source code, which is then easier, or on binary code, which is much more difficult. In both cases, automatic and semi-automatic tools exist to help the analyst. Attackers anticipating static analysis might take precautions to hinder access to code by encrypting (also known as "packing") it. This is reversible but, in most cases, not automatable with current solutions which decreases the analysis speed. Examples of static analysis include:

- Reverse engineering;
- Data flow analysis; and
- Control flow analysis.

Dynamic program analysis is performed by executing program on a real or virtual processor. For dynamic analysis to be effective, the program must be executed with sufficient test scenarios. The generation of test scenarios depend on the given situation. Sometimes, the tester will manually change the test scenarios to analyze certain aspects of the program. In other situations, tests are automated with fuzzers which modify a given test scenario and re-run it. Attackers anticipating dynamic analysis might take precautions to hinder the analysis. The success of those precautions has been demonstrated by the malware community for years as they almost trivially bypass anti-virus solutions and security appliances. Examples of dynamic analysis include:

- Penetration testing;
- Fuzzing; and
- Sandbox tests.

C.2 PENETRATION TESTING

Penetration testing, or penetration test or pentest, is an authorized cyberattack on a computer system. It is performed to evaluate the security in a realistic environment and can be classified in dynamic analysis. Typical penetration tests follow a multi-step approach. The first step is the definition of the scope and information gathering. This step is very important as it defines which systems will be evaluated and what information will be available to the testers. In the context of this report, it is important to distinguish the scope from the process' step 1 and the penetration testing's scope. The penetration test is one part of the overall process, and the scope will be much smaller. Too broad scopes prevent the testers from focusing and will lead to minimal results due to a lack of time. Another important aspect is the testers' level of information access. We can distinguish two extremes: White Box and Black Box. In a White Box Test the testers get access to all relevant information, e.g., system architecture, source code, documentation. This might even contain working login credentials if insider attacks are in scope. On the other end are Black Box Tests. In such tests, pentesters have no access to any information. Part of the test is the collection of system information. Grey Box Tests are between the two extremes. Although Black Box Tests seem to be much more realistic compared to real attacks, it is typically a waste of resources as defenders should always assume that attackers have all information they need, that they can gather that information. The usual penetration test steps are: reconnaissance, scanning, vulnerability assessment, exploitation, and documentation.

Several guides and standards have been developed over time, e.g., NIST 800-115 [1], OWASP Testing Guide [2], PCI DSS Penetration Testing Guide [3] and BSI Durchführungskonzept für Penetrationstest [4]. However, penetration testing is far from being an easy standard procedure and requires highly skilled personal, especially if the system under assessment is a non-standard system.

Most penetration testers use a mix of standard and per-test-specific tools. Standard tools like OpenVAS [5], Nessus [6] and Metasploit [7] scan systems and identify possible security risks almost automatically. It is common to identify hundreds of security problems in larger systems with such tools. Some of the findings might be false positives such that penetration testers have to analyze the results. However, those results are a good starting point as they provide an overview of possible attack vectors.

It is important to note that a penetration test without any findings is a non-conclusive security proof. The result is that the penetration testers did not find any problems with respect to the scope of the test, the given time, and their skills. More advanced attackers or systems out of scope may pose serious security risks. However, penetration tests are a good method to identify weaknesses in the system. The high value of penetration tests is recognized for example by the payment card industry which established the Payment Card Industry Data Security Standard (PCI DSS) to increase the IT security of merchants accepting their cards by extensive penetration tests and further security requirements.

Penetration tests are typically performed on real production systems and penetration testers operate carefully, such that identified weaknesses and exploitation of them will not interrupt the production process. However, there is always a small risk for unexpected problems. Keep in mind that penetration testers try to improve the system security while attackers will not take care. Nevertheless, our experience showed that previously untested systems might have so many security problems, that a first penetration test should be done in a very controlled manner or on a replicated test system.

C.3 FIRMWARE ANALYSIS

Firmware is a specific class of computer software that provides the low level control for the device's specific hardware (Figure C-1). Firmware can either provide a standardized operating environment for the device's more complex software (allowing more hardware-independence like done by the PC BIOS), or, for less complex devices, act as the device's complete operating system, performing all control, monitoring and data manipulation functions. It is important to note that even less complex devices may contain highly complex firmware. Increased processing power and large storage allows a higher degree of abstraction and reuse of code such that even less complex devices may run multiuser operating systems with capabilities not unlike classical PCs. Firmware is omnipresent as it is part of computer processors, Engine Control Units (ECU), smartphones, memory cards, hard drives, and almost every electronic device used nowadays.

Firmware is held in non-volatile memory such as EPROM or flash memory. Changing the firmware of a device may rarely or never be done during its lifetime. Common reasons for updating firmware include fixing bugs or adding features to the device. This may require ROM integrated circuits to be physically replaced, or flash memory to be reprogrammed through a special procedure.

Most firmware cannot be accessed by the user, the operating system, or maintenance staff. Access is limited to the vendor unless special acquiring and analysis methods are used. In general, there are two options to acquire firmware:

1) In many cases vendors provide firmware updates. However, vendors might protect the update files utilizing encryption to prevent the analysis. Or they provide incremental updates which do not contain the full firmware but only patches for certain parts of the firmware.

2) Alternatively, the firmware can be extracted from the device itself. The extraction process might include desoldering of the flash storage or the utilization of debug ports like JTAG, UART or Telnet. However, there is a moderate risk of bricking the device.

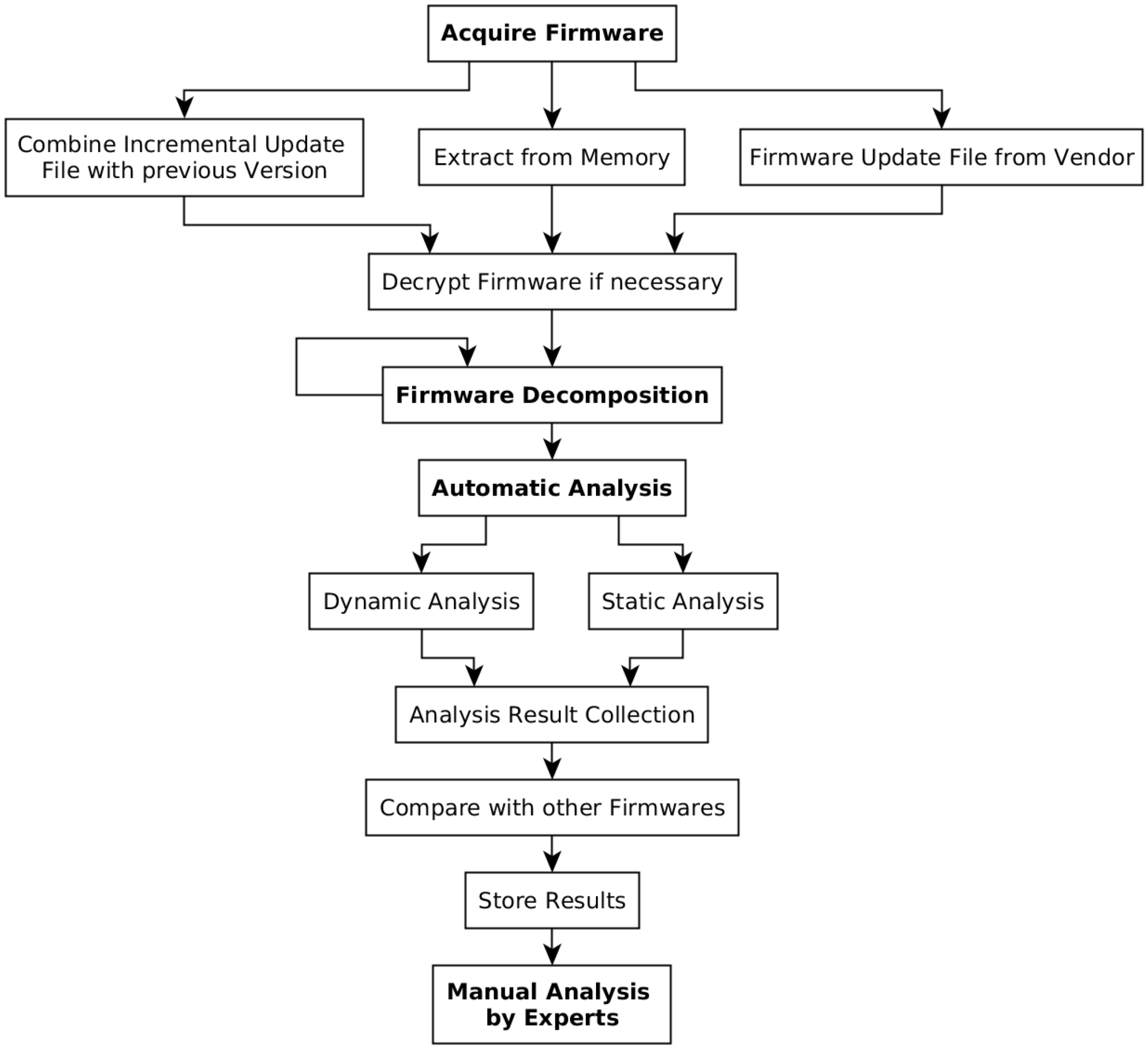

Acquire Firmware
Combine Incremental Update File with previous Version
Extract from Memory
Firmware Update File from Vendor
Decrypt Firmware if necessary
Firmware Decomposition
Automatic Analysis
Dynamic Analysis
Static Analysis
Analysis Result Collection
Compare with other Firmwares
Store Results
Manual Analysis by Experts

Figure C-1: Firmware Analysis Process.

After the acquisition of the firmware, the analysis process can be started. Typically, this starts with a decomposition of the acquired firmware image into its parts (e.g., bootloader, firmware core software, file systems). Following analysis steps can be classified as dynamic (executing the complete firmware of software parts) or static analysis (analysis that is performed without executing programs). Typical examples of dynamic analyses are penetration tests, fuzzing, or running the code inside a debugger. Although, these methods need some adaptations to support firmware analysis, they are very similar to regular software analysis. Typical static firmware analyses gather further information on the firmware itself. This might be information on the used software and detailed software version information, used cryptographic material (e.g., public and private keys) or functions, hard coded user credentials, etc. This information is typically used by experts analyzing the firmware with less automatable methods like reverse engineering.

The described firmware analysis approach is a preliminary step for a holistic firmware analysis where firmware information is collected (as described here) and follow up tasks are planned as needed (e.g., reverse engineering or penetration testing). The required expertise and time depend on the firmware. The Firmware Analysis and Compare Tool (FACT) automates almost all steps and instruments further analysis tools. No special expertise is needed to use FACT and the analysis is automated as long as the firmware type is supported. The analysis time depends on the firmware type and the processing power of the analysis system. Typical firmwares are analyzed within a few minutes. However, new firmware types or protected firmware images require at least some basic, sometimes expert knowledge to extend FACT to support these types. Non-automated or partially automated analyses based on tools like Binwalk [8] are much more time consuming (days or weeks), require expert knowledge and tools on a wide range of topics. Some of them are a good understanding of filesystem properties, different cryptographic standards, and user management in operating systems. Therefore, it is highly advisable to rely on automatic, extendable analysis tools with a (small) group of experts able to add required features or firmware types as needed.

Firmware images are composed of a few common sections (Figure C-2).

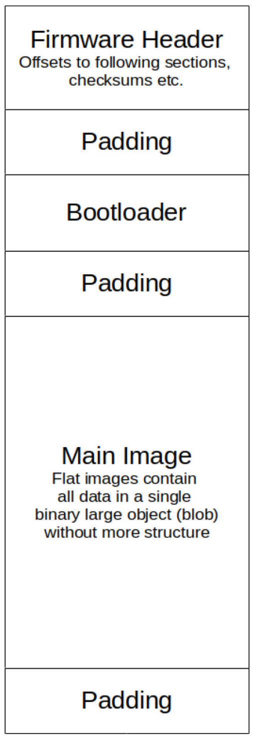

Firmware Header
Offsets to following sections, checksums etc.
Padding
Bootloader
Padding
Main Image
Flat images contain all data in a single binary large object (blob) without more structure
Padding

Figure C-2: Flat Firmware Image Sections.

A leading firmware header contains structural information about the firmware file and checksums to detected unintentional bit errors. Padding is used to align the following sections to certain positions. This simplifies the update process. The bootloader is the first executed software when the device starts. It starts the kernel and hands over control of the hardware to the OS. This process is very similar to standard office computers.

Flat firmware images compile the OS into a single file. This might also include non-executable data as images and text.

The firmware of more powerful devices often resembles standard office computers with OS components, applications, and file systems. Such images require a recursive approach to decompose the firmware into its parts (Figure C-3).

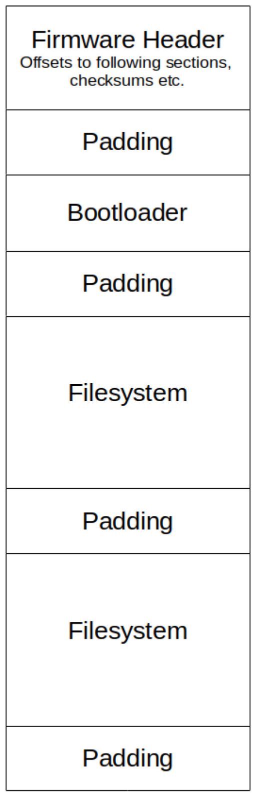


Figure C-3: Complex Firmware Image Sections.

C.4 REVERSE ENGINEERING

Reverse Engineering (RE) is the process of understanding how a system is build or put together. Such system can be a hardware device, a circuit or software, such as a program or a protocol. In simple words, RE is a problem-solving process with the final goal of obtaining as much knowledge as possible about the design and functionality of a system.

In the context of software, reverse engineering focus on understanding how a program was written, operates and behaves in the target environment. Typically, this process starts with an unknown system and works backward towards recreating all its useful functions. Common use cases are malware analysis and security analysis of closed source software. RE is also used to "crack" Digital Rights Management (DRM) or copy protections of software. The process of reverse engineering software can be as simple as observing behavior in a controlled test environment or as complex as decompiling a program and analyzing its components to learn how it was written – a process which is highly dependent on the original programming language (e.g., Java, C++).

RE, therefore, encompasses a wide range of interests and techniques. Fundamentally, RE may focus as well on exposing the underlying source code when it is not available. The recovered code can be used then for maintenance and improvement; creating additional interoperable modules or adding features to original software; comparing similar solutions or the competition; "cracking" software and media for bypassing security or copy protection in digital consumer products; detect, analyze and eliminate malicious code (malware) and much more.

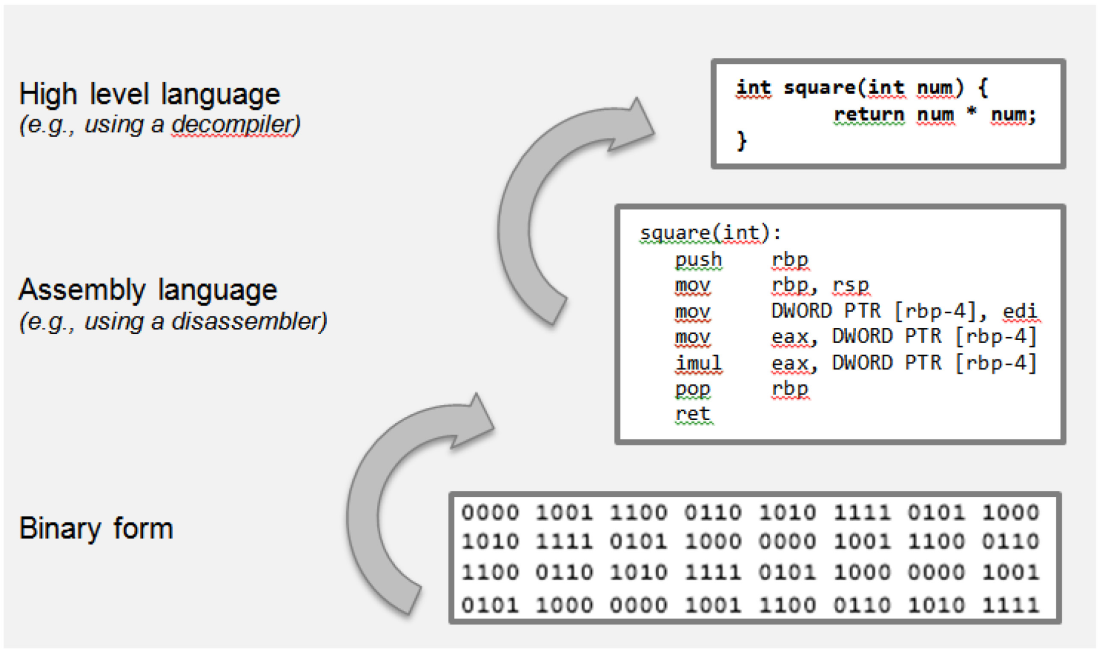
RE nowadays is considered part of a competitive intelligence field but practices of reverse engineering are not new and appear back in history in the form of military and commercial espionage. For instance, during the Second World War "Enigma" encryption machines were intercepted and reverse engineered to expose their weaknesses to consequently break their coded messages [9]. Another example was Chiasmus [10], originally a secret German government tool used for encryption but now leaked after performing RE and re-implementing its algorithm. Re-Implementations in the context of RE are not unusual – in fact, entire models have been derived from RE efforts, e.g., the Samba [11] and ReactOS [12] are well-known projects derived from the SMB networking protocol and Windows NT respectively.

C.4.1 RE Process

The most common tools used to reverse engineer – but not limited to only those – are debuggers, disassemblers, sandboxes, and network protocol analyzers (packet sniffers). Their essential role is to expose interesting data and reveal the precise functions and nitty-gritty of a program. In most scenarios, these functions are later reproduced without any previous knowledge and without access to the original source code.

Figure C-4 depicts the multiple representations of a software piece. During RE, an executable (code in binary form) gets translated into a representation of a higher-level language using a disassembler and/or a decompiler. The disassembler translates the binary code into an Assembly language while the decompiler translates into a high level language such as C. Translation to assembly is quite reliable but very hard to read and understand as even simple functions like printing a string are translated into multiple machine code instructions. Although decompiled code is easier to read as it is given in a high level language, it is far from typical source code quality. Compilers translating the original source code into the binary code optimize the given code. Optimizations include replacing variable names with simple place holders, inlining functions into the code, unrolling short loops into multiple executions of the same code and many more. Thus, the decompiled high level code cannot reconstruct the original source code. Therefore, expert knowledge is needed to read and understand assembly and high level code.

Furthermore, in most cases, artifacts available to abstract knowledge about a program are limited or deliberately hidden (obfuscated) to an analyst. For instance, many commercial programs and digital right products include protection against unauthorized access or software duplication. In such cases this translation is not feasible. Another example is malware that protect their code against detection engines (e.g., anti-virus) and reverse engineers.



```
int square(int num) {
        return num * num;
}
```

```
square(int):
   push    rbp
   mov     rbp, rsp
   mov     DWORD PTR [rbp-4], edi
   mov     eax, DWORD PTR [rbp-4]
   imul    eax, DWORD PTR [rbp-4]
   pop     rbp
   ret
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

Figure C-4: Example of Language Translation.

Protection can appear in many shapes and flavors such as custom file compression mechanisms (packers), modifying the original structure of the code while being executed, encryption or a combination of them; all with the same goal of avoiding the scrutiny of the program and therefore its analysis. Additionally, the process can be burdensome and time consuming requiring certain level of expertise and good understanding of low level components about architectures and operating systems. Fortunately, nowadays there are multiple reverse engineering methods and tools available that can be used not only to speed up the process but also sometimes to guide the analysis. Therefore, many of the RE tasks can be automated, either with new approaches or by incorporating additional features to existing tools.

Depending on the complexity of their target, and most significantly in the context of malware analysis, reverse engineers use a variety of system and network monitoring utilities to perform behavioral analysis in conjunction with examining static properties (the binary code) of suspicious programs [13].

Many of these tools would focus on different approaches that usually complement each other. For example, following the program control flow to understand decision points during execution. Sandboxes as well, like Cuckoo Sandbox will provide an analyst with full reports describing the behavior of files during their execution in a realistic isolated environment. Many of these tools can recognize common malware characteristics (e.g., registry manipulation, keylogging, network communication, etc.) based on the interaction with lower level components of the Operating System. Other alternatives may in turn work on static properties that aid on finding more specific security weaknesses based on the binary code. Disassemblers like the Interactive Disassembler (IDA) Pro [14], Binary Ninja [15], Radare2 [16] in conjunction with decompilers like the Hex-Rays [17] or the recently released NSA Ghidra [18] are some of the primary members in every reverse engineer's toolkit. Many of these tools offer additional APIs for instructing and automating analysis tasks. Fortunately, many of these extensions and newer approaches usually become open source projects which in turn make their contribution to the reverse engineering community.

C.4.2 Required Expertise

Although RE is a very active field with a huge arsenal of tools to improve the quality and speed of the work, it is still a field where expert knowledge is essential and it takes a very long time to master even basic RE. Therefore, RE experts are a rare resource and they can only process a very limited amount of data in a short period of time.

C.5 FUZZING

Fuzzing is a dynamic analysis technique to automatically find program errors. A fuzzer is a program that automatically feeds unexpected input into a target program and observes the reaction of it, i.e., if it crashes. From a security point of view, every program crash is at least a denial-of-service attack, often it may even be exploitable. Such exploits may allow attackers to gain further privileges or attack other systems. Fuzzing has gained a lot of media attention lately, partly because companies like Google and Microsoft comprise large fuzzing clusters to fuzz software products finding critical bugs.

C.5.1 Applications

There are two major application areas of fuzzing: first, finding bugs and hardening an application with access to the source code. This is a typical scenario for developers. Second, finding security issues in an application without access to the source code. This is a typical scenario for attackers or when security experts analyze closed source software.

C.5.1.1 General Application Hardening

Developers may utilize fuzzing to find crashes in their programs. This is not necessary as developers can analyze all possible inputs. However, edge cases may be overlooked but detected by a fuzzer. Therefore, fuzzing is a proactive method to increasing the stability of their programs and to mitigate security issues. The fuzzer AFL (see Section Fuzzing Software) is very popular with developers because it is easy to setup and use as well as very efficient. Therefore, many developers utilize AFL to harden their application.

C.5.1.2 Finding Security Issues

Fuzzing is very valuable to attackers. Today it is a standard technique of hackers, security researchers, and alike to find vulnerabilities in binary programs. Attackers utilize fuzzing because in most cases they can quickly set it up and let it run in the background. Furthermore, it is easy to scale, just add more servers. While a fuzzer stress-tests a target program, attackers can work with others analysis techniques on it.

C.5.2 Fuzzing Process

The fuzzing process is rather trivial. It can be summarized as a trial and error process, where an error is a crashing input. A fuzzer continuously feeds test cases into a target program in order to provoke a crashing condition. If the program crashes, then this is considered as success and a program error is found. Some of these program errors may be exploitable and therefore serious security vulnerabilities. From a high level point of view, we differentiate between the "test case creation", the "target program execution and observation", and an "optional feedback loop" to improve in the next iteration.

C.5.2.1 Test Case Creation

There are two ways how a fuzzer can create a new test case. First, it may generate one based on a grammar definition. The advantage is that this yields many close to valid inputs that are not directly discarded by the target program. The disadvantage is that a human expert has to create such a grammar definition. Second, the fuzzer may mutate known valid/non-valid inputs of the program. The advantage is that often these initial inputs are easy to create/gather. This can also be done by non-experts. The disadvantage is that initial test cases may exhibit a low code coverage and the fuzzer starts off very slowly.

C.5.2.2 Target Program Execution and Observation

For each test case the fuzzer hands the test case over to the program. Depending on the fuzzing target, it is restarted between each test case (e.g., userland program) or it continuously runs (e.g., operating system). This is a performance decision, since restarting a huge program is time consuming. However, if a fuzzer does not restart a program then this may lead to non-deterministic crashes because previous test cases may have already corrupted the internal state.

After the target program has been started, it processes the provided input. The fuzzer must observe the program somehow in order to detect crashes. In the worst case, the fuzzer has to treat the program as a black box without any knowledge about its internal state. Therefore, it can only detect a crash, for example, based on its return value or its availability. If the source code of the target program is available, compiler instrumentation can be used to extract information of the internal state, e.g., code coverage. Code coverage in particular is the most important metric in fuzzing.

C.5.2.3 Optional Feedback Loop

Especially in the case of mutation-based fuzzers, an optional feedback loop may help to increase performance. A metric like code coverage is required to measure how much of the target program a test case explores. The test cases that exhibit high values according to the metric are fed-back into the test case corpus and are further mutated. By doing so, the fuzzer steadily explores the target program and as a result it yields more bugs.

C.5.2.4 Fuzzing Limitations

As every dynamic analysis technique, fuzzing suffers from the problem of code coverage. It is important to measure and observe the code coverage. If a fuzzer hits an obstacle (e.g., a checksum) then human intervention is required to overcome it and dive deeper into the program. Even though there are academic approaches like Driller that enhance fuzzing with symbolic execution to find a solution to overcome an obstacle, their practicality is questionable.

Another disadvantage of fuzzing may be its long runtime. Sometimes it takes weeks to hit on a bug, which may be very expensive when considering the electricity that was needed to hit on this bug.

C.5.3 Discussion

The gateway hurdle is very low thanks to fuzzers like AFL (see Section Fuzzing Software). This software is easy to setup and easy to use. Hence, thousands have utilized AFL to fuzz their own or foreign software. Even though the gateway hurdle is very low due to accessible fuzzers, writing a custom fuzzer may be very complicated and requires expert knowledge. For instance, an expert is required to understand a specific part of a program to interact with it, i.e., fuzz it. This may be done with the help of reverse engineering.

Fuzzing benefits from hardware quality and quantity. The more computing power is available, the more executions can be achieved and the more likely are crashes. Therefore, it is recommended to run fuzzing jobs on as many powerful computing nodes as possible. However, fuzzing wears off hardware. Starting a program billions of times from a hard disk or running 24/7 with 100% CPU workload are just two reasons why fuzzing leads to early outages of hardware components.

C.5.4 Fuzzing Software

Due to the popularity of fuzzing, there are many fuzzers publicly available. In the following, we introduce three fuzzers that target three different target program types: parsers, network services, and operating systems.

C.5.4.1 AFL

AFL (http://lcamtuf.coredump.cx/afl) is a general application fuzzer for Unix-based systems. It can be considered as the program that brought fuzzing to the masses. This fuzzer is mutation-based, it starts with a corpus of test cases and tries to find program errors by mutating them. It incorporates a feedback-driven logic based on code coverage. If a test case increases the code coverage, then it is considered relevant and AFL continues to mutate it. Furthermore, AFL introduced many sophisticated optimizations to increase execution speed of a program.

AFL works with and without the source code of a target program. However, AFL is more efficient if the source code is available. It comes with a wrapper around the compiler's GCC as well as Clang and utilizes compiler instrumentation to measure, for example, the code coverage of a program. If the source code is not available, then the code coverage can be measured by emulating the program with the help of the emulator QEMU.

There are many forks of AFL due to its efficiency and popularity. These forks apply AFL to special cases such as system call fuzzing or fuzzing of Windows binaries.

C.5.4.2 BooFuzz

BooFuzz (https://github.com/jtpereyda/boofuzz) is a fuzzer for network services (clients and servers). It is grammar-based, which means that a security analyst must first specify the communication protocol. While this is very tedious and time consuming, especially in the case of binary programs, this greatly improves the efficiency of the fuzzing process. There are less nonsense inputs that are directly discarded by the target program. However, the whole endeavor stands and falls with the quality of the input grammar.

C.5.4.3 Syzkaller

Syzkaller (https://github.com/google/syzkaller) is an operating system fuzzer. It stress-tests system calls, which cross a trust boundary (userland to kernelland). Therefore, they are a high profile target. Syzkaller generates programs comprising of many systemcalls. It executes them and observes their code coverage. Similar to AFL, it mutates programs in a feedback loop based on the code coverage. Many operating systems such as Linux and FreeBSD continuously fuzz their source code with Syzkaller.

C.5.5 Alternatives

In the following, we present alternatives to fuzzing. Note that some of them could also be considered as complimentary approaches.

C.5.5.1 Static Analysis Tools

As code coverage is a major problem of fuzzing, static analysis tools are an alternative. They can inspect the whole code base of a target program and therefore they may find program errors in regions that are not trivial to explore with a fuzzer. Furthermore, such tools also detect bug classes like information leaks, on which typical fuzzers not focus.

C.5.5.2 Manual Code Audits

Manual code audits still yield to high profile vulnerabilities. This especially includes logical bugs, on which fuzzers typically not focus. In addition, software security experts may also find the same bugs as a fuzzer. However, these experts have high hourly rates and are a scarce resource.

C.5.5.3 Telemetry Data

A possible alternative if a product comprises a sufficiently large user base is to utilize telemetry data like crash reports to further improve the product's security or to detect ongoing exploitation campaigns. Since users can be very creative when utilizing an application, this may lead to special corner cases engineers may never have thought of. However, we advise to think about telemetry data not as a drop-in replacement for fuzzing but rather a complement.

C.6 REFERENCES

[1] NIST SP 800-115 NIST Special Publication 800-115, "Technical Guide to Information Security Testing and Assessment", September 2008.

[2] "OWASP Testing Guide, v.4.0", https://www.owasp.org/images/1/19/OTGv4.pdf, April 2013.

[3] PCI Data Security Standard (PCI DSS), "Penetration Testing Guidance, v.1.0", March 2015, https://www.pcisecuritystandards.org/documents/Penetration_Testing_Guidance_March_2015.pdf.

[4] Federal Office for Information Security, BSI Durchführungskonzept für Penetrationstest, "Study – A Penetration Testing Model", Federal Office for Information Security, Bonn, Germany. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration_pdf.pdf, Retrieved on July 2021.

[5] OpenVAS, https://www.openvas.org/, Retrieved on July 2021.

[6] Nessus, https://www.tenable.com/products/nessus, Retrieved on July 2021.

[7] Metasploit, https://www.metasploit.com/, Retrieved on July 2021.

[8] Binwalk, https://github.com/ReFirmLabs/binwalk, Retrieved on July 2021.

[9] Lycett, A. "Breaking Germany's Enigma Code". Online Resource: http://www.bbc.co.uk/history/worldwars/wwtwo/enigma_01.shtml, Retrieved on July 2021.

[10] Chiasmus™ for Windows/Linux. https://www.bsi.bund.de/EN/Topics/OtherTopics/Chiasmus/Chiasmus_node.html, Retrieved on July 2021.

[11] Samba, https://www.samba.org/, Retrieved on July 2021.

[12] ReactOS, https://reactos.org/, Retrieved on July 2021.

[13] Duchene, D., Le Guernic, C., Alata, E., Nicomette, V., Kaâniche, M. "State of the Art of Network Protocol Reverse Engineering Tools", Journal of Computer Virology and Hacking Techniques, Springer, 14 (1), 53-68, 2018.

[14] IDA Pro, https://www.hex-rays.com/products/ida/, Retrieved on July 2021.

[15] Binary Ninja, https://binary.ninja/, Retrieved on July 2021.

[16] Radare2, https://rada.re/n/, Retrieved on July 2021.

[17] Hex-Rays, https://www.hex-rays.com/products/decompiler/, Retrieved on July 2021.

[18] NSA Ghidra, https://www.nsa.gov/resources/everyone/ghidra/, Retrieved on July 2021.

REPORT DOCUMENTATION PAGE

1. Recipient's Reference	**2. Originator's References** STO-TR-IST-151 AC/323(IST-151)TP/1030	**3. Further Reference** ISBN 978-92-837-2350-9	**4. Security Classification of Document** PUBLIC RELEASE
5. Originator	Science and Technology Organization North Atlantic Treaty Organization BP 25, F-92201 Neuilly-sur-Seine Cedex, France		
6. Title	Cyber Security Risk Assessment Process for Military Systems		
7. Presented at/Sponsored by	This document presents a cyber security risk assessment process tailored to military systems.		
8. Author(s)/Editor(s)	Multiple	**9. Date**	October 2021
10. Author's/Editor's Address	Multiple	**11. Pages**	70
12. Distribution Statement	There are no restrictions on the distribution of this document. Information about the availability of this and other STO unclassified publications is given on the back cover.		
13. Keywords/Descriptors	Cyber security; Military platforms; Military systems; Risk assessment; Risk management		

14. Abstract

Military platforms are more computerized, networked and processor-driven than ever. The consequence is an increased exposure to cyber attacks and thus, an amplified risk. However, the continuous and stable operation of these platforms is critical to the success of military missions and public safety.

Military systems and platforms are targets of choice for cyber attacks not because of their prevalence, like consumer electronics, but because of their potential strategic impact. Once compromised, all sorts of short-term and long-term effects can be achieved, ranging from denying a capability to covertly reducing its effectiveness or efficiency, on demand. Therefore, military forces must address cyber security at all levels: strategic, while acquiring platforms and systems, operational, while planning military missions and tactical, while in operations.

Perfect cyber security does not exist. Cyber security must be continuously managed through iterative risk assessments. Many cyber security risk management frameworks and processes exist for traditional IT systems. However, this is far from being the case when it comes to military systems. This document presents a cyber security risk assessment process tailored to military systems. The process can be applied to both traditional IT and firmware-based embedded systems, which are everywhere in military platforms and systems.

ISBN 978-92-837-2350-9