# RETHINKING INTRUSION SOFTWARE

## Ideas for a more sustainable approach

Abstract

The Wassenaar Arrangement's first foray into export control of cybersecurity has created unintended consequences and implementation challenges that the IT sector seeks to undo. The Wassenaar Arrangement's first foray into export control of cybersecurity has created unintended consequences and implementation challenges that the IT sector seeks to address.

Cristin Flynn Goodwin, Anne Marie Griffin, Thomas Peltier & John Walton

## Contents

## Rethinking “Intrusion Software” Control and Regulation

In anticipation of additional technical discussions that Wassenaar Arrangement Participating States will be having on the “intrusion software” control, we offer these thoughts publicly to government policymakers engaged in those discussions and welcome engagement on this topic from the cybersecurity community worldwide.

In this paper, we offer views on how to refocus the current intrusion software export control debate to a more sustainable approach going forward. We then look at how the security community and software developers are impacted by the control, and offer a table of examples of products that may fall in scope of the intrusion software control. We then provide a narrower approach to controlling products aimed at intrusion, and provide recommended language for Participating States to consider prior to the September experts group meeting of Participating States.

We hope that with continued commitment from the Wassenaar participating states, we can evolve the intrusion software control over time to a narrowly tailored and well understood control that can help protect those involved in protecting the fight for human rights, and protecting our security online.

## Focusing on the Goal

The ongoing Wassenaar discussions about the intrusion software and technology controls are a very positive development. Wassenaar members should continue to discuss this control and work towards the ultimate goal of either substitution of the control in its entirety, or a significant modification of the definition of “intrusion software,” which may take several years to accomplish. At its core, the definition of intrusion software remains the primary challenge of this control, and its substitution is essential to creating a narrowly tailored and meaningful technology control.

As the Participating States contemplate changes to the current language in the “intrusion software” control in advance of continued negotiations, those involved in the discussion should work towards three goals:

1. A Publicly Articulated Problem. At present, the definition of “intrusion software” remains fatally overbroad. The goal of Participating States should be to articulate the underlying problem they are trying to solve publicly and ensure that a definition for a control meets that publicly articulated problem.

2. A New Control to Address the Articulated Problem. Once articulated, the Participating States should agree publicly to work with the security community to create a new control that is clearly understood, narrowly scoped, and implementable in a meaningful way by Participating States.

3. A Commitment to Transparency on Cybersecurity. As has been made clear by the outcry over the Intrusion Software control, the ability of the export control community to craft a cybersecurity control is significantly hampered without sustained input and partnership from the security community. This will require rethinking some of the classifications and security restrictions on discussions in this space. Going forward, Participating States should commit to a public vetting of cybersecurity controls prior to adoption to ensure accuracy, appropriate tailoring, and context.

Addressing these goals would eliminate the current challenges around the Intrusion Software control and would enable a more stable and robust process to address future concerns, as it is inevitable that they will arise. Implementing Participating States are either excepting out large swaths of industry in broad domestic exceptions (some of which have been made public, but many of which have been decided in a non-transparent manner to a handful of companies) or are not creating clear regimes to enforce its terms and are thus seeing either little or no licensing activity in relation to the export of these products and technologies. This lack of conformity begs the question as to whether the control fits the purpose for which it was proposed. Given the pervasiveness of technology and the need to ensure security practitioners have the flexibility and agility to respond quickly, we must rethink the current "intrusion software" definition and control approach.

## From the Perspective of Software Developers and Security Practitioners

As has been stated in comments on the Intrusion Software control, the challenge of the underlying definition of "intrusion software" is that it incorporates the very essence of software development – taking code that already exists for a purpose, and modifying it or integrating it in a way to make it do something different or new that was not intended or permitted when it was created. While "intrusion software" itself is not controlled, the definition is confusing and is central to the successful implementation of the control.[1]

Ideas can sometimes start small. Security practitioners focused on defence (the "Blue Team" to the attacker's "Red Team", in security parlance) may want to build some new attack tools to help defend against new classes of attacks. He or she may go to well-known software development platforms, like GitHub, and pull down some files that already provide obfuscation, or evasion, or modification techniques. Simply because that Blue Team'er or Red Team'er is thinking about new ideas by pulling code and building on it to test out a new idea for a defence or an attack tool that helps create a defence, that cobbling together of (or in export control parlance, "specially designing") a new software product based in part on existing code would be subject to controls. That garage-level innovation, the "what if" inquisitiveness we need to create the building blocks for tomorrow's defences, is chilled by the prospect of an export control review or licensing at each step of the "what if" process.

Large corporations are also sharing information internally, across teams and across borders. This type of security collaboration also happens around the globe and around the clock. Any transfer of technology internally of these types of technologies, that security teams may need to investigate an incident in a rapidly evolving environment, would also be subject to license in some nations that have already adopted the control.[2] Bug bounties – a growing and common tool that large companies often use to solicit sophisticated vulnerabilities – may also be subject to license.[3]

---

[1] See, e.g., "Intrusion software tools and export control," UK Department of Business, Innovation and Skills (BIS), available at http://blogs.bis.gov.uk/exportcontrol/files/2015/08/Intrusion-Software-Tools-and-Export-Control1.pdf (including the definition of "intrusion software" and providing examples of potential UK license obligations).

[2] Id. at 9.

[3] Id.

Let's be clear – this type of work happens constantly. Files are created and shared, and the security community encourages re-use and "recycling" of code or techniques to defeat protective countermeasures, modify the intended path of a file, or extract data. The security community's use and re-use of code and projects is intentional, and the same type of sharing occurs inside companies between teams looking to innovate and leverage work already done – with a wide range of license terms, or perhaps no license at all.

All of this organic sharing and exchange is a result of software engineers and innovators asking "what if?". If an export control review is required before that "what if?" conversation can continue, or if ideas are shelved because of the lack of clarity around the export control review process, the cascading impacts on security cannot be understated.

It may be tempting for a Participating State to say "that's not what we mean, we were only targeting a limited number of products." It's important to keep in mind that although these types of security innovations (or creating or sharing Blue Team, Red Team or "bad guy" tools) may not have been intended to be captured, they unambiguously meet the current definition and control groups. Many (if not most) small innovators and security researchers may not have the resources to seek export counsel, and are either under-reporting their obligations, unaware of those obligations, or waiting to see how governments enforce those obligations. Fixing the definition of "intrusion software" and narrowing the scope of the control will help remove that uncertainty and provided a more sustainable approach going forward.

## Identifying the Technologies, Approaches and Products Impacted

Microsoft strongly encourages the Wassenaar Participating States to reconsider the underlying definition of "intrusion software." If the current ambiguous term is left unchanged, governments seeking to impose the Intrusion Software control will need to ensure that appropriate groupings of activities or approaches, and their relevant technologies or products, are adequately excluded from any implementation. This will be a challenge, given that software engineering generally requires modification of a system, or the modification of a standard execution path of a program or process allowing for external instructions, in order to create new software. Re-use of code, either internally developed or in the open source space, will fall into the definition of controlled code, because monitoring tools or protective countermeasures are often broken in order to create new functionalities or to enable new innovations.

As we begin to identify tools and techniques that may be impacted by the control, it's important to keep in mind the context of how security researchers and responders work. Discovering, investigating, recovering and understanding common intrusions, or software used by intruders, requires security practioners to gather data and examine evidence of both malicious activity, malicious software as well as common software used by defenders and adversaries alike. A number of open source, proprietary and commercial tools exist for security and software developers, or open source (or even internal) code projects are assembled to help address an unmet need. Intrusion platform and defender software often share the same or similar techniques or capabilities but with opposing intentions – the defenders seeks to detect and respond to indicator(s) of an adversaries access, presence or malicious intent while the adversary seeks to hide, masquerade or disrupt evidence of their tools and activity.

The table below is designed to articulate multiple types and classes of work currently caught by the Intrusion Software control, which we have modified from a version provided in Microsoft's Comments to the US Department of Commerce Bureau of Import Security in 2015 to provide (1) examples of products or technologies that potentially fall under the control; and (2) additional activities that may remain impacted. Only governments can make a determination as to what is covered, or what a government may choose to exempt in an advisory opinion. It is important to keep in mind that new categories or techniques may emerge as cloud computing, virtualization and Internet of Things (IOT) continue to advance. Attacks against hypervisors may not use the same processes as those focusing on broader Operating System exploits. Tools or techniques used to enable an intrusion against an endpoint – for example, a home user's PC, may be different from those used to attack a hypervisor running a particular aspect of a cloud service, but containing no end user information.

Following the table, we provide further examples of products and capabilities impacted by the Intrusion Software control. We note that the examples listed are simply that, and are not intended to describe or endorse a particular product or imply its use inside Microsoft, except where we list a Microsoft tool, product or service.

## Table of Potentially Impacted Products, Technologies or Approaches

| Area | Description | Used For | Software Examples |
|---|---|---|---|
| **Penetration Testing** | Software created or used to evaluate and improve the security of services and software that many companies develop and operate. Includes proprietary software and open-source software that many companies specially design or modify for particular purposes. It may also involve controlled tools used for offense (i.e. Ethical Hacking). Can also simulate attacks to test internal detection and response capabilities. | Used to monitor internal systems, look for vulnerabilities, ensure compliance with security policies, and help protect systems. Companies also reverse engineer tools used by bad actors in order to protect customers. Also includes "Red team" toolsets for establishing and maintaining access (aka backdoor, process execution, Command and Control, data exfiltration, etc.). | Nmap, Burp Suite, Rapid7's Metasploit Pro, Immunity's Canvas, Core Impact, SysInternals, other internal / external tools[4] |
| **Malware Research** | Malware, exploit code, and reported vulnerabilities, including malware that meets | Microsoft performs extensive analysis on malware, including reverse engineering the code to identify how it was put together. Microsoft | Treasurehunt, Maltego, UltraEdit, OllyDbg, IDA Pro ADV, HexRays, Far, .NET Reflector, 010 Hex |

[4] See, e.g., "Penetration Testing Tools Cheat Sheet" for a list of penetration testing areas or scenarios that are relevant to the intrusion software space. https://highon.coffee/blog/penetration-testing-tools-cheat-sheet/

| | | | |
|---|---|---|---|
| | the definition of intrusion software. | also creates *new* code, including new intrusion software, to illustrate the risks of the particular malware or malware family. Can include sandbox or "detonation" technologies that are designed to watch program behavior to ascertain whether a running program is malware. | Editor, other internal / external tools |
| **Vulnerability Scanning** | Similar to penetration testing, private sector entities use both proprietary tools and open source tools that are specially designed or modified in response to specific intrusion software-related attacks. | Mitigating impacts of vulnerabilities, identifying new vulnerabilities, and enabling software engineers to reproduce and test software patches, updates, and upgrades. | Nessus, Qualys, Nexpose, WebInspect, other internal / external tools |
| **Incident Response, Network Security, Host Intrusion Prevention Systems (HIPS), and Forensic Analysis Tools** | Products or technologies used to help triage or respond to an incident, or help identify and remediate an attacker. This is a wide range of products, services and approaches that enable security responders to do their jobs. | Remote collection, network mapping, packet sniffers, log analysis, scanning, host intrusion prevention, forensics, etc. | Encase Suite, Access Data, XWays, F-Response, Slueth Kit, WireShark, Maltego, EMET, Sysinternals, other internal / external tools |
| **Sandboxing**[5] | Sandbox or "detonation" technologies designed to watch program behaviour to ascertain whether a running program is malware. | Because malware attempts to detect whether it is running in sandboxes, this common approach applies deception practices to *lie to the malware* about the environment it's in—almost rootkit like behavior to convince malware that our monitoring software isn't present. The intent is to avoid detection or monitoring by potential malware, not by | Office365 Advanced Threat Protection; Defender Advanced Threat Protection, internal / external tools |

[5] For more information on sandboxing, please see: https://msdn.microsoft.com/en-us/library/hh673561(v=vs.85).aspx.

| | | | |
|---|---|---|---|
| | | our protective network defences. | |
| **Security Engineering Tools** | This is a broader class of tools used in security, including debuggers, file fuzzers, and other automation used to support security. | Identifying vulnerabilities, modifying software to enhance operability or decreasing security risks, software development tools used to create elements of various tools into an integrated and new product. | Debugging tools for WinDbg, KD, CBD, NTSD, OllyDbg, Immunity Debugger, SDL MiniFuzz, Codenomicon, Peach Fuzzer, etc., this is a wide space with many different types of tools. |
| **Application Compatibility, Interoperability and Work-Arounds** | Many companies develop or deploy "shims" which are technology "work-arounds" to aid in the compatibility of software programs with an operating system. | Shims or work-arounds modify the intended function or path of a file in order to enable compatibility with other devices or interoperability with other software. Helpful when an old application needs to work with a new operating system or back-end system, for example. | Application Compatibility Toolkit, Compatibility mode capabilities, Detours, shims as compatibility mitigation[6] |
| **Information Sharing** | Receiving and sharing threat reports, vulnerability issues, and other security related issues on products and services and third party products and services in the computing ecosystem. Collaborating on planned and ad hoc issues that arise on security. | Incident response, mitigating vulnerabilities, investigating new issues, sharing information to help raise security awareness amongst others, and generally protecting the computing ecosystem. | Often on a case-by-case basis, includes proof of concept (POC) code or exploit code, and maybe a tool that proves the POC. Critical to incident response, can also include technical mitigation techniques. |

[66] https://technet.microsoft.com/en-us/library/dd837645(v=ws.10).aspx

| | | | |
|---|---|---|---|
| **Supporting Customers** | Security Consulting Services provides technical and other services on-site with customers around the world leveraging a wide range of internal and external tools and technologies. This involves they use of controlled forensics tools. | Used to investigate breach responses, conduct penetration tests, review software and security issues, and create recommendations on improving security. | Encase Suite, Access Data, XWays, F-Response, WireShark, Maltego, Core Impact, Burpsuite, PSExec, internal tools and other many other solutions (see Penetration testing, Security Engineering, and IR tools) |
| **Engaging the Security Community** | Working directly with security researchers, third-party companies, hosting competitions, participating in conferences, and engaging on difficult security issues to improve product and services security. | Includes sharing information, technology, tools, ideas, and collaboration; can include hosting "bug bashes" or awarding prizes,[7] paying for "bug bounties," publishing research,[8] attending conferences, and creating new tools, technologies, and tactics to improve security. | Pwn2Own competition, technical conferences and information exchanges, etc. |
| **Automated Exports and Re-Exports or Unintentional Re-Exports** | Automation is the future state of security and is continuing to change the security landscape. Machine to machine information sharing allows automation and machine learning to make adjustments without human interaction, although the information can move between company and non-company servers. | Companies are engaging in a growing use of automated software programs and custom developed tools, which can include software that automatically exports and reexports items; the draft does not contemplate machine to machine exports and re-exports.<br><br>If a government requires under its laws incident reporting of some kind, and that information is shared | Incident reporting obligations under the EU NIS Directive are still being formed but could be relevant in this space. Machine to machine export can also trigger this issue. |

[7] See, e.g., Microsoft's Blue Hat Prize: http://www.microsoft.com/securitv/bluehatprize/.

[8] "UK Student's Research a Wassenaar Casualty," Michael Mimoso, threatpost.com, July 6, 2015, available at: https://threatpost.com/uk-students-research-a-wassenaar-casualty/113625/ (highlighting a restricted portion of the student's dissertation on expanding bypasses for Microsoft's Enhanced Mitigation Experience Toolkit).

|  |  |  |  |
|---|---|---|---|
|  |  | with another government, a re-export could occur. |  |

Many of the tools and techniques listed above cross multiple categories, while others do not fit neatly into one particular area but could, if modified or used in a particular way (as they often are), be swept in under the definition of "intrusion software". Thus while the original purpose may not meet the intrusion software definition of "specifically created" when combined together the question of intent becomes less clear, and the likelihood of use in scenarios for which the Participating States would seek control could be present. Examples of these types of software (and other tooling security or defensive areas) are easily found on common code sharing platforms, such as GitHub or sectools.org.

## Solving the Problem

The definition of "intrusion software" remains overbroad, and any implementation will either result in massive exceptions or confusing and not-well-understood regimes. As noted above, that stems in large part from the foundation – the definition of intrusion software as currently written is fatally flawed. If we are focused on the issue of protecting human rights workers and civil advocates from software that allows a third party to conduct surveillance, extract information, or install or change software for the purpose of enabling those types of behaviors, starting with a more accurate definition is critical to success.

### Intrusion Delivery Platforms

When an attacker targets an individual, the attacker needs to find a way to get a foothold into the victim's technology. That can be done via a number of ways, but the point isn't that the intrusion is done via software (of course it is!) – the point is that the target is identified, there is intent to commit an attack against an un-consenting victim, the intrusion is delivered, and it is delivered by a tool or platform that enables the intrusion to be successful.

Microsoft believes that by scoping the issue to address the problem that has been articulated publicly – attacks against human rights workers and defenders of civil society – to identify the most common attributes of the means of attack is a smarter approach to regulation and licensing of those technologies. Microsoft has introduced the idea of requiring licenses for "intrusion delivery platforms" which we define as software that meets each of the following three criteria:

1. **Exploits a process to obtain access to a system**
    a. Includes exploits for vulnerabilities for which a patch, update or mitigation is not widely available in the public domain; or
    b. Includes executables to obtain access to a system; or
    c. Includes software that exploits vulnerabilities in any cryptographic algorithm or intentionally weakens the cryptographic implementation on the target device or system; and

2. **Exhibits evasion capabilities**
    a. Includes anti-disassembly technical mechanisms to prevent reverse engineering or to avoid protective countermeasures; and
3. **Enables subversion or destruction**
    a. Is capable of enabling or re-enabling access on a target device or system without authorization; or
    b. Includes software which irretrievably destroys the functionality of the device or system without consent of the owner of the device or system.

In order to be specific about the use cases and applicability of Intrusion Delivery Platforms, it is necessary to detail certain exceptions which would not require a license. Many of the items listed below are common technologies or security tools that enable engineers, consultants and responders to continue their work unabated. The following could be workable standard exceptions to any licensing obligation for Intrusion Delivery Platforms:

- Non-commercial use or sales of Intrusion Delivery Platforms under $1m USD
- Hypervisors, debuggers or Software Reverse Engineering (SRE) tools;
- Sweep patch validation or assessment tools;
- Digital Rights Management (DRM) ‘‘software’’;
- Software or code designed to be installed by manufacturers, administrators or users, for the purposes of asset tracking, incident response, and recovery;
- Capabilities developed for and/or used, operated, or installed for the purposes of:
    - Adding to or modifying the security or functionality of systems, equipment, components, software, or technology;
    - Adding to, testing, or modifying security or functionality of any hardware, software or technology where informed consent with knowledge has been given;
    - Securing data, systems, and networks;
    - Creating reports and analyses for customers for the purpose of remediating security issues
- Network-capable devices including mobile devices and smart meters.

The non-commercial use or sales under a particular dollar amount is an important exception. Given the large number of security researchers building tools from GitHub, SourceForge, and other project repositories, a non-commercial exception ensures that researchers and initial ideas can be shared and shaped unabated. While the $1 million USD exception limit can be subject to debate, the point in setting the bar at that level is to help parse out the delta between smaller purchases and large government deals for the purpose of buying intrusion delivery platforms used against large numbers of human rights or civil society advocates. Government experts in criminal law and organized crime should be consulted on the appropriate level of transaction costs that tip from the hobbyist, researcher, or legitimate user to a broader use of intrusions delivered by a platform enabled for causing harm.

In support of the above language, we also propose the following definitions, for context and discussion purposes:

- ‘Offensive intrusion’ is the capability of compromising or circumventing security features, and obtaining access to systems, equipment, component or software without consent.
- ‘Monitoring tools’ are ‘‘software’’ or hardware devices, that monitor system behaviors or processes running on a device. This includes antivirus (AV) products, end point security products, Personal Security Products (PSP), Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS) or firewalls.
- ‘Protective countermeasures’ are the techniques designed to ensure the safe execution of code, including but not limited to Data Execution Prevention (DEP), Address Space Layout Randomization (ASLR), or sandboxing.

## Conclusion

In order to create clear use cases and exceptions that can be understood and implemented by the security community worldwide, Microsoft recommends that the Participating States revisit the scope of the original control and the definition of “intrusion software.” Given the interconnectedness of technologies and license obligations that can be created when software engineers and security experts are experimenting, building, and defending, an exception-based list of software and technology will quickly become unmanageable and outdated. As the Internet of Things and machine learning continue to advance our ability to innovate, our first foray into cybersecurity licensing should be discrete and specific. The Wassenaar Participating States have the opportunity to remedy the intrusion software situation and start anew, consistent with the three goals we outline above. We remain ready to engage in that dialogue.

# Appendix – Language for Use in Negotiation

## Substitute Definition for Intrusion Software – Intrusion Delivery Platforms

An intrusion delivery platform is defined as software that meets the following three criteria:

1. **Exploits a process to obtain access to a system**

    a. Includes exploits for vulnerabilities for which a patch, update or mitigation is not widely available in the public domain; or

    b. Includes executables to obtain access to a system; or

    c. Includes software that exploits vulnerabilities in any cryptographic algorithm or intentionally weakens the cryptographic implementation on the target device or system; and

2. **Exhibits evasion capabilities**

    b. Includes anti-disassembly technical mechanisms to prevent reverse engineering or to avoid protective countermeasures; and

3. **Enables subversion or destruction**

    c. Is capable of enabling or re-enabling access on a target device or system without authorization; or

    d. Includes software which irretrievably destroys the functionality of the device or system without consent of the owner of the device or system.

## Exceptions

- Non-commercial use or sales of Intrusion Delivery Platforms under $1m USD
- Hypervisors, debuggers or Software Reverse Engineering (SRE) tools;
- Sweep patch validation or assessment tools;
- Digital Rights Management (DRM) ‘‘software’’;
- Software or code designed to be installed by manufacturers, administrators or users, for the purposes of asset tracking, incident response, and recovery;
- Capabilities developed for and/or used, operated, or installed for the purposes of:
    - Adding to or modifying the security or functionality of systems, equipment, components, software, or technology;
    - Adding to, testing, or modifying security or functionality of any hardware, software or technology where informed consent with knowledge has been given;

    - Securing data, systems, and networks;
    - Creating reports and analyses for customers for the purpose of remediating security issues
- Network-capable devices including mobile devices and smart meters.

## Definitions

- ‘Offensive intrusion’ is the capability of compromising or circumventing security features, and obtaining access to systems, equipment, component or software without consent.
- ‘Monitoring tools’ are ‘‘software’’ or hardware devices, that monitor system behaviors or processes running on a device. This includes antivirus (AV) products, end point security products, Personal Security Products (PSP), Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS) or firewalls.
- ‘Protective countermeasures’ are the techniques designed to ensure the safe execution of code, including but not limited to Data Execution Prevention (DEP), Address Space Layout Randomization (ASLR), or sandboxing.