Sergey Bratus
6211 Sudikoff Laboratory
Dartmouth College
Hanover, NH 03755
Ph.: (603) 646-9224

July 20, 2015

Dear Sir/Madam,

I am writing to provide public comment on the BIS proposed rules regarding the Wassenaar Arrangement's "intrusion software" and related items,[1] and BIS' subsequent FAQ on the subject.[2]

Executive summary: As an academic security researcher with strong ties to industry, I believe that the proposed rules will directly affect my research activities. As a cybersecurity expert and educator, I believe that the language of the proposed rule is overbroad, vague, and *will damage US cyber-defense* companies and researchers. The attempted clarifications in the BIS FAQ unfortunately resolve neither the excessive breadth nor ambiguity of the language. The use of industry jargon terms "zero-day exploits" and "rootkits"—which are meaningless outside of the context of professional conversation—further complicates the matter. I respectfully urge the BIS to *revise* this language, and to *allow additional time* for comment from affected industry, to establish narrow and unambiguous definitions compatible with Wassenaar's intent.

In the following I first describe the effects of the proposed rules on my security research activities, then comment on the general issues with the rules and the FAQ.

0. My professional activities. I am a Research Associate Professor at Dartmouth College, and Chief Security Technology Advisor to Dartmouth's Institute for Security, Technology, and Society. My focus is on practical cyber-security, both defensive and offensive. In 2014 I served as a co-chair of the USENIX Workshop on Offensive Technologies, the leading forum dedicated to offensive cybersecurity research that brings together industry and academia. Since 2007, I gave over 50 talks at security industry conferences, including several keynotes.

I lead, advise, and mentor cybersecurity research by graduate students, many of whom are foreign nationals.

1. "Deemed export" rules threaten our research communications.

Our research requires close communications between myself, graduate student researchers I lead, and industry. The subject of these communications, typically undertaken under NDAs, is technical information (research) "on the vulnerabilities and exploitation of computers and network-capable devices". It includes technical information about "zero-day exploit capabilities" and "rootkit capabilities". Moreover, an NDA on the specific details and results of vulnerability research is often the necessary condition for our interactions with a vendor. For example, our work on security of "smart grid" devices would have been impossible without an NDA, because we would not have been able to access these devices.

Proposed rules that control "proprietary research on the vulnerabilities and exploitation of computers and network-capable devices" would thus control our private research communications, and would place a heavy burden on industry researchers and vendors, complicating and potentially cutting off our access to necessary technical information.

We note that a similar problem exists on the industry side: often, specific results of vulnerability and exploitation research are only communicated to the client or vendor, and are not published in open venues, thus lacking publishing exemption.

[1] `https://www.federalregister.gov/articles/2015/05/20/2015-11642/wassenaar-arrangement-2013-plenary-agreements-implementation-intrusion-and-surveillance-items`

[2] `https://www.bis.doc.gov/index.php/policy-guidance/faqs`

2. "Presumptive denial for items that have or support rootkit or zero-day exploit capabilities" undermines our defensive research.

Our defensive research relies on collecting technical information (offensive research) about "non-standard execution paths" of software and hardware, and about ways such deviation is achieved and controlled. We then generalize this technical information items to derive general models for offensive computation, and develop protective measures against these generalized threats. Generalization is the crucial step, because it creates the capability to protect against broad classes of attacks, not just particular samples of malware.

To achieve productive generalization, we must work with detailed analysis results and descriptions of attacks, which create "capabilities" for these "exploits" or "rootkits". Mere receipt of samples or proof-of-concept code not accompanied by analysis will place the burden of such analysis on us (as well as unproductively and unjustifiably duplicating effort). Indeed, to avoid such inefficiencies, the majority of researcher communications traditionally goes beyond proof-of-concept and includes analysis and tools to create the exploitation capability; we have produced and distributed such tools with our publications and received them from other researchers before these were publicly available.

Vague language controlling such tools and technical information that may be deemed to create an "exploit capability" or "rootkit capability" will create a substantial burden on our research.

3. Controlling tools specially designed to "communicate with intrusion software" threatens our research and industry training activities.

Although "intrusion software" itself is not controlled, technology "specifically designed" to "communicate" with it is controlled (as per original Wassenaar language and the proposed BIS rules). As we make frequent use of such software, as specified below, our research will be impacted.

Unlike the previous items, which pertain to the BIS-specific rules, these controls originate in the Wassenaar language, which BIS rules incorporate and do not restrict. This language—apparently unintentionally—captures software such as (1) malware sandboxes, (2) threat intelligence software designed to develop such sandboxes, (3) intentionally vulnerable test software with weak protection mechanisms to test the skill of attackers, and (4) intentionally vulnerable software to evaluate students. In particular:

1. Malware sandboxes are containers specially designed to communicate with intrusion software. By running captured intrusion software in a sandbox and communicating with it, sandboxes allow for deeper analysis of malware and the gathering of threat indicators. Malware sandboxes occur in the FireEye defensive appliance, most antivirus engines, the Cuckoo Sandbox, and a litany of defensive software projects. Malware sandboxes would be controlled as "software or system specially designed to generate, operate, deliver, or communicate with intrusion software".
2. Information Sharing and Threat Intelligence software is specially designed to develop effective malware sandboxes (1). By providing Threat Indicators to malware sandboxes (such as malware decryption keys, command and control sequences, etc.), these special design features allow for the distributed gathering and sharing of threat intelligence. Such software would be controlled as "specially designed or modified for the development" of (1).

3-4. Both cybersecurity competitions[3] and security training courses use intentionally vulnerable test software with weak protection mechanisms to test the skill of competitors and evaluate students. An example is the Hacksys "Extreme Vulnerable Driver", a system component filled with many known security mistakes used for educational purposes.[4] These intentionally vulnerable software samples (challenges) are operated on test systems for training purposes and do not guard the privacy or data of any person. Training material and contest participant submissions defeat the intentionally weak protections and deliver instructions or modify data, making them intrusion software. Thus, cybersecurity competitions and training courses must communicate with, generate, and deliver intrusion software, and would be controlled as (1).

[3] http://www.theguardian.com/technology/2014/jan/30/top-uk-hackers-compete-cybersecurity-challenge-gchq

[4] http://www.payatu.com/hacksys-extreme-vulnerable-driver/

4. BIS FAQ process does not remove ambiguity and excessive breadth from the proposed controls.

Some of the above concerns were intended to be mitigated by the BIS FAQ process. However, as I demonstrate in the attached notes, these answers unfortunately do not remove the ambiguity. Apparent contradictions in these notes demonstrate that the underlying language needs substantial revisions and narrowing to avoid negative impacts on US cyber-defense companies and researchers.

Best regards,

Sergey Bratus, Ph.D.
Research Associate Professor
Department of Computer Science
Dartmouth College

Notes on the FAQ for the BIS proposed Wassenaar rules

Sergey Bratus, Anna Shubina

These notes cover the BIS FAQ published at `https://www.bis.doc.gov/index.php/policy-guidance/faqs` as of July 2015. In these notes, selected FAQ questions are followed by the indented block quotes of the relevant answering text, and then by the notes proper.

Q1. Does the rule BIS is proposing control "intrusion software", malware, exploits, etc.?

> No, the proposed rule would not control any "intrusion software," which may also be referred to as malware or exploits. The Category 4 control entries would control the command and delivery platforms for generating, operating, delivering, and communicating with "intrusion software." It would also control the technology for developing "intrusion software," but it does not control the "intrusion software" itself.
>
> Thus, transferring or exporting exploit samples, exploit proof of concepts, or other forms of malware would not be included in the new control list entries and would not require a license under the proposed rule.

This separation would appear to protect researcher communication, under the model in which this communication occurs primarily by transferring "samples" or malware specimens. This model is, however, factually wrong, because such sample transfer—although indeed necessary for effective progress of security—is by far not sufficient.

To understand malware, researchers need to exchange not just "samples," but insights into what makes these "samples" of unexpected computation work, the descriptions of *mechanisms* that make these computations possible, and methods for generalizing and finding them. Since the classic "Smashing the stack for fun and profit" paper—by now an obligatory citation in any academic work on exploitation—researcher communications included detailed descriptions of unexpected uses of computing systems and mechanisms, not mere proof-of-concept code for discussed phenomena but methods of creating more such code for a variety of related phenomena. To claim otherwise is to ignore prevailing practice in both the industry and the academic security research.

Yet this additional technical information that has driven the progress of security since at least 1990s is apparently **not** protected in the BIS rules, judging by the "presumptive denial for items that have or support rootkit or zero-day exploit capabilities." An in-depth technical description inherent in high-quality researcher communication—the description of the previously unknown mechanisms behind the unexpected execution paths and scenarios—is the knowledge that *creates* "zero-day exploit capabilities," in the same way as understanding the algebraic formula for the quadratic equation creates the ability to solve such equations.

BIS rules may aim to protect researcher communications, but in fact will stifle them. Without detailed technical information, effective security knowledge transfer is impossible, just like the ability to communicate with prominent foreign researchers is impeded.

Q2. Doesn't the rule potentially criminalize hacking?

> No. The rule would control the export of hardware and software delivery tools, as well as the export of technical data for developing exploits ("intrusion software"). The rule as proposed would not control the export of exploits to a target system since "intrusion software" would not be controlled.

This is not correct. The essence of hacking is not mere manipulation of some software or computing devices in unexpected ways; it is also automation of such manipulation. In fact, the most highly regarded

hacker presentations, which confer on their authors authority in the field, deal with automation tools and frameworks—which pass under the control definitions.

Thus, if public presentations at the leading InfoSec conferences such as Defcon, BlackHat, etc. are representative of private hacking activities, then BIS rules would potentially criminalize these activities.

For example, key security research and penetration testing tools such as Metasploit, Nmap, and Scapy (to name a few free tools) are actually frameworks that offer ways to automate "intrusion software"-related tasks with scripts that generate the necessary execution path modification and/or countermeasure avoidance payloads ("intrusion software") on the fly, with a use of a scripting framework. The automation framework architecture for such tools has been pioneered in industry research by companies such as Immunity and Core Security Technologies in their proprietary tools; eventually, this pioneering functionality became available and standard in free tools, expanding the capabilities of researchers and pentesters. Under the proposed BIS rules, however, it would be controlled and its development would likely be stymied.

Q3. Does the rule inadvertently capture defensive products as well as offensive products?

> The proposed rule would control the command and delivery platforms "specially designed" or modified for generating, operating, delivering, or communicating with "intrusion software" [...] BIS is not aware of other defensive products [besides penetration testing products] that would be caught by the proposed rule, but would welcome comments on this.

A variety of defensive tools are "specially designed" to "deliver and communicate" "intrusion software", and also to "generate" it. These include but are not limited to: (1) sandboxes for malware analysis, (2) rescue tools for systems compromised by malware and ransomware such as crypto-lockers, and (3) tools that automate vulnerability finding by generating exploits for the target software. Examples follow.

1. Malware sandboxes are containers specially designed to communicate with intrusion software. By running captured intrusion software in a sandbox and communicating with it, sandboxes allow for deeper analysis of malware and the gathering of Threat Indicators that enable Information Sharing. Malware sandboxes occur in the Fireeye defensive appliance, most antivirus engines, the Cuckoo Sandbox, and a litany of defensive software projects.

Malware sandboxes would be controlled as software specifically designed to communicate with "intrusion software" such as malware.

The same applies to other virtualization test environments for "intrusion software", as well as emulators and operating systems modules designed for communicating with "intrusion software." These tools are essential for capturing and analyzing malware, as well as for testing of defensive measures.

2. "Intrusion software", including "zero-day exploits", and accompanying software to deliver it and to communicate with it can be released for defensive and/or forensic purposes. For instance, Cisco recently released a zero-day exploit in the Teslacrypt ransomware (http://blogs.cisco.com/security/talos/teslacrypt), which allows users to defeat the protections of the computer running Teslacrypt, extract the decryption key, and recover their files. Zero-day exploits that defeat the intended execution path of malware include virus/rootkit uninstallers, ransomware key recovery tools, and other such software (e.g., `http://www.malware.lu/assets/files/articles/RAP002_APT1_Technical_backstage/1.0.pdf`)

These tools are classic exploits and meet every checklist item in the intrusion software definition. Any software that is specially designed to generate, operate, deliver, or communicate with these tools appears to be controlled. This includes but is not limited to most adware removal suites, networked ransomware remediation, and some forms of antivirus that allow web-only operation.

3. In recent years, major advances have been made in automating the finding of vulnerabilities with tools and methodologies that generate the proof-of-concept exploits (i.e., "intrusion software") for the vulnerable software. The value of these tools to software vendors and software owners is in the proof-positive that the vulnerability in question is exploitable, and is not already mitigated by existing security measures. Research in this promising field of Automatic Exploit Generation (AEG) is carried out both in academia (e.g., Prof. David Brumley's group at CMU) and in industry, from industry giants such at Microsoft to a number of startup companies. Industry research is primarily proprietary and can only be shared with academia under Non-Disclosure Agreements (NDAs).

Since these research tools generate "intrusion software", they will be controlled under the proposed BIS rules. These rules would effectively stifle the development of AEG tools and methodologies by startups that

must rely on NDAs to protect their research investment.

These types of defensive software do not by far exhaust the range of defensive tools captured by the proposed rules. The reason for this lies in the tremendous breadth of the current definitions. In modern software engineering, progress is made by automating computing tasks and automatically generating critical parts of software. A control on automation ("generation" of and "communication with") effectively freezes progress in software engineering by severely reducing the scale at which analysis can be applied. Since defenders succeed by operating at large organization-wide or Internet scales, and the proposed rule would impede scaling, the rule would actually favor attackers rather than defenders.

Q4. Will the rule control vulnerability research as well as research on exploits?

> The rule would control the export of technology for the "development" of "intrusion software", as well as the technology for the "development" or "production" of the command and delivery platforms themselves. A license would not be required simply to conduct research or analyze code, unless there was an associated transfer or deemed export of controlled technology, executable software, or source code.
>
> As a clarification, the proposed rule would control the following, among other things:
>
> Information "required for" developing, testing, refining, and evaluating "intrusion software", in order, for example, technical data to create a controllable exploit that can reliably and predictably defeat protective countermeasures and extract information. Information on how to prepare the exploit for delivery or integrate it into a command and delivery platform. The development or production of the command and delivery platform itself.

The proposed rule would significantly devalue the outcomes of non-controlled vulnerability research for vendors. Specifically, vulnerabilities that *cannot* be reliably exploited in the presence of modern protective countermeasures such as DEP, ASLR, EMET, etc., are not a high priority for software vendors, OS vendors, and asset owners. These vulnerabilities are already effectively mitigated by the investment into the deployed protective measures (which include significant investments into functional testing of software under these measures).

The only research findings that deliver value to vendors and asset owners are those of vulnerabilities and weaknesses that can still be reliably exploited despite the protective measures. Thus the proposed rules will control the kinds of research that is most relevant and of most value to the stakeholders.

> The proposed rule would not control the following:
>
> Information on how to search for, discover or identify a vulnerability in a system, including vulnerability scanning; Information about the vulnerability, including causes of the vulnerability; and Information on testing the vulnerability, including fuzzing or otherwise trying different inputs to determine what happens; and Information on analyzing the execution or functionality of programs and processes running on a computer, including decompiling or disassembling code and dumping memory.

In this, the rules appear to contradict the "policy of presumptive denial for items that have or support rootkit or zero-day exploit capabilities." The detailed technical information items on the causes of a previously unknown vulnerability (a "zero-day") is exactly the knowledge that creates the capability to exploit it; a "zero-day exploit" is the means of testing for the existence of such vulnerability.

> In addition, there are two further limitations to controls on technology:
>
> First, not all malware and exploits meet the definition of "intrusion software." The definition specifies only intrusion software that is capable of extracting or modifying data or modifying the standard execution path of software in order to allow the execution of externally provided instructions. Thus, technology for the development of malware that is designed to do other things, such as damage or destroy systems or infrastructure, would not be controlled under the proposed rule.

This proposed rule unfortunately captures the absolute majority of proof-of-concept exploits that demonstrate the existence of a vulnerability. Specifically, the point of a proof-of-concept is to demonstrate that unexpected computation happens in the vulnerable software. It proves so by diverting the execution from its standard path towards an unexpected result, which is typically a "root shell" program in Linux and the Windows Calculator under Windows. Neither are on the standard path (as the authors of the original program do not intend that these programs be invoked on the standard path), and both consist of instructions external to the vulnerable program and to the exploit.

Thus the rule as written captures each and every traditional proof-of-concept on Linux and Windows as "intrusion software," and, consequently, any tools for producing those as controlled items.

> Second, the only technology controlled is the technology that is "required for" and peculiarly responsible for achieving or exceeding the control level. See BISs 3/25/14 advisory opinion at http://www.bis.doc.gov/index.php/policy-guidance/advisory-opinions.
>
> Third, export controls do not apply to any technology or software that is "published" or otherwise made publicly available.
>
> Thus, only that part of the technology that is peculiarly responsible for meeting the definition of "intrusion software" , and is not publicly available, would be controlled.

The "publishing" exception, although a step in the right direction, misses the primary revenue source for the leading top-tier industry conferences such as BlackHat, Recon, Hack-in-the-Box, and others, namely, private trainings offered by conference speakers to paying attendees.

These trainings communicate specific technical information in one-to-several days of full-day sessions, to the extent impossible to cover during a hour-long conference presentation. The materials of these trainings are typically not publicly released, and would not be covered by academic educational materials exceptions. Controls on these materials would preclude foreign researchers, students, and employees from attending these trainings, and would put the burden on avoiding "deemed exports" on both the trainers and the conference organizers.

Q5. Doesnt the rule expose researchers to criminal prosecution if they carry information on exploits to a public conference, unless they publish it before the conference?

> Under Section 734.7 of the EAR, information that is published, or released at an open conference, is not subject to the EAR. That section also specifies that it would not be an export to transfer the technical data to conference organizers with the intent that it will be published at the conference.

Security industry conference trainings in the formats as described above—with additional information imparted to a much smaller group of paying attendees—may still subject the trainers to criminal prosecution, if the imparted technical information is not deemed to be covered in the publicly presented talk and slides.

As such, this rule may criminalize the key activity that sustains the security industry conference ecosystem. A brief examination of the leading industry conference agendas will demonstrate this point.

It should be noted that the trainings are typically offered by the conference speakers on the same topic of their expertise as their actual conference presentations; however, they contain more detail and allow the attending industry professionals new to the topic—the absolute majority of them are cyber-defenders—to catch up with the technical level of the public presentations. The proposed rules may criminalize this critical knowledge transfer.

Q8. Does the rule capture auto-updaters and anti-virus software?

> No. Software that permits automatic updates and anti-virus tools are not described in proposed ECCN 4D004. ECCN 4D004 software must be specially designed or modified for the generation, operation or delivery of, of communication with, "intrusion software," which is separately defined. Anti-virus software is a monitoring tool that is explicitly excluded from the definition of "intrusion software. Further, software that automatically updates itself may need to interact with installed monitoring tools" and protective countermeasures in order to properly execute, but they are not defeating (or otherwise subverting) the system or generating, operating, delivering, or communicating with "intrusion software."

This rule captures third-party update tools for software that does not include a vendor-native update capability. This class of software includes mission-critical software that is no longer supported by its original vendor (mission-critical software operates assets such as industrial control systems that age more slowly than typical software, and may span twenty years or more for power and other utility systems, as well as financial systems). These third-party updates technically violate the integrity of the original software, protected by a variety of protective measures that the update must defeat to succeed.

These third-party update tools must modify the "standard execution paths" of the updated software with newly supplied "external instruction"; for this purpose, they must bypass any integrity protective measures that the software being updated may employ. This squarely puts such third-party update tools into the controlled category, as the means of delivering "intrusion software" that accomplishes the update.

For example, the first commercially successful Personal Firewall for Windows, the defensive tool BlackIce, installed itself by patching the Windows kernel and diverting relevant execution paths to its own logic instructions, external to the Windows kernel and to Microsoft. (`http://blog.erratasec.com/2013/03/the-debate-over-evil-code.html`) The installation program from the BlackIce core—the execution path-altering "intrusion software"—would be controlled under the proposed BIS rules.

Q10. If an IT security researcher had done an analysis on a software application to find a vulnerability in the code, had written up code to then take advantage of the vulnerability and then sent that code to an anti-virus company or the software manufacturer, would that code require an export license?

> No, what is described is the creation of an "exploit." Exploits are not described in the text of the proposed control list entries. The code that takes advantage of the vulnerability would not require a license. As stated above, "intrusion software" itself would not be controlled by the proposed rule.
>
> For any associated technology for the "development" of "intrusion software," under section 734.7 of the EAR, any technical data sent to an anti-virus company or software manufacturer with the understanding that the information will be made publicly available, would not be subject to the EAR. However, "technology" that is not intended to be published would be subject to the control see question #4.

Although the proof-of-concept code for an "exploit" is not controlled, this approach places a burden on the anti-virus vendor or the original software vendor to recover the technical information necessary for the understanding of the exploit mechanism, its replication outside of the researcher's limited environment, and the scope of the overall problem. Thus under the proposed rules the vendor must bear the costs of re-discovering the "associated technology" and "technical information" for the development of the exploit. This disadvantages both the vendor and the software asset owners.

Moreover, it is in the vendor's and software owners' interests—especially when the software is mission-critical such as SCADA and ICS software—that the "technology" of the exploit does not reach public circulation, as as the vendor's or the owners' capability for mitigating the vulnerability in question may be technically limited. Accordingly, many disclosures are never made public.

It should be noted that lacking the general description of the underlying exploit mechanism and thus the means of "developing" other exploits of the same vulnerability class, a vendor typically lacks the skill to fix the root cause of the problem, and instead issues a patch that merely breaks the proof-of-concept code, without actually addressing the actual larger vulnerability.

The MITRE's Common Vulnerability Enumeration (CVEs) contains a history of amendments for multiple bugs, in which the vendor initially failed to correctly estimate the impact of a vulnerability, rated it as low-impact, and was forced to upgrade this rating all the way to critical over a number of iterations with the security researcher. In these cases, the researchers were not discouraged from sharing further technical details and technologies with the vendor, unlike under the proposed rules.

Q11. The first FAQ basically equates "intrusion software" with malware and exploits. Is this the intent?

> The definition of "intrusion software" is meant to include a subset of all the malware (exploits, viruses, etc.) that are out there. The definition describes only "intrusion software" that is specially designed to extract or modify data or modify the standard execution path of software in

order to allow the execution of externally provided instructions. Other types of malware, including software that only leaves evidence of a successful security breach without further compromising or controlled the system, or is designed to destroy data or systems would not be included in the definition of "intrusion software."

This statement is self-contradictory. The only way for an exploit or proof-of-concept to leave evidence of a successful security breach is to cause the vulnerable software to leave the "standard execution path" and execute instructions it does not normally execute on this path. Since creating evidence of a security breach is not within the normal scope of software operation, instructions to create this evidence must be externally provided. Thus creation of such evidence falls under the definition of "intrusion software." These instructions can take many forms, but by their nature they are not a part of the standard execution path, and the evidence-creating path would not be taken by the software except for their effects.

Thus any malware that "only leaves evidence of a successful security breach" already qualifies as "intrusion software."

Q15. The answer to FAQ #1 says "exploit samples, exploit proof of concepts, or other forms of malware would not be included" yet the answer to FAQ #7 appears to keep the door open for "zero-day exploits." Can you please clarify your definition of "zero-day exploit" and provide discriminating characteristics to differentiate it from the "exploit samples, exploit proof of concepts" that are explicitly excluded by question 1?

This is a two-part answer. First, the answer to FAQ #1 states that the proposed rule does not control the export of exploits and other forms of malware. Zero-day exploits are included in this answer. The export of zero-day exploits, however the term "zero-day" is defined, is not subject to any requirements under the proposed rule.

Second, FAQ #7 asks whether companies will be required to share their zero-day exploits with the government in order to get a license. The answer to FAQ #7 states that when an export license application is filed, BIS can request a copy of the part of the software or source code that implements the controlled functionality. To expand this answer, the export license requirement applies to the system, equipment, component or software that would generate, operate, deliver or communicate with an exploit. *The only regulatory distinction involving zero- day exploits in the proposed rule regards the possibility that a delivery tool could either have (e.g., incorporate) or support (e.g., be specially designed" or modified to operate, deliver or communicate with) zero-day exploits. If the system, equipment component or software at issue has or supports zero-day or rootkit capabilities, then BIS could request the part of the software or source code that implements that capability.* BIS does not anticipate receiving many, or any, export license applications for products having or supporting zero-day capabilities.

The term "zero-day exploit" is a jargon term that is meaningless outside of the context of any given professional conversation. Thus the definition of what constitutes a "zero-day capability" is equally dependent on context and cannot be applied as a structural characteristic of software.

In the context of a computer science publication, "zero-day" means "new, previously not described in literature, therefore worth publishing". In this sense, the only scientific results worth publishing are "zero-day", and vice versa. In the context of a particular attacker operation, "zero-day" usually means "not known to the target's defender", or "not known to the targeted organization's security vendor", or "not known to the broader community although known in some of its parts". Similarly, any novel security tool is "zero-day", or creates a "zero-day capability" by definition, so long as it is novel in some context. Both in academic circles and in industry venues, the novelty of a particular tool is often contested; the peer review mechanism is used to judge this novelty, but often fails. Both modes of failure, designating a non-novel capability as novel, and failing to recognize a novel methodology as one, are common.

Thus without a clear definition of (non-controlled) "zero-day exploit", the definition of controlled "zero-day exploit capability" is vague and beyond any individual researcher's ability to judge, in the same way as scientific novelty is in more established areas of computer science.

Similarly, a "rootkit" is a jargon term that is meaningless outside of a particular context. A "rootkit" is typically a kind of software instrumentation that modifies certain execution paths of the instrumented

software to take actions not supported by the original software. As such, defensive software such as an anti-virus could be a "rootkit", and indeed "rootkit" techniques ("capabilities") have been employed by the antivirus software such as Kaspersky's.[1] Similarly, DRM software such as the "Sony rootkit" revealed by the noted Microsoft researcher Mark Russinovich has been termed a "rootkit". In each case, technical information revealing the particular "rootkit" techniques created a "rootkit capability", easily gleaned from the technical descriptions.

While the functionality of a particular "rootkit" may vary, the capability for it is created by identifying the set of locations in the original software where its execution can be reliably modified and additional ("rootkit") functionality can be injected. Thus any software security instrumentation framework creates a "rootkit capability", including Linux's Security Modules (LSM), the basis for NSA's SELinux software that hardens the Linux kernel against exploits, the DTraces software on MacOS and Solaris operating systems, and many kinds of MS Windows instrumentation such as MinWin, Nirvana, iDNA, TTT, etc.

Transparent instrumentation is the core enabler of both security research and protective measures, yet it also created a "rootkit capability", and would thus be controlled under the proposed BIS rules, to a disastrous effect for cyber-defense.

Q16. Is the United States legally bound to implement the December 2013 Wassenaar Arrangement changes to its control list, or does BIS have discretion?

> The United States, as a Participating State in the Wassenaar Arrangement, has agreed to maintain national export controls on items included in the Wassenaar Arrangements control lists, implemented via national legislation and/or regulation. The U.S. implementation process includes determining reason(s) for control, which carry with them license requirements by destination, licensing policy, and license exceptions.

As demonstrated above, the Wassenaar definitions of "intrusion software" and related control items are exceedingly broad and capture most of meaningful and promising defensive security research. The Wassenaar Arrangement was apparently motivated to curb commoditization of mass surveillance software, but, in an apparent effort to capture all varieties of such software, described most meaningful phenomena and artifacts of security research instead.

Thus the US implementors are urged to narrow the scope of the Wassenaar definition to the mass surveillance software originally envisioned.

Q17. How would the proposed rule affect software used by multinational companies that monitor their overseas networks?

> Under the proposed rule, all exports of specified systems, equipment, components or software that would generate, operate, deliver or communicate with "intrusion software" would require an export license. There is no license exception for intra-company transfers or internal use by a company headquartered in the United States under the proposed rule.

This interpretation, and in particular the controlled status of "intra-company transfers" critically undermines defensive operations of international organizations.

I turned to Iván Arce, co-founder and former CTO of Core Security Technologies, and a noted expert on operational security, for the explanations on this item. Quoting from his response:

> *Proposed regulation will have direct negative impact on security operations at many organizations worldwide.*
>
> *Delivery of offensive tools across national boundaries will require an export license thus delaying arrival. This regulatory imposed delay can take from days to up to several weeks. In the meantime actual exploitation by malicious actors will happen.*

[1] A technical analysis of "rootkit" techniques employed by defensive software and the compositional challenges of such deployment can be found in "Bickering In-Depth: Rethinking the Composition of Competing Security Systems", by Locasto, Bratus, and Schulte, IEEE Security & Privacy Journal, Volume 7, Issue 6, 2009.

Think about a subsidiary of a US based corporation that wants to run Canvas to find out if their local infrastructure is vulnerable (that is exploitable) to a recently disclosed vuln. The US HQ has purchased Canvas, the exploit for the vuln is available (but it isn't in the public domain), the local security admin at the subsidiary wants to check her network of 3000 devices and figure out if the vuln is present and exploitable. There are only two options:

- *Have HQ get an export license to be able to transfer Canvas to the subsidiary.*
- *Run Canvas from HQ. This likely requires downgrading some security controls to allow traffic and the requirement for an export license hinges on what the post-exploitation payload is. For example, it would still require an export license if it supports further "deploying of, or communication with intrusion software".*

Furthermore, this interpretation will also hamper forensic analysis in such organization. Continuing the quote from Iván Arce:

Think about mobile forensics software that runs exploits to gain access to a mobile device and dump its data. During the course of a criminal investigation it is determined that target of the investigation has used a new cellphone for which data acquisition is not supported by current software. A new version of the mobile forensics software needs to be acquired, export license is required to ship it, meanwhile the opportunity to acquire evidence is lost.

Q18. Security professionals use exploit toolkits (e.g. Neosploit, Blackhole, Phoenix, Crimepack &c.) to test patches and harden systems employing the same tools as a potential criminal adversary. Distribution and licensing of such toolkits is tightly controlled in order to preserve their offensive edge. When security professionals succeed in accessing or acquiring these toolkits, they often share with one another across corporate and international boundaries; would such sharing of exploit toolkits be subject to control?

> Exploit toolkits would be described in proposed ECCN 4D004 if they are "specially designed" or modified for the generation of "intrusion software." There are no end user or end use license exceptions in the proposed rule.

This answer underscores the essential ambiguity between the non-controlled "intrusion software" such as exploits and controlled "exploit toolkits". Being a *toolkit* is not a structural description of software; a toolkit may simply be a collection of "exploits"—not to be individually controlled, as per Wassenaar apparent intentions, since each exploit can serve as a sample to be exchanged freely by malware researchers—with trivial customization scripts. The situation when a collection is controlled but its individual items are not controlled creates an ambiguity that requires clarification to be uniformly applied.

Q22. BIS has adopted the definition of 'intrusion software' from the Wassenaar Arrangement language, but has elsewhere indicated a policy of 'presumptive denial' for cybersecurity items 'that incorporate or otherwise support rootkit or zero-day exploit functionality'. Could BIS explain what threshold of severity is meant to be indicated by 'rootkit or zero-day exploit functionality', and could BIS make clear their understanding of those terms?

> Rootkit and zero-day exploit functionality are features more likely to be found in offensive systems or products. A zero-day exploit is not itself controlled. However, when a rootkit or a zero-day exploit is incorporated into a product or system that is described in the new Category 4 control list entries, or if an exploit delivery tool is specially programmed to deliver or command this specialized malware, that product or system is presumed to be offensive by design.

The initial statement of this answer is not correct. So-called "rootkit" functionality—that is, the capability to intercept, examine, and modify flows of data through the operating system to either the targeted software or all software—is in fact a part of a majority of defensive products such as antivirus software, personal firewalls, and many others. More details are given in our notes to Q15 and the paper "Bickering

In-Depth: Rethinking the Composition of Competing Security Systems", by Locasto, Bratus, and Schulte, IEEE Security & Privacy Journal, Volume 7, Issue 6, 2010.[2]

The terms "rootkit" and "zero-day exploit" here are apparently understood in the sense of belonging primarily to the "offensive" realm. However, these are in fact core techniques and artifacts of security work, and are equally important for defense. A "rootkit" technique used by an anti-virus or a security monitoring agent is likely to be structurally indistinguishable from that of malware, differing only in the end-case of use. Accordingly, the jargon term "rootkit" should not be used to structurally describe controlled software, as it is meaningless in this context.

If the distinction between the offensive and defensive uses of these core techniques and artifacts is important in the context of regulation, it should be further clarified.

Q24. Would prior BIS authorization be required for a researcher to privately disclose an exploit to a vendor outside the US with the understanding that the information would NOT be published?

> No. The exploit itself is not described in the new control list entries. Please see the answer to question #1. For this question, a vulnerability is a weakness in a vendors software or hardware. Exploit code could be written to take advantage of the vulnerability or to prove that the vulnerability can be exploited. The exploit code itself may be considered "intrusion software." Neither the disclosure of the vulnerability nor the disclosure of the exploit code would be controlled under the proposed rule. However, information for the development of "intrusion software" that may accompany the disclosure of the exploit may be described in proposed new ECCN 4E001.c.

This interpretation places undue burden on the vendor. Restricting the information that the researcher can supply without triggering a potential violation means that the vendor will need to duplicate effort by analyzing the proof-of-concept to recover an understanding of its mechanism—the understanding that the researcher already has. Further, such recovery by the vendor may be incomplete, resulting in an inefficient mitigation that blocks only the proof-of-concept but not its broader class of attacks.

In order to deploy efficient mitigations, the vendor should receive as detailed and broad description of the exploitation mechanism as possible. The depth of such technical description may provide the vendor with the "capability" to (re)develop the attack, but exactly the same depth also allows the vendor to develop efficient mitigations against such attacks.

The ambiguity of the proposed rule here will damage both researchers and vendors, and, if vigorously enforced, will degrade rather than improve the ecosystem of security mitigations.

Q26. Mobile phone jailbreaking tools include platforms for delivering intrusion software to the phone. These generally include fully operational exploits including the delivery code. Are such tools subject to control?

> This response divides the question into two parts:
>
> i) Does this regulation make it illegal to jailbreak a phone?
>
> No. The Commerce regulation controls exports of certain software, and downloading jailbreaking software to a computer within the United States and using it to jailbreak a phone does not involve an export of software. The proposed rule does not limit the ability of owners to modify their devices.

Although this regulation may not criminalize jailbreaking, it will likely bring it outside of the reach of a typical consumer, as explained below. Thus, while not criminalizing user modification per se, it will chill such modification and any related activities.

> ii) What if the jailbreak software includes a platform for delivering intrusion software to the phone–is the jailbreak software subject to control?
>
> If particular jailbreak software did meet all the requirements for classification under ECCN 4D004 (such as a commercially sold delivery tool "specially designed" to deliver jailbreaking exploits)

[2] `http://www.computer.org/cms/ComputingNow/homepage/2010/0110/W_SP_BickeringIn-Depth.pdf`

then it would be subject to control and a license would be required to export it from the United States. Note that if such software were "publicly available," it would not be subject to the Export Administration Regulations.

The answer (ii) ignores the technological reality of deploying jailbreaking tools to consumers. For example, several Apple iPhone jailbreaks were delivered via a web page, for ease of consumer use—and such web pages were "specifically designed" to deliver the jailbreaking exploit, making them controlled if not "publicly available".

Delivery of new firmware to phones—involved in most kinds of jailbreaking—is a complicated technical process that a typical consumer does not expect to understand; if fact, few security professionals do. Thus, in order to be consumer-accessible, delivery of a jailbreak to a phone must be automated with software for such delivery. Controlling such software will bring jailbreaking, and, consequently, user customizations and modifications, outside of the mass consumer's usability reach.

Q32. Will technology and source code classified under the new control list entries be subject to deemed export requirements?

> Yes, the proposed rule does not provide for any exceptions to deemed export license requirements for release of technology and source code that will be classified under ECCNs 4D004, 4E001.a or .c, 5D001 or 5E001.

This interpretation will severely limit the participation of academic researchers—who typically work in open lab environments with many foreign students, including foreign M.S. and Ph.D. researchers—in interactions with industry where the industry needs to protects its proprietary information, and cannot agree to full publication of research results.

To the best of our knowledge, many such interactions are governed and made possible by such partial restrictions on publication of results. The proposed rule will limit both the industry access to academic research groups and vice versa, damaging both sides and weakening defense beyond the apparent original intentions of the Wassanaar Arrangement.