FANCY BEAR EM AÇÃO

FANCY BEAR EM AÇÃO

Hackers e Guerra Cibernética

SCOTT J. SHAPIRO

ALTA BOOKS
GRUPO EDITORIAL
Rio de Janeiro, 2026

Fancy Bear em Ação

Alta Books é uma Editora do Grupo Editorial Alta Books.

ISBN: 978-85-508-2981-4

Translated from the original Fancy Bear Goes Phishing © 2023 by Scott J. Shapiro. All rights reserved. ISBN 9780374601171. Published by arrangement with Farrar, Straus and Giroux. PORTUGUESE language edition published by Starlin Alta Editora e Consultoria LTDA, Copyright © 2026 by STARLIN ALTA EDITORA E CONSULTORIA LTDA.

Impresso no Brasil — 1ª Edição, 2026 — Edição revisada conforme o Acordo Ortográfico da Língua Portuguesa de 2009.

Dados Internacionais de Catalogação na Publicação (CIP)

S529f
Shapiro, Scott J.
Fancy Bear em Ação: Hackers e Guerra Cibernética / Scott J. Shapiro. - Rio de Janeiro : Alta Books, 2026.
452 p.; il. 15,7 x 23 cm.

Titulo original: *Fancy Bear Goes Phishing: The dark history of the information age, in five extraordinary hacks.*
ISBN 978-85-508-2981-4

1. Cibersegurança. 2. Hackers. 3. Segurança de computadores. 4. Guerra cibernética. 5. Crimes informáticos. I. Titulo.

CDD 005.8

Índice para catálogo sistemático:

1. Segurança de computadores / cibersegurança — 005.8

Produção Editorial: Grupo Editorial Alta Books
Diretor Editorial: Anderson Vieira
Editor da Obra: J. A. Ruggeri
Vendas Governamentais: Cristiane Mutüs

Produtor Editorial: Tentáculos Editorial
Revisão Técnica:
Antônio Carlos Pereira Borge
Daniel Sadoc Menasche
Henrique Rabelo de Andrade

ALTA BOOKS
GRUPO EDITORIAL

Rua Viúva Cláudio, 291 — Bairro Industrial do Jacaré
CEP: 20.970-031 — Rio de Janeiro (RJ)
Tels.: (21) 3278-8069 / 3278-8419
www.altabooks.com.br — altabooks@altabooks.com.br
Ouvidoria: ouvidoria@altabooks.com.br

Editora
afiliada à:

Para minha mãe, Elaine Shapiro, por tudo,
especialmente por aquela conversa na 110th Street.

Sumário

Prefácio da Edição Brasileira

É com **muito orgulho** que a Clavis Segurança da Informação finaliza mais um projeto de tradução de uma obra relevante sobre cibersegurança juntamente com a Alta Books. Desta vez, foi escolhido o livro *Fancy Bear em Ação: Hackers e Guerra Cibernética*, de Scott J. Shapiro.

Este lançamento consolida a Clavis como a principal curadora e produtora de literatura técnica e estratégica sobre cibersegurança no Brasil. Nossa missão vai além da tecnologia; buscamos democratizar o conhecimento de alto nível, servindo como ponte entre os maiores pensadores do mundo e os profissionais brasileiros. Esta obra se junta ao nosso prestigiado catálogo de traduções:

Guerra Cibernética: *A próxima ameaça à segurança e o que fazer a respeito*, de Richard A. Clarke e Robert K. Knake.

Fundamentos de Segurança da Informação: *Com base na ISO 27001 e na ISO 27002,* de Jule Hintzbergen, Kees Hintzbergen e outros.

Contagem Regressiva Até Zero Day: *Stuxnet e o lançamento da primeira arma digital do mundo*, de Kim Zetter.

Clique Aqui para Matar Todo Mundo: *Como sobreviver em um mundo hiperconectado,* de Bruce Schneier.

Sandworm: *Uma nova era na guerra cibernética e a caça pelos hackers mais perigosos do Kremlin*, de Andy Greenberg.

> ***O Quinto Domínio:*** *Defendendo nosso país, nossas companhias e nós mesmos na era das ameaças cibernéticas*, de Richard A. Clarke e Robert K. Knake.

Agradeço profundamente à nossa equipe técnica de tradução — **Antonio Carlos Pereira Borge, Bruno Salgado, Daniel Sadoc Menasche, Henrique Andrade e Raphael Machado** — pelo rigor e excelência na adaptação desta obra.

Convido você a mergulhar nesta narrativa que revela as engrenagens reais de um poder invisível que molda a geopolítica moderna. Mais do que um relato histórico, esta obra funciona como um manual de sobrevivência para o século XXI; entender a mentalidade e as táticas desses adversários é o primeiro passo para garantir que a nossa soberania e a nossa privacidade permaneçam protegidas em um mundo em constante conflito digital.

Sobre a Clavis Segurança da Informação

A Clavis é uma plataforma de cibersegurança e privacidade que combina uma mentalidade jovem e inovadora com mais de 22 anos de experiência no mercado, integrando tecnologia proprietária, serviços gerenciados e um time altamente especializado para oferecer uma abordagem acessível, ágil e abrangente para a proteção dos riscos cibernéticos. Oferece uma plataforma de segurança, SOC 24/7, pentesting e consultoria em GRC (Governança, Riscos e Compliance) para empresas de todos os tamanhos. A Clavis é a espinha dorsal da resiliência digital brasileira, protegendo mais de **130 milhões de pessoas.**

Atuamos como parceiro estratégico das organizações na prevenção, detecção e resposta a riscos cibernéticos, permitindo que nossos clientes foquem exclusivamente em seus negócios enquanto cuidamos da segurança, conformidade e continuidade operacional. Atualmente, a Clavis protege mais de 350 organizações em diferentes setores estratégicos, contando com uma equipe multidisciplinar formada por mais de 200 colaboradores especializados em cibersegurança, privacidade, infraestrutura crítica e inteligência contra ameaças.

Reconhecida pelo Ministério da Defesa como uma **Empresa Estratégica de Defesa (EED)**, a Clavis também possui produtos classificados como **Produtos Estratégicos de Defesa (PED)**, reforçando sua relevância para a proteção de infraestruturas digitais e para a soberania nacional no âmbito da segurança cibernética. Além disso, sua plataforma possui certificação AWS Foundational Technical Review (FTR), validando aderência a elevados padrões de segurança, confiabilidade e boas práticas de arquitetura em nuvem.

Ao longo de sua trajetória, a Clavis consolidou-se como referência em monitoramento de segurança, gestão de vulnerabilidades, inteligência contra ameaças, segurança ofensiva, proteção de ambientes em nuvem e capacitação profissional por meio da Academia Clavis. Combinando inovação tecnológica, atuação estratégica e forte compromisso com a excelência técnica, a empresa contribui ativamente para o fortalecimento da resiliência digital no Brasil e na América Latina.

Boa leitura!

— **Victor Santos,** Cofundador e CEO da Clavis Segurança da Informação

FANCY BEAR EM AÇÃO

Introdução:

O Projeto Brilhante

"Acho que fiz uma grande merda." Paul sabia que Robert estava em sérios apuros. O quieto estudante de pós-graduação de 22 anos, de óculos, não costumava falar palavrão. Mais tarde, Paul testemunhou no julgamento de Robert que seu amigo "costumava ser um tanto puritano na forma de falar. Por isso achei que havia algo de muito grave acontecendo".[1]

Havia realmente algo de *muito* grave. O amigo de Paul havia acabado de derrubar a internet.

A ligação chegou às 23 horas, em 2 de novembro de 1988.[2] Robert Morris Jr., doutorando em ciência da computação em Cornell, descreveu o desastre em andamento a Paul Graham, um estudante de pós-graduação em Harvard.

Mais cedo naquela noite, por volta das 20 horas, Robert se sentou diante de um terminal na Sala 4160 do Upson Hall, então sede do Departamento de Ciência da Computação de Cornell, em Ithaca, Nova York,[3] e fez login remoto em prep.ai.mit.edu, um computador VAX 11/750 do Laboratório de Inteligência Artificial do MIT, em Cambridge, Massachusetts. Ele transferiu e executou três arquivos, liberando assim aquilo que ele e Paul haviam chamado de "o projeto brilhante" — um programa autorreplicante... um "worm", ou "verme" de computador.[4]

O worm havia sido programado para infectar computadores na então incipiente internet. Após a infiltração, o computador serviria como base para infectar outros. A cada novo alvo, o worm se autorreplicava e despachava seu clone para um novo lar. Trabalhando em conjunto, o worm e seus clones continuariam se multiplicando até completar sua missão e colonizar toda a internet.

A motivação de Robert era puramente científica; ele queria criar um programa que pudesse explorar o ciberespaço. Ele estava tentando infectar o maior número possível de computadores *simplesmente para ver quantos conseguiria infectar*, não para derrubá-los e causar estragos. Mas quando Robert voltou do jantar para verificar o progresso de seu experimento, percebeu que a

rede estava lenta. Havia um lag[5] evidente entre digitar e os caracteres aparecerem na tela, e entre o comando e a execução. O worm estava se espalhando rápido demais e consumindo recursos demais. Ele havia feito o caminho de volta de Cambridge para Ithaca em menos de três horas e estava tomando conta da rede de seu departamento. E isso era só o começo.

O worm de Robert não apenas paralisou a rede de Cornell; ele avançava pela internet, atropelando tudo em seu caminho. Apenas alguns minutos após seu lançamento no MIT, a primeira infecção conhecida do worm e ocorreu na Universidade de Pittsburgh.[6] De Pittsburgh, o worm cruzou o país em alta velocidade e atingiu rand.org, a rede da RAND Corporation em Santa Monica, Califórnia, às 20h24. Dentro de uma hora, os gerentes de informática da RAND notaram que a rede estava ficando lenta; vários nós estavam parados. Às 21 horas, o worm foi identificado circulando pelo Stanford Research Institute. Às 21h30, estava na Universidade de Minnesota. Às 22h04, infiltrou-se na máquina de entrada de Berkeley, o computador que servia como portal da universidade para a internet. Quase de imediato, os administradores notaram uma carga excepcionalmente alta na máquina e um acúmulo no sistema. À meia-noite, administradores do MIT voltaram de uma pausa para um sorvete e descobriram que a rede deles também estava falhando. À 1h05, o worm penetrou no Lawrence Livermore National Laboratory, um local responsável por proteger o arsenal nuclear do país. Logo ele havia se infiltrado no Los Alamos National Laboratory, no Novo México, lar do Projeto Manhattan e das primeiras bombas atômicas do mundo. O projeto brilhante de Robert já não parecia tão brilhante.

A situação na Universidade de Utah era típica. O primeiro ataque a cs.utah.edu ocorreu logo após a meia-noite, pelo sistema de email, às 0h09. Em onze minutos, a carga da rede — a quantidade de dados transportados — chegou a 5. Numa noite normal, a carga variava entre 0,5 e 2. Um 5 significava lentidão; um 20 significaria um colapso. Às 0h41, a carga em Utah aumentou para 7. Vinte minutos depois, 16. Cinco minutos depois, a rede inteira caiu. Jeff Forys, o administrador de Utah, derrotou os invasores um a um até que todos desapareceram — apenas para retornarem com força total menos de uma hora depois. A carga chegou a 27. À 1h49, Forys desligou a rede, o que matou os novos intrusos. Mas quando ele a religou, outro enxame atacou. A carga disparou para 37, e Forys não conseguiu reduzi-la. O jogo de gato e rato[7] não estava funcionando.

O telefone acordou Dean Krafft, chefe das instalações de computação do Upson Hall, onde Robert Morris havia lançado seu malfadado experimento. "À 1h30, recebi uma ligação de um estudante veterano de pós-graduação do departamento me dizendo que parecia haver um problema de segurança e que várias das máquinas estavam travando",[8] testemunhou Krafft mais tarde. Vinte por cento dos computadores do departamento de Cornell estavam

travados. Quando as máquinas eram desligadas e reiniciadas, funcionavam por um curto período, mas logo voltavam a travar. Krafft disse ao estudante de pós-graduação para desconectar os computadores do departamento da rede principal do campus.[9] (Cornell teve sorte. Na Carnegie Mellon, oitenta de cem computadores foram afetados; na Universidade de Wisconsin, duzentos de trezentos. O Bell Labs,[10] o braço de pesquisa e desenvolvimento da gigante telefônica AT&T, não foi afetado.)

Às 2h38, Peter Yee, do NASA Ames Research Center, publicou o primeiro alerta público na lista de emails TCP-IP, o principal quadro de avisos[11] (público para notícias relacionadas à internet: "Estamos atualmente sob ataque de um VÍRUS da internet.[12] Ele atingiu UC Berkeley, UC San Diego, Lawrence Livermore, Stanford e NASA Ames." Ele aconselhou todos a desligarem certos serviços de rede, como email, para conter a propagação.

Especialistas em segurança da computação temiam esse dia havia anos. A internet estava crescendo a uma taxa tão explosiva, conectando redes de computadores por todo o país — na verdade, pelo globo — que eles temiam que fosse atacada por uma potência estrangeira hostil.[13] E em 2 de novembro de 1988, eles presumiram que o momento havia chegado. Stevan Milunovic, diretor de sistemas de informação do Stanford Research Institute, disse ao *New York Times*: "Pensei: 'Esta é a catástrofe que vínhamos antecipando, e ela finalmente chegou.'"[14]

Esses primeiros respondentes não faziam ideia de que o ataque vinha de um estudante do primeiro ano da pós-graduação, de Millington, Nova Jersey, que havia ido dormir naquela noite aterrorizado, esperando que, de alguma forma, pela manhã, o pesadelo tivesse acabado. Mas, quando ele acordou, não tinha.

Crescendo em Nova Jersey

É difícil não sentir empatia por Robert Morris Jr. Segundo todos os relatos, ele era um jovem brilhante, mas tímido e desajeitado.[15] Deve ter sido aterrador derrubar a internet e se tornar matéria de destaque nacional — um vilão atrapalhado. Eu nem consigo imaginar a humilhação. (Na verdade, quase consigo. Eu fiz um bar mitzvá.)

Escrevo sobre muitos hackers neste livro, mas é com Robert que sinto a conexão mais profunda, provavelmente pela simples razão de termos a mesma idade e virmos de origens muito semelhantes. Não sei se nossos pais se conheciam, mas eles trabalharam no Bell Labs, em Morristown, Nova Jersey, ao mesmo tempo e ambos eram matemáticos. Robert e eu costumávamos visitar "o Labs". Talvez tenhamos participado dos mesmos dias de "leve seu filho para o trabalho". Ambos éramos obcecados pelo sistema operacional UNIX e líamos os manuais por diversão. Ambos estudamos ciência da computação

na faculdade. E ambos seguimos para o doutorado e hoje somos professores titulares. Robert é professor de ciência da computação no MIT; eu desviei do caminho e acabei como filósofo na Yale Law School.

Robert e eu fomos apresentados aos computadores por nossos pais. Robert Morris Sr., pai de Robert, instalou um terminal remoto na casa da família,[16] na área rural de Nova Jersey ainda em 1964. Robert usava o terminal para se conectar à rede do Bell Labs pela linha telefônica. Meu pai não instalou um terminal em nossa casa em Nova Jersey, um prédio multifamiliar em Paterson, mas me trazia uma variedade infinita de microchips, resistores, capacitores, diodos, LEDs e "breadboards" (plataformas reutilizáveis de encaixe para esses componentes eletrônicos). Eu usava essas diversas peças para montar computadores rudimentares capazes de resolver problemas simples de matemática. Nosso passeio anual de pai e filho era um dia na convenção do IEEE (Institute of Electrical and Electronics Engineers), no decadente New York Coliseum, em Manhattan, onde os brindes vinham na forma de microchips ultrapassados. Eu vasculhava grandes caixas em busca desses chips e os levava para casa empolgado, onde os encaixava nas breadboards, curioso para ver o que, se é que algo, aconteceria.

Alguns anos depois, meu colega de classe Ritchie Seligson me apresentou à programação de computadores. Um dia, na aula de biologia do nono ano, notei que ele examinava atentamente uma impressão de computador. Ritchie estava calculando os horários do pôr do sol de todas as sextas-feiras daquele ano. Essa era uma informação importante. Eu estudava em uma escola judaica, e os pores do sol de sexta-feira marcavam o início do Shabat judaico, quando entravam em vigor rigorosas regras de observância. Mas eu estava confuso. A programação dos pores do sol era tão importante que aparecia impressa com destaque em nossos livros de orações. Então por que Ritchie estava recalculando tudo?

Ele disse que era divertido. Tive minhas dúvidas. Quão divertido poderia ser recalcular um calendário religioso? Mas isso mudou quando ele me mostrou o código. Nossa sala de biologia tinha um TRS-80, o primeiro computador pessoal produzido em massa. No terminal, Ritchie digitou "For x = 1 to 10; Print x; Next x". Em seguida, apertou Enter, e os números de 1 a 10 apareceram magicamente na tela.

> **12345678910**

Fiquei maravilhado. Para ser sincero, gostaria que tivesse sido algo mais impressionante ou sofisticado que me conquistasse para a programação. Mas bastou um código curto que alinhava os numerais de 1 a 10 na tela. Fiquei obcecado por programação de computadores pela década seguinte.

No ensino médio, Robert, ao contrário de mim, se envolveu com hacking. Seu pai era especialista em criptografia, o estudo das comunicações seguras por meio de códigos. Ele passava horas conversando com Robert sobre segurança de computadores. Meu pai era especialista em linhas de transmissão de alta tensão, com ênfase em transformadores elevadores de tensão. Ele não tinha interesse em cibersegurança, e eu também não. (Francamente, eu também não tinha interesse em transformadores elevadores de tensão.)

Assim, ao contrário dos hackers precoces deste livro, cheguei tarde à cibersegurança. Todos sobre quem escrevo aqui começaram a invadir computadores ainda adolescentes — em geral por volta dos 14 anos. Sempre fui alguém que amadurece mais tarde. Invadi meu primeiro computador aos 52 anos.

O fim dos anos 1970 foi uma boa época para crescer se você fosse filho de um engenheiro eletricista. Era o início da revolução do computador pessoal, aquele momento eletrizante em que startups como Apple e Microsoft enfrentavam a gigante IBM vendendo microcomputadores e software diretamente aos consumidores. O TRS-80 da minha sala de biologia era vendido pela Radio Shack,[17] uma rede nacional de lojas de eletrônicos, hoje em grande parte extinta. Pela primeira vez na história do mundo, qualquer pessoa podia entrar em uma loja e comprar um computador digital de uso geral. O TRS-80 custava US$399,[18] o equivalente a aproximadamente US$1.700 em valores de 2023.

Quando ficou claro que programar era a minha paixão, meus pais me compraram meu próprio computador Apple II. O Apple II custava US$1.298, cerca de US$5.500 em valores de 2023, e isso não incluía monitor, unidade de disquete nem impressora — apenas 4 kilobytes de RAM (memória de acesso aleatório). Meu iPhone, em contraste, tem 4 gigabytes de RAM, um milhão de vezes mais capacidade de memória (4.000.000.000 contra 4.000 bytes). O resto era improvisado. Eu usava uma velha TV preto e branco como monitor, que conseguia exibir 40 caracteres por linha. (A Apple vendia uma placa de vídeo que dobrava a largura para 80 caracteres, mas meus pais traçaram o limite em 40.) Eu armazenava programas em fitas cassete.[19] Para carregar um programa, eu reproduzia aqueles chiados irritantes — pense em aparelhos de fax — diretamente no Apple II. Incrivelmente, uma em cada três vezes, funcionava.

As coisas ganharam ritmo nos anos 1980, quando me tornei aluno de ciência da computação na Columbia College, passando dias e noites intermináveis em porões muito iluminados, programando em PASCAL e FORTRAN — linguagens de programação antigas que hoje raramente são ensinadas. Por um curto período, cheguei até a ser um empreendedor de tecnologia. Abri uma

empresa de informática especializada na construção de bancos de dados e lhe dei o nome pomposo de "Scott Shapiro Consultants". Entre meus clientes estavam o banco de investimentos Donaldson, Lufkin & Jenrette e a Time-Life Books. Naquela época, habilidades em construção de bancos de dados eram raras.

Porém, com o tempo, perdi o interesse por computadores. Depois de concluir ciência da computação, fui para a faculdade de direito em Yale e, em seguida, voltei para Columbia, onde comecei a me dedicar a um doutorado em filosofia. Decidi encerrar minha empresa de informática no início dos anos 1990, justamente quando a World Wide Web foi inventada. Perdi o contato com a tecnologia digital e, com isso, minha chance de ganhar bilhões.

Não voltei a pensar seriamente em computação por quase três décadas. Cerca de sete anos atrás, concluí um longo projeto de livro, *The Internationalists*, escrito em coautoria com minha colega Oona Hathaway, sobre a história da guerra ao longo dos últimos quatro séculos e os diversos esforços para detê-la. Pesquisar e escrever *The Internationalists* despertou uma série de questões sobre o futuro da guerra — a próxima fase que os especialistas estavam chamando de guerra cibernética. A guerra cibernética representa uma ruptura em relação à guerra tradicional, ou ambas são guerra, apenas com armas diferentes? As regras do direito estabelecidas por batalhas antigas e aperfeiçoadas ao longo de séculos de combates terrestres e navais fazem sentido para o novo mundo da guerra cibernética? Os especialistas estão certos ao anunciar a guerra cibernética como a maior ameaça isolada à nossa segurança?[20] Diante da minha ampla formação técnica em ciência da computação, imaginei que não levaria muito tempo para me atualizar.

Mas eu estava errado. Muito errado.

• • • • • • • • • • •

Como Rip Van Winkle, eu havia dormido durante a revolução e acordei, várias décadas depois, desorientado e sem entender nada. *Linux? Apache? Python? JavaScript?* Eu não fazia ideia do que eram. A internet já existia quando eu era estudante universitário, mas eu quase não a usava. A World Wide Web foi criada em 1989, portanto, não havia websites para visitar antes disso, e o primeiro navegador gráfico para acessar websites só foi desenvolvido em 1992.[21] Eu usava email, mas quase sempre para me comunicar com colegas de classe. Nunca me ocorreu enviar um *ping*[22] para alguém fora da universidade. Redes sociais, comércio eletrônico, celulares acessíveis — tudo isso ainda estava a anos de distância.

Ainda mais confuso era o mundo do hacking, um lugar repleto de jargões traiçoeiros. *Honeypots? Sinkholing? Fuzzing? Shellcode? Mimikatz? Ataques do tipo "evil maid"?*[23] Que diabos é um ataque do tipo "evil maid"?! Tudo parecia

obscuro, ininteligível e impossivelmente abstrato. Mas eu estava cada vez mais consciente de que não conseguiria fazer meu trabalho principal, que era estudar a guerra cibernética, se não me atualizasse.

Adaptando a famosa frase de Leon Trótski sobre a guerra, talvez você não esteja interessado em hacking, mas o hacking está interessado em você. O hacking agora faz parte da vida cotidiana. Pesquisadores estimam que metade de todos os crimes contra a propriedade ocorre na internet.[24] O crime está, de forma lenta, mas contínua, se tornando *ciber*crime. O setor privado, em particular, está aflito com o custo crescente. As estimativas de prejuízo variam muito, de US$600 bilhões a US$6 trilhões por ano.[25] Segundo Ginni Rometty, ex-CEO da IBM, "o cibercrime é a maior ameaça[26] a todas as empresas do mundo". Ironicamente, a produção deste livro foi interrompida por um ataque de ransomware à empresa controladora da minha editora, a Macmillan.[27] Durante cerca de uma semana, meu livro sobre hacking foi hackeado.

Considere a espionagem. Ela é uma característica central do Estado moderno, e a ciberespionagem é sua encarnação mais recente. Em dezembro de 2020, para citar apenas um exemplo recente, o *Washington Post* noticiou que um hacker patrocinado por um Estado-nação — hoje considerado como sendo a inteligência russa — violou os servidores da SolarWinds, uma empresa do Texas que vende software para que organizações monitorem suas redes de computadores.[28] A SolarWinds tem uma vasta base de clientes, com 300 mil clientes privados e 32 agências-chave do governo dos EUA, incluindo o Pentágono, o Cyber Command, o FBI, o Tesouro e os Departamentos de Segurança Interna, de Comércio e de Saúde e Serviços Humanos.

Em março de 2020, a SolarWinds havia distribuído um "patch" que tinha como objetivo corrigir vulnerabilidades de segurança, mas que acabou implantando malware[29] em seus clientes. Conhecido como um ataque à *supply-chain*, o hack[30] se infiltrou em dezoito mil redes. Não apenas grandes agências do governo dos EUA foram comprometidas, incluindo o Pentágono, o Departamento de Justiça e o Departamento do Tesouro, como o alcance global da SolarWinds fez com que a OTAN, o governo do Reino Unido e o Parlamento Europeu também fossem afetados. Até a Microsoft foi comprometida.[31] Segundo Brad Smith, presidente da Microsoft, os hacks da SolarWinds foram "o maior e mais sofisticado ataque que o mundo já viu".[32]

Governos estrangeiros não são os únicos hackers por aí. Em 2013, Edward Snowden revelou que a NSA estava espionando americanos de diversas formas não divulgadas. (Falarei delas em detalhes mais adiante.) Mas não precisávamos de Snowden para saber que Estados espionam seus próprios cidadãos. A legislação americana é bastante clara ao estabelecer que o FBI e a NSA têm o direito de vigiar americanos em uma ampla gama de situações. Como muitos de nós, eu queria entender melhor essa vigilância doméstica. Até que ponto havia abusos e quão assustado ou indignado eu deveria estar? Mas, mais uma vez,

sem compreender como esses esforços se davam na prática e como a tecnologia funciona, avançar seria difícil, talvez até impossível.

E não era só eu. Fiquei chocado com a quantidade de pessoas — inclusive especialistas — que me disseram não ter a menor ideia do que realmente são guerra cibernética, cibercrime e ciberespionagem. Passadas décadas na era da internet, meus alunos são todos nativos digitais, que passaram uma grande parte da vida em uma plataforma online ou outra. Ainda assim, a maioria deles não tem nenhuma noção de como a internet funciona — nem de como funcionam os computadores. Muitos desses estudantes altamente motivados, curiosos e capazes acabarão trabalhando no governo, onde irão elaborar e implementar leis e regulamentos. Outros ingressarão em startups ou escritórios de advocacia cujos clientes incluem grandes empresas de tecnologia. Mas como eles vão entender o novo "cenário de ameaças",[33] para usar um termo do hacking, quando não há ninguém para explicá-lo? Mesmo quando entram no aquecido setor de cibersegurança, é provável que nunca aprendam os fundamentos do hacking. Muitos advogados de cibersegurança que conheci admitem que, boa parte do tempo, não fazem ideia do que diabos seus clientes estão dizendo. Ainda assim, suas decisões afetarão a segurança das empresas desses clientes. E essas decisões nos afetam por sua vez, porque são esses clientes que controlam *nossos* dados.

Vivemos em uma sociedade da informação em que riqueza, status e vida social dependem do armazenamento, da manipulação e da transmissão de informações. O número de dispositivos digitais no mundo hoje supera em muito o número de seres humanos; existem pelo menos 15 bilhões[34] de computadores para apenas 8 bilhões de pessoas. A segurança — seja ela pessoal, econômica, nacional ou internacional — envolve necessariamente uma cibersegurança eficaz.[35] Ainda assim, nós, cidadãos dessa nova sociedade da informação, quase não temos ideia de como nossas informações são armazenadas, usadas, protegidas e exploradas.

• • • • • • • • • • •

Iniciei este projeto com três perguntas básicas. Primeiro, eu queria saber por que a internet é tão insegura. Eu conseguia entender por que a internet costumava ser insegura. Afinal, ela foi projetada no fim da década de 1960. Certamente, havia ajustes a serem feitos. Mas por que ainda existem tantas vulnerabilidades várias décadas depois?

Em segundo lugar, eu queria saber como os hackers fazem o que fazem. Quando eu olhava para o meu computador, tudo o que via era a tela de login. E, se eu não soubesse minha senha, estava sem saída. Como hackers do outro lado

do mundo conseguiam contornar o sistema de segurança do meu computador e roubar meus dados?

Por fim, eu queria saber o que poderia ser feito. O fato de eu não dominar os problemas centrais significava que eu não estava em posição de pensar em soluções. Tornar a internet mais segura seria simplesmente uma questão de senhas mais fortes? Ou de letramento dos usuários? Se as pessoas entenderem como os computadores funcionam, passarão então a praticar uma melhor higiene cibernética e se tornarão menos vulneráveis ao cibercrime? Outra possibilidade seria construir tecnologias melhores, como softwares antivírus mais potentes e criptografia mais robusta[36] para manter nossos dados em segredo. Ainda mais extremo seria criar grandes firewalls nacionais para impedir que malwares atravessem fronteiras internacionais, à semelhança do que China e Rússia fizeram para bloquear conteúdo político online. Eu estava até aberto à possibilidade de que nossos problemas atuais sejam tão intratáveis que precisemos redesenhar e reconstruir a internet do zero, tendo a segurança como prioridade.

Despertar do meu longo sono digital significou voltar ao básico. Precisei reaprender C (uma linguagem de programação padrão) e código Assembly x86 (uma linguagem de programação irritante, mas poderosa), pois fazia trinta anos desde a última vez que eu os havia usado. Aprendi Linux, um sistema operacional gratuito baseado em UNIX, que eu conhecia dos tempos da graduação. Também tive de entender como a internet funcionava.

Mas o básico só me levou até certo ponto. Eu precisava aprender a "hackear o kernel". O "kernel" é a parte mais interna do sistema operacional e o Santo Graal do hacking. Quem "controla" o kernel controla o sistema operacional. Então, assisti como ouvinte a uma disciplina de pós-graduação sobre sistemas operacionais no departamento de Ciência da Computação de Yale, no qual aprendi a construir um kernel. Tornei-me presença constante em grandes convenções de hackers como DEFCON, Black Hat e Enigma. Inscrevi-me em boot camps (curso intensivo) de cibersegurança para administradores de sistemas. E hackeei o site da Faculdade de Direito de Yale, uma façanha que não agradou o diretor.

Também mergulhei na história do hacking. Além de consumir montanhas de mídia e relatórios técnicos sobre os últimos cinquenta anos de hacks, eu precisava decifrar os programas maliciosos usados nesses ataques. Por isso, contratei um graduando extremamente inteligente, Daniel Urke, como assistente de pesquisa para me ajudar. Juntos, examinamos minuciosamente milhares de linhas de código de malware que possibilitaram os hacks mais infames da história.

Os malwares que estudei são exemplos do que chamo de downcode. Downcode é código técnico de computador. Pense nele como o código que está literalmente sob as pontas de nossos dedos quando digitamos em um

teclado. O downcode vai desde microcódigo incorporado em microchips e drivers de dispositivos que vêm com sua impressora, até sistemas operacionais como Windows, Linux e iOS, passando por código de aplicações escrito em linguagens de programação de alto nível como C e Java, código de websites que usam JavaScript e SQL, e softwares de comunicação que utilizam protocolos de rede como TCP/IP e HTTPS. (Não se preocupe, explicarei essas siglas mais adiante no livro.)

Se o downcode é o que está literalmente sob a ponta de nossos dedos, a instrução que digitamos, o *upcode*, é o que acontece além — desde as operações internas do cérebro humano até as forças sociais, políticas e institucionais externas que definem o mundo ao nosso redor. O upcode inclui os códigos mentais que moldam o pensamento e o comportamento humanos internamente e os códigos culturais que atuam sobre nós, muitas vezes de forma invisível, externamente: moralidade pessoal, rituais religiosos, normas sociais, regras legais, políticas corporativas, ética profissional, termos de serviço de websites. O downcode é executado por computadores; o upcode, por humanos.

Se eu pretendia aprender como o hacking funciona, não bastava aprender apenas o downcode do hacking. Eu precisava entender o upcode também — não apenas as leis formais que regulam o hacking de cima para baixo, mas as normas que os hackers criaram informalmente, as propensões incomuns da mente humana e os incentivos que impulsionam o mercado de software.

O upcode é fundamental para entender o hacking por um motivo simples: o upcode molda o downcode. Bill Gates não *descobriu* o Windows — a Microsoft, sua empresa, o construiu. As 50 milhões de linhas de código do Windows 10 são o produto de funcionários da Microsoft respondendo a muitas camadas de upcode.[37] Os programadores foram trabalhar na Microsoft porque ela é um empregador de colarinho branco estimulante e prestigioso (normas sociais); eles foram orientados a desenvolver o downcode por seus gerentes (políticas corporativas); eles foram pagos por seu trabalho porque a Microsoft detém a propriedade intelectual do downcode e gera receita a partir dele (regras legais); eles iam trabalhar todos os dias por uma combinação de motivações pessoais e normas e expectativas sociais (moralidade pessoal); e conseguiam seguir planos porque os humanos são muito bons em planejar (psicologia). O upcode molda o downcode, em outras palavras, porque o upcode molda o comportamento humano, e o downcode é um produto desse comportamento humano.

Além de upcode e downcode, estudei a filosofia da computação. Hackers, como mostrarei, não hackeiam apenas o downcode — eles exploram princípios filosóficos, que chamo de "metacódigo". Metacódigo refere-se àqueles princípios fundamentais que controlam todas as formas de computação. Eles determinam o que é a computação e como ela deve funcionar. O metacódigo, em outras palavras, é o código do código — o código que precisa "rodar" antes que as instruções do computador possam ser executadas.

O metacódigo foi descoberto por Alan Turing, o engenhoso matemático cuja vida trágica é retratada no filme vencedor do Oscar *O Jogo da Imitação*. Turing é mais conhecido por ter ajudado a decifrar o código alemão Enigma durante a Segunda Guerra Mundial e por desenvolver um teste para a inteligência artificial, hoje conhecido como Teste de Turing.[38] O Teste de Turing afirma que um computador possui inteligência quando consegue enganar um humano, fazendo-o acreditar que ele é humano. Apesar de suas muitas contribuições para seu país e para a humanidade, Turing foi processado e punido pelo governo britânico por ter mantido relações sexuais com outro homem. Ele cometeu suicídio em 1954, comendo uma maçã contaminada com arsênico.

Alan Turing tinha apenas 24 anos em 1936 quando publicou seu artigo seminal, "On Computable Numbers", no qual estabeleceu os princípios do metacódigo.[39] Turing mostrou, por exemplo, que a computação é um processo físico. Quando sua calculadora soma 2 + 2, quando a Amazon.com busca um livro em seu banco de dados, quando a companhia telefônica encaminha sua ligação ou mesmo quando seu córtex visual processa estas palavras, mecanismos físicos estão em funcionamento: comutação de circuitos, envio de pulsos de luz, formação de reações neuroquímicas, e muito mais.

Como a computação é um processo físico, Turing demonstrou como seria possível construir um dispositivo físico de computação, isto é, um computador. Desde que uma máquina consiga executar certas tarefas básicas, como ler e escrever símbolos, ela pode resolver um problema solucionável. Mas Turing fez uma descoberta ainda mais profunda. Ele manipulou o metacódigo não apenas para construir um computador capaz de resolver problemas específicos — ele mostrou como construir um computador *programável* capaz de resolver *qualquer* problema solucionável.[40]

Sem o metacódigo de Turing, como veremos, nosso mundo digital não teria se desenvolvido. Não haveria computadores capazes de executar o código que fornecemos a eles. O metacódigo torna possível a internet, websites, email, redes sociais, iPhones, laptops, filmes da Pixar, a economia de trabalhos sob demanda, mísseis de precisão guiada, naves espaciais, e-books, videogames, Bitcoin, reuniões no Zoom, apresentações em PowerPoint, planilhas, processamento de texto, torradeiras inteligentes, até mesmo o meu pobre, mas querido, Apple II com um gravador de fitas cassete para armazenamento.

Os próprios princípios que possibilitam o nosso mundo digital, no entanto, também possibilitam o hacking. Hackers não apenas abusam do downcode e se aproveitam do upcode — eles também exploram o metacódigo. Como veremos, Robert Morris construiu seu worm para manipular esses princípios filosóficos da computação. De fato, ele foi tão bem-sucedido ao explorar o metacódigo que se tornou o primeiro hacker a derrubar a internet.

O resultado mais surpreendente da minha imersão prolongada, quase obsessiva, na tecnologia, na história e na filosofia do hacking é eu não estar em pânico. Pelo contrário, concluí que muito do que se diz sobre hacking é errado, enganoso ou exagerado. Decidi escrever este livro porque fiquei empolgado com tudo o que descobri. Mas também para corrigir esses equívocos.

A imagem popular dos hackers é um bom exemplo disso. Políticos e comentaristas dão a impressão de que hacking é um ato invisível praticado com intenção maliciosa por jovens brilhantes e desajustados que usam moletons com capuz e pijamas o dia todo, vivem nos porões de seus pais e subsistem inteiramente de Red Bull. A estrela homônima da série de televisão *Mr. Robot*, por exemplo, é um hacker mentalmente doente que sofre de transtorno de personalidade múltipla.

A verdade, como descobri, é mais banal. O hacking não é uma arte obscura, e aqueles que o praticam não são magos de 180 quilos nem prodígios idiotas. Tampouco são sombras anônimas. Hackers têm nomes e rostos, mães e pais, professores, amigos, namoradas, inimigos-amigos, colegas e rivais. Eles são figuras familiares do cotidiano: adolescentes imaturos, engenheiros subempregados e pouco estimulados, pequenos criminosos, supergeeks e funcionários do governo que trabalham das 9 às 17 horas. É verdade que os hackers que você encontrará neste livro tendem a ser um pouco estranhos e socialmente desajeitados. Mas, convenhamos, quem não é?

O cibercrime é um negócio,[41] e negócios existem para gerar lucro. Cibercriminosos não querem ler seus emails nem usar sua webcam para espioná-lo enquanto você faz o jantar.[42] Em sua maioria, são pessoas racionais tentando ganhar a vida. E, embora possam praticar atos maliciosos, como roubar as informações do seu cartão de crédito ou criptografar seus dados, eles não querem desperdiçar um tempo precioso invadindo o *seu* computador. Se você tomar ao menos precauções mínimas, como evitar links que vêm de pessoas que você não conhece, talvez o cibercriminoso comum conclua que invadir o seu computador simplesmente não vale o esforço.

A mídia alimenta nossa insegurança cibernética com um fluxo interminável de histórias assustadoras. Quando o Departamento de Segurança Interna anunciou, em 2019, que vários marca-passos populares eram vulneráveis a hacking, embora nenhum caso tenha sido relatado, a *Healthline* publicou uma reportagem com uma machete arrepiante: "Criminosos podem hackear o seu coração".[43] Em 2017, a CNN noticiou o caso de um adolescente alemão que explorou uma vulnerabilidade em um aplicativo instalado em certos modelos da Tesla, o que lhe permitiu manipular alguns dos recursos

não relacionados à condução do veículo, como as travas das portas e as luzes.[44] A mídia fez grande alarde quando, em 2016, pesquisadores anunciaram que haviam hackeado o We-Vibe — o primeiro vibrador inteligente do mundo. Eles conseguiram controlar o vibrador sem o consentimento da usuária (potencialmente cometendo uma forma remota e aterradora de agressão sexual criminosa).

Todos esses são cenários aterradores. Produtos com falhas de segurança escancaradas não deveriam chegar ao mercado. Alguns criminosos hackeiam por diversão ou pelo desafio. Outros têm motivações mais sombrias. Mas a grande maioria do cibercrime tem motivação financeira. Assim, o fato de vulnerabilidades serem ou não exploradas em geral depende da possibilidade de ganho. Em muitos casos, não há. É difícil ganhar a vida hackeando dispositivos médicos ou brinquedos sexuais a meio mundo de distância.

O tema que inspirou meu fascínio pelo hacking — a ciberguerra — é especialmente propenso a exageros. Há décadas, analistas de ameaças e filmes de Hollywood vêm alertando para um suposto Pearl Harbor Digital ou um 11 de Setembro Cibernético.[45] Segundo afirma, hackers seriam capazes de derrubar as redes militares do Pentágono, provocar explosões em refinarias de petróleo, liberar gás cloro de fábricas químicas, impedir a operação de aviões e helicópteros ao desativar o controle de tráfego aéreo, apagar informações financeiras do sistema bancário e mergulhar os Estados Unidos na escuridão ao desligar a rede elétrica, matando milhares de pessoas no processo. Nas palavras do jornalista do *New York Times* David Sanger, uma arma cibernética é a "arma perfeita".[46] A ciberguerra é inevitável, alertam os especialistas. Não podemos impedi-la; só podemos tentar nos preparar.

Mas, felizmente, a verdade é menos dramática.[47] Explorações computacionais não são armas perfeitas. Pelo contrário, são armas hiperespecializadas, frustradas pelos mesmos problemas de compatibilidade — ou interoperabilidade — que todos nós enfrentamos. Da mesma forma que um aplicativo que funciona em um iPhone não funciona em um telefone Android, um malware que opera em desktops com Windows quase nunca funciona em Macs.[48] Do mesmo modo, ataques capazes de infiltrar PDFs feitos com o Acrobat 9.3 podem ser inúteis contra PDFs criados com o Acrobat 9.4. Tudo isso significa que um ataque sistêmico bem-sucedido à infraestrutura digital de um país tecnologicamente avançado como os Estados Unidos, com sua diversidade de computadores, sistemas operacionais, configurações de rede e aplicações, exigiria não apenas um arsenal cibernético de proporções inimagináveis. Uma falha digital em larga escala também exigiria uma quantidade sobrenatural de sorte.

O alarmismo é, até certo ponto, inevitável. Muitos grupos têm interesse em inflar as ameaças cibernéticas. Autores vendem livros, jornalistas ganham cliques, empresas promovem seus produtos, consultores vendem seus

serviços, e autoridades governamentais protegem a própria pele. Cenários sensacionalistas de ciberguerra atraem atenção e rendem entretenimento empolgante. Histórias de terror sobre a tecnologia se voltando contra seus criadores e saindo do controle são um elemento constante da literatura e do cinema modernos, começando pelo clássico *Frankenstein*, de Mary Shelley, publicado em 1818.

A terminologia da cibersegurança, com suas metáforas características de poluição e doença, não ajuda. Falhas de software são conhecidas como *bugs* (insetos). Malware é composto por vírus e *worms* (vermes). Código contagioso se *replica* e se *dissemina* por meio de *vetores de infecção* para *contaminar hospedeiros*. Quando capturado por softwares antivírus, o malware é colocado em *quarentena* e *desinfetado* para evitar contágio adicional. Essas metáforas biológicas são intuitivas. Como veremos no capítulo 3, muitos tipos de malware de fato se assemelham à contaminação biológica. Mas a linguagem de poluição e doença também evoca sentimentos viscerais de nojo e repulsa. Ansiamos para evitar o contato com o objeto do nosso asco, com medo de que ele nos contamine. Associar um malware com excremento, vômito, mau hálito, furúnculos cheios de pus, lixo, carne apodrecida, ratos, baratas, larvas e deformidades corporais torna algo assustador ainda mais tenebroso.

Dito isso, não pretendo minimizar os danos nem os riscos do hacking. Em 2021, a Colonial Pipeline, que opera o maior sistema de oleodutos de petróleo refinado dos Estados Unidos, foi atingida por um ataque de ransomware que levou à interrupção do fornecimento de combustível por vários dias e a um aumento nos preços da gasolina. O ransomware também tem sido um flagelo para governos locais, hospitais e escolas. De fato, este livro está repleto de exemplos de ciberataques direcionados e prejudiciais.

É por isso que profissionais de cibersegurança são essenciais para qualquer empresa moderna. Ainda assim, é comum que esses profissionais trabalhem demais, e muitos também sejam mal remunerados. Depressão, ansiedade e abuso de substâncias são problemas sérios na comunidade de cibersegurança, na qual especialistas têm a tarefa de defender redes sobrecarregadas.[49] Estima-se que haja um *déficit de 3,5 milhões de profissionais na área de cibersegurança*. Se quisermos permanecer vigilantes diante dessas novas ameaças do século XXI, precisamos enfrentar o enorme abismo entre oferta e demanda.

Porém exagerar o risco é contraproducente. Ao provocar o pânico, a comunidade de cibersegurança acabou induzindo, de forma não intencional, o que psicólogos chamam de impotência aprendida.[50] Quando sentimos que não temos controle sobre as circunstâncias, quando nada do que fazemos faz diferença, ficamos paralisados, como o proverbial cervo diante dos faróis, incapazes de dar os poucos passos necessários para desviar do perigo. Esse sentimento de resignação impotente é uma das razões pelas quais usuários de computador praticam uma ciber-higiene tão precária — como clicar em links

em emails de pessoas desconhecidas ou usar senhas de seis dígitos que começam com 1 e terminam com 6. Por que se preocupar em ser cuidadoso com o email ou usar uma senha longa se o Armagedom é iminente?

• • • • • • • • • • •

Livros sobre cibersegurança — e há muitos — tendem a se enquadrar em uma de duas categorias. Ou adotam um estilo sem graça, do tipo "coma seus vegetais", ou então um tom alarmista e desesperado, "corra para as colinas agora". *Fancy Bear em Ação* procura evitar esses dois extremos. Não é um manual nem um guia do usuário, tampouco uma obra de profecia sombria. Minha esperança é que ele capacite os leitores, fornecendo ferramentas para responder às três perguntas que despertaram meu interesse por esse tema: por que a internet é tão vulnerável? Como os hackers exploram essas vulnerabilidades? O que empresas, Estados e o resto de nós podemos fazer em resposta?

Abordo essas questões por meio de histórias de cinco hacks.[51] O livro começa com o primeiro hack da internet — o chamado Morris Worm, que o então estudante de pós-graduação de Cornell, Robert Morris Jr., pretendia que fosse um experimento científico sofisticado, mas terminou mal, derrubando acidentalmente a internet e culminando na primeira condenação federal por hacking. A partir daí, passamos pela paixão não correspondida que levou ao primeiro motor de vírus mutante. Um hacker búlgaro conhecido pelo codinome Dark Avenger criou um vírus na forma de uma carta de amor para a pesquisadora de cibersegurança Sarah Gordon, uma demonstração de amor geek que ameaçou paralisar a então incipiente indústria de antivírus. O terceiro hack é ainda mais dramático. Veremos como um jovem de 16 anos do sul de Boston hackeou o celular da socialite celebridade Paris Hilton e vazou fotos nuas, apenas para que Paris Hilton fosse acusada de empregar um hack semelhante contra sua rival famosa, Lindsay Lohan. Em seguida, descrevo como o Fancy Bear, uma unidade de hacking dentro da inteligência militar russa, invadiu a rede de computadores do Comitê Nacional Democrata (DNC, na sigla em inglês) e, supostamente, ajudou a eleger Donald Trump presidente dos Estados Unidos. Por fim, explico como o "botnet Mirai", um supercomputador de hacking distribuído que um estudante de graduação da Rutgers criou para fugir de uma prova de cálculo e interromper o jogo online *Minecraft*, e quase destruiu a internet no processo.

Ao me aprofundar nesses cinco hacks épicos, também vou expor a tecnologia que os tornou possíveis. Minha esperança é que essas histórias de crime real — algumas acidentais, outras não — despertem leitores que tenham pouco ou nenhum interesse prévio em tecnologia e os capacitem a ir além das manchetes.

Ao mesmo tempo, essas cinco histórias também ilustram minha mensagem justamente porque mostram que as questões mais interessantes colocadas por nosso novo mundo turbulento têm pouco ou nada a ver com tecnologia em si. Compreender o que está acontecendo no ciberespaço, em larga escala, e por que nossas redes continuam inseguras, significa manter o foco nos seres humanos e nas normas e forças institucionais que os orientam. Ao longo deste livro, alternarei entre as peculiaridades da tomada de decisões humanas e essas forças mais amplas. Explicarei por que o mercado continua produzindo softwares de baixa qualidade e como a lei transformou o ciberespaço em uma vasta área de impunidade, além de destrinchar os fundamentos filosóficos da computação. Upcode, Downcode e Metacódigo interagem o tempo todo. Mas, ao longo de todo o livro, permanecerei focado nas pessoas. Hacking diz respeito a seres humanos, e meu objetivo é abordá-lo dessa forma.

1

O Grande Worm

Quando Robert Morris Jr. liberou seu worm às 20h, ele não fazia ideia de que talvez tivesse cometido um crime. Sua preocupação naquela noite era a reação de outros geeks: muitos administradores de UNIX ficariam furiosos quando descobrissem o que ele tinha feito. Como Cliff Stoll, um especialista em segurança computacional de Harvard, disse ao *New York Times*: "Não há um único administrador de sistemas que não esteja arrancando os cabelos. Isso está causando dores de cabeça enormes."[1] Quando o worm surgiu, os administradores não sabiam por que ele havia sido lançado nem que danos estava causando. Eles temiam o pior — que o worm estivesse apagando ou corrompendo os arquivos das máquinas que tinha infectado. (Não estava, como logo descobririam.)

Depois de confessar pela primeira vez a "merda feita" ao amigo Paul Graham, Robert soube que precisava fazer alguma coisa. Infelizmente, ele não podia enviar nenhum email de aviso, porque Dean Krafft havia ordenado que as máquinas do departamento fossem desconectadas da rede principal do campus e, portanto, da internet pública. Às 2h30, Robert ligou para Andy Sudduth, o administrador de sistemas do Aiken Computation Lab de Harvard,e pediu que ele enviasse uma mensagem de alerta a outros administradores, com instruções sobre como proteger suas redes. Embora não estivesse pronto para se expor, Robert também queria expressar remorso pelos problemas que estava causando. Andy enviou a seguinte mensagem:[2]

> De: foo%bar.arpa@RELAY.CS.NET
> Para: tcp-ip@SRI-NIC
> Data: Terça-feira 03:34:13 03/11/1988 EST
> Assunto: [sem assunto]
>
> ———————————————————————

> Comunicado de possível vírus:
> Pode haver um vírus circulando pela internet.
> Segue um resumo da mensagem que recebi:
> Sinto muito.
> Aqui estão algumas etapas para evitar novas transmissões:
> 1) Não execute o fingerd, ou corrija-o para que não cause stack overflow ao ler argumentos.
> 2) Recompile o sendmail sem o DEBUG definido
> 3) Não execute o rexecd
> Espero que isso ajude, mas, mais ainda, espero que seja uma farsa.

Andy sabia que o worm não era uma farsa e não queria que a mensagem fosse rastreada até ele. Ele havia passado a hora anterior elaborando uma forma de postar a mensagem anonimamente, decidindo enviá-la a partir da Brown University para uma Listserv popular da internet, usando um nome de usuário falso (foo%bar.arpa). Esperar até as 3h34 foi um azar. O worm se reproduziu com tanta rapidez que derrubou os roteadores que gerenciavam a comunicação da internet. A mensagem de Andy ficou presa em um engarrafamento digital e só chegaria ao destino 48 horas depois.[3] Os administradores de sistemas tiveram de se virar sozinhos.

Em 3 de novembro, enquanto os administradores com certeza estavam arrancando os cabelos, Robert ficou em casa, em Ithaca, dedicando-se às tarefas acadêmicas e evitando a internet. Às 23h, ligou para Paul para pedir uma atualização. Para seu horror, Paul contou que o worm havia se tornado uma sensação na mídia. Era uma das principais notícias do telejornal noturno de todas as emissoras; Robert não fazia ideia, já que não tinha televisão. Os jornais passaram o dia todo ligando para tentar descobrir o culpado. O *New York Times* publicou a história na parte superior da primeira página. Quando perguntaram o que ele pretendia fazer, Robert respondeu: "Não faço a menor ideia."

Dez minutos depois, Robert sabia que precisava ligar para o cientista-chefe responsável pela cibersegurança na Agência de Segurança Nacional (NSA, na sigla em inglês). Então pegou o telefone e ligou para Maryland. Uma mulher atendeu. "Posso falar com meu pai?",[4] perguntou Robert.

Os Primórdios da Cibersegurança

Especialistas em segurança vinham antecipando ataques cibernéticos havia muito tempo — mesmo antes de a internet ser inventada. A NSA organizou o primeiro painel sobre cibersegurança em 1967, dois anos antes de ser criado o primeiro link da ARPANET, o protótipo da internet. Foi há tanto tempo que a conferência aconteceu nos áureos tempos de Atlantic City.[5]

A preocupação da NSA cresceu com a evolução dos sistemas computacionais. Antes da década de 1960, os computadores eram máquinas titânicas alojadas em salas especiais próprias. Para submeter um programa — conhecido como um *job* (ou "trabalho") — o usuário entregava uma pilha de cartões perfurados a um operador de computador. O operador reunia essas tarefas em "lotes" (*batches*) e os passava por um leitor de cartões. Outro operador pegava os programas lidos pelo leitor e os armazenava em grandes fitas magnéticas. As fitas então alimentavam esse lote de programas no computador, muitas vezes em outra sala conectada por linhas telefônicas, para processamento por mais um operador.

Na era do processamento em lote, como era chamada, a segurança computacional era bastante literal: o próprio computador precisava ser protegido. Esses gigantes descomunais eram surpreendentemente delicados. O IBM 7090, que ocupava uma sala do tamanho de um campo de futebol americano[6] no Computation Center do MIT, era composto por milhares de frágeis válvulas a vácuo e quilômetros de fios de cobre intrincadamente enrolados. As válvulas irradiavam tanto calor que o derretimento dos fios era uma ameaça constante. A sala de computadores do MIT tinha seu próprio sistema de ar-condicionado. Esses computadores "mainframe"[7] — provavelmente assim chamados porque seus circuitos ficavam armazenados em grandes estruturas metálicas que se abriam para manutenção — também eram caros. O IBM 7094 custava US$3 milhões em 1963 (cerca de US$30 milhões em valores de 2023).[8] A IBM concedeu um desconto ao MIT, desde que a instituição reservasse 8 horas por dia para uso corporativo. O presidente da IBM, que gostava de velejar no estreito de Long Island, usava o computador do MIT para calcular os handicaps nas corridas.[9]

Regras burocráticas elaboradas determinavam quem podia entrar em cada uma das salas. Apenas alguns estudantes de pós-graduação tinham permissão para entregar cartões perfurados ao operador do processamento em lote. O critério para entrar na sala do mainframe era ainda mais rigoroso. A regra mais importante de todas era que ninguém podia tocar no computador, exceto o operador. Muitas vezes, uma corda o isolava, por via das dúvidas.

Nos primórdios da computação, portanto, cibersegurança significava proteger o hardware, não o software — o computador, não o usuário. Afinal, havia pouca necessidade de proteger o código e os dados do usuário. Como o computador executava apenas um job por vez, os usuários não conseguiam ler nem roubar as informações uns dos outros. No momento em que o job de um usuário rodava no computador, os dados do usuário anterior já haviam sido apagados.[10]

Porém os usuários odiavam o processamento em lote com a mesma intensidade de uma válvula a vácuo incandescente. Programadores achavam frustrante ter de esperar até que todos os jobs do lote fossem concluídos para

obter seus resultados. Pior ainda: reexecutar o programa, com ajustes no código ou com dados diferentes, significava voltar para o fim da fila e esperar a execução do próximo lote. Levava dias apenas para identificar bugs simples e fazer os programas funcionarem. Tampouco os programadores podiam interagir com o mainframe. Uma vez que os cartões perfurados eram entregues ao operador do computador, o envolvimento do programador terminava. Como descreveu o pioneiro da computação Fernando "Corby" Corbató,[11] o processamento em lote "tinha todo o glamour e a empolgação de levar roupas a uma lavanderia automática".

Corby decidiu mudar isso. Em 1961, trabalhando no MIT com outros dois programadores, ele desenvolveu o CTSS, Compatible Time-Sharing System (ou "Sistema Compatível de Compartilhamento de Tempo").[12] O CTSS foi projetado como um sistema multiusuário. Os usuários passariam a armazenar seus arquivos privados no mesmo computador e a executar seus próprios programas, time-sharingsem a necessidade de submeter cartões perfurados a operadores, já que cada usuário tinha acesso direto ao mainframe. Sentados em terminais próprios, conectados ao mainframe por linhas telefônicas, atuavam como os próprios operadores. Se dois programadores submetessem jobs ao mesmo tempo, o CTSS realizava um truque engenhoso: executava uma pequena parte do job 1, depois uma pequena parte do job 2, e voltava ao job 1. Alternava entre eles até que ambos fossem concluídos. Como o CTSS alternava muito rápido, os usuários mal percebiam a intercalação. Ficavam na ilusão de que tinham o mainframe só para si. Corby chamou esse sistema de "time-sharing" (compartilhamento de tempo).[13] Em 1963, o MIT tinha 24 terminais de time-sharing, conectados por seu sistema telefônico ao IBM 7094.[14]

O inferno, como escreveu Jean-Paul Sartre de forma célebre, são os outros.[15] E, como o CTSS era um sistema multiusuário, ele criou uma espécie de inferno da cibersegurança. Embora os mainframes agora estivessem seguros, já que ninguém precisava tocar no computador, nos leitores de cartões ou nas fitas magnéticas para executar seus programas, aqueles que criavam ou utilizavam esses programas passaram a ficar vulneráveis.

Um sistema de time-sharing funciona carregando vários programas na memória e alternando rapidamente entre os jobs para criar a ilusão de uso exclusivo.[16] O sistema aloca cada job em partes diferentes da memória — o que cientistas da computação chamam de "espaços de memória". Quando o CTSS alternava entre os jobs, ele passava de um espaço de memória para outro. Embora carregar o código e os dados de vários usuários no mesmo computador otimizasse recursos preciosos, isso também criava enorme insegurança. O job nº 1, em execução em um espaço de memória, podia tentar acessar o código ou os dados no espaço de memória do job nº 2.

Ao compartilhar o mesmo sistema computacional, as informações dos usuários passaram a ficar acessíveis a mãos e olhos curiosos. Para proteger a segurança de seu código e de seus dados, o CTSS concedia a cada usuário uma conta protegida por um "nome de usuário" exclusivo e uma "senha" de quatro letras. Usuários que faziam login em uma conta só podiam acessar o código ou as informações do espaço de endereçamento correspondente; o restante da memória do computador ficava fora de alcance. Corby escolheu senhas para autenticação por economia de espaço: armazenar uma senha de quatro letras consumia menos da preciosa memória computacional[17] do que guardar a resposta a uma pergunta de segurança como "Qual é o sobrenome de solteira da sua mãe?". As senhas eram mantidas em um arquivo chamado UACCNT.SECRET.[18]

Nos primórdios do time-sharing, o uso de senhas tinha menos a ver com confidencialidade e mais com o racionamento do tempo de computação. No MIT, por exemplo, cada usuário recebia 4 horas de tempo de computação por semestre. Quando Allan Scherr, um pesquisador de doutorado, quis mais tempo, solicitou que o arquivo UACCNT.SECRET fosse impresso. Quando seu pedido foi aceito, ele usou a lista de senhas para "pegar emprestadas" as contas de seus colegas. Em outra ocasião, uma falha de software exibiu a senha de todos os usuários, em vez da mensagem de login "Message of the Day". Os usuários foram obrigados a trocar suas senhas.

De Multics a UNIX

Embora limitado em funcionalidade, o CTSS demonstrou que o time-sharing não era apenas possível em termos tecnológicos, mas também muito popular. Os programadores gostavam do retorno imediato e da possibilidade de interagir com o computador em tempo real. Um grande grupo do MIT, do Bell Labs e da General Electric decidiu, portanto, desenvolver um sistema operacional multiusuário completo como substituto do processamento em lote. Eles o chamaram de Multics, sigla de Multiplexed Information and Computing Service.

A equipe do Multics projetou seu sistema de time-sharing com foco em segurança. O Multics foi pioneiro em muitos controles de segurança ainda usados hoje — um deles foi armazenar senhas de forma embaralhada, para que os usuários não pudessem repetir o truque simples de Allan Scherr. Após seis anos de desenvolvimento, o Multics foi lançado em 1969.[19]

Os militares viram potencial no Multics. Em vez de comprar computadores separados para lidar com informações não classificadas, classificadas, secretas e ultrassecretas, o Pentágono poderia comprar um único sistema e configurar o sistema operacional de modo que os usuários só pudessem acessar informações

para as quais tivessem autorização. Os militares estimaram que economizariam US$100 milhões ao migrar para o time-sharing.[20]

Antes de adquirir o Multics, a Força Aérea decidiu testá-lo. O resultado foi um desastre. Bastaram trinta minutos para descobrir como violar o Multics e mais duas horas para escrever um programa para executar a invasão. "Um usuário malicioso pode invadir o sistema sem problemas com um esforço relativamente mínimo", concluiu a avaliação.[21]

A comunidade de pesquisa também não gostou do Multics. Menos preocupados com sua segurança deficiente, os cientistas da computação estavam insatisfeitos com o projeto. O Multics era complexo e inchado — um resultado típico de decisões tomadas por um comitê. Em 1969, parte do grupo do Multics se separou e decidiu recomeçar do zero. A nova equipe, liderada por Dennis Ritchie e Ken Thompson, trabalhava em um sótão do Bell Labs usando um PDP-7 reserva, um "minicomputador" construído pela Digital Equipment Corporation (DEC) que custava dez vezes menos do que um mainframe da IBM.[22]

A equipe do Bell Labs havia aprendido a lição do fracasso do Multics: *keep it simple, stupid*.[23] Sua filosofia era construir um novo sistema multiusuário baseado no conceito de *modularidade*: cada programa deveria fazer bem uma única coisa e, em vez de adicionar funcionalidades a programas existentes, os desenvolvedores deveriam encadear programas simples para formar "scripts"[24] capazes de executar tarefas mais complexas. O nome UNIX começou como um trocadilho: como as primeiras versões do sistema operacional suportavam apenas um usuário — Ken Thompson —, Peter Neumann, um pesquisador de segurança do Stanford Research International, brincou dizendo que se tratava de um "Multics emasculado", ou "UNICS". A grafia acabou sendo alterada para UNIX.[25]

Em 1971, quando a primeira versão ficou pronta, o UNIX foi um sucesso estrondoso.[26] O sistema operacional versátil atraiu legiões de seguidores leais, com uma devoção quase cultuada, e rapidamente se tornou padrão em universidades e laboratórios. De fato, o UNIX desde então alcançou dominação global. Macs e iPhones, por exemplo, rodam um descendente direto do UNIX do Bell Labs.[27] Os servidores do Google, do Facebook, da Amazon e do Twitter rodam Linux, um sistema operacional que, como o nome sugere, é explicitamente modelado a partir do UNIX (embora, por razões de propriedade intelectual, tenha sido reescrito com código diferente). Roteadores domésticos, alto-falantes Alexa e até torradeiras inteligentes também rodam Linux. Por décadas, a Microsoft foi a única resistente. Mas, em 2018, a Microsoft lançou o Windows 10 com um kernel Linux completo. O UNIX tornou-se tão dominante que faz parte de todos os sistemas computacionais do planeta.

Como Dennis Ritchie reconheceu em 1979: "O primeiro ponto a considerar é que o UNIX não foi desenvolvido com a segurança em mente, em qualquer

sentido realista; esse fato, por si só, garante um grande número de brechas."[28] Algumas dessas vulnerabilidades eram erros de programação involuntários. Outras surgiram porque o UNIX concedia aos usuários privilégios maiores do que o estritamente necessário,[29] mas tornava suas vidas mais fáceis. Thompson e Ritchie, afinal, construíram o sistema operacional para permitir que pesquisadores compartilhassem recursos, não para impedir que fossem roubados.

O downcode do UNIX, portanto, foi moldado pelo upcode da comunidade de pesquisa — um upcode que incluía a competição por sistemas operacionais fáceis de usar, normas culturais distintivas da pesquisa científica e os valores que o próprio Thompson e Ritchie defendiam. Todos esses fatores se combinaram para criar um sistema operacional que valorizava a conveniência e a colaboração em detrimento da segurança — e o enorme número de brechas de segurança levou alguns a se perguntarem se o UNIX, que havia conquistado a comunidade de pesquisa, poderia um dia ser alvo de ataques.

Jogos de Guerra

Em 1983, a empresa de pesquisas Louis Harris & Associates informou que apenas 10% dos adultos tinham um computador pessoal em casa.[30] Desses, 14% disseram usar um modem para enviar e receber informações. Quando perguntados: "O fato de você poder enviar e receber mensagens de outras pessoas... no seu computador doméstico seria muito útil para você?", 45% desses primeiros usuários disseram que isso não seria muito útil.

Os americanos logo descobririam o poder impressionante das redes de computadores. O filme *Jogos de Guerra*,[31] lançado em 1983, conta a história de David Lightman, um adolescente suburbano interpretado por Matthew Broderick, que passa a maior parte do tempo em seu quarto, no computador, sem supervisão dos pais, como um Ferris Bueller nerd. Para impressionar uma garota, interpretada por Ally Sheedy, ele invade o computador da escola e altera a nota dela de B para A. Ele também aprende a encontrar computadores com os quais se conectar via modem ligando para números aleatórios — uma prática hoje conhecida como *war-dialing* (ou "discagem de guerra", em referência ao filme). David, por acidente, disca aleatoriamente até chegar ao sistema de computadores do Pentágono. Pensando ter encontrado um jogo de computador ainda não lançado, David pede ao programa, chamado Joshua, que simule um cenário de guerra. Quando Joshua responde: "Você não preferiria um bom jogo de xadrez?", David diz a Joshua: "Vamos jogar Guerra Termonuclear Global". No entanto, David não está em um jogo — Joshua é um computador da NORAD e controla o arsenal nuclear dos Estados

Unidos. Ao mandar Joshua armar mísseis e mobilizar submarinos, a invasão de David leva o mundo à beira de um confronto nuclear. O filme termina quando David interrompe o "jogo" antes que seja tarde demais. Joshua, o programa de computador, conclui sabiamente: "A única jogada vencedora é não jogar."

Jogos de Guerra arrecadou US$80 milhões nas bilheterias e foi indicado a três prêmios da Academia.[32] O filme apresentou aos americanos não apenas o ciberespaço, mas também a insegurança cibernética. A imprensa explorou esse tema mais sombrio ao se perguntar se uma pessoa com um computador, um telefone e um modem — talvez até um adolescente — poderia invadir computadores militares e iniciar a Terceira Guerra Mundial.

Todas as três grandes redes de televisão exibiram trechos do filme em seus telejornais noturnos. A ABC News abriu sua reportagem comparando Jogos de Guerra à comédia sobre a Guerra Fria de Stanley Kubrick, *Dr. Fantástico*. A reportagem sugeria que, longe de ser um brinquedo para adolescentes suburbanos entediados, a internet era uma arma do juízo final capaz de iniciar um Armagedom nuclear. Em um esforço para tranquilizar o público, o porta-voz da NORAD, o general Thomas Brandt, disse à ABC News que erros de computador como os retratados no filme eram impossíveis. Nesses sistemas, afirmou Brandt, "o ser humano está no circuito. O ser humano toma as decisões. Na NORAD, os computadores não tomam decisões".[33] Embora a NBC News tenha descrito o filme como dotado de "autenticidade assustadora", concluiu aconselhando "todos vocês, gênios da computação com seus computadores, modems e discadores automáticos" a desistirem. "Não há como vocês jogarem guerra termonuclear global com a NORAD, o que significa que o resto de nós pode relaxar e aproveitar o filme."[34]

Nem todos ficaram tranquilos. O presidente Ronald Reagan havia assistido ao filme em Camp David e ficou perturbado com a trama. No meio de uma reunião sobre mísseis nucleares e controle de armamentos, da qual participavam os chefes do Estado-Maior Conjunto, os secretários de Estado, da Defesa e do Tesouro, a equipe de segurança nacional e 16 congressistas influentes, Reagan interrompeu a apresentação e perguntou à sala se alguém vira o filme. Ninguém havia visto — o filme havia estreado na semana anterior. Reagan, então, fez um resumo detalhado da trama. Em seguida, voltou-se ao general John Vessey Jr., presidente do Estado-Maior Conjunto, e perguntou: "Algo assim poderia realmente acontecer?". Vessey disse que iria averiguar.[35]

Uma semana depois, Vessey voltou com sua resposta, presumivelmente após descobrir que os militares vinham estudando essa questão havia quase duas décadas: "Senhor presidente, o problema é muito pior do que o senhor imagina". Reagan orientou sua equipe a enfrentar o problema. Quinze meses depois, eles retornaram com a NSDD-145,[36] uma Diretiva de Decisão de Segurança Nacional. A diretiva concedia à NSA poderes para proteger a segurança das redes de computadores domésticas contra "nações estrangeiras...

grupos terroristas e elementos criminosos". Reagan assinou a ordem executiva confidencial em 17 de setembro de 1984.

Enquanto *Jogos de Guerra* levou a Casa Branca a tratar do que mais tarde seria chamado de "guerra cibernética", também levou o Congresso a tratar do "cibercrime".[37] Tanto a Câmara quanto o Senado iniciaram audiências em subcomissões sobre segurança computacional exibindo trechos do filme. O deputado Dan Glickman, democrata do Kansas, anunciou: "Vamos mostrar cerca de quatro minutos do filme *Jogos de Guerra*, que acho que delineia o problema de forma bastante clara".[38] Essas audiências acabaram resultando na primeira legislação abrangente do país sobre a internet e na primeira legislação federal — o *upcode jurídico*, em nossa terminologia — sobre crimes computacionais: a Counterfeit Access Device and Computer Fraud and Abuse Act (Lei sobre Dispositivos de Acesso Falsificados, Fraude e Abuso Computacional) de 1984, promulgada em outubro de 1984.[39]

Os políticos não foram os únicos a se preocupar. Em 1984, quando Ken Thompson recebeu o Turing Lifetime Achievement Award (o Prêmio Turing pelas suas contribuições ao longo de toda a carreira), a mais alta honraria da comunidade de ciência da computação, pelo desenvolvimento do UNIX, o tema de sua palestra foi a cibersegurança[40] — algo inédito na história da Conferência Turing. Na primeira metade da apresentação, Thompson descreveu um hack engenhoso usado pela primeira vez por testadores da Força Aérea quando eles invadiram o sistema Multics em 1974.[41] Eles mostraram como inserir um "backdoor" indetectável no Multics. Um backdoor é uma entrada oculta em um sistema computacional que contorna os mecanismos de segurança, o equivalente digital a uma estante que também funciona como porta para uma passagem secreta. Thompson mostrou como alguém poderia fazer o mesmo, de forma sorrateira, no UNIX.[42] (Thompson inseriu um backdoor em uma versão do UNIX usada no Bell Labs e, como ele previu, ela nunca foi detectada. O hack seria repetido pela inteligência russa no ataque SolarWinds de 2020, que comprometeu milhões de computadores americanos antes de ser descoberto.) A lição extraída por Thompson foi contundente: o "único programa em que se pode realmente confiar é aquele que você mesmo escreveu".[43] Mas, como não é possível escrever todo seu software, é necessário confiar em terceiros — o que é inerentemente arriscado.

Thompson então assumiu uma postura moralizante. Como cocriador do UNIX, que entendia muito bem como hackers podiam explorar sistemas multiusuário, ele denunciou os hackers como "nada mais do que motoristas bêbados". Thompson encerrou seu discurso alertando para "uma situação explosiva em formação". Filmes e jornais haviam começado a exaltar hackers adolescentes, transformando "vândalos em heróis ao chamá-los de gênios da informática". Thompson não estava se referindo apenas ao hype em torno de *Jogos de Guerra*. Ao mesmo tempo em que o filme chegava aos cinemas, um

grupo de seis jovens hackers, com idades entre 16 e 22 anos, conhecido como Clube 414, invadiu diversos sistemas computacionais de alto perfil, incluindo os do Los Alamos National Laboratories e do Security Pacific National Bank. O porta-voz do Clube 414, Neal Patrick, de 17 anos, aproveitou seus quinze minutos de fama, aparecendo no *Today Show*, no *The Phil Donahue Show* e na capa da *Newsweek* de 5 de setembro de 1983, [44] embora ele e seus amigos fossem apenas novatos.[45]

Thompson apontou para o que chamou de uma "lacuna cultural" na forma como a sociedade compreende o hacking. "O ato de invadir um sistema computacional precisa carregar o mesmo estigma social que invadir a casa de um vizinho. Não deveria importar que a porta do vizinho esteja destrancada."

Com estigma social ou não, o Congresso logo aprovou leis criminais mais rigorosas para o hacking. O Computer Fraud and Abuse Act (CFAA), de 1986, tornou crime federal o "acesso não autorizado" a qualquer computador do governo que causasse mais de US$1.000 em danos. Os condenados com base nessa lei podiam enfrentar até 20 anos de prisão e uma multa de US$250 mil.[46]

Infiltrar milhares de computadores do governo e derrubar a internet era exatamente o tipo de ataque hollywoodiano que a nova lei havia sido criada para punir. Quem quer que tivesse construído e liberado o worm autorreplicante na noite de 2 de novembro de 1988 estava em sérios apuros.

Bob Morris

Quando Robert Morris Jr. ligou para casa pedindo para falar com o pai no fim da noite de 3 de novembro, a mãe respondeu que ele estava dormindo. "É importante?"

"Bem, eu realmente gostaria de falar com ele", respondeu Robert.

Robert Morris Sr. era o cientista-chefe do National Computer Security Center, na NSA.[47] Com 56 anos, "Bob" era um criptógrafo matemático. Tinha uma barba longa e já grisalha,[48] cabelos indomáveis, olhos vivos e um sorriso travesso — o tipo de excêntrico que nunca parece totalmente adequado dentro de um terno. Ele havia se mudado para Washington havia pouco tempo, depois de passar 26 anos no Bell Labs. Nesse período, criara muitos dos utilitários centrais do UNIX. Na verdade, era amigo próximo de Ken Thompson, que alguns anos antes havia denunciado publicamente os hackers prodígios como motoristas bêbados.

Dada sua posição, Bob entendia que o filho corria um risco legal. Embora o FBI ainda não tivesse descoberto quem havia liberado o worm, era apenas uma questão de tempo até que descobrissem. Bob, portanto, aconselhou Robert a permanecer em silêncio e seguir com seus planos de encontrar a namorada na Filadélfia no dia seguinte.

Sem dúvida, Bob estava aterrorizado pelo filho, mas a trapalhada do filho também devia ser profundamente constrangedora. Bob ainda era novo em Washington. Ele chegara à sede da NSA, em Fort Meade, apenas 2 anos antes. O filho do chefe da segurança computacional hackeando os sistemas de computadores da nação era... constrangedor.

A ligação também veio em um momento inoportuno. Bob adorava estar na NSA. "Para um criptógrafo como ele, era como ir a Meca",[49] disse Marvin Schaefer, antecessor de Bob na NSA, que o recomendara para o cargo. Na verdade, Bob estava fazendo cursos para ser promovido a um cargo gerencial mais sigiloso. Como admitiu com ironia ao *New York Times*, ter menções noturnas do filho no noticiário "não era um ponto a favor na carreira".[50]

Bob mal conseguia ficar bravo com Robert. Mesmo antes de Robert nascer, Bob já havia instalado um terminal em casa para se conectar ao mainframe do Bell Labs. Logo, Robert passou a usar o computador sozinho para explorar o mainframe e, nesse processo, aprender UNIX. Tanto ele quanto o pai eram obcecados por cibersegurança, especialmente por encontrar brechas em softwares aparentemente seguros, e conversavam com frequência sobre o assunto. Como colocou o repórter do *New York Times* John Markoff: "O caso, com todas as suas reviravoltas bizarras, revela o mundo cerebral de um pai e um filho — e, na verdade, de toda uma subcultura moderna — obcecados pelo desafio intelectual de explorar os recantos mais internos das máquinas poderosas que, nas últimas três décadas, passaram a controlar grande parte da sociedade".[51] Quando a mãe de Robert foi questionada se pai e filho percebiam suas semelhanças, ela respondeu: "Claro que eles têm consciência disso. Como não teriam?"

Markoff descobriu que Robert Morris Jr. havia escrito o worm quando falou com Paul Graham. Ao longo de várias conversas, Paul se referira ao culpado como "Sr. X", mas depois escorregou e, por acidente, passou a chamá-lo de "rtm". Por meio do Finger, um serviço UNIX hoje extinto que funcionava como uma lista telefônica da internet, Markoff descobriu que "rtm" significava Robert Tappan Morris.[52]

Quando Markoff entrou em contato, Bob não viu sentido em fingir o contrário e, assim, confessou pelo filho. Bob declarou ao *New York Times* que o worm era "obra de um estudante de pós-graduação entediado". Mesmo com o medo e a raiva crescendo contra o estudante de pós-graduação entediado, o pai não conseguiu deixar de se gabar do filho — e de si mesmo. "Conheço algumas dezenas de pessoas no país que poderiam ter feito isso. Eu poderia ter feito, mas sou um programador muito bom." Bob também tinha um senso de humor espirituoso. Ao falar sobre a obsessão do filho por segurança computacional, ele disse: "Eu tinha a sensação de que esse interesse chegaria ao fim no dia em que ele descobrisse as garotas. Garotas são um desafio maior."[53]

O FBI abriu uma investigação criminal contra Robert Morris Jr. O caso recebeu "prioridade muito alta", mas a agência não sabia bem o que fazer. O porta-voz do FBI, Mickey Drake, admitiu: "Não temos histórico nessa área." Hackers já haviam sido processados antes, mas predominantemente no âmbito estadual.[54] Um ano antes, um funcionário insatisfeito em Fort Worth, no Texas, havia sido condenado por um crime de terceiro grau por apagar 168 mil registros de folha de pagamento depois de ser demitido pelo empregador, uma seguradora. Mas o Departamento de Justiça nunca havia levado ninguém a julgamento sob a nova lei Fraude Computacional de 1986 diante de um júri. O departamento não conseguia decidir se deveria processar o ato de Morris como uma contravenção, punível com multa e até 1 ano de prisão, ou como um crime, o que poderia resultar em 10 anos atrás das grades.

Bob Morris contratou o advogado criminalista Thomas Guidoboni em preparação para uma possível acusação. No primeiro encontro entre eles, Guidoboni ficou impressionado com a falta de noção de Robert — ele ainda não tinha consciência de que poderia ser acusado de um crime. Sua maior preocupação era ser expulso de Cornell. Guidoboni também observou: "Robert talvez seja o jovem de 22 anos mais imaturo que já conheci".[55] No caminho de volta para casa depois da reunião, Robert desmaiou no metrô.

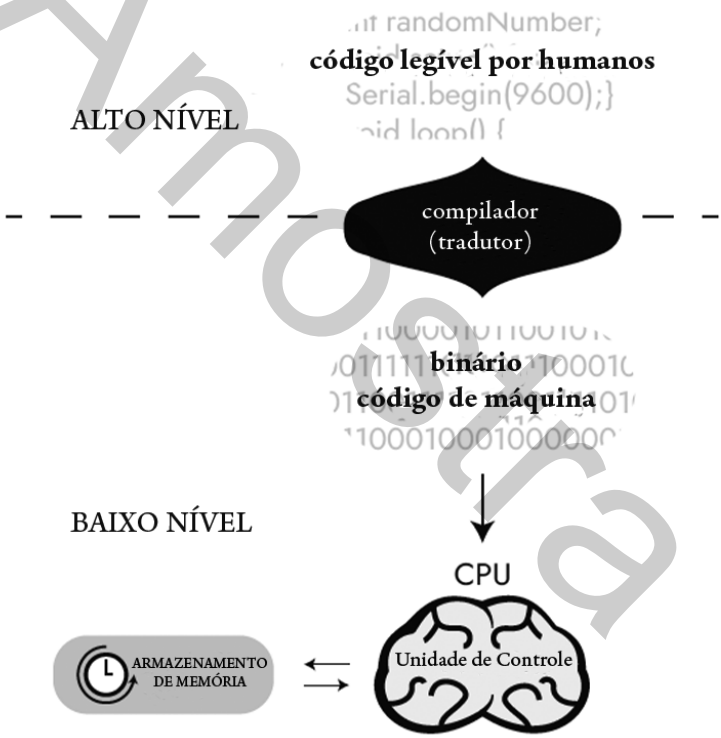
Dissecando o Worm

Há várias modalidades de downcode. As mais familiares são aquelas escritas em linguagens de programação de alto nível, com nomes como C (sucessora da linguagem B), C++ (sucessora do C, sendo ++ o modo, em C, de dizer "adicionar 1"), Python (batizada em homenagem ao grupo britânico de comédia Monty Python) e JavaScript (batizada a partir da popular linguagem de programação Java como um truque de marketing, embora tenham pouco a ver uma com a outra). Os humanos programam nessas linguagens porque elas são fáceis de usar. Suas instruções são escritas em inglês, empregam símbolos aritméticos básicos e têm uma gramática simples.

As unidades centrais de processamento (Central Processing Units ou CPUs) — os "cérebros" do computador — não entendem linguagens de alto nível. Em vez de usar palavras em inglês, as CPUs se comunicam em binário, a linguagem de zeros e uns. No código de máquina, como é conhecido, diferentes cadeias binárias representam instruções específicas (por exemplo, 01100000000000100000010000000010 significa "SOMAR 2+2").[56] Programas especiais, conhecidos como compiladores, pegam o código de alto nível escrito por humanos e o traduzem em código de máquina para que as CPUs possam executá-lo. O código de máquina normalmente é colocado em um arquivo binário para que possa ser executado rapidamente. (No capítulo

3, encontraremos um terceiro tipo de downcode, conhecido como linguagem Assembly, ou de montagem.)

Robert Morris Jr. escreveu seu worm em C, mas não liberou essa versão. Em vez disso, lançou a versão compilada, o código de máquina composto apenas de zeros e uns. Enquanto Robert entrava em estado de choque, os administradores de sistemas não tiveram escolha a não ser "descompilar" o worm. Eles tiveram, em outras palavras, de traduzir cuidadosamente a linguagem primitiva de zeros e uns enviada às unidades centrais de processamento dos computadores para o código simbólico de alto nível que Robert havia escrito originalmente. Eles estavam revertendo o processo de compilação que traduziu o código em C em uma cadeia binária de zeros e uns que os computadores conseguem entender, mas os humanos não.[57]



Com todo o infortúnio que se abateu sobre esses administradores, eles tiveram um golpe de sorte: naquela semana, acontecia em Berkeley uma conferência sobre UNIX. Os maiores especialistas do mundo estavam reunidos no campo de batalha enquanto o ataque se desenrolava. Trabalhando dia e noite, os gurus do UNIX decodificaram o worm na manhã de 4 de novembro. Eles descobriram que o worm não usava apenas um método para invadir

computadores. O worm era tão eficaz porque atacava usando quatro métodos diferentes, ou, na terminologia dos hackers, *vetores de ataque*.

O primeiro vetor de ataque era muito simples. O UNIX permite que os usuários selecionem "hosts confiáveis". Se você tivesse uma conta em uma rede, poderia dizer ao UNIX para "confiar" em determinadas máquinas dessa rede. Se você selecionasse a máquina nº 1 como confiável, poderia trabalhar, por exemplo, na máquina nº 2 e acessar a nº 1 sem precisar digitar sua senha novamente. Os hosts (ou computadores)[58] confiáveis eram úteis porque permitiam que os usuários trabalhassem em várias máquinas de forma simultânea sem ter de fazer login a cada vez.

Quando o worm chegava pela primeira vez a uma máquina, ele verificava se havia algum host confiável. Se encontrasse algum, o worm tentaria estabelecer uma conexão de rede. O worm, por assim dizer, ligava para o host confiável e via se ele atendia. Se o host atendesse, uma nova conexão de rede se formaria. O worm usaria essa conexão para enviar um pequeno programa — conhecido como código *bootstrap*. O código *bootstrap* então buscaria uma cópia do worm, a armazenaria na nova máquina e a colocaria em funcionamento. O worm-pai seguiria para o próximo host confiável, enquanto sua prole iniciava o mesmo processo em seu novo lar.

O segundo vetor de ataque tinha como alvo o SENDMAIL, um programa de email escrito por Eric Allman, então um estudante de graduação em ciência da computação na UC Berkeley, em 1975. Como Allman trabalhava para um administrador de sistemas que era mesquinho com o tempo de computador, Allman construiu um backdoor no SENDMAIL. Quando o SENDMAIL era instalado com a opção de depuração ativada, ele permitia que Allman enviasse um programa pelo correio eletrônico para o computador do administrador. Assim que a mensagem de email chegava à caixa de entrada do administrador, o SENDMAIL executava automaticamente o programa anexado em sua máquina, permitindo que Allman conseguisse um tempo extra de uso do computador. Quando concluiu seu projeto, Allman se esqueceu do backdoor que havia instalado no SENDMAIL.

Enquanto isso, o SENDMAIL tornou-se o programa de email padrão do UNIX. Quando era instalado com a opção de depuração ativada, o backdoor podia ser aberto — exatamente o que o worm de Morris fez. Quando o worm esgotou todos os hosts confiáveis, ele enviou por email uma cópia do código *bootstrap* para outros nós da rede usando o SENDMAIL.[59]

O terceiro ataque explorava as vulnerabilidades das senhas. Quando você escolhe uma nova senha para seu notebook ou um código de acesso para seu telefone, o sistema operacional não a armazena. Em vez disso, o sistema operacional passa sua palavra ou frase secreta por um programa que a embaralha usando matemática complexa. Como o sistema operacional armazena apenas a versão embaralhada, ele não sabe qual é a sua senha real. Isso é uma má