Can Software Licenses Contribute to Cyberarms Control?

Steve Dierker
Freie Universität Berlin
Berlin, Germany
dierker.steve@fu-berlin.de

Volker Roth
Freie Universität Berlin
Berlin, Germany
volker.roth@fu-berlin.de

ABSTRACT

We discuss the potential role that software licenses can play in cyberarms control, the attribution of cyber attacks and the adherence to international humanitarian law and treaties that stipulate that the effects of war on the civilian population shall be minimized. We consider the increasing reliance of civilian and military institutions on FOSS and conduct a thought experiment: what would happen if a fraction of FOSS migrated to a license with non-military use clauses? If this caused civilian and military systems to diverge in design, and exploits and malware reflect their targets, then erroneous targeting of civilian institutions could be ruled out. This idea led us to perform an initial analysis of software dependencies and their relationship to the propagation of copyleft licenses. We analyzed packet manager data for seven different programming languages, based on data from *Libraries.io*. Among other things we found that a small number of packages accounts for the majority of the dependencies. The number of dependent packages varies from language to language and ranges from 79% in the case of Cargo to merely 3.9% in the case of Pypi. We also review existing non-military licenses and identify areas that need further research in order to understand the potential and the applicability of non-military licensing regimes.

CCS CONCEPTS

• **Social and professional topics → Licensing**; **Governmental surveillance**; *Codes of ethics*; • **Applied computing → Cyberwarfare**; *Investigation techniques*;

KEYWORDS

Software licenses, cyberwar, arms control, Geneva Conventions, FOSS

ACM Reference Format:
Steve Dierker and Volker Roth. 2018. Can Software Licenses Contribute to Cyberarms Control?. In *New Security Paradigms Workshop (NSPW '18), August 28–31, 2018, Windsor, United Kingdom.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3285002.3285009

1 INTRODUCTION

Computer systems and networks have played a role in warfare at least since the introduction of the *AirLand Battle Doctrine* [12] in the 1980's by the United States and its adoption by NATO. It was seen as a means to rapidly disseminate information to where it was needed in order to achieve situational awareness. In addition, computerized smart weapons were developed in order to improve the efficiency and precision of military operations. The trend to computerize and automate military conflict continues with the development of *Lethal Autonomous Weapon Systems* [25].

However, computer systems are not only a means to enhance and extend traditional military capabilities, they are targets of attacks themselves. Consequently, computer networks, specifically the internet, have become a battle ground on which *computer network operations* (CNO) take place. The term *cyberwarfare* is used to refer to offensive operations such as *computer network attacks* (CNA) and *exploitation* (CNE). These operations may take place even during peace time as a preparation for future conflicts, for example, by planting malware that remains dormant until activated during an outbreak of hostilities (cf. [6]). Information about software vulnerabilities, exploits and malware are the principal weapons of cyberwar.

The internet is primarily a civilian infrastructure and connects civilian institutions, municipal infrastructure and systems of military, particularly strategic, value alike. This puts CNO at odds with international humanitarian law and established military doctrine [9] that seek to minimize the effects of military conflicts on the civilian population. Beyond risking collateral damage, parties to a conflict may simply not adhere to international humanitarian law and may target civilian institutions and infrastructure deliberately with CNO. The potential effects are considerable while the execution can be relatively easy and cost-effective. For example, the Bush administration reportedly considered CNO on Iraq's banking system before the invasion of 2003, but “rejected the idea, fearing an unintended impact on global financial markets” [21].

Whistleblower Edward Snowden revealed that US and UK intelligence agencies target civilian institutions and infrastructure aggressively and extensively with CNE, in a clandestine fashion, in order to extend their intelligence collection. Other intelligence agencies certainly attempt the same and differ only according to their capabilities. CNE involves implanting malware that seizes control of its host computer and can be updated dynamically with arbitrary functions. Hence, it becomes almost impossible to draw a line between what has been, historically, accepted conduct between competing nation states, that is, espionage, and laying the groundwork for military operations during times of conflict. For this reason, we conflate military and intelligence institutions in our discussion.

A persistent risk of CNO is that the tools and techniques used for it may fall into the hands of criminal actors or terrorists. These tools need to be copied from compartmentalized systems to staging servers in order to be used against targets. Targets and third parties who have compromised a staging server may be able to obtain these tools. Worse still, CNO tools and techniques may be exfiltrated from the source and publicized. This happened to the NSA, whose tools were quickly picked up by criminal actors [18].

1.1 Limiting Armament

The UN Charter stipulates in Article 2 that *Members shall settle their international disputes by peaceful means* and that *Members shall refrain [. . .] from the threat or use of force against the territorial integrity or political independence of any state.* General and complete disarmament has been a declared goal of the international community but remains elusive [1]. Instead, states continue to maintain military forces to ensure their ability to self-defend, which the UN Charter permits in Article 51. However, armament increases the risk of conflict and the potential for escalation [1]. This is the security dilemma. Cyberweapons are particularly concerning because their use is difficult to prevent and difficult to attribute. They may be deployed in low-intensity conflicts for which conventional retaliation is disproportionate and likely to escalate the conflict further. Classic mechanisms for arms control, such as treaties, are difficult to apply here because it is unclear how the crucial element of verification may be implemented.

Besides treaties, a limiting factor of armament is the cost of weapons systems. Among the most expensive weapons are nuclear ones. Their producers are funded by financial institutions with hundreds of billions of Dollars [3]. Organizations such as PAX in the Netherlands campaign for financial institutions to divest from producers of nuclear weapons, they hope that this leads producers to cut nuclear weapons production from their business strategies [3]. Cyberweapons are cheap compared to kinetic weapons and delivery systems. Of course, skilled labor is needed as in other specialized areas but in terms of material investment a few computers are sufficient to get started. Countless developer-years worth of software is available to be leveraged and is continuously developed. Much of that software is free of costs and is developed and maintained by volunteers, for example, by enthusiasts and academic researchers. Even commercial software and its development often depends on free and open software. Without the benefit of free and open software, the costs of cyberweapons would increase, as would the costs for operations and for the development and maintenance of other types of weapons that rely on software components.

1.2 Diversification of Systems

Since civilian institutions and military organizations rely on a common set of operating systems, applications and other software, exploits developed for military purposes may work against civilian targets and vice versa. If military and civilian systems were developed independently then we could reasonably assume that exploits for one type of system would not readily work on the other type. This would have compelling conceptual benefits:

(1) The type of exploit (it works for a military system versus it works for a civilian system) would indicate the type of target intended by the creator and the user of the exploit. Observing military use of exploits against a civilian system would be indicative of a violation of international humanitarian law. In contrast to bombing a hospital [19] this can not be dismissed as an accident because exploit development requires deliberation and a precise target specification.
(2) A stolen cache of exploits developed for military targets could not be applied readily to attack civilian systems with criminal intent.

Of course, a complete diversification may not be possible. For example, the MIL-SPEC reform initiated in 1994 by then United States Secretary of Defense William Perry [14] sought a consolidation and reduction of military specifications in order to leverage greater cost efficiencies of COTS products and a recognition that COTS products may meet or even exceed military expectations in the field. MIL-SPECs focused on physical and operational requirements of military equipment as opposed to software. As a consequence, military computing equipment often contains COTS computing hardware and firmware that shares vulnerabilities with its civilian counterparts. *Meltdown*[1] and *Spectre*[2] are recent examples that come to mind and that affect CPU families produced by Intel and AMD. On the other hand, exploits often require the use of several vulnerabilities in conjunction. While some vulnerabilities may be shared between military and civilian systems, others may not. The fact that some vulnerabilities are shared therefore does not necessarily preclude all benefits that may be derived from diversified systems.

1.3 Non-Commercial Software Plays a Role

What if free and open source software was not available to the military and the intelligence community, directly or indirectly? They would have to procure their software entirely from commercial sources (commercial off-the shelf) or they would have to develop their own software in-house (government off-the-shelf). Commercial providers of military software in turn would have to raise prices if they cannot leverage free software. The costs of selling into both, the civilian and defense, markets would therefore increase, which benefits the competitiveness of companies that focus on either market. This increases the likelihood that development of civilian and military systems diverges and the two worlds become increasingly incompatible to each other, at least with regard to the exploits and the malware that affect the two types of systems.

The commercial sector does not have an incentive to cause such a market separation. For software companies, it is efficient to invest in the development of one product that is sold into both markets. At best one can hope that the product is customized to the specific needs of either market segment. However, these considerations play a lesser role in the case of software that is made available for free. Instead, there is room to base one's decision to whom the software is made available on ideological or humanitarian grounds. The principal means to limit who may use a software is the software license.

In this paper, we make a thought-experiment: what would happen if a fraction of free and open source software developers decided

[1]CVE-2017-5753, CVE-2017-5715
[2]CVE-2017-5754

to move their software to a license with clauses that prohibit the use of the software for military and intelligence purposes? This thought-experiment immediately leads to a number of questions:

(1) What would a non-military license have to look like and would it be enforceable?
(2) In what cases, if any, is it possible to move software from one license to a non-military one?
(3) How many developers would have to make such a switch and for which projects in order to have an effect?

In what follows, we begin to explore answers to these questions and steps that need to be taken to arrive at answers. Once we have answers we can begin to ask whether a necessary fraction of developers would indeed be willing to make such a license switch.

2 SOFTWARE LICENSE BACKGROUND

We can distinguish between proprietary software licenses and *free and open source* (FOSS) licenses. Proprietary licenses govern the use of proprietary software, which tends to be closed source, for example, Microsoft Windows. The license allows the licensee to use the licensed software in certain narrowly defined ways. Usually, this excludes the right to inspect or further develop the software or to make it available to third parties. We are not interested in this type of license and hence do not discuss it further. We rather focus on FOSS licenses.

The first FOSS license was written by Richard Stallmann and led to the foundation of the *Free Software Foundation* (FSF) on October 4th, 1985. The FSF is a nonprofit organization dedicated to advocating FOSS and it maintains a definition of FOSS. The initial definition [22] was given by Stallmann as follows:

> The word "free" in our name does not refer to price; it refers to freedom. First, the freedom to copy a program and redistribute it to your neighbors, so that they can use it as well as you. Second, the freedom to change a program, so that you can control it instead of it controlling you; for this, the source code must be made available to you.

The second major nonprofit advocacy group for FOSS is the *Open Source Initiative* (OSI). It was founded by Bruce Perens and Eric S. Raymond in February 1998. The OSI maintains its own *Open Source Definition*[3] (OSD) and approves licenses that it deems compliant. In order to comply with the OSD, the license must not discriminate against persons, groups or fields of endeavor, among other criteria. Whereas the FSF appears more focused on the moral issues of FOSS, the OSI takes a pragmatic view point [23, p. 31].

Multiple flavors of FOSS licenses exist. They are differentiated by the strictness of copyleft the license imposes on derived works. Copyleft ensures that derived works have to be published under the same license or with the same degree of freedom. The *Institute for Legal Questions on Free and Open Source Software*[4] distinguishes between five types of licenses, that is, licenses:

(1) without copyleft
(2) with strict copyleft
(3) with restricted copyleft
(4) with providing a certain choice
(5) with particular privileges

[3]https://opensource.org/docs/osd
[4]http://www.ifross.org

The first type, licenses without copyleft, provide licensees with all the freedom of a FOSS license but without any obligation to publish derivations of the software under the same license as the original. Licensees may modify the software and may publish derived works under a license of their own choosing. The *2-Clause BSD License* and the *MIT License* are examples of this type.

The second type are licenses with strict copyleft. They provide licensees with all the freedom of a FOSS license but require that derived works are published under the same license as the original. The best-known example of such a license is the *GNU General Public License* (GPL).

The third type are licenses with restricted copyleft. Such a license provides the freedom and obligations of a FOSS license with strict copyleft. Additionally it allows the distribution of derived works under a different license as long as the files of the original software remain unchanged. This type of license is meant to enable the bundling of FOSS software with non-free software. The best-known example of such a license is the *GNU Lesser General Public License* (LGPL).

The fourth type are FOSS licenses that provide licensees with choices of how to distribute derived works. The fifth type are FOSS licensees that reserve privileges for the licensor in case that the licensee produces derived works. This type of license is often used when a proprietary software is turned into FOSS. The last two types of FOSS licenses are used the least.

3 NON-MILITARY LICENSES

Probably the best-know license with a non-military clause is the *JSON License,* published in 2002 by Douglas Crockford.[5] It stipulates that "The software shall be used for Good, not for Evil." This vague phrasing caused lawyers headaches because neither "good" nor ”evil" is particularly well-defined [7, from 16:00]. Therefore, it is necessary to clarify with the licensor on an individual basis whether the intended use of the software is indeed "good" and not ”evil." The well-known FOSS project Debian refuses to distribute software under the JSON License because it is not in compliance with the OSD. It violates the criterion of not discriminating against a field of endeavor of which doing evil apparently is one. In that spirit it is remarkable that Google (now Alphabet) refuses to host software under the JSON License as well, for the same reason. Google is the company that famously chose "Don't do evil" as a motto for its corporate code of conduct. One might think that the JSON License reflects that motto and is compatible with what Google strives to achieve but apparently it is not. More recently, Google is making headlines because several thousand Google employees wrote a letter of protest to Google's Chief Executive Officer Sundar Pichai [20], urging Google to withdraw from a project with the Pentagon that is said to build "warfare technology."

An example of a software license with an explicit non-military clause is the *Software Libre para Uso Civil* (SLUC) published 2006 in Spain.[6] The SLUC builds on the GPL but prohibits military personnel from using the software, it prohibits the use of the software

[5]https://www.json.org/license.html
[6]https://en.wikipedia.org/wiki/SLUC

for the manufacture of offensive weapons (not for defense) and it prohibits the use of the software by organizations that are either controlled by military organizations (more than a 51% of shares) or whose customers are dominantly military (more than 51% of clients).

Another example is the *Light++ License*,[7] which "prohibits use of the software by employees of military or defense-related organizations, or within facilities producing weapons or conducting research on weapon design." Yet another example is the *Qabel Public License Version 0.2*,[8] which excludes the use of the software for "military, intelligence or related purposes, including but not limited to intelligence and military research."

Another non-military license is due to Phillip Rogaway.[9] It defines military use as "any Use by, in cooperation with, on behalf of, or paid for by" a number of explicitly stated U.S. military and intelligence institutions and their foreign counterparts.

The SLUC, Light++ and Qabel licenses are all copyleft licenses. The license of Phillip Rogaway is for use of intellectual property rather than a specific software implementation. Therefore, the notion of copyleft does not apply to his license. The license the second author used for his *Easy Encrypt* tool builds on a BSD license and thus is an example of a license without copyleft. It stipulates the following:

> Military and Intelligence organizations, their Affiliates and individual Members of these organizations and Affiliates are excluded under penalty of law from this License in regard to this code irrespective of their purpose, including for private, non-commercial, commercial, governmental, research and educational purposes.

The pertinent terms are defined as follows:

> Affiliates means all entities which are controlled by a military or intelligence organization, whether directly or through one or more intermediaries. For purposes of this definition "controlled" means ownership of securities representing more than fifty percent of the voting capital stock or other interest having voting rights with respect to the election of the board of directors or similar governing authority, or any other power by contract or in any other form which entitles such named entity to the respective voting rights.

In what follows we point out questions about non-military licenses that need to be answered. Some answers appear straightforward, others require further research.

3.1 Can there be Non-Military FOSS Licenses?

The first question is whether a license with a non-military clause can be a FOSS license. The answer is "No" when judging by the criteria of the FSF and the OSI. The FSF intends to provide "the freedom to run the program as you wish, for any purpose." The OSD specifies as its fifth criterion that "the license must not discriminate against any person or any group of persons." The sixth criterion stipulates that "the license must not restrict anyone from making use of the program in a specified field of endeavor." Excluding the military or its contractors from the license discriminates against a group and it discriminates against a field of endeavor. Hence, a non-military clause cannot comply to the FOSS definitions of the FSF or the OSI. Thus, non-military licenses need a novel "brand" of their own that captures the salient points of FOSS licenses while restricting military use in suitable ways.

3.2 Can FOSS become Non-Military?

The second question we need to consider is whether existing FOSS can migrate to a non-military license and, if so, how. Of course, any copyright holder can decide at his discretion to publish new versions of his software under a new license. The situation becomes more complicated if the copyright is shared, for example, if a software is a derivation of another software that is governed by a FOSS license.

Migrating FOSS licenses without copyleft to a non-military license is straightforward. It suffices to publish a derivation of the software under the new license. Of course, a non-military license without copyleft can as easily be converted back to a license without the non-military clause. This return path can be closed by publishing the software under a non-military license with copyleft. Although, the original FOSS version remains available under the FOSS license and may be developed further under that license. It is therefore necessary that the developer community commits to maintaining the non-military version rather than the original FOSS version.

There is not a general migration strategy for FOSS with copyleft because the copyleft principle requires that derived works are published under the same license. However, there is a theoretical loophole. Some FOSS is licensed according to "GPL version x or later," where x is a version number. The phrase "or later" means that the applicable license is always the most recent version of the GPL. The most recent version of the GPL is 3. If the FSF decided to create a version 4 GPL that includes a non-military clause then all software licensed according to a prior version "or later" can be distributed under the terms of that clause. Compelling as this might be, it is unlikely that the FSF would do that – and it would without doubt cause an intense debate – because the clause still violates the FOSS principle of non-discrimination and the FSF freedom to distribute copies of the software to others.

It needs to be determined on an individual basis whether or not there exists a migration path for FOSS licenses of types 3 (restricted copyleft) to 5 (particular privileges).

3.3 What is a Good Non-Military Clause?

The third question in need of an answer is what needs to be in the non-military clause of a software license. In any case, the clause must be legally sound. In what follows, we analyze the licenses we described in Section 3 and sort them loosely into different categories according to what the basis for the exclusion is.

Based on intent (end goal). The approach taken by the JSON License is too vague because neither "good" nor "evil" is easily defined. The terms would have to be interpreted in the light of each individual licensing case, jointly with the licensor. Even worse, courts would interpret vague terms against the licensor according

[7] https://www.openhub.net/licenses/LightPlusPlusLicense
[8] https://qabel.de/en/license
[9] http://web.cs.ucdavis.edu/~rogaway/ocb/license2.pdf

to the *contra proferentem* principle.[10] One could choose "intention to harm a human" as a more precise formulation of intent. This wording covers harm inflicted by anyone, not just harm inflicted by the military. However, establishing intent (as a legal standard) can be difficult to meet in a legal dispute and the question arises as to who has to carry the burden of proof. Requiring the licensor to carry that burden does not seem to scale well. On the other hand, how would a licensee prove that it had no intention to harm a human? In the case of the military, harming humans (and objects) is an intended effect of military force, that is, the application of violence to achieve political goals. Based on that reasoning, military organizations should always be excluded. Arguably, it is preferable to have a simple legal test that can be made unequivocally over creating legal uncertainties that risk missing the larger goal, which is limiting preparations for large wars and a reduction of risks to civilians.

Based on organization type. The exclusions defined in the Light++ License and the Qabel License are more explicit and more precise than the JSON license. They speak of military and intelligence organizations and purposes and of uses for weapons research and development. Still, there exists a continuum of involvement with the military and intelligence. Without further interpretation on a case-by-case basis one risks an exclusion that is exceedingly broad.

The SLUC License is already very precise in the definitions of organizations that are excluded from the license. For example, it excludes organizations of which the military holds more than a 51% share, and it excludes organizations with a customer base of which more than 51% are military organizations. A military organization is defined as being subject to military legislation.[11] However, this definition does not account for indirect control through holding companies. This would enable military organizations to launder their controlling position for the purpose of licensing. Furthermore, shares can sometimes be distinguished into voting stock and preferred (non-voting) stock. By not distinguishing between the two in a license we risk an exclusion that is broader than intended because the military might have obtained a majority economic interest in a company without having a controlling (voting) majority. On the other hand, the military may have a controlling majority without having a majority of the economic interest, which means that the definitions is not strict enough to achieve its intended effect.

The exclusions in the Easy Encrypt license are perhaps the most precise we encountered. They cover the cases of indirect control and voting stock. The exclusion is also the broadest of the examples we encountered. It excludes even the private use of the software by excluded individuals, for example, the employee of a defense contractor. It remains to be seen to what degree such an exclusion is compatible with existing law. For example, Directive 2009/24/EC of the European Union declares that any contractual provisions that prohibit a person from "performing acts necessary to observe, study or test the functioning of the program" are null and void [13]. However, the Directive grants these exceptions only to persons who have a right to use the software, that is, users who have *legally licensed* the software. Since excluded individuals are not and cannot be licensees they do not enjoy the provisions of the Directive. U.S. Code, Title 17, § 1201 (Circumvention of copyright protection systems) contains similar provisions, which grant limited exceptions from end-user license agreements to legal users of a software. However, in contrast to the Directive it contains a broad exception for intelligence and other government activities in Section (e). Therefore it seems that U.S. Government affiliates do not violate 17 U.S. Code § 1201 by inspecting the software governed by the license, as long as they act on behalf of the U.S. Government. Although, the license may still exclude private uses of the software by affiliates of the U.S military and intelligence. Other jurisdictions may have different laws and different exceptions, still.

Based on application. References to weapons and weapons production, as found in the SLUC and Light++ Licenses, are ambiguous as well. For example, one might ask whether intrusion software such as *Metasploit* counts as a weapon. This does make sense given its application to CNO. If exploits are not considered weapons then the license does not cover a core ingredient of cyberwarfare. Furthermore, "intrusion software" is a controlled technology according to the *Wassenaar Arrangement* [24]. On the other hand, researchers and pen testers are users of *Metasploit.* Furthermore, the Wassenaar Arrangement exempts technology "in the public domain" and "basic scientific research." By this standard, *Metasploit* would be exempt because versions of it are FOSS and hence in the public domain. Which interpretation should one follow?

Another question that arises is whether any support system should be considered a part of the weapons production system. Consider for example a software that is used to keep track of the working hours of cleaning personnel who cleans the offices of engineers involved in the production of weapons. The cleaners are employed by a service company that services civilian and military facilities. One might argue that the Light++ License excludes use of the software if it runs inside the facility. If the software is governed by the Qabel License then its use appears permissible. The upside of focusing on weapon systems is that such an exclusion still allows for positive uses of the licensed software within the military. For example, the *US Army Corps of Engineers*[12] provides public engineering services along with military ones.

Exclude the Military or their Targets? The language of existing non-military licenses seeks to limit the uses of a given software by the military. This matter of principle approach speaks to the idealism of the licensors. In this paper, we hypothesized a specific effect that can be sought by a non-military licensing regime, that is, the diversification of systems and their attack vectors so that legitimate and illegitimate target systems of cyberwar operations are independent. From that perspective, it may make sense to *limit the use of a software by legitimate targets of military operations* rather than the military itself, with the understanding that every military is also a legitimate target of its opponent's forces. This brings about the challenge of defining tie-breakers for dual-use goods and services. For example, a large internet service provider (ISP) may provide internet access to large parts of the civilian population but also to military installations. Should the ISP be allowed to use a software that is not meant to be operated by a legitimate target of

[10] UNIDROIT Principles 2016, Article 4.6.

[11] We used Google Translate to translate the Spanish original into English for better interpretation.

[12] http://www.usace.army.mil/

a military? Similar considerations hold for any essential support service such as food, utilities and power, whose denial has grave effects on civilians.

Take-away. In Figure 1, we contrast the three license design approaches we discussed before using two arguments in favor and against each approach. None of the licenses we discussed offers any provisions that would allow exceptions for humanitarian missions. We also shared the idea that the targets of military operations might be a worthwhile "target" of license designers rather than just the military. The question of what a good non-military clause is, remains open. A debate is needed on two levels. First, a debate about what the intent of the clause should be and, second, how it must be formulated to achieve this intent.

3.4 Exemptions for Humanitarian Missions

Humanitarian missions can be divided into humanitarian aid and humanitarian intervention. The former consists of material and logistic assistance after a humanitarian crisis, for example, natural disasters and disasters caused by humans. The latter involves the deployment of armed forces to sovereign states with the intention to end or alleviate mass human suffering within that states' borders. In order to exempt humanitarian missions it is necessary to determine whether a mission is indeed humanitarian. In fact, three questions arise immediately:

(1) Who determines whether a mission is humanitarian?
(2) When is the determination being made?
(3) Does the determination meet legal standards?

Neither of these questions is easy to answer. The humanitarian status of several interventions is controversial in hindsight. For example, the US government justified its Iraq war of 2003 based on claims that Iraq held weapons of mass destruction (WMD). Only after the invasion did it turn out that these claims had been falsified [16]. The Libya intervention of 2011 was based on UN Resolution 1973. A discussion is still ongoing whether the intervention was just [2]. In the beginning of 2018, an investigation began to look into the ties that Nikolas Sarkozy, then president France, had to Libya before 2011 because he may have had a personal interest in the intervention [5]. If we accept, for example, the UN as the authoritative and legally binding source of the determination of a mission's humanitarian status, this status may still be challenged post-hoc and intervening states may turn out not to have acted in good faith. If the prior determination of humanitarian status is binding then no remedy exists against states not operating in good faith, which is unsatisfactory. On the other hand, if a state acts in good faith then it must sill consider the risk of being sued for license violations post-hoc if the humanitarian status of the mission is challenged. All this creates considerable uncertainties for the licensees and licensors of software governed by a non-military license with humanitarian exemptions.

It is worth asking whether such exemptions would play a role in practice. Humanitarian missions are a small part of the tasks of armed forces. On the other hand, serious risks arise from the preparation of large wars, specifically those including cyber attacks, with the associated destabilization of international security and world peace. We argue that the larger goal of working towards demilitarization should take precedence over humanitarian applications of military force when it comes to deciding upon a non-military licensing regime. This does not necessarily imply a limitation on humanitarian missions. The military can still carry out humanitarian missions using the systems it has which, by assumption, are not governed by a non-military license. Hence not much would be lost by avoiding the legal uncertainties that surround exemptions for non-military clauses for humanitarian purposes.

3.5 Are Non-Military Clauses Enforceable?

The fourth question we must address is whether a non-military license is actually enforceable. Sceptics might worry that military and intelligence institutions may violate licenses with impunity and that their illegitimate use of unlicensed software may be difficult to prove because of the secrecy that shrouds them. However, we doubt that these are fundamental concerns in liberal democracies.

Court cases in the U.S. suggest that the Department of Defense indeed engages in software piracy occasionally but it does pay fines for it or pays for settlements. In 2012, U.S. company Apptricity Corp. filed a lawsuit against the U.S. Army for software piracy and finally settled the case for $50 million [11]. In another case, the U.S. Navy obtained 38 licenses of a software copyrighted by the German company *Bitmanagement Software GmbH.* Subsequently, the U.S. Navy copied and used the software in an unauthorized fashion in more than half a million cases, according to a lawsuit[13] filed in the U.S. by *Bitmanagement* [10]. The U.S. Navy admitted the numbers but denies that this occurred in an unauthorized fashion. The case appears to be still ongoing.

The aforementioned cases pertain to proprietary software licenses rather than FOSS. However, U.S. court cases clarified that the GPL is an enforceable set of copyright terms *(Jacobsen vs. Katzer)* and an enforceable contract *(Artifex vs. Hancom).* A pending lawsuit between Artifex Software, Inc. and Hancom, Inc. over a violation of the GPL has reached a settlement meanwhile.[14]

Of course, the aforementioned cases do not answer affirmatively the question whether a non-military license with FOSS characteristics would be enforceable. Moreover, it is an open question what the fines would be for violating the license of a software that is otherwise available without cost, and whether these fines are a sufficient deterrent. However, judging by unclassified internal communication, the U.S. Department of Defense takes FOSS and the associated licenses seriously [26]. Of course it stands to reason that military institutions in other countries may not have to fear comparable scrutiny from their own countries' judicial system.

On a final note, a German idiom says that "wo kein Kläger, da kein Richter," which means that someone has to sue before a judge can come to a verdict. For non-military licenses to be enforceable, particularly if the licensed software is otherwise free of cost, there still has to be someone, an individual or an organization, motivated to challenge the license violation in a court of justice.

[13]U.S. Court of Federal Claims, case 16-840 C, filed July 15, 2016.

[14]https://artifex.com/news/artifex-and-hancom-reach-settlement-over-ghostscript-open-source-dispute/

Who uses it?
Example: Military and contractors
In favor:

(1) Focused, broad
(2) Unambiguous test

Against:

(1) Arbitrary thresholds for control
(2) Setting threshold is challenging

What is made with it?
Example: Weapons
In favor:

(1) Focused, narrow
(2) Positive uses possible, e.g., USACE

Against:

(1) Dual-use software ambiguous
(2) Support functions ambiguous

What is intended by its use?
Example: Harming humans, doing "evil"
In favor:

(1) Independent of who uses it
(2) Independent of what is made with it

Against:

(1) Intention difficult to prove
(2) Exceptions warranted, e.g., surgery

Figure 1: Contrasts reasons in favor and against various approaches to define military exclusions in software licenses. The acronym USACE refers to the United States Army Corps of Engineers, which provides public engineering services along with military ones.

3.6 Can License Violations be Detected?

Yet another question is how violations of a non-military license can be uncovered despite the secrecy that shrouds military and intelligence organizations. In the U.S., the *Freedom of Information Act* can probably be used to obtain information on the kinds of software used by the Department of Defense. In Germany, the *Informationsfreiheitsgesetz* can probably be invoked to inquire about the FOSS software being used by government entities. We are unaware to what degree other countries offer comparable information freedom laws.

3.7 Are License Violations Likely?

Military institutions tend to procure systems from larger "mature" companies with the express or implied expectation that supplies and support for these systems will be available for a prolonged time. Of course, smaller and younger companies may still sell products and services to the military, for example, if their products and services are either unique or interchangeable. Mature companies tend to have a certification process that all their software products need to pass. In the course of that process, legal staff assesses whether the licenses that govern the product's components are compatible and aligned with the company's business objectives and those of their customers. Large customers often have a matching process that makes the same determination, independently. Therefore, mature companies will likely avoid a dependency on software whose license prevents them from meeting their military customers' requirements or exposes them to a legal risk [15]. Hence, the question whether non-military clauses are enforceable may not likely pose itself often in practice because non-military software will not be integrated into commercial products to be sold to the military in the first place. Once again, this is probably true for democratic rule-of-law countries. We cannot predict whether this will be true, for example, in autocratic countries unless international pressure assures compliance.

3.8 Effects of License Non-Compliance

It stands to reason that some state A may gain an advantage over state B by ignoring the provisions of non-military licenses if state B remains compliant. If this is so then state B has incentives to not comply with the license provisions either, which puts the value of what we propose into question. The hypothetical advantage manifests from one of two actions:

(1) Civilian systems of state A use software licensed for use in the military.
(2) Military systems of state A use software it is prohibited to use under a non-military provision.

The expectation of state A would be that its civilian or military systems are better protected than they would be if A remained compliant.

It is unclear how action 1 would lead to an advantage. Clearly, state B has an incentive to develop exploits against the military systems of A. Collecting information on these systems would certainly be easier rather than more difficult compared to the alternative. Furthermore, exploits would work against civilian and military systems of A, which is more cost-effective than the alternative and provides exactly the pretext for attacks on civilian systems that we seek to minimize.

Action 2 is more promising. State A may be able to develop its military systems faster and in a more cost-effective fashion than in the alternative case. This may yield a strategic advantage. At the same time, its military systems inherit the weaknesses of the civilian systems and obtaining information on the military systems becomes easier, as in the case of the first action. Crucially, state B once again has a pretext to seek cyberweapons against the civilian systems of state A, which renders the civilian population of state A less safe. Furthermore, the same cyberweapons would work against the civilian systems of a third state C whose civilians rely on the same software, even if C complies with the non-military provisions of software. At least to us it is unclear how A would benefit from this consequence, strategically.

It appears that action 2 may be beneficial to A if the strategic advantage gained from the use of non-military software in military systems outweighs the risk this decision poses to the civilian systems of A. Of course, autocrats may have little regard for their civilians when it comes to obtaining military power and they may trust that their opponent does not intend to target their civilian systems. However, the price they pay is less security for their own military systems. First, state B has easier access to the information necessary to develop exploits. Second, criminals will still develop exploits against civilian systems and will likely sell their tools and exploits to state B if the demand arises and the price is right. On top of that, state A exposes its military systems to the very same criminals even in the absence of any demand by B, that is, during peace times. Lastly there is a risk that military-grade exploits are lost or stolen and are used by criminals on civilian targets, as has happened before [18].

3.9 Future Work on Non-Military Licenses

A number of non-military software licenses exist, which take different approaches towards a similar goal. The impression they give is still ad hoc. We consider it an interesting question to think this topic through and to develop a "second generation" non-military license portfolio. Different developers may have different preferences when deciding upon their licensing choices. The license portfolio needs to capture the majority of these preferences, which poses the question of what they are. Examining the distribution of existing licenses may yield useful insights. For example, is the license distribution of software that is of interest to the military biased towards particular licenses, for example, GPL versus MIT?

A second question of interest is how existing licenses can be combined with flavors of non-military licenses and what approach should be taken to combine them. There appear to be two general approaches, as Nathan Reitlinger pointed out in the course of a collaboration that ensued from NSPW 2018:

(1) **Modular.** Non-military licenses may augment and modify established licenses in the form of an add-on. When present, the add-on would have to take precedence over conflicting clauses in the modified license.
(2) **Stand-alone.** Non-military licenses may include desired subsets of established licenses while leaving out clauses that conflict with the non-military provisions.

It seems that stand-alone licenses would be harder to maintain because changes to established licenses would have to be tracked and ported into the corresponding non-military license whereas modular licenses may simply latch on to established licences.

Another important question to be addressed is the ways in which non-military licenses combine in the case of software dependencies. In a simple case, the licenses have a hierarchical ordering of "strictness." In other words, software *A* and *B* can be combined under license *B* if and only if everything that is excluded under license *B* is also excluded under license *A*. A more general case would impose a lattice structure on licenses, that is, for licenses *A* and *B* there would have to be a license *C* (the least upper-bound) such that *C* excludes everything that *A* or *B* excludes. The ways in which licenses combine needs to be "psychologically acceptable" (cf. [17]). Hence, license portfolio design also poses usability challenges that need to be addressed in suitable ways.

A challenge in the design of non-military licenses is to carefully delineate desired and undesired uses of a software. At the same time, the license language must be applicable internationally, that is, the language cannot be tied to the institutional structure of a particular state.

Another interesting idea that was raised by Eireann Leverett at NSPW 2018 is to specifically furnish the community of hackers and security professionals who develop exploits (and pentesting tools) with licenses that prohibit the use of their exploits by the military. Anecdotal evidence of a demand along those lines can be found in the form of a disclaimer in the Github repository of *Hydra*,[15] a tool to audit the password security of a plethora of different services. On September 22th, 2018, the disclaimer reads "Please do not use in military or secret service organizations, or for illegal purposes." Penetration testing tools are of particular interest because some of them are used by military and intelligence organizations and they have an immediate impact on cyberwarfare.

[15] https://github.com/vanhauser-thc/thc-hydra

4 RIPPLE EFFECTS DUE TO DEPENDENCIES

One important aspect of the *copyleft* provision is that it requires derived works to be licensed under the same license. A derived work is software that extends other licensed work or simply includes it as a dependency. One may therefore ask how many software projects must convert to a non-military license with copyleft in order to cause a particular fraction of all dependent projects to convert to a non-military license as well.

Towards this end we decided to perform an analysis of current open source projects. We used public datasets from *Libraries.io*, which cover the projects from 36 package managers and public repositories from 3 social coding platforms. In total these are over 2.7 million unique open source projects, 33 million repositories and 235 million interdependencies between them.

The dataset is released as an archive containing multiple CSV files, each representing one table of a relational database. We used release 1.2.0 from March 13, 2018, which is 52G in size. For our analysis only two files were of interest, the projects file and the dependencies between projects (*dependencies.csv, project_dependencies.csv,*). Since each file contains additional information, for example, the *Homepage URL,* we filtered them and extracted only the information on license use and dependencies (attributes *Project ID* and *Dependent Project IDs*).

We analyzed license usage for the entire dataset and selected seven popular package managers for a detailed analysis: Cargo for Rust, CPAN for Perl, Maven for Java, NPM for NodeJs, NuGet for .NET, Packagist for PHP and Pypi for Python.

In our analysis we interpret a package as a vertex in a directed graph. Two vertices α and β are connected by the directed edge $\alpha \rightarrow \beta$ if and only if package β depends on package α. We also say that α *captures* β and β is a *captive* because β may have to comply with the license terms of α. The transitive hull of α is the maximal subgraph that can be reached from α by following outgoing edges. In this graph we can label each vertex with its associated license if its license can be converted to a non-military one, and give it an empty label otherwise. A vertex is particularly attractive if it has a non-empty label and a large outdegree.

4.1 Dataset Analysis

Table 2 shows the statistics of the most relevant licenses in our dataset. The most relevant licenses without copyleft were MIT, Apache v2.0, ISC, BSD 3-Clause and BSD 2-Clause. The most relevant licenses with copyleft were the GPL v2 and GPL v3. Overall, 58.3% of the packages are governed by a license without copyleft and could be migrated to a non-military license. All other packages are governed by a license of types 2 to 5 and cannot easily be migrated to a non-military license.

For our detailed analysis, given a graph *G*, we extracted all vertices with an outdegree greater than zero, that is, all projects on which other projects depend. We ranked these vertices in a greedy fashion. The greedy algorithm begins with a set of selectable vertices $\mathcal{P}$ and an empty set of selected vertices $\mathcal{D}$. In each step, the algorithm chooses a vertex π from $\mathcal{P}$ that has a transitive hull of maximum size. It removes the transitive hull of π from $\mathcal{P}$ and adds it to $\mathcal{D}$, outputs $|\mathcal{D}|$, recalculates the transitive hulls of all vertices in $\mathcal{P}$, and repeats until $\mathcal{P}$ only contains vertices with an outdegree of zero. The vertices in $\mathcal{D}$ are the "captives," that is, the set of vertices that can be captured by converting fewer than $|\mathcal{D}|$ vertices. A *core* of $\mathcal{D}$ is a set of vertices that is necessary and sufficient to capture all of $\mathcal{D}$.

Note that the greedy algorithm does not necessarily find the optimal solution, that is, the minimal selection necessary to compute the captives. Finding the optimal solution amounts to solving an instance of the *maximum coverage problem,* which is known to be $\mathcal{NP}$-hard. Therefore, our distribution only yields a sufficient number of packages, not the necessary number, that one must migrate to a non-military license in order to convert any given fraction of the captives.

Figure 2 shows the resulting distributions. The ordinate specifies the fraction of captives that is captured by the number of vertices given on the abscissa. Note that the graphs do not show the fraction of all packages, which includes packages without dependent ones. The tail end of the distribution must be captured on a 1-for-1 basis. Table 1 gives descriptive statistics of the datasets, for example, the number of captives, the size of the core, percentiles and the overall number of packages.

For all package managers, the graphs show that there exists a comparatively small number of influential packages with large outdegrees. These packages capture a large fraction of dependent packages. In the case of Pypi, 10 packages capture more than 65% of all captives, and 208 packages capture more than 90%. In the case of NuGet, 10 packages capture more than 93% of the captives, and only 4 packages are necessary to capture more than 90%. For all other package managers, one package already captures more then 90%. More than 77% of Cargo (79%) CPAN (78%) and NPM (77%) packages are captive, and less than than 59% of Maven (42%), Packagist (51%) and NuGet (58%) packages are captive. For Pypi packages, only 3.6% are captive. The maintainers of the remaining (noncaptive) packages must be convinced to adopt non-military licenses on an individual basis.

Manager	90%	95%	core	used	captives	all
CPAN	1	1	106	10978	27488	35132
Cargo	1	1	176	5046	11382	14499
NPM	1	1	167	269325	540445	699243
Packagist	1	1	320	35913	100533	198815
Maven	1	23	175	39379	57117	135486
NuGet	4	27	187	28754	69034	118623
Pypi	208	320	432	2646	4494	125719

Table 1: Shows dataset descriptive statistics. *All* indicates all packages. *Used* indicates packages that are used by at least one other package. *Captive* indicates the maximum number of packages that can be captured with cost less than one. A cost of one means one package has to be converted to capture one package (no uncaptured dependent packages are left). *Core* indicates the maximum number of packages necessary to capture all captives. The percentiles indicate how many packages must be converted to capture x% of all captives.

License	percent	License	percent
MIT	36.5	BSD 3-Clause	3.9
Apache v2.0	10.9	GPL v3	1.7
ISC	6.1	BSD 2-Clause	0.9
Other	4.6	GPL v2	0.8

Table 2: Shows the distribution of licenses among the projects in our dataset in percent.

5 DISCUSSION AND CONCLUSIONS

Note that the *Association of Computing Machinery* (ACM) expects of its members that they behave ethically and take a moral stand if need be.[16] One of the moral imperatives explicitly mentioned in the ACM's Code is "avoid harm to others." While the Code does not speak about war in general or cyberwar in particular, some ACM members might still want to position themselves on these issues by way of the licenses they choose for their own software. Using an accepted and well-crafted non-military license that otherwise seeks to maximize freedom in the spirit of FOSS, but is distinct from FOSS licenses, would be one way to do that. The idea to do that might be viewed as symbolic or even naïve. However, judging by the existing examples of non-military licenses it appears that some individuals and groups do have a desire for their software not to be associated with military and intelligence activities. Whether an idea is naïve or an idea whose time has yet to come is often only clear in hindsight. What is clear is that FOSS is a force to be reckoned with inside and outside the military. Already in 2003, MITRE conducted a study [4] on behalf of the Defense Information Systems Agency (DISA) on the importance of FOSS for the U.S. Department of Defense (DoD). We quote liberally from the executive summary:

> "To help analyze the resulting data, the hypothetical question was posed of what would happen if FOSS software were banned in the DoD."
>
> "The main conclusion of the analysis was that FOSS software plays a more critical role in the DoD than has generally been recognized."
>
> "Taken together, these factors imply that banning FOSS would have immediate, broad, and strongly negative impacts on the ability of many sensitive and security-focused DoD groups to defend against cyberattacks."
>
> "Research would be impacted by a large to very large increase in support costs, and by loss of the unique ability of FOSS to support sharing of research results in the form of executable software."
>
> "To the contrary, the combination of an ambiguous status and largely ungrounded fears that it cannot be used with other types of software are keeping FOSS from reaching optimal levels of use."

[16]https://www.acm.org/about-acm/acm-code-of-ethics-and-professional-conduct

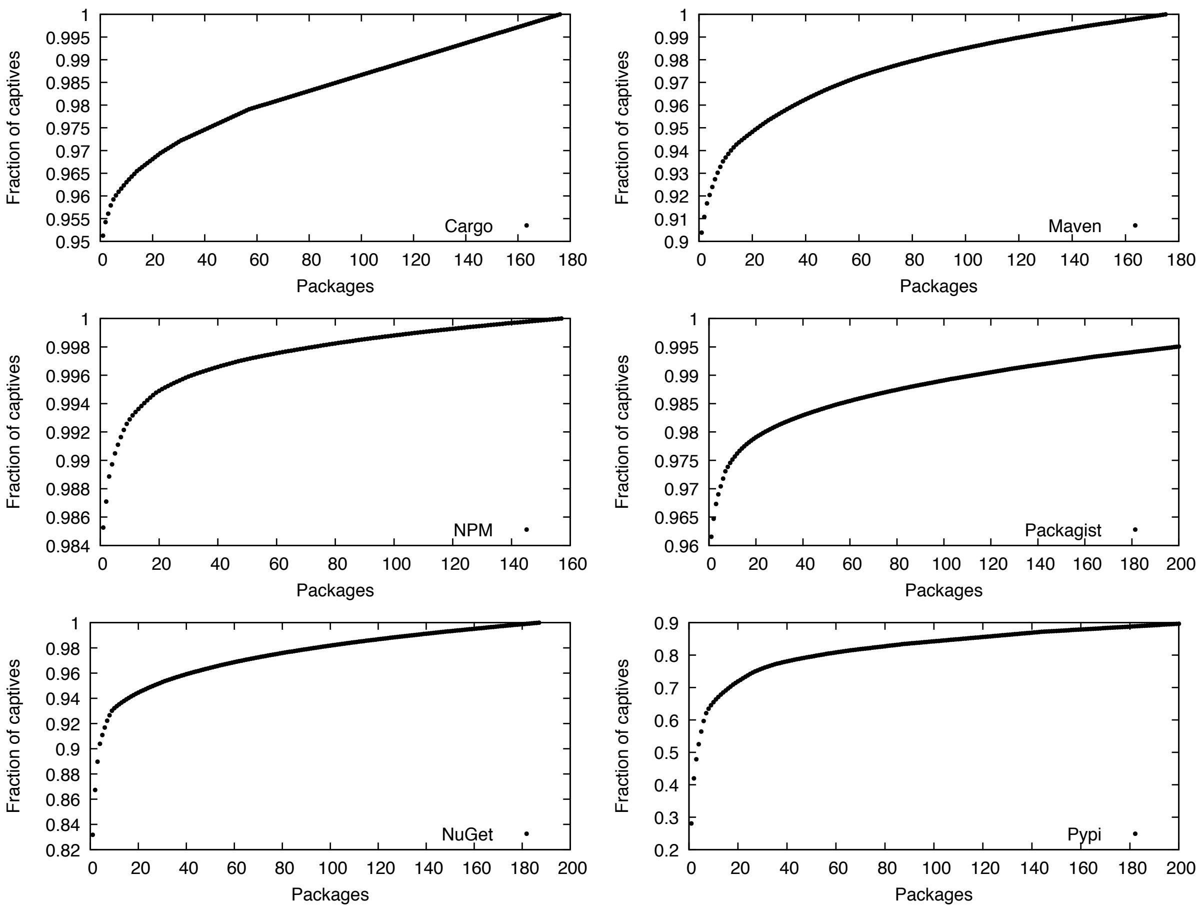

Figure 2: Shows what fraction y of captive projects (ordinate) can be affected by changes in x projects (abscissa). Projects are ranked in a greedy fashion. For any given point (x, y) the project at $x + 1$ produces the greatest increase of y. We show graphs for six of the seven dependency managers we analyzed in order to keep the columns balanced. We omitted the graph for CPAN because it has the smallest core and is very similar to the graph of Cargo.

In 2018, fifteen years later, it is probably safe to assume that the reliance on FOSS has increased rather than decreased, particularly since the U.S. government meanwhile embarked on a broad FOSS strategy itself.[17] According to a recent report of the U.S. Government Accountability Office [8], "DOD reports state that many weapon systems rely on commercial and open source software." We contend that if some of the FOSS ecosystem migrated to non-military licenses then this may have considerable impact on the DoD and perhaps its counterparts worldwide who certainly leverage FOSS as well.

Towards an assessment of the impact that license changes may have on FOSS software we analyzed popular package managers. Our analysis of package dependencies necessarily looks only at one case because the number of possible configurations is 2^n for n core packages. Since the worst case is not very interesting we analyzed the best case configuration, that is, the most influential packages convert to a non-military license with copyleft. This yields the greatest effect with the least number of conversions. Since computing exact solutions is $\mathcal{NP}$-hard we settled for a heuristic solution, the greedy computation. The results show that a small number of influential packages suffices to capture most captive packages. However, the fraction of all packages that is captive in this fashion varies among package managers, which suggests that a significant number of packages would have to be converted to a non-military license on a one-by-one basis (almost 97% in the case of Pypi). However, this does not yet answer the question what packages are the most *interesting* ones to convert, that is, the packages of greatest interest to military users and military targets. It would therefore make sense to look specifically at the dependencies of projects with known origin in defense departments. Prominent examples would be the projects listed at code.gov, which are provided by the US Government and which are hosted at Github. The *Libraries.io* dataset does contain information on Github repositories and their dependencies on packages. In order to analyze these dependencies, the repository graph must be merged with the individual package

[17]https://code.gov

graphs and analyzed jointly. This yields a fairly large graph that we have not proccessed at the time of writing.

It is important to note that the odds of converting even a small number of specific packages to another license, say 10 of them, are significantly smaller than converting an arbitrary set of 10 packages. It appears that, in order to have an effect on the diversification of systems, non-military licensing would have to become a trend not unlike FOSS itself. On the other hand, we have only started to investigate the impact that software dependencies have on software licensing. More research on this subject will be helpful to better understand the potential and limitations of "contagious" licenses in today's software ecosystem. In other words, perhaps non-military licensing can be more than merely an idealistic statement if a modest number of influential developers decide to "flip the switch." We believe this could justifiably be called a paradigm shift. A motivation to doing so, besides signaling a moral position on an important societal issue, might be the expected divergence of civil and military systems. Combatants in a cyberwar would have to commit to their types of targets in a fashion that is amenable to forensic analysis and a firm conclusion. Because exploits and malware necessarily reflect the specifics of the systems they are designed to attack there would be little room to dismiss the targeting of civilian institutions as accidents. This ups the ante for successful attribution because international humanitarian law might be applied more effectively as a consequence.

Besides further analysis of software and license dependencies a number of principal and legal questions arise. For example, in what terms should military use be defined? Should it be where a software is used, by whom it is used, on whose behalf it is used, for what it is used or any combination thereof?

ACKNOWLEDGMENTS

The *Easy Encrypt License* has been developed jointly by Volker Roth and Dr. jur. Andreas Schoberth, Head of Legal Affairs in Research & Transfer at Freie Universität Berlin. We thank Jürgen Altmann, Ingo Ruhmann and Ute Bernhardt for reading drafts of our paper and for suggesting improvements. We would also like to thank the anonymous reviewers of NSPW for their insights and comments, and our shepherds Cormac Herley and Olgierd Pieczul for their guidance on revising our paper. We benefitted greatly from the many discussions at NSPW 2018, particularly those with Eireann Leverett, Nathan Reitinger and Olgierd Pieczul. Specific thanks go to Eireann Leverett for having pointed us to libraries.io. Their datasets prompted us to replace our original and more limited analysis of Github data with the current analysis. We would also like to thank the scribes, Bob Blakely and Matt Bishop, for their meticulous notes, which were a great asset when finishing the post-proceedings version of our paper.

REFERENCES

[1] Jürgen Altmann. 2013. Arms Control for Armed Uninhabited Vehicles: an Ethical Issue. *Ethics Inf. Technol.* 15, 2 (March 2013), 137–152.

[2] Jeff Bachman. 2017. Revisiting the "Humanitarian" Intervention in Libya. Huffington Post. (March 13, 2017). Online at https://www.huffingtonpost.com/jeff-bachman/revisiting-the-humanitari_b_9445270.html.

[3] Maaike Beenes and Susi Snyder. 2018. *Don't Bank on the Bomb: A Global Report on the Financing of Nuclear Weapons Producers.* Technical Report. PAX, Utrecht, Netherlands. ISBN 978-94-92487-25-4 NUR 689.

[4] Terry Bollinger, Frank Petroski, Fred Schultz, and Flayo Kirk. 2003. *Use of Free and Open-Source Software (FOSS) in the U.S. Department of Defense.* Report MP 02 W0000101. The MITRE Corporation. Version 1.2.04, prepared for the Defense Information Systems Agency (DISA) under contract No. DAAB07-01-C-N200.

[5] Anqelique Chrisafis. 2018. Nicolas Sarkozy Denies 'Crazy, Monstrous' Libya Funding Allegations. The Guardian. (March 22, 2018). Online at https://www.theguardian.com/world/2018/mar/22/nicolas-sarkozy-denies-crazy-monstrous-libya-funding-allegations.

[6] Gordon Corera. 2018. Could Russia and West be Heading for Cyber-War? BBC News. (April 16, 2018). Online at http://www.bbc.com/news/technology-43788114.

[7] Douglas Crockford. 2018. The JSON Saga. Online at https://www.youtube.com/watch?v=x92vbAN_j1k. (April 2018).

[8] GAO. 2018. *Weapon Systems Cybersecurity – DOD Just Beginning to Grapple with Scale of Vulnerabilities.* Report to the Committee on Armed Services, U.S. Senate. United States Government Accountability Office.

[9] International Committee of the Red Cross 1977. *Protocol Additional to the Geneva Conventions of August 12, 1949, and Relating to the Protection of Victims of International Armed Conicts (Protocol I).* International Committee of the Red Cross, Geneva.

[10] Sebastian Jannasch. 2016. Bayerische Firma will 600 Millionen Dollar von der US-Marine. Süddeutsche Zeitung. (July 22, 2016). Online at http://www.sueddeutsche.de/digital/klage-gegen-raubkopien-bayerische-firma-will-millionen-dollar-von-der-us-marine-1.3090442.

[11] Michael A. Lindenberger. 2013. Irving Software Firm Settles Suit with U.S. Army for $50 million. The Dallas Morning News. (Nov. 2013). Online at https://www.dallasnews.com/business/business/2013/11/24/irving-software-firm-settles-suit-with-u.s.-army-for-50-million.

[12] United States. Dept. of the Army. 1982. *FM 100-5, Operations.* Headquarters, Department of the Army, Washington, D.C., USA.

[13] The European Parliament and the European Council. 2009. Directive 2009/24/EC on the Legal Protection of Computer Programs. Official Journal of the European Union. (April 23, 2009), 111/16-111/22 pages.

[14] William J. Perry. 1994. *Specifications and Standards - A New Way of Doing Business.* Memorandum for Secretaries of the Military Departments. United States of America Department of Defense, Washington, D.C., USA.

[15] Olgierd Pieczul. 2018. Personal Communication at NSPW. (Aug. 2018).

[16] Kenneth Roth. 2006. Was the Iraq War a Humanitarian Intervention? *Journal of Military Ethics* 5, 2 (2006), 84–92. https://doi.org/10.1080/15027570600711864 arXiv:https://doi.org/10.1080/15027570600711864

[17] Jerome H. Saltzer and Michael D. Schroeder. 1975. The Protection of Information in Computer Systems. *Proc. IEEE* 63, 9 (1975), 1278–1308.

[18] David E. Sanger. 2017. Malware Case Is Major Blow For the N.S.A. The New York Times. (May 17, 2017). New York edition, A1, online at https://www.nytimes.com/2017/05/16/us/nsa-malware-case-shadow-brokers.html.

[19] John Schwarz. 2015. A Short History of U.S. Bombing of Civilian Facilities. The Intercept. (October 7, 2015). Online at https://theintercept.com/2015/10/07/a-short-history-of-u-s-bombing-of-civilian-facilities/.

[20] Scott Shane and Daisuke Wakabayashi. 2018. A Google Military Project Fuels Internal Dissent. The New York Times. (April 5, 2018). New York edition, A1, online at https://www.nytimes.com/2018/04/04/technology/google-letter-ceo-pentagon-project.html.

[21] Tom Shanker. 2010. General Nominated to Lead Cyberspace War Unit Sees Gaps in Laws. The New York Times. (April 15, 2010). New York edition, A10, online at https://www.nytimes.com/2010/04/15/world/15military.html.

[22] Richard Stallman. 1986. What is the Free Software Foundation? *GNU Bulletin* 1, 1 (Feb. 1986), 8–9. Online at https://www.gnu.org/bulletins/bull1.txt, accessed June 16, 2017.

[23] Victor van Reijswoud and Arjan de Jager. 2008. Free and Open Source Software for Development. *CoRR* cs.GL, Article 0808.3717 (2008), 113 pages. http://arxiv.org/abs/0808.3717

[24] Wassenaar Arrangement on Export Controls for Conventional Arms and Dual-Use Goods and Technologies 2017. *List of Dual-Use Goods and Technologies and Munitions List.* Wassenaar Arrangement on Export Controls for Conventional Arms and Dual-Use Goods and Technologies, Vienna, Austria. Online at https://www.wassenaar.org.

[25] Human Rights Watch. 2012. *Losing Humanity: The Case against Killer Robots.* Human Rights Watch, 350 Fifth Avenue, 34th Floor, New York, NY 10118-3299, USA. ISBN 1-56432-964-X.

[26] David M. Wennergren. 2009. *Clarifying Guidance Regarding Open Source Software (OSS).* Memorandum. Department of the Defense. Attachment 2, online at https://dodcio.defense.gov/Portals/0/Documents/FOSS/2009OSS.pdf.