COMMENTS ON BUREAU OF INDUSTRY AND SECURITY PROPOSED RULE

WASSENAAR ARRANGEMENT 2013 PLENARY AGREEMENTS IMPLEMENTATION: INTRUSION AND SURVEILLANCE ITEMS

Allen Householder <adh@cert.org>

Art Manion <amanion@cert.org>

2015-07-20

About CERT

The CERT Coordination Center[1] (CERT/CC) is part of the Software Engineering Institute (SEI), a Federally Funded Research and Development Center (FFRDC) based at Carnegie Mellon University. The CERT/CC was established in 1988 and has decades of experience in software and network security. We engage regularly with the software development ("vendor") and security research communities in two primary areas of focus:

- Developing tools and techniques for finding vulnerabilities, generally intended for use by software development organizations to improve their security testing
- Coordinating the public disclosure of vulnerabilities among the security research and vendor communities, with the goals of improving processes and developing longer term mitigation strategies

We offer the following comments in response to the proposed rule with request for comments, on the *Wassenaar Arrangement 2013 Plenary Agreements Implementation: Intrusion and Surveillance Items,* published in 80 FR 28853 [Docket No. 150304218–5218–01] RIN 0694–AG49 BIS-2015-0011.

[1] http://www.cert.org/

Summary of Recommendations

In summary, we recommend the following:

1. We're experienced in the security research field, but not in the export control field. We reviewed the proposed rules carefully, but we don't understand some important aspects of them. We recommend creating a second draft and establishing a corresponding comment period.

2. We are concerned about chilling effects on vulnerability discovery and disclosure. Such chilling effects could impair vulnerability remediation and management.

3. Difficulty and ambiguity in defining what software (technology) is meant to be covered is likely to have the unintended consequence of chilling beneficial public security research. To ease this risk, we recommend the following:

    a. Avoid the use of the terms "zero-day exploit capability" and "rootkit capability" entirely as (1) they are not well defined and (2) they do not sufficiently define a certain class of intrusion software.

    b. Define "carrier class IP network" clearly using well-defined technical metrics.

    c. Clarify what is meant by "externally provided instructions."

4. We offer our assistance to further refine the proposed rules.

General Comments

Fundamentally, the Wassenaar Arrangement (WA) definition of “intrusion software” is overly broad. We generally agree with the argument and proposed alternative (“exfiltration software”) described by Bratus et al.[2]

Our comments align most closely with question #3:

> *Would the rule have negative effects on your legitimate vulnerability research, audits, testing or screening and your company’s ability to protect your own or your client’s networks? If so, explain how.*

Our government sponsors have tasked us with coordinating the public disclosure of vulnerabilities among the security research and vendor communities, with the goals of improving processes and developing longer-term mitigation strategies. We are concerned that overly general and imprecise language in the proposed rules could have unintended chilling effects on beneficial security research, thereby inhibiting our ability to perform this important task.

The Intrusion and Surveillance Items FAQ[3] provided by the Bureau of Industry and Security (BIS) after the initial FR posting helped to clarify a number of our concerns, for example, by listing exemptions for published research and the exchange of proof-of-concept exploits, malware, and some non-public vulnerability information.

Unfortunately, even with the FAQ and considerable effort, we do not understand some important aspects of the proposed rules. We are also aware of confusion and concern within the security research community, and suggest an additional draft of the proposed rules and a corresponding public comment period.

[2] http://www.cs.dartmouth.edu/~sergey/drafts/wassenaar-public-comment.pdf

[3] https://www.bis.doc.gov/index.php/policy-guidance/faqs?view=category&id=114#subcat200

Chilling Effects on Beneficial Security Research

Open, public research in software security benefits the United States and society in general, including the software industry, the defense industrial base, and many elements of the nation's critical, financial, and social infrastructure. Greater use of commodity and commercial off-the shelf (COTS) components in emerging, non-traditional computing markets (e.g., Internet of Things, industrial control systems, medical devices, transportation) increases system-wide susceptibility to vulnerabilities.

Public security research leads to improved software through the resolution, mitigation, and notification of vulnerabilities. Vulnerabilities cannot be resolved or mitigated unless they are discovered and disclosed. In federal government vulnerability management terms, the following steps are involved:

1. Security research is performed.
2. Researchers privately share information about vulnerabilities, exploits, and associated technology with vendors.
3. The results of security research are published, often in the form of public vulnerability disclosures.
4. Common Vulnerability and Enumeration (CVE)[4] assigns identifiers (CVE-IDs) to publicly disclosed vulnerabilities, and entries are made in the National Vulnerability Database (NVD)[5].
5. Federal civilian vulnerability management programs such as Continuous Diagnostics and Mitigation (CDM)[6] and military programs like Information Assurance Vulnerability Management (IAVM)[7] track vulnerabilities using CVE-IDs.
6. Vulnerabilities are identified, detected, and resolved in deployed systems.

This process is common practice (with different programs for step 5) and is not limited to the U.S. government. Decreasing public research output (steps 1-3) would almost certainly lead to decreased vulnerability resolution (step 6).

Vulnerability research is performed by a wide range of participants from students, hobbyists, and amateurs through professionals with or without advanced degrees, and by both individuals and organiza-

[4] http://cve.mitre.org/

[5] https://nvd.nist.gov/

[6] http://www.dhs.gov/cdm

[7] IAVM overview: http://www.prim.osd.mil/cap/iavm_req.html

IAVM to CVE mapping: http://iase.disa.mil/stigs/Pages/iavm-cve.aspx tions. Vulnerability research is often performed outside the context of any contractual obligations. Serious vulnerabilities have been found because someone noticed something interesting, pursued it, and took action, not because they were hired to do so. Publication venues range from simple email to a public list, tweets, and blog posts up to rigorous peer-reviewed academic conferences or journals.

Some security research is reimbursed after the fact by bug bounty programs that pay out once a vulnerability has been reported and fixed. Bug bounty programs show promise in helping to identify and remove vulnerabilities. The global nature of the Internet coupled with geographical arbitrage results in many vendors receiving a significant number of vulnerability reports from researchers in other countries.

Legal, regulatory, and policy-making efforts face challenges in keeping up with rapidly evolving technical developments, including security research. Imprecise or outdated legal and regulatory language can lead to chilling effects, as those who would otherwise conduct research may choose not to engage in, share, or publish research due to concerns about legal repercussions.

Concerns Regarding Vulnerability Disclosure for Platform Security

Some vendors offer higher bug bounty payments for reports of vulnerabilities in their platform-wide security features (e.g., exploit mitigation[8] or sandbox bypasses[9]). To demonstrate the existence of such vulnerabilities, it is often necessary for the researcher to create a proof-of-concept exploit that is indistinguishable from the WA definition of intrusion software, specifically, “Software specially designed or modified … to defeat ‘protective countermeasures’ … of a computer or network-capable device, and performing … the modification of system or user data.” Thus, by providing the vendor with detailed information about the platform security vulnerability the researcher would be transferring “technology required for the development of intrusion software.”

We believe this interpretation to be consistent with the answer to FAQs #19 and #24. We quote the latter for discussion purposes:

> *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.*

[8] Microsoft Mitigation Bypass Bounty and BlueHat Bonus for Defense Program
https://technet.microsoft.com/en-us/Library/dn425049.aspx

[9] Chrome Reward Program Rules (Sandbox Escape)
https://www.google.com/about/appsecurity/chrome-rewards/

We are concerned that tools used for research and development of proof-of-concept exploits against platform security (e.g., exploit mitigation or sandbox bypass) might qualify as “technology required for the development or production of intrusion software.”

The proposed rules appear to make it illegal, absent the possession of a license, for

1. a U.S. researcher to provide such vulnerability information to a foreign vendor
2. a U.S. researcher to provide such vulnerability information to a foreign citizen employed by a U.S. vendor
3. a U.S. employee of a U.S. vendor to provide such vulnerability information to a non-U.S. employee at the vendor’s international division
4. a U.S. employee of a U.S. vendor to provide such vulnerability information to a foreign citizen employed in the same U.S. location
5. a U.S. researcher traveling to another country (for example to attend a conference), while in possession of vulnerability research prototypes, meeting the definition of “technology required for the development of intrusion software”

In particular, the second scenario describes the situation in which a researcher discovers an exploit mitigation bypass technique and reports all the details they have (including code) to Microsoft’s Mitigation Bypass Bounty. Would sharing the details with the product security incident response team (PSIRT), which may include non-U.S. citizens, qualify as a deemed export? Is it incumbent on the researcher to first ascertain the citizenship of the PSIRT members who will receive those details? Is it Microsoft’s responsibility to acquire a license for each of their foreign citizens working in their PSIRT prior to receiving the details? Should vendors instead only employ U.S. citizens in their PSIRTs?

It is not clear that the proposed rules are intended to require licensing for the cases described above. We are concerned about the potential for the misinterpretation of the BIS’ intent given the wording of the proposed rules. Such misinterpretation will likely lead many small organizations and individual security researchers to avoid an ambiguous legal context and simply cease performing or privately sharing the results of their research.

We recommend BIS provides further clarification.

Questionable Applicability of Publication Exception

We understand that intrusion software itself is not covered by the proposed rules. Furthermore, we understand that technology required for the development of intrusion software is covered, unless that technology is published. There are common security research scenarios in which both intrusion software and technology required for the development of intrusion software are exported without publication or intent to publish in the future. For example, vendors often release fixed software and publish summary information about vulnerabilities, but they do not publish the proof-of-concept exploits or detailed vulnerability information.

Several FAQ entries (i.e., #4, #10, #19, #24, #25) discuss these scenarios. In particular, FAQ #25 states:

> *…if technology "required for the development of intrusion software" (as described in the proposed control list entry ECCN 4E001.c.) exported with the functional proof of concept/"intrusion software" would be described in new control list entry ECCN 4E001.c and would, under the proposed rule, require a license…*

Consider the following additional scenario: A U.S. researcher reports a technique for exploit mitigation bypass to a non-U.S. vendor with the intent that the vendor will improve the exploit mitigation techniques in their products. From the example in the previous section, we understand that this would constitute “technology required for the development of intrusion software.” However, in such cases often neither the researcher nor the vendor has any intention to publish the technical details or proof-of-concept code the researcher provided. The vendor improves their exploit mitigation technology in a future version. Did the U.S. researcher need to acquire a license prior to reporting to the vendor?

As the above scenario illustrates, vulnerability reports are often accompanied with more information than is actually published upon their resolution. This fact, coupled with the ambiguity of what specifically is and is not covered is concerning. From FAQ #19:

> *…it is possible that certain technology associated with the exploit would be "technology required for the development or production of intrusion software" under proposed ECCN 4E001.c. As stated in the answer to FAQ #10, any technical data that is transferred with the intent that it be published would not be controlled. However, as the question recognizes, not all technical data is intended to be made public, and some of it may be controlled.*

The distinction between proof-of-concept exploit code that qualifies as intrusion software, and associated “technology required for the development or production of intrusion software” is unclear.

We recommend BIS provide further clarification.

Imprecise Language

Our remaining concerns chiefly involve imprecise terminology and language that, left open for interpretation, is likely to cause chilling effects on legitimate, beneficial security research.

The following terms, as they appear in the proposed rules, the WA text itself, or the BIS FAQ, are offered without definitions:

- "Items that have or support rootkit or zero-day exploit capabilities"
- "Describe how rootkit or zero-day exploit functionality is precluded from the item"
- "Support rootkit or zero-day exploit functionality"
- "Carrier class IP network (e.g., national grade IP backbone)"
- "Externally provided instructions"

The absence of definitions as part of the proposed rules significantly inhibits our ability to reason about the rules themselves. Because of this confusion, we request that the BIS revise the proposed rules and corresponding guidance based on the feedback received in this comment period and offer the public a second comment period on the revisions.

Ambiguous: "Rootkit and Zero-Day Exploit Capabilities"

Section 742.6(b)(5) [excerpted for clarity] states

> *Applications for exports, reexports and transfers of cybersecurity items [list of specifics removed] will be reviewed favorably if [a number of caveats for regional stability], except that there is a policy of presumptive denial for items that have or support rootkit or zero-day exploit capabilities.*

Also, in Supplement No. 2 to Part 748, paragraph (z), question (1)(iii)(C)

> *(C) For items related to "intrusion software," describe how rootkit or zero-day exploit functionality is precluded from the item. Otherwise, for items that incorporate or otherwise support rootkit or zero-day exploit functionality, this must be explicitly stated in the application.*

The adjective "zero-day," used to modify "exploit," is generally, loosely understood by security professionals to indicate a sense of surprise by stakeholders (typically vendors), the general security community, or the public. "Zero-day" is not, however, sufficiently precise for use in standards, law, or regulation, including the BIS proposed rules. Security experts do not precisely agree on what "zero-day" means. Appendix A lists ten definitions for "zero-day" from ten different expert sources.

We interpret that the BIS may use "zero-day" to mean exploitation of a vulnerability that is not known to some combination of the victim, the vendor, or the general public, and for which a patch or update does not exist (i.e., the victim would be very unlikely to be able to defend against the exploit, and the exploit would be very likely to succeed).

However, we observe that from the perspective of someone creating a tool to perform penetration testing, vulnerability scanning, vulnerability discovery, or even intrusion software (as defined), there is nothing intrinsic to what the tool does or the knowledge that it embodies that lets its author distinguish between its *zero-day exploit capability* and its *exploit capability*. This ambiguity is entirely because any definition of "zero-day" is an assertion about specific humans' (vendors, the security community, or the public, depending on the specific usage) ignorance at a particular point in time, and not an assertion about software, its vulnerabilities, their exploits, or any other intrinsic property of the code.

In other words, it is not possible to meaningfully differentiate software that exploits *known* vulnerabilities from software that exploits *unknown* ones. If the intrusion software uses exploits, those exploits may or may not be known to a given party or the general public. Nothing about the intrusion software changes based on others' state of knowledge.

Contrary to the BIS assertion in FAQ #15, "BIS does not anticipate receiving many, or any, export license applications for products having or supporting zero-day capabilities," it is our opinion that any technology that supports or provides *exploit capability* must also be presumed to include *zero-day exploit capability* since the differentiating factors occur in the world, not in the technology.

As a result, we strongly recommend that the BIS avoid the term "zero-day exploit" in both the final rules and the supplementary information that accompanies them.

Likewise the term "rootkit" is similarly imprecise and subject to widely variable interpretation. We interpret that the BIS may use "rootkit" to mean a specific kind of intrusion software (as defined) with the features of persistence and stealth. We note that stealth (in the form of avoiding detection) is already part of the definition of "intrusion software."

We recommend that the BIS avoid the term "rootkit" in both the final rules and the supplementary information that accompanies them.

We further suggest that the rules and supplementary information could be improved by expressing the distinguishing features in terms of some combination of persistence, stealth, exfiltration, remote control, robustness of the exploitation techniques, and reliability of the of the intrusion software. We offer the following definitional sketches that may be useful:

- *Persistence* - designed to persist on an affected system or in an affected network beyond the termination of its delivery mechanism (This could include mechanisms for recovering after a machine reboots, or for reinfection of systems across a network.)
- *Stealth* - designed to evade detection, whether in transit, during execution, or at rest
- *Exfiltration* - designed to surreptitiously send data from the affected system, software, or device without the knowledge of its owner or operator

- *Remote control* - accepts commands from a remote operator
- *Robustness of exploitation techniques* - the ability of an exploit to consistently function and succeed in the presence of platform exploit mitigation technology (ASLR[10], DEP[11], etc.)
- *Reliability of intrusion software* - the ability to ensure predictable behavior and performance of the intrusion software

We do not claim that these definitions are sufficiently precise as written. Nor do we offer any specific suggestion for how these terms might be combined into guidance to differentiate export-controlled technologies from non-controlled ones. Our reticence to make more detailed recommendations stems from our lack of understanding of which distinguishing features BIS had in mind. The next section discusses this in more detail.

Did BIS Intend to Discriminate Between Offensive and Defensive Technologies?

We interpret that the BIS intended to define a particular class of technology that is not only controlled but for which no license would be granted ("presumptive denial"). FAQ #22 sheds some light on this intention:

> *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.*

If BIS' concern is to distinguish between offensive and defensive security products, with the intent to catch the former while releasing the latter, we believe there is significant work to be done in clarifying the difference and defining the distinguishing characteristics. As it stands the rules, supplementary information, and FAQ left us unable to discern what the discriminating characteristics might be for this class of technology.

To that point, FAQ #13 states, "It is BIS's understanding that there is no technical basis to distinguish defensive products from offensive products (i.e., a defensive product may be used offensively)." We agree. We are unaware of material technical differences between legitimate penetration testing tools, which incorporate the above features and the class of attack tools that we believe the BIS is trying to define.

For this reason we posit that the "policy of presumptive denial for items that have or support rootkit or zero-day exploit capabilities" is untenable in its present state. Given the significant number and depth

---

[10] Address Space Layout Randomization

[11] Data Execution Prevention

of concerns described here, we request that the BIS revise the proposed rules and offer the public a second comment period on the revisions.

Undefined: “Carrier Class IP Network”

From 5A001

> *j. IP network communications surveillance “systems” or “equipment”, and “specially designed” components therefore, having all of the following:*
>
> *j.1. Performing all of the following on a carrier class IP network (e.g., national grade IP backbone):*
>
> *j.1.a. Analysis at the application layer (e.g., Layer 7 of Open Systems Interconnection (OSI) model (ISO/IEC 7498–1));*
>
> *j.1.b. Extraction of selected metadata and application content (e.g., voice, video, messages, attachments); and*
>
> *j.1.c. Indexing of extracted data; and*
>
> *j.2. Being “specially designed” to carry out all of the following:*
>
> *j.2.a. Execution of searches on the basis of ‘hard selectors’; and*
>
> *j.2.b. Mapping of the relational network of an individual or of a group of people.*
>
> *Note: 5A001.j does not apply to “systems” or “equipment”, “specially designed” for any of the following:*
>
> *a. Marketing purpose;*
>
> *b. Network Quality of Service (QoS); or*
>
> *c. Quality of Experience (QoE).*
>
> *Technical Note: ‘Hard selectors’: data or set of data, related to an individual (e.g., family name, given name, email or street address, phone number or group affiliations).*

The term “carrier class IP network (e.g., national grade IP backbone)” is undefined. The BIS discusses this in FAQ #14:

> *The term “carrier class IP network” is meant to specify systems that sit at a national level (or large regional) IP backbone and handle data from an entire city or country. In terms of IP network surveillance systems, this is meant to exclude systems that can only handle smaller data streams or networks, such as those for a campus or a neighborhood. This control does not capture systems that can only analyze data from one person or a small group of people at a time. The term “carrier class IP network” was not defined because it was difficult to put precise technical parameters around this concept.*

We note “The term ‘carrier class IP network’ was not defined because it was difficult to put precise technical parameters around this concept” as a critical weakness in the proposed rules. We further observe that it is incongruous to use the term “smaller” without some scale or point of comparison to measure against.

It’s likely that some enterprise or provider networks are significantly larger than the “national grade IP backbones” of some nations. Internet service in some nations is provided directly or indirectly by the government, while service in other nations is provided by private industry, all with varying degrees of regulation, including provisions for legal surveillance.

We suggest that the BIS rules include modifications or notes to more narrowly define “carrier class IP network,” either based on providing service to the general public or having common carrier status. These changes, however, do not consider the size or throughput of the network, which seems to be the intent of *j.1*. If size or throughput are significant defining characteristics, we suggest that the BIS define some scale in terms of data throughput, whether at a network interconnect or equipment interface level, or the potential available processing power or constructed graph analysis capacity (in terms of number of nodes or links it can reason over).[12]

Additional Exceptions to IP Network Surveillance Software

We note the exceptions for “systems” or “equipment”, “specially designed” for any of the following:

a. Marketing purpose;

b. Network Quality of Service (QoS); or

c. Quality of Experience (QoE).

We also suggest that similar consideration be made explicit for

d) Network availability monitoring

e) Network performance monitoring

[12] We note that any such definition will necessarily have a limited time horizon in which it remains useful before available technology overtakes it. Thus, continued vigilance on the part of the BIS would be required to maintain the proper differentiating parameters given advances in available technology. We also speculate that horizontal scaling techniques may make any such definition dependent on choices made at the time of such a system's deployment (e.g., how many machines to deploy) rather than being an intrinsic property of the technology (i.e., the same technology deployed as a 100-node instance may be permissible but not as a 1000-node installation). All this is obviously suboptimal to some other decision criteria based on explicitly defined features that such a system offers; however, it seems unlikely that such a definition is possible without reference to system capacity and scaling. We hope to be proven incorrect on this last point.

f) Network Intrusion Detection[13] (IDS)

g) Network Intrusion Prevention (IPS)

h) Data Loss Prevention (DLP)

i) Cross-Domain Solutions (CDS)

j) Network troubleshooting

Ambiguous: "Externally Provided Instructions"

§772.1 defines "intrusion software" using language from the WA with additional notes from the BIS.

> *(b) The modification of the standard execution path of a program or process in order to allow the execution of externally provided instructions.*

Our concern with this clause is that the terms "externally provided" and "instructions" may be misleading.

"Instructions" may refer to either (1) machine instructions such as the Intel 64 and IA-32 instruction sets or (2) arbitrary code or commands issued by a remote operator. In the context of the execution of a control-flow exploit such as a buffer overflow leading to provision of a shell, an attacker may use the former to enable the latter.

We interpret that the BIS meant to use the second definition above, and therefore suggest adding an explanatory note.

Furthermore, "externally provided" could refer to either (1) external to the running program or process being exploited, yet still within the same machine, device, or system or (2) external to the machine, device, system, network, or other infrastructure in which the exploited program or process is running.

Again, we interpret that the BIS meant to use the second definition, and therefore suggest adding an explanatory note. If or when there is an opportunity to modify the WA language, we might suggest "…execution of instructions provided by a remote operator," assuming that is the correct intent.

[13] Also see Dr. Nicholas Weaver's comment http://www.regulations.gov/#!documentDetail;D=BIS-2015-0011-0089

Security Assurance for the Licensing Process Itself

Paragraph (z) Part 748, question (2) states:

> *(2) Upon request, include a copy of the sections of source code and other software (e.g., libraries and header files) that implement or invoke the controlled cybersecurity functionality.*

Software development organizations may be reluctant to share sensitive intellectual property (IP) without safeguards or recourse in the event that the information is leaked or stolen in the course of meeting the proposed rules.

We note the importance of providing sufficient protection to information submitted and subsequently handled in the licensing process. Such protection is important for data both in transit and at rest, whether on the part of the potential licensee, systems involved in the BIS licensing process, or transfer of information to or from reviewers. Given the highly sensitive nature of such submissions, we expect any centralized repositories used in the licensing process to be high-value targets for adversarial entities both foreign and domestic.

We recognize that safeguarding practices may already be well established in other controlled software items such as cryptography. We emphasize its importance here in light of recent significant security events, where the concentration of high-value data has led to catastrophic breaches resulting in precisely the types of knowledge transfer that such processes are specifically intended to avoid.

Appendix A

The following text is reproduced from https://www.cert.org/blogs/certcc/post.cfm?EntryID=247

Like Nailing Jelly to the Wall: Difficulties in Defining "Zero-Day Exploit"

By Allen Householder on 07/07/2015

During the Watergate hearings, Senator Howard Baker asked John Dean a now-famous question: "My primary thesis is still: What did the president know, and when did he know it?" If you understand why that question was important, you have some sense as to why I am very concerned that *"zero-day exploit capability"* appears as an operative phrase in the Department of Commerce Bureau of Industry and Security (BIS) proposed rules to implement the Wassenaar Arrangement 2013 Plenary Agreements regarding Intrusion and Surveillance Items.

Background: BIS, Wassenaar, and "Zero-Day Exploit Capability"

The United States Department of Commerce Bureau of Industry and Security recently proposed a set of rules to implement the agreements by the Wassenaar Arrangement (WA) at the Plenary meeting in December 2013 regarding tools and technologies surrounding "intrusion software."

One particular comment in that proposal is relevant to this blog post: *"Note that there is a policy of presumptive denial for items that have or support rootkit or zero-day exploit capabilities."* This policy is implemented in the following line proposed for inclusion in section 742.6(b)(5) [excerpted for clarity]:

> *Applications for exports, reexports and transfers of cybersecurity items [list of specifics removed] will be reviewed favorably if [a number of caveats for regional stability], except that there is a policy of presumptive denial for items that have or support rootkit or zero-day exploit capabilities.*

Further, Supplement No. 2 to Part 748—Unique Application and Submission Requirements—notes the following:

> *(iii) If the cybersecurity item has not been previously classified or included in a license application, then:*
>
> *[Other requirements removed for clarity]*
>
> *(C) For items related to "intrusion software," describe how rootkit or zero-day exploit functionality is precluded from the item. Otherwise, for items that incorporate or otherwise support rootkit or zero-day exploit functionality, this must be explicitly stated in the application.*

The answer to question 1 of the BIS FAQ on Intrusion and Surveillance Items reads in part

> *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.*

Later, the answer to question 15 includes this statement:

> *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.*

In writing this post, I attempted to draw a flowchart to map the decision process to discriminate between tools having *zero-day exploit capabilities* but not *exploit samples, exploit proof of concepts*. I also tried to generate the possible combinations of both the existence of and vendor and public awareness of vulnerabilities, exploits, and patches, and map those onto whether or not they qualified as "zero-day exploits" or "zero-day vulnerabilities." I failed in both cases.

The reasons I failed are twofold:

1. There are many definitions of *zero-day exploit* available. These definitions are not merely diverse wordings that map onto the same concepts; they refer to distinct (albeit related) concepts. In other words, given the same state of affairs in the world, they yield different answers as to whether or not that state meets the definition.
2. Common to all the definitions is a sense of history, summarized as "Who knew what, and when did they know it?" Note its resemblance to Senator Baker's query. The problem is that some information relevant to the definition only becomes available after certain decisions have been acted upon, and thus that information can not have a causal relationship to the decision in the first place.

I cover both points in more detail below following a brief introduction to why this topic is so relevant now.

You Keep Using That Word; I Don't Think It Means What You Think It Means

Many discussions that touch on vulnerability disclosure involve phrases like "zero-day vulnerability," "zero-day exploit," or simply "zero day" or "0day." However, I've noticed that there is a good deal of confusion as to the meaning of these terms. Security professionals have used these terms inconsistently, or at least they've done so in ways that make it unclear about which meaning they're using. Furthermore, inconsistent use of terms in media reports exacerbates confusion and concern among individuals, network defenders, and decision and policy makers. Finally, in the context of laws and regulations, inconsistent definitions of terminology can become the distinguishing factor as to whether or not one has committed a crime.

Normally I wouldn't write an entire blog post on the definition of terms since for most conversational purposes, loose definitions will suffice. However, when those loosely defined terms become the basis for decisions, policies, and regulations, it's important to get it right.

Like Nailing Jelly to the Wall

The BIS proposed rules that specifically refer to *zero-day exploit capability*. Setting aside what it means for something to have *X capability*, I'd like to demonstrate the difficulty in defining this particular *X*: What's a *zero-day exploit*? (For if we can't define *X*, then *X capability* must also remain undefined.) I went looking for definitions, and found a few:

1. "A zero-day exploit is one that takes advantage of a security vulnerability on the same day that the vulnerability becomes generally known. There are zero days between the time the vulnerability is discovered and the first attack." —SearchSecurity

The first definition is fairly specific, even if it doesn't really explain what "generally known" means. (Known to whom? What subset of the population must know about it for it to count as "generally known"?) But the rest of it is pretty clear: if the exploit is used on the same day that the vulnerability became "generally known," then it's a zero-day exploit.

Oh, but wait, does *same day* mean the same calendar day? In what time zone? Like the song says, "It's Five O'Clock Somewhere." So if the vulnerability is reported at 11:59 p.m. in your time zone and an exploit is reported five minutes later, is it still a *zero-day exploit*? Maybe?

What if we replace *same day* with *within 24 hours*. At least then we can say for certain that if the vulnerability is made public at 8:00 a.m. UTC on day 0 and the exploit is reported at 8:01 a.m. UTC on day 1, it's not a *zero-day exploit*. I don't know about you, but that strikes me as arbitrary and unsatisfying.

By the way, nothing in this definition talks about patch availability. We'll come back to that in a moment.

2. "A zero day exploit attack occurs on the same day a weakness is discovered in software. At that point, it's exploited before a fix becomes available from its creator." —Kaspersky

There's that *same day* again. I'll grant that *weakness* here is equivalent to *vulnerability* in definition 1. But this definition goes beyond just talking about a vulnerability and its exploit; it mentions a *fix* that *becomes available*.

Stating it explicitly: if the following events occur (a) a vulnerability is announced by a vendor, (b) a patch is provided along with the announcement, and (c) it is exploited on the *same day* (whatever you decide that means, just be consistent), definition 1 says it's a zero-day exploit while definition 2 says it isn't.

3. "An attack on a software flaw that occurs before the software's developers have had time to develop a patch for the flaw is often known as a zero-day exploit. The term "zero-day" denotes that developers have had zero days to fix the vulnerability. It can also refer to attacks that occur on the same day (day zero) a vulnerability is disclosed. In fact, some zero-day exploits are the first indication that the associated vulnerability exists at all." —Tom's Guide

There are two distinct definitions here: one is in the first sentence, and one is in the third. The third sentence equates to definition 1 above, so let's focus on the one in the first sentence.

Here we find that the definition hinges on the existence of a patch. A strict interpretation of this definition would permit someone to apply the zero-day exploit label even if the vulnerability is known to the vendor and the public long before the first attack. The vulnerability may have been known to the vendor for months, and a patch is in development but not does not yet exist. Thus definition 3 directly conflicts with both definitions 1 and 2 above. Definition 1 says nothing of patches. Definition 2 talks about patch availability, not existence.

4. "Zero-day attacks...software or hardware vulnerabilities that have been exploited by an attacker where there is no prior knowledge of the flaw in the general information security community, and, therefore, no vendor fix or software patch available for it." —FireEye

Granted, this definition is for a *zero-day attack*, but since it mentions exploitation, I think we are justified to include it here. FireEye adds hardware to our growing list of definitions. Further, they discriminate based on the state of knowledge of the *general information security community,* with the implication that if that community is unaware of the vulnerability, there must not be a patch available. From context, this *general information security community* appears to be larger than the affected vendor(s) yet smaller than the public. So while it shares some degree of overlap with the other definitions discussed above, it remains distinct in its referents.

"But," you say, "these are informal definitions that aren't meant to be interpreted as strictly as you're doing so here." Criticism acknowledged. Using colloquial definitions in a technically focused context may be inappropriate when there are important yet subtle distinctions at play. So let's review the academic literature.

5. "A zero-day attack is a cyber attack exploiting a vulnerability that has not been disclosed publicly. There is almost no defense against a zero-day attack: while the vulnerability remains unknown, the software affected cannot be patched and anti-virus products cannot detect the attack through signature-based scanning." —Leyla Bilge and Tudor Dumitras, Before we knew it: an empirical study of zero-day attacks in the real world

Again, we make the bridge from *attack* to *exploit*. Interestingly, this definition equates *disclosed publicly* with *unknown*. Yet we know that vendors are continuously made aware of vulnerabilities in their products that the public does not know about: coordinated disclosures are things that happen (and that we here at CERT/CC are often involved in facilitating them).

In this case, *cannot be patched* is not an assertion about the creation of a patch; rather it refers to the application of that patch to deployed vulnerable systems. Also, that point is presented as an implication of the definition rather than a part of the definition.

Interpreting definition 5 strictly, neither of the scenarios presented under definitions 2 or 3 above would qualify as *zero-day attacks*. Definition 4 differs from definition 5 in that it refers to the *general information security community* while definition 5 refers to public disclosure.

6. "A zero-day exploit is a new attack that an organization is not prepared for and can't stop. But there are conflicting definitions of zero-day, and different understandings regarding dates and times when an exploit becomes and/or ceases to be a zero-day exploit. The most practical definition of a zero-day exploit: An exploit that has no corresponding patch to counteract it. Technically, if the exploit code exists before the vulnerability is made public, it's a zero-day exploit -- regardless of how long the software vendor may have been aware of the vulnerability." —Brian T. Contos, Enemy at the water cooler: True stories of insider threats and enterprise security management countermeasures

Here we have a definition that at least acknowledges that other definitions exist, then hews fairly closely to definition 3 above.

7. "Zero-day exploit: An attack that exploits a zero-day vulnerability." —David A. Mundie and David M. McIntire, The MAL: A Malware Analysis Lexicon

Hmm. Is this definition talking about different things than those presented in definitions 1-4? I can't tell. I suppose we'll have to define *zero-day vulnerability* to figure that out. Conveniently, the MAL defines it for us:

8. "Zero-day vulnerability: A vulnerability that has not been disclosed to the general public and so can be exploited before patches are available."

Exploited prior to public disclosure. Easy enough. Everybody can agree to that, right? Wrong. Keep reading.

9. "A zero-day vulnerability is one that is unpublished. By definition, all vulnerabilities are zero-day before they are disclosed to the world, but practitioners in the art commonly use the term to refer to unpublished vulnerabilities that are actively exploited in the wild. We further distinguish zero-day vulnerabilities from published vulnerabilities as those for which no patch, upgrade, or solution is yet available from the responsible vendor, although some fail to make this distinction. " —Elias Levy, Approaching zero

Three different definitions appear here: (a) unpublished, (b) unpublished and exploited, (c) no patch available (regardless of exploitation status). Ugh. One more try?

10. "For the purposes of this paper, we formally define a 0Day vulnerability as any vulnerability, in deployed software, that has been discovered by at least one person but has not yet been publicly announced or patched." —Miles A. McQueen and colleagues, Empirical estimates and observations of 0day vulnerabilities

Given this definition we can describe the number of people who know about the vulnerability as greater than or equal to 1 but (significantly) less than *the public*. Also, patch status matters.

Lucky for us, this paper actually prefixes the above definition with the following caveat:

> *There is no generally accepted formal definition for "0Day (also known as zero-day) vulnerability." The term has been used to refer to flaws in software that no one knows about except the*

> *attacker. Sometimes the term is used to mean a vulnerability for which no patch is yet available.*

I'm going to take the hint here and stop trying to pin this down further. You can probably see why I failed in my attempt to map this out in a concise flowchart.

Who Knew What, When?

The thing that is most clear to me from the above is that all definitions of *zero-day exploit* and *zero-day vulnerability* hinge on the state of knowledge of some subset of humanity at some point in time. Once discovered, there is always at least one person who is aware of the existence of the vulnerability. Beyond that the definitions largely vary based on who knows what, and when. This is the connection to Baker's question.

So far, we've established that all the definitions of zero-day exploit and zero-day vulnerability are time dependent. Moreover, they all incorporate the notion of surprise: in order for a vulnerability or its exploit to meet any of the definitions above, its existence must be surprising to someone. Furthermore, the definitions don't simply state that someone has to be surprised they indicate specific subsets of humans that must experience that surprise: either vendors, the security community, or the public, depending on which definition you prefer.

Now, think about that for a moment: What observable property intrinsic to a vulnerability could you point to that tells you this? Nothing. Why? Because surprise arises inside human skulls, not in the software, nor in the vulnerability report, nor in the exploit code, nor in any of the tools that support the discovery or development of these things. The adjective phrase *zero-day* is an assertion about human ignorance *at a particular moment in time*. It isn't an assertion about an intrinsic attribute of software, a vulnerability in that software, or an exploit for that vulnerability.

Complicating things further, not every vulnerability has exactly one vendor responsible for providing a patch. In the CERT/CC, our vulnerability disclosure coordination efforts often require us to work with multiple vendors as we try to synchronize the publication of vulnerability information with the release of patches. In situations where a vulnerability affects multiple vendors' products, public disclosure of one product's vulnerability can lead directly to the users of other products being put at risk because they are exploitable without recourse until a patch for their software is provided.

Even this scenario is too simple though. Some vulnerabilities affect *multiple products* from multiple vendors. This is a common occurrence for vulnerabilities that arise above the code level (e.g., protocol vulnerabilities) or when code is shared across products (e.g., third party libraries, example code that everybody copied and pasted, even a single developer who recreated the same error in multiple projects). So now we have a number of vendors and potentially distinct user groups that could be surprised by the existence of a vulnerability or its exploit. Should a vulnerability that affects 100 vendors' products be considered a zero-day if 99 vendors announce patches while one doesn't? What if 50 vendors patch and 50 don't? What if one vendor provides patches but 99 don't? What if that one vendor accounts for 90% of the users? 80%? 50%? 20%? 2%?

Most extant definitions of *zero-day exploits* and *zero-day vulnerabilities* completely fail to acknowledge this sort of multiparty process, and assume (naively) that a vulnerability report is between one vendor and one finder.

Conclusion

If you discover a vulnerability in a product and you want that vulnerability to get fixed, there's really no way around telling the vendor about it. At the point you make that decision though, you don't (and can't) actually know whether this particular vulnerability is new to them or not. If you find a vulnerability and for whatever reason you don't want it to get fixed, you still don't (and can't) know whether it is unknown to the vendor. You might have some degree of belief about that proposition, but the available facts are limited.

Likewise, the decisions you make to defend your network may be different given your knowledge (or lack thereof) of vulnerabilities, their exploits, and patches. If you have been exploited, you have work to do regardless of the availability of a patch. Whether the vulnerability or exploit deserved the *zero-day* prefix does nothing to help you clean up (although it might help you save face when you get called onto the carpet to explain the attack). Similarly, the availability of a patch gives you a clear course of action regardless of whether you have been exploited or not.

However, from the perspective of someone creating a tool to perform penetration testing, vulnerability scanning, or vulnerability discovery, there is nothing intrinsic to what the tool does or the knowledge that it embodies that lets you distinguish between its *zero-day exploit capability* and its *exploit capability*. As I've shown, all the relevant definitions that could be brought to bear depend on extrinsic factors involving the state of knowledge of others.

In technical contexts, we eschew the use of *zero-day anything* not because it is colloquial but because it is imprecise. Imprecision leads to confusion in technical discussions, and in the current situation, laws and regulations count as technical discussions. Confusion increases costs by creating a drag on decision making. Confusion also leads to a chilling effect as would-be security researchers will avoid performing research that leads to ambiguous legal outcomes and risk of prosecution.

At best, the phrase *zero-day exploit* serves as an attention grabber since it implies that you should pay attention and take some sort of action in response. Using that phrase as a discriminating term as in "a policy of presumptive denial for items that have or support rootkit or zero-day exploit capabilities" puts individuals and businesses at risk of noncompliance due not to their malicious intent but rather to the incomprehensible wording of the regulation.

It is my conclusion that no definition of *zero-day exploit* is possible that refers only to concepts intrinsic to vulnerabilities, their exploits, and their patches. Thus any tool that supports or provides *exploit capability* must also be presumed to include *zero-day exploit capability* since the differentiating factors occur in the world, not in the tool.

Contact Us

CERT Coordination Center
Software Engineering Institute
4500 Fifth Avenue, Pittsburgh, PA 15213-2612

Phone: 412-268-7090
Web: www.cert.org
Email: cert@cert.org