Open Source, Freie Software – Conditio sine qua non für digitale Selbstverteidigung

Übersicht:

- zwei Vorbemerkungen
- eine Annäherung über die Begrifflichkeiten Open Source, Freie Software
- eine historische Herleitung der These, die in der Überschrift steckt
- zum Schluss: meine Perspektive für das Handeln

zum Handout:

- zwei Listen aus dem Vortrag, spart das Mitschreiben.
- kommentierte Literaturliste
- Linkliste
- der Link, wo ich diesen Vortrag zum Nachlesen veröffentlichen werde

Zur ersten Vorbemerkung:

1 Digitalisierung ist analog

Digitale Selbstverteidigung ist keine technische Angelegenheit. Und - es geht auch um weit mehr als nur den Schutz **digitaler** Güter. Privatsphäre ist mittlerweile tief in ihre analogen Aspekte hinein durch digitale Überwachung und Manipulation (“Profilauswertung zur Prozessoptimierung”) berührt.
*Beispiel: Wenn **digitale Routenplaner** den Verkehrsfluss zwecks Stauvermeidung vorübergehend durch eine kleine Nebenstraße legen, dann verändert sich dort ganz konkret und analog die Verkehrssituation und -sicherheit. Eine sonst für Kleinkinder bewältigbare Straße kann dann relativ plötzlich und ohne Vorwarnung zur gefährlichen Hauptverkehrsader werden.*

Es macht also einen Unterschied, ob eine Maschine nach sehr abstrakten Kriterien (Fahrtzeit, Entfernung) entscheidet oder ein erfahrener Verkehrsplaner oder auch nur ein Verkehrspolizist mit lokalem Hintergrundwissen. Und, ebenso wichtig: Die individuelle Routenplanung wird über ihre massenhafte digitale Akkumulation zum öffentlichen Phänomen, um es neutral zu formulieren.

Bei digitaler Selbstverteidigung geht es also um den Schutz von Gütern – analoger und digitaler Güter – vor den negativen Auswirkungen elektronischer Datenverarbeitung. Die beiden Bereiche Analog und Digital sind über Ursache-Wirkung-Zusammenhänge und Rückkopplungseffekte miteinander verbunden. So zu tun als seien sie voneinander zu trennen, verschleiert wesentliche Aspekte der Digitalisierung. Digitalisierung kann nur als Wirkungsverhältnis zwischen analog und digital begriffen werden. Diese Wirkungen haben sich in den letzten Jahrzehnten durch drei Tendenzen potenziert:

1. Digitalisierung greift um sich. Immer weitere, tendenziell alle Lebensverhältnisse sind einbezogen: Die digitale Verdopplung der analogen Welt ist zu beobachten.
2. Das alles in Echtzeit. Die netzbasierte Übermittlung von Daten findet global und quasi in Echtzeit statt.
   Das ist buchstäblich zu spüren bei minimalen Verzögerungen von Übertragungen, z.B. von Fußballspielen.
3. Automatisierung, Autonomisierung, Maschinenlernen: Die algorithmusbasierte Verarbeitung und Auswertung der ansonsten gar nicht bewältigbaren Menge anfallender Daten. Sie führt zu Anhäufungen von Gegenständen, zur Verdichtung von Verhältnissen, die qualitativ etwas anderes darstellen als die Summe der Einzelereignisse ohne ihre digitale Gleichschaltung, *vgl. das Routenplanerbeispiel: wenn alle sich individuell ihre Umleitung suchen würden, würde sich der Umleitungsverkehr mehr oder weniger gleichmäßig auf die umliegenden Straßen verteilen, mit ganz anderen Auswirkungen für die Straßenanlieger_innen.*

Schlagwörter wie Industrie 4.0, Big Data, Smart Irgendwas oder aktuell Künstliche Intelligenz betonen einzelne Aspekte dieses Zusammenhangs. Meist handelt es sich dabei um interessengeprägte Begrifflichkeiten. Die Debatten bilden nichts anderes ab als ganz traditionelle gesellschaftliche Konflikte - auf mindestens vier Ebenen:

- Um Ressourcenverteilung, etwa: Wer bekommt über das KI-Ticket mehr Forschungsgelder in der nächsten Finanzierungsrunde?
- Um Staatsfinanzierung: Wer wird besteuert (weil er sich nicht digital verflüchtigen kann) und wer bekommt Steuererlass (für die Ansiedlung von IT-Industrie)?
- Um die Regulierung von Märkten: Unter welchen Regulierungsbedingungen legalisieren wir den Datenhandel?
- Um Arbeitsverhältnisse: Wessen Jobs übernehmen die Automaten und wer muss mehr Arbeiten fürs gleiche schlechte Geld?

Noch einmal zusammengefasst: Das Digitale ist analog. Wer analog und digital trennt, kann nicht verstehen, was passiert. Wer Technik und Politik – die *gemeinsame* Regelung *öffentlicher* Angelegenheiten – trennt, kann nichts mehr tun. Ganauer: Wer Technikpolitik an Techniker oder Technologiekonzerne delegiert, verliert Souveränität und schließlich Handlungsfähigkeit auf den gesellschatlichen Feldern, die mit Technik in einem gegenseitigen Wirkungsverhältnis stehen.

2 Zweite Vorbemerkung: Das verlorene Individuum

Die Debattenebenen aus dem vorigen Punkt (Verteilung, Finanzierung des Öffentlichen, Wirtschaftsregulierung, Arbeits-verhältnisse) zeigen: Digitalisierung ist sowohl materieller Prozess als auch Diskurs mit Auswirkungen auf das normative Gefüge.

In der Regel bedeutet Digitalisierung Auflösung etablierter Verhältnisse und Institutionen. Die damit einhergehenden Neuzusammensetzungen sind instabil. Es herrscht eine Art Wild-West-Situation. Verloren sind darin die Individuen, überwältigt schon im einfachen Reproduktionsalltag, der allmählich quasi wie von Geisterhand immer mehr Digitalisierung erzwingt. *Internet für die schnelleren Informationen, Online-Banking fürs Geld, Smartphone für die Sozialkontakte. Die Werkzeuge auf Arbeit. Die Verwaltungsschnittstellen z.B. wenn es um Gesundheit geht, usw. undsofort.* Jetzt sollen die Vereinzelten auch noch *das* meistern, sich darin verantwortlich und "gut" verhalten, "das Richtige" tun. Denn der digitale Wilde Westen erzwing die digitale Selbstverteidigung nicht nur, sondern macht sie gleichzeitig zur Privatangelegenheit: Alle gegen alle. Jeder für sich.

In der Regel spüren wir an diesem Punkt die Notwendigkeit, uns zu informieren und so in die Debatte zu bewegen. Die dreht sich dann um den "sichersten" Messenger oder das beste Memory-Stick-Verschlüsselungsprogramm. Debatten um Werkzeuge, bestenfalls *technisch* aufgeklärt. Der Zeitschriftenmarkt hält eine Unzahl von Ratgeber-Magazinen vor, um unsere Verzweiflung in Wert zu setzen. Die Frage nach den Werkzeugen und Programmen zur Realisierung von digitaler Selbstverteidigung zäumt das Pferd jedoch von hinten auf. Außerdem führt sie in den meisten Fällen zu Überforderungsgefühlen und damit *nicht* zur Abhilfe, sondern im Gegenteil, zur Verstärkung der Verzweiflung. Die mehr oder weniger bewußte Schlussfolgerung der Vereinzelten lautet zugespitzt: Führe mich sanft, denn ich kann ja eh nichts tun.

Soweit meine Vorbemerkungen.

3 Zur begrifflichen Annäherung: Freie Software vs. Proprietäre

Mein Interesse ist es, dieser fatalistischen, im Kern autoritär-unterwerfungsbereiten Haltung eine Alternative entgegenzusetzen. Denn diese Alternative existiert: Freie Software.

- Software, deren Programmcode menschenlesbar vorlieg.
  Hier will ich etwas ausholen mit einer Erläuterung: Software verleiht Hardware einen Zustand und Prozessfähigkeit. Die Maschinenlesbarkeit ermöglicht das Zusammenspiel von Hard- und Software. Klassische siliziumbasierte Maschinen lesen Maschinensprache. *Solche Hardware kodiert* binär*: Strom ein, Strom aus, ein Code mit zwei Buchstaben, Eins und Null.*[1]
  Übersetzungsprogramme (fachsprachlich: Compiler) können diese maschinenlesbaren Texte in menschenlesbare *Texte übersetzen: Stelle Ausgangsituation A dar, nehme Input B, tue damit C, dann gibt es noch Bedingungsstrukturen wie z.B. wenn-dann oder solange-bis und schließlich: liefere Output X an Schnittstelle Y und mache weiter bei Z.*
  Menschenlesbarer Programmtext lässt sich studieren, verstehen, verändern und per Compiler wieder in maschinenlesbaren Text übersetzen. Z.B. um die Maschine probelaufen zu lassen. Immer hin und her.
  In diesem Hin- und Her auf der Basis von menschenlesbarem Code haben wir eine Maschine:
- Deren Funktionsweise wir erforschen können und dürfen.
- Die wir nach unseren Bedürfnissen verändern können.
- Die jeder zu jedem beliebigen Zweck benutzen kann.
  (Es bleiben „nur“ die ethischen Schranken. Sie sind *politisch* auszuhandeln.)
- Und die wir weitergeben dürfen an andere, damit nicht jeder mit seiner Problemlösungsversuchen in der Ursuppe anfangen muss.

1 Eiweißbasierte biologische Maschinen, wie sie im Falle irdischen Lebens üblich sind, lesen einen Code mit vier Buchstaben, den vier Nukleinbasen Adenin, Guanin, Cytosin und Thymin, A, G, C und T, die sich zur DNA zusammensetzuen. Sie bilden das sogenannte biologische Erbgut. Aus informationeller Sicht: Die komplette Zustands- und Betriebsinformation des Lebens.

Noch einmal, die vier Freiheiten Freier Software: Freier, sprich menschenlesbarer Code, freie Verwendung und Erforschung, freie Abwandlung und freie Weitergabe.

Eine kurze Begriffsklärung in diesem Zusammenhang: Oft ist auch einfach nur von «Open-Source» die Rede. Aber es handelt sich bei Open Source und Freier Software nicht um Synonyme oder den englischen Begriff und die deutschsprachige Übersetzung: Freie Software (Free Software) ist das strengere, weniger beliebige Konzept, das alle vier Freiheiten *erfordert*, nicht nur den offenen Quellcode (Open Source: Offene Quelle). Der Sammelbegriff, der etwas künstlich daherkommt, lautet FOSS, Free and Open Source Software. In der Regel lohnt Nachfragen in den Debatten um Open Source im weitesten Sinne, wenn nicht klar ist, was gesagt wird und was damit tatsächlich gemeint ist.

Freie Software gibt es für jeden Anwendungsbereich. In vielen Anwendungsbereichen ist sie konkurrenzlos, da lohnt es sich für kein gewinnorientiertes Unternehmen, mit einer sogenannten proprietären Software überhaupt anzufangen. *Kurioses Beispiel: Betriebssysteme für Superrechner. In der Liste der 100 schnellsten Rechner der Welt laufen alle, und zwar* alle*, mit Varianten von Linux, dem Freie-Software-Betriebssystem.*

Proprietäre Software hingegen

- verheimlicht ihren Programmcode.
- besteht auf dem Geistigen Eigentum und
- verbietet die Weitergabe des Programms ohne finanzielle Kompensation.
- Macht abhängig vom Hersteller, was Sicherheitsupdates und Feature-Erweiterungen angeht.
- Und – wichtig im Zusammenhang mit digitaler Selbstverteidigung – kann unerkannt Hintertüren enthalten und weitere Schadsoftware (Stichwort: Trojaner) ins System einspielen.

Hintertüren ermöglichen dem Hersteller und Dritten Überwachung und Manipulation. Die Maschinen (Rechner, Telefone, Autos, aber auch immer mehr Gebrauchselektrik wie Kühlschränke, Heizungssteuerungen usw.) sind zunehmend nicht nur an den Strom, sondern auch ans Internet angeschlossen. Darauf basieren Konzepte der Komplettüberwachung des Alltagslebens. Seitens der IT- und Social-Media-Konzerne ist die Rede von der "Auswertung des Nutzungsverhaltens zur Produkt- und Prozessoptimierung". Bei proprietären Maschinen, Plattformen und Systemen passiert das intransparent und ohne Eingriffsmöglichkeit durch die Betroffenen. Daher halte ich die Konzernrede für verharmlosend und die harten Vokabeln von Überwachung und Manipulation für angebracht.[2]

Werbung, das heißt also Datenhandel, das Geschäft mit den Nutzungsprofilen von Software, Webdiensten, vernetzten Geräten und mit den Bewegungsmustern on- und offline,[3] stellt ein relevantes Geschäftsfeld dar. Das ist so und wird absehbar eher mehr als weniger werden. Solange gibt es keinen Grund, einem kommerziellen Anbieter von Geräten und Webdiensten mit proprietärer Software zu vertrauen.

---

2 Die Betroffenen können entweder einwilligen in die Nutzungsbedingungen oder auf den Gebrauch verzichten. Der Preis für Nichtgebrauch ist mittlerweile oft sehr hoch: Verlust von Arbeitsplatz, Verlust von Sozialkontakten, hohe finanzielle Kosten für das Recht der Weiternutzung der traditionellen analogen Maschinen bzw. Dienste und Verfahren. Beispiel Heizkostenablesung… https://netzfueralle.blog.rosalux.de/2017/01/26/wenn-der-heizungsableser-zweimal-klingelt/

3 Oder zum Beispiel Werbung. Werbung ist unter den Bedingungen von immer weniger reguliertem Kapitalismus eben nicht Produktinformation. Informationen, die tatsächlich den Verkauf des Werbetreibenden fördern, lassen sich als "gute" Werbung bezeichnen. Gute Werbung erreicht ihr Ziel im Zeitalter der Digitalisierung durch personalisierte Ansprache der Werbesubjekte. Gute Werbung ist nichts anderes als die individuelle Manipulation zum Zwecke höherer Umsätze und Gewinne. Für personalisierte Werbung braucht es Nutzungs-, Konsum- und Bewegungsprofile. Die werden durch die Hintertüren erhoben – trotz AGBs und Anonymisierungsversprechen, denn: Die Profile selbst bzw. Teilprofile müssen nicht persönlich sein, sondern nur möglichst genau dem Persönlichkeitstyp entsprechen. Es klingt widersprüchlich: Auch mit anonymisierten Daten lässt sich personalisierte Werbung optimieren. Das zeigt: Die Möglichkeiten der algorithmischen Mustererkennung und die schieren Massen anfallender Daten hebeln jedes klassische auf das konkrete Individuum zielende Datenschutzrecht aus. Die Profile werden über die Speicherung und Akkumulation aller digitalen Spuren im Netz hergestellt, ständig anonymisiert und wieder re-personalisiert und hin und her verkauft. Vgl. https://netzpolitik.org/2015/personalisierte-werbung-wie-unsere-person-selbst-an-verschiedenen-geraeten-identifizierbar-wird/

Einem Anbieter, der nur den Gebrauchswert seines Produkts bejubelt, aber nicht auch den Code offenlegt, dem darf ich unterstellen, dass er über Hintertüren und Schnittstellen zur Datensammelei verfügt. Für Windows 10 hat es das Bayerische Landesamt für Datenschutzaufsicht herausgearbeitet: Die bundesweiten und europäischen Datenschutz- und -sicherheitsziele öffentlicher Verwaltungen und privater Firmen seien mit Windows 10 gar nicht mehr erreichbar (so das Bayerische Landesamt für Datenschutzaufsicht nach eigenen Tests und Messungen im September 2017).[4]

Dennoch kommt flächendeckend Microsoft Windows und Office zum Einsatz, dennoch wird massenhaft geliked und getwittert, dennoch gelten die Geräte mit dem Apfel als Statussymbol für den jungen, dynamischen, kurz: besseren Menschen. Und das, obwohl in Wirklichkeit FOSS besser ist und zwar aus strukturellen Gründen. - Das hört sich an wie eine harte Behauptung. Ich will sie plausibel machen, indem ich die Geschichte dieser real existierenden Alternative erzähle.

4 Ein kleiner historischer Überblick über die Produktion von Software – am Beispiel von GNU/Linux

Warum ist Geschichte wichtig für erfolgreiche digitale Selbstverteidigung?

Weil sie zeigt: Technik fällt nicht vom Himmel, sondern sie wird produziert und diese Produktion, ihre Bedingungen und Auswirkungen, sogar die Zwecke, für die produziert wird, sind gesellschaftlich umkämpft.

Weil sie zeigt: Technik ist nicht neutral. Technische Entwicklung ist in gesellschaftliche Auseinandersetzungen eingebettet. Daher schreiben sich z.B. gesellschaftliche Methoden oder Zielsetzungen in die Technik ein.

4 https://www.lda.bayern.de/media/windows_10_report.pdf#page=19

Ein Beispiel für solche Methoden: Konkurrenz oder Kooperation oder eben ein gesellschaftlich ermittelter Kompromiss zwischen beidem. Ein Beispiel für gesellschaftliche Zielsetzungen: Profitmaximierung zugunsten privater Eigentümer versus Gebrauchswertoptimierung zur allgemeinen Lebensverbesserung.

Und weil ein solcher geschichtlicher Rückblick, geeignet ist, *Vertrauen* zu begründen. Ohne Vertrauen gibt es keine digitale Selbstverteidigung. Denn es genügt nicht, alleine dem Gegenüber in der Kommunikation zu vertrauen. Es braucht gemeinsame Kriterien, die die Werkzeugwahl ermöglichen. Der historische Blick hilft, diese Kriterien zu erarbeiten.

4.1 Es begann mit Hippies und Großrechnern

In einer Zeit, in der es ganz wenige Computer gab: Sie waren groß und schwer, standen in Universitäten, großen Technikkonzernen und Militärforschungsabteilungen und konnten so gut wie nichts. Die – überwiegend tatsächlich männlichen, so die berechtigte Kritik[5] – Helden dieser Zeit waren die Programmierer, die Programme für diese Computer schrieben, damit sie überhaupt etwas machten: aus einer Eingabe nach der programmgesteuerten Verarbeitung eine Ausgabe. Die Programmierer programmierten für ihre Computer Betriebsprogramme und ließen sie damit für ihre jeweiligen Zwecke rechnen: wissenschaftliche Simulationen, Geschäftskalkulationen und Raketenflugbahnen. Sie nannten diese Programme «Betriebssysteme».

5 Vgl. z. B. Meyer, Silke: «Free Software, Free Society»? Über die Reproduktion von Differenz in der Praxis von Free/Libre Open Source Software-Communities, Diss. FU Berlin 2013, http://vbly.us/meyer

Für unsere Geschichte ist das Betriebssystem Unix wichtig.[6] Es wurde 1969 an der US-amerikanischen Ostküste in den Bell Laboratories, der Forschungsabteilung von Western Electric und AT&T, zwei Elektronik- und Telekommunikationskonzernen, geschrieben und sollte die Entwicklung aufgabenspezifischer Anwendungsprogramme erleichtern. Unix war ein Programm zur Erleichterung der Entwicklung von Programmen. Als solches war es in der Programmierergemeinschaft von Anfang an sehr beliebt. Zufälligerweise waren es gerade die wilden 1968er Jahre, in denen Unix entstand, sodass auch die Programmierer vom grassierenden Hippietum nicht verschont blieben. Auch weil sie mit ihrer Arbeit nicht immer wieder von vorne anfangen wollten und weil sie sich alle kannten – egal ob sie für die Wissenschaft, die Konzerne oder das Militär arbeiteten –, tauschten sie ihre Betriebssysteme untereinander aus und verwendeten die gelungensten Teile ihrer Kollegen, um selbst dazuzulernen und ihren eigenen Code zu verbessern. Das Konzept des «geistigen Eigentums» an Programmquelltexten, dem «Code», wäre ihnen nicht in den Sinn gekommen, sie hätten es vermutlich zunächst gar nicht verstanden, dann aber höchst irritiert zurückgewiesen: Wie soll man da vernünftig arbeiten, wenn man seinen Programmcode nicht mal schnell weitergeben oder fremden einbauen kann, weil die Code-Schnipsel den Firmen gehören? Absurd.

Spätestens in den frühen 1980er Jahren hatte sich der Wind gedreht:

4.2 Die Entwicklung des Personal Computers (PC) - als technologischer Ausdruck der neoliberalen Reaktion

Die Neoliberalen waren auf dem Weg zur Weltherrschaft, von den Hippies war nur noch ein Style auf dem Markt der Identitäten übriggeblieben und Software wurde von oben, von den Geschäftsführungen, angeeignet und eingemauert.

6 Vgl. zur Geschichte des Internets kurzgefasst, kenntnisreich und wohlstrukturiert: Lang, Susanne: Eine kurze Geschichte des Internets. Die Inkorporation des Internets in kapitalistische Verhältnisse ist keinesfalls abgeschlossen und noch immer umkämpft, in: Prokla 186, 2017, http://vbly.us/lang, S. 7 ff.

Zwischen den von den Konzernführungen errichteten Copyright-Burgen fanden die Unix-Wars statt: Jede Firma versuchte, ihre eigene Unix-Variante zum Produkt umzugestalten und mittels eigener, geheimer Schnittstellen und Formate als Standard am Markt durchzusetzen. Da begann Richard Stallman – ein universitärer Programmierer, verärgert über diese Proprietarisierung[7] von Unix – mit der Arbeit an einem eigenen, unixoiden Betriebssystem und verankerte die Freiheit, den Code auszutauschen und wiederzuverwenden, in einem Manifest. Sein Unix nannte er GNU, Gnu is Not Unix. Aus dem Manifest entstand später mithilfe eines Anwalts die GNU GPL, die GNU General Public Licence, eine Copyright-Lizenz zum Schutz dieser Freiheit vor der Aneignung durch die Verfechter exklusiver geistiger Eigentumsrechte auf juristischem Weg. Die GPL hat diese Freiheit bis heute in vielen Gerichtsverfahren geschützt. Sie bildet eine schräge Anomalie im bürgerlichen Eigentumsregime: Sie schützt das Nicht-Eigentum mit den juristischen Mitteln des Zivilrechts, das eigentlich historisch entstanden und zweckbestimmt ist zur Durchsetzung (nicht Verhinderung) des individuellen Privateigentums. Stallman ist bis heute der Evangelist der Bewegung für Freie Software, das GNU-Manifest und die GPL sind ihre kanonischen Schriften.

Während die festangestellten Programmierer in Universitäten, Konzernen und beim Militär sich also darum streiten mussten, wer jetzt welches Unix auf welchem Großrechner benutzen darf, trieben ehemalige Garagenbastler von der US-Westküste ihre Kleinrechner (PC: Personal Computer) zur Marktreife und begannen, einen Massenmarkt damit zu versorgen. Für Apple und Microsoft war die Lizenzfrage keine Frage.

7 Proprietäre Software (lateinisch propriē: eigentümlich, eigen, ausschließlich) bezeichnet eine Software, die das Recht und die Möglichkeiten der Wieder- und Weiterverwendung sowie Änderung und Anpassung durch Nutzer_innen und Dritte stark einschränkt. Es gibt einige Mechanismen, die eine Software proprietär machen und halten können: Softwarepatente, das Urheberrecht, Lizenzbedingungen (EULAs), das Aufbauen der Software auf herstellerspezifischen, nicht veröffentlichten Standards und die Behandlung des Quelltextes als Betriebsgeheimnis (englisch closed source).

Die gerade erst entstandenen Firmen steckten Zeit in die Entwicklung von Hard- und Software und wollten mit dem Verkauf ihrer Geräte in erster Linie Geld verdienen (und nicht wissenschaftliche Fragen lösen, ihre eigentlichen Produkte entwickeln oder Raketen steuern). Apple setzte auf den Verkauf seiner Hardware und mauerte eigentumstechnisch schon auf dieser Ebene: Nur Apple stellte Apple-Computer her, die Software war voraufgespielt und im Bündel mit der Hardware zu erwerben. Microsoft verschaffte sich den entscheidenden Konkurrenzvorteil auf diesem Markt durch eine geniale Idee: Sie legten die Hardware-Spezifikationen für den Nachbau kompatibler Rechner offen, sodass jeder Hardwarehersteller einen Microsoft-PC bauen konnte. Er musste nur eine Herstellerlizenz für die Software – das Betriebssystem und die zentralen Programme wie etwa Office – bei Microsoft bezahlen und konnte die Kosten an seine Endkund_innen weitergeben. Das führte dazu, dass die PCs immer billiger produziert wurden und sich Anfang der 1990er Jahre sogar ein finnischer Informatikstudent namens Linus Torvalds die Anzahlung für einen PC leisten konnte.

Seine Geschichte sollte zeigen:

4.3 Verlorene Freiheiten lassen sich auch wieder aneignen

Torvalds genügten weder das mitgelieferte Microsoft-DOS-Betriebssystem noch das in seiner Vorlesung verwendete Übungs-Unix namens Minix, das er mit einer freien Lizenz als Fachbuchbeilage bekommen hatte. Vorlesungsbegleitend ergänzte er Minix um die Elemente, die er für seine Zwecke brauchte: Er programmierte sich eine Verbindung zum Universitäts-Netzwerk, fügte einen Möglichkeit hinzu, um E-Mails zu lesen und um Dateien hoch- und runterzuladen. Da er über die Minix-Mailingliste Fragen und Antworten zu seiner Arbeit austauschte, wurden andere darauf aufmerksam, dass Torvalds im Prinzip einen eigenen Betriebssystemkern produzierte.

Genau das, was dem GNU-Projekt zu diesem Zeitpunkt als zentrales Element noch fehlte.

Ein Kernel (englisch für Kern) ist der zentrale Bestandteil eines Betriebssystems. Er schafft die Grundlagen für die Prozess- und Datenorganisation, mit der alle anderen Betriebssystemelemente und die Anwendungsprogramme arbeiten können. Der Kernel bildet die unterste Softwareschicht. Das ist unmittelbar wichtig für die digitale Selbstverteidigung: Wer den Kernel programmiert, der kann auch kontrollieren, wie er arbeitet, was er im einzelnen tut und nicht tut und wer ihn kontrollieren kann. Kernelfunktionen haben Zugriff auf das, was die Programme tun. Da kann mein Programm noch so sicher sein, wenn der Kernel eine Hintertür hat, dann geht der Zugriff auch durch auf das, was mein vermeintlich sicheres Programm tut. Denn der Kernel hat direkten Zugriff auf die Hardware: Eingabe- und Ausgabegeräte, Recheneinheit (Prozessor, CPU) und wichtig in unserem Zusammenhang: vollen Zugriff auf den Arbeitsspeicher (RAM). Ohne den Kernel gäbe es den Arbeitsspeicher gar nicht. Und jedes Passwort, muss, damit es verarbeitet werden kann, unverschlüsselt durch den Arbeitsspeicher. Jeder entschlüsselte Inhalt muss, damit er verarbeitet werden kann, durch den Arbeitsspeicher. Daher stellt sich Frage: Warum sollte ich der Funktionsweise meines Kernels vertrauen?

Zurück zu unserem finnischen Studenten: Einige von denen, die über die Mailingliste von Torvalds' Projekt erfahren hatten, übernahmen die letzten Raten für seinen neuen PC und verschafften ihm Online-Speicherplatz für die öffentliche Weiterentwicklung seines Kernels, sodass Menschen weltweit seine Arbeit verfolgen konnten und bald eine eigene Mailingliste zur Kernel-Entwicklung anlegten. 1992, ziemlich genau ein Jahr, nachdem Torvalds seinen PC angezahlt hatte, stellte er den Linux-Kernel unter die GPL-Lizenz. Damit war GNU/Linux als freies unixoides Betriebssystem komplett.

Mit der Vermassung von PCs im laufenden Jahrzehnt fand GNU/Linux zunehmend Verbreitung – nicht nur, aber eben auch unter politisch eher progressiv eingestellten Computernutzer_innen, die der neuen Technologie aus den Händen der Konzerne bisher sehr ablehnend gegenübergestanden hatten.

Das führt uns zu einer wichtigen Episode der Geschichte starker Verschlüsselung, nämlich:

4.4 Wie PGP starke Verschlüsselung für alle zugänglich machte

Letztere hatten die teuren Rechner, die sich nur Konzerne und staatliche Dienste leisten konnten, bisher eher als Abstraktions- und alles kontrollieren wollende Monstermaschinen gesehen. Mit dem Preisverfall konnte jedoch ein Aneignungsprozess aus den sogenannten Grassroots-Bewegungen einsetzen. Von heute aus gesehen und im Zusammenhang mit Digitaler Selbstverteidigung ist hier unbedingt die Geschichte von Phil Zimmermann und Pritty Good Privacy (PGP, “Ziemlich guter Vertraulichkeit”) anzureißen. Phil Zimmermann konnte als Sohn eines Betonmischerfahrers aus New Jersey (über seine Mutter ist nichts zu finden, aller gesellschaftlichen Wahrscheinlichkeit nach hat sie sich um den Haushalt gekümmert) an einer staatlichen Universität Informatik studieren. In den 80er Jahren verdiente er sein Geld als Programmierer und war den Rest seiner Zeit in der US-Friedensbewegung aktiv gegen Atomwaffen. Dort spürte er die Notwendigkeit sich und seine Genoss_innen vor den Spionageattacken der Geheimdienste zu schützen. 1991 veröffentlichte er den Programmcode von PGP auf ähnliche Weise und mit ähnlichem Effekt wie fast zeitgleich Torvalds seinen Kernel und einige Jahre zuvor Stallman seine GNU-Tools: Andere konnten die Verschlüsselung ohne Einschränkung nutzen, ihre Funktionsweise prüfen und zu ihrer Verbesserung beitragen, in dem sie Fehler fanden und vielleicht sogar gleich den “Patch” mitlieferten.

Eine Begriffsklärung zwischendurch: Als „stark“ gilt allgemein gesprochen Verschlüsselung, die *nicht* entziffert werden kann. Selbstverständlich kann aus starker Verschlüsselung mit höheren Rechenkapazitäten oder wissenschaftlichem Fortschritt schwache Verschlüsselung werden. Dennoch gilt PGP aus mathematischen Gründen bis heute als stark. Den nach wie vor steigenden Rechenkapazitäten lässt sich mit längeren Schlüsseln und Passwörtern begegnen.

Damit war PGP war die erste Implementation starker Verschlüsselung auf dem neuesten Stand der Wissenschaft, die der Allgemeinheit leicht zugänglich war und tatsächlich schnell Verbreitung fand. 1993 (in dem Jahr, in dem das WWW sichtbar wurde durch die Veröffentlichung des ersten graphischen Browsers: Mosaic, dem Vorläufer von Netscape und Mozilla) versuchten die US-Zollbehörden wenigstens den Export von PGP zu verhindern und ermittelten gegen Zimmermann wegen Verstoß gegen das Waffenexportgesetz. Eine beispiellose Bewegung erhob sich gegen diesen Kriminalisierungsversuch. IT-Konzerne und Grassroots-Gruppen sorgten mit ihren je eigenen Mitteln für politischen Druck. Und Zimmermann druckte den Quellcode von PGP komplett in einem knapp 1000seitigen Buch ab: “PGP Source Code and Internals”. Als Buch konnte der Code legal exportiert werden - unter dem Schutz des “First Amendments”, das in den USA die Meinungs- und Publikationsfreiheit schützt. Hacker außerhalb der USA tippten den Code ab und veröffentlichten eine “internationalisierte” Version von PGP. Der Damm war gebrochen: Starke verschlüsselte Kommunikation war nun ohne Spezialkenntnisse und in Echtzeit überall auf der Welt mit einem Internetanschluss zu haben. Ein Privatisierungsversuch Ende der 1990er Jahre führt dazu, dass eine Variante unter Stallmans nicht privatisierbarer GNU-Lizenz veröffentlicht wurde: GnuPG, GPG. Weder das staatliche Exportverbot noch der Privatisierungsversuch konnten die Verallgemeinerung starker Verschlüsselungstechnologie verhindern.

Verschiedene Regierungen versuchen seitdem immer wieder, direkte gesetzliche Verbote starker Verschlüsselung durchzusetzen – und scheitern dabei bisher am Widerstand von Konzernen, die ihre Geschäftsgeheimnisse schützen wollen, und Bürgerrechtsbewegungen.[8] Zuletzt kam eine solche Offensive 2015 vom britischen Premier, der damit beim US-Präsidenten Unterstützung fand – und das war *vor* Trump. Aber die Luft wird dünner in den sogenannten *Crypto Wars*: Wer in Großbritannien sein Passwort nicht "freiwillig" rausgibt, der geht derzeit bis zu sechs Monate ins Gefängnis („Erzwingungshaft").

Aber gut: GPG ist *noch* legal und bildet bis heute den Standard starker Email-Verschlüsselung. Es gibt diverse *freie* Emailprogramme, die GPG/PGP können und alle sind untereinander kompatibel. Einfach einzurichten und zu bedienen ist meines Erachtens das recht weit verbreitete Emailprogramm Thunderbird mit dem GPG-Plugin Enigmail. Es ist für alle Betriebssysteme erhältlich. Auch wenn es natürlich empfehlenswert ist, sein Verschlüsselungssystem innerhalb eines technischen Rahmens zu betreiben, der möglichst wenig von Konzernen kontrolliert wird und möglicherweise mit Hintertüren versehen ist. Auch auf Kernel-Ebene nicht.

Bei Linux als freiem Betriebssystem auf UNIX-Basis waren und sind Privacy-Erwägungen Teil des Entwicklungsprozesses. Die Entwicklung fand in einem Zusammenhang statt, der auf strategischer Ebene nicht geprägt war von Profitorientierung, wie es in Konzernen der Fall ist. Stichwort *security by design*:

8 Die ganze Absurdität des Vorhabens zeigt der Algorithmus Ciphersaber bzw. Ciphersaber2 für eine starke symmetrische Verschlüsselung. Er ist so einfach gehalten, dass jede_r mit Grundkenntnissen im Programmieren ihn im Kopf behalten und jederzeit *from the scratch* implementieren kann. http://ciphersaber.gurus.org/faq.html (beachte aber: https://de.wikipedia.org/wiki/RC4#Spritz)
Dazu passend Phil Zimmermann: "If privacy is outlawed, only outlaws will have privacy." (dt: „Wenn Privatsphäre gesetzlich verboten wird, haben nur Gesetzlose Privatsphäre."), Quelle: https://www.philzimmermann.com/EN/essays/WhyIWrotePGP.html

Auch wenn es größeren Entwicklungsaufwand bedeutet, waren Nutzer- und Rechteverwaltung schon auf Prozessebene von Anfang an selbstverständlich. Das macht sog. isolierte Umgebungen möglich, in denen Programme vom Rest des Systems abgeschirmt laufen können. Passwörter und Schlüssel, die auf dem Rechner liegen und die vielleicht sogar gleichzeitig verarbeitet werden, können sie so nicht sehen. Und *security by default*: Vorhandene Sicherheitseinstellungen sollten von Anfang an automatisch aktiviert sein. Dort wo das zur Einschränkung des Funktionsumfangs führen könnte, erwarte ich ein Infofenster, das mir die Optionen erklärt und mich vor die Wahl stellt, ob ich die Einstellung beibehalten will (“Empfohlen”) oder ob ich sie wirklich deaktivieren will (“Nicht empfohlen”).

Die Kernelemente eines freien Betriebssystems für alle lagen also seit Anfang der 1990er Jahre vor. Die darauf folgende Dekade war geprägt durch die

4.5 Vermassung durch „Anwenderfreundlichkeit“

Zu Beginn der 2000er Jahre war es im Linux-Kosmos so weit: Es gab es ausgereifte vorgefertigte, auf CD oder DVD erhältliche oder im Internet downloadbare Installationspakete (Distributionen) für GNU/Linux, leistungsfähige Plattformen für die dezentrale, nicht-lineare Weiterentwicklung von GNU/Linux selbst[9] und ein Ökosystem kleinerer Dienstleistungsfirmen rund um den PC- und Netzwerkbetrieb auf GNU/Linux-Basis. Damit war dieses Betriebssystem auch für fortgeschrittene Heimanwender_innen und progressive Entscheider_innen in Verwaltung und Wirtschaft zur ernst zu nehmenden Option neben den kommerziellen Angeboten der großen IT-Konzerne – damals allen voran Microsofts Windows – geworden.

9 Git ist eine solche paradigmatische Plattform. Die Theoretisierung der Arbeits- und Kooperationsweise auf Systemen wie Git hat zum Konzept der «commons-basierten Peer-Produktion» geführt und wird mitunter als «Keimform» einer neuen Produktionsweise diskutiert, die noch im Schoß der alten entsteht, aber das Potenzial hat, sich und die mit ihr entstehende neue Gesellschaft aus dieser heraus und über diese hinaus zu entwickeln: http://vbly.us/keime

Das Beispiel der Stadt München ist hier interessant: Die Münchener Stadtverwaltung ist zwischen 2003 und 2013 auf den Beschluss einer rot-grünen Mehrheit erfolgreich von Windows auf Linux umgestiegen. 2017 hat der mittlerweile rot-schwarz dominierte Stadtrat dann die Rückabwicklung beschlossen. Wie es dazu kam, habe ich in meiner Studie untersucht, die online als PDF zu haben ist und die ich auch mitgebracht habe.
So viel kann ich verraten: Es waren nicht die technischen *Gründe, die zur Wiedereinführung von Windows geführt haben.*

Seitdem sind wieder fast 20 Jahre vergangen. Im Kosmos der Freien Software ein Zeitalter der Konsolidierung. Ubuntu zum Beispiel startete als strategischer Ansatz eines südafrikanischen IT-Milliardärs und Idealisten. Er wollte auch den nicht-zahlungsfähigen Teilen der Weltbevölkerung die Möglichkeit verschaffen, die immer leichter zugängliche Computer-Hardware eigenwillig und kreativ zu nutzen. Er sah die umfassende Digitalisierung kommen und wollte die Menschen zu einer *Gemeinschaft* von *Nutzerinnen und Nutzern* machen. Statt es den Konzernen zu überlassen sie zu einer *Masse* von *Kunden* zu machen – und zwar nur falls es sich lohnt, was mit Blick auf nennenswerte Teile der Weltbevölkerung (gerade auch in Afrika) gar nicht selbstverständlich ist. Ubuntu ist heute eine benutzerfreundliche Gnu/Linux-Distribution und bringt umfassende Hardware-Tauglichkeit mit für Desktop-Rechner und mobile Computer. Ubuntu ist anpassbar in der Oberfläche durch die Benutzer_innen und bietet alle vier Freiheiten. Es gibt umfassende und durch die Mitarbeit der Nutzer_innen ständig aktualisierte und erweiterte Gebrauchsanleitungen. Ich kann mich an kein Computerproblem in den letzten 20 Jahren erinnern, dass nicht schon andere vor mir hatten und mit deren veröffentlichter Problemlösung ich dann auch irgendwie klar kam.

Was ich damit sagen will:

Technologische Souveränität ist nicht nur nötig – sie ist auch möglich: Mit Hilfe der Werkzeuge aus dem Kosmos der Freien Software. Als Basis für den ersten Ausstieg aus der Konzern- und Überwachungssoftware halte ich Ubuntu für unbedingt empfehlenswert.
Ähnliches gilt für Smartphones: Google-Apps lassen sich aus Android entfernen. Es gibt freiere (im oben geklärten Sinne) Smartphonebetriebssysteme (CyanogenMod/LineageOS). Es gibt sogar Bemühungen, die Hardware *fair* zu produzieren, Arbeitsbedingungen am anderen Ende der globalen Werkbank und in den Rohstoffminen zu berücksichtigen. Auch da ist Open bzw. “frei” ein Beurteilungskriterium: Wo Fair draufsteht, aber nicht mindestens Open drin ist, ist meines Erachtens insgesamt Skepsis angebracht.

Ich komme zum Schluss.
Eine Perspektive zum Handeln habe ich in Aussicht gestellt:

5 Digiale Selbstverteidigung als Aneignung digitaler Infrastruktur

Die Softwarekonzerne selbst setzen in ihrer Software*produktion intern* auf freie Software. Dennoch geben sie sich alle Mühe mit ihren proprietären Produkten die Geräte und Köpfe ihrer Kunden zu dominieren. Damit sind sie sehr erfolgreich. Ihre Kontroll- und Verfügungsmacht ist in die Eigentumsverhältnisse der Software eingeschrieben, die sie verkaufen. Der Umgang damit gilt allgemein als technisches Nerd-Thema. Das reduziert die Technik zur Blackbox. Diese Reduktion der Technik auf eine Blackbox wird in der gesellschaftlichen Arbeitsteilung ständig reproduziert. In der bürgerlichen Gesellschaft mit ihrer kapitalistischen Produktionsweise ist die Arbeitsteilung ebenso wie die Eigentumsordnung herrschaftsförmig festgelegt. Daher lässt sich die Blackbox auch kaum öffnen: Ihr Inhalt, die Technik, ihre Funktionsweise und damit auch ihre gesellschaftliche Zwecksetzung, sind nur sehr schwer verhandelbar. Das beraubt die Gesellschaft ihrer Souveränität in den betroffenen Lebens- und Arbeitsbereichen.

Es dürfte klar geworden sein: Der Kompass, den ich hier anzubieten habe, schlägt aus in Richtung freier Software. Die grundsätzliche Frage: Wem gehören die Programme und Werkzeuge der digitalen Selbstverteidigung? Politisch formuliert: Wer kontrolliert die Werkzeuge und wer bestimmt, wie sie funktionieren? Diese Frage hilft mir, die strukturell sicherere Variante einer bestimmten softwareförmigen Problemlösung zu finden.

Im Alltag ist den meisten Benutzer_innen elektronischer Geräte gar nicht klar, wenn sie Open-Source-Software verwenden. Etwa wenn sie mit ihrem Android-Smartphone telefonieren oder andere Funktionen nutzen, wenn sie mit der Konfigurationsoberfläche des Routers[10] ihre Internetverbindung verwalten, wenn sie in ihrem Wordpress-basierten Blog schreiben oder mit Firefox oder Chrome im Internet unterwegs sind. Die Eigentumsverhältnisse der digitalen Gegenstände, mit denen wir Tag für Tag hantieren, bleiben uns meist verborgen. Sie interessieren uns nicht weiter. Dabei sollten sie es. Diese IT-Systeme sind streng genommen kritische Infrastrukturen, ohne die wir nicht leben oder für ein besseres Leben kämpfen können.

Die Perspektive besteht in der Frage nach den Eigentumsbedingungen und den Produktionsweisen. Mit dieser Frage verschaffe ich mir Orientierung und werde wieder handlungsfähig. Denn ich weiß: Wer qua Eigentumsrecht am Code über die Funktionsweise der Software bestimmen kann, der kann auch bestimmen, für wen es wirksame digitale Selbstverteidigung gibt und für wen nicht. Das meine ich, wenn ich von der Wiederaneignung der kritischen Infrastrukturen spreche.

10 Vgl. zu Linux in elektronischen Kleingeräten, sog. embedded systems, Henkel, Joachim/Tins, Mark: Die industrielle Nutzung und Entwicklung von Open-Source-Software: Embedded Linux, in: Lutterbeck, Bernd/Bärwolff, Matthias/Gehring, Robert A. (Hrsg.): Open Source Jahrbuch 2005, Berlin 2005, www.opensourcejahrbuch.de/download/jb2005/index.html, S. 123 ff.

Als Antwort auf die Frage „Was tun?“ heißt das: den Umstieg auf Freie Software *gemeinsam* anpacken! Dazu plädieren wir für ein *nicht-technisches* Vorgehen: Ein Set von Arbeitsfragen, anhand derer wir überhaupt erst einmal diskutieren, was wir mit welchen Mitteln vor wem schützen wollen. Diese Arbeitsfragen stellen sich sowohl im Hinblick auf die Sicherung digitaler Güter (z.B. ein Fotoarchiv) als auch im Hinblick auf netzbasierte Kommunikation (also etwa Text- oder Video-Chat). Die fünf Fragen bauen logisch aufeinander auf:

1. Was habe ich bzw. was haben wir, das es zu verteidigen lohnt?
2. Vor wem will ich bzw. wollen wir diese Güter schützen?
3. Wie wahrscheinlich wird dieser Schutz notwendig?
4. Was ist der Schaden, wenn ich bzw. wenn wir mit unseren Schutzmaßnahmen scheitern?
5. Wie viel Aufwand will oder kann ich bzw. wollen oder können wir treiben, um diesen Schaden zu vermeiden?

Anhand solcher Arbeitsfragen gilt es mit unseren Kommunikationspartnern und -partnerinnen bzw. in unserer Bezugsgruppe (Familie, Gemeinde, Freundeskreis, Verein, Firma etc.) möglichst hierarchiefrei und konsensorientert zu diskutieren. Nur so kommen alle zur Sprache, tragen die unterschiedlichen Sichtweisen und Talente aller zur gemeinsamen Sicherheit bei. Nur so lassen sich Bedrohungsszenarien realistisch einschätzen, gemeinsame Sicherheitsbedürfnisse ermitteln und verlässlich abgleichen. Lassen wir uns dann vom Kompass der Eigentumsrechte an den digitalen Werkzeugen (Linke sagen hier oft: „Produktionsmitteln“) leiten, dann ergeben sich die Antworten auf die Frage nach den einzelnen Werkzeugen fast alle automatisch.

Danke.

Verständnisfragen? Sonst kann Susanne direkt anschließen.