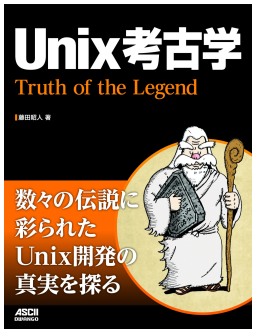
ニコニコ超会議 2017 ／ 超技術書典 特別企画

『Unix 考古学』2017 年春スペシャル



Ken Thompson のチューリング賞授賞記念講演 “Reflections on Trusting Trust”

A. Fujita

拙著『Unix 考古学』の出版 1 周年を記念して、小さな書き下ろしを試みています。そもそもこの本は月刊連載向けの書き下ろしとして始めたこともあって執筆は常に時間との勝負でした。その過程では書いておきたいけど資料を読み込むのに時間を要するために泣く泣くボツにしたネタが多数存在します。

チューリング賞受賞記念講演

“Reflections on Trusting Trust”[*1] はそのうちでも最も尾を引いてるネタの 1 つです。これは、皆さんもよくご存知の Ken Thompson が密かに実装していた（初期の）Unix のバックドアの話で、1983 年の ACM チューリング賞の受賞記念講演においてコンピュータセキュリティに関する重要な注意喚起「ソースコードに不審な点が見つからなかったとしても、それだけでは安全ではないかもしれない」を示す具体的な事例として Thompson 自身が暴露しました。

『Unix 考古学』の読者の方はご存知でしょうが、そもそも Ken Thompson はプログラミングの天才ではありましたが、執筆や講演はそれほどでもないようで、彼自身がクレジットされている文献は実はあまり多くありません。Unix の開発においてそういった物書きの仕事は多筆で雄弁な Dennis Ritchie の仕事だったようです。しかし、そんな彼もチューリング賞の受賞記念講演ともなると誰かに押し付けるわけにもいかなかったのでしょう。彼の講演はハッカーらしい型破りな内容でしたが、それも晴れがましい席での講演に不慣れだった故の賜物であり、その講演録は彼のクレジットの入った貴重な文献となっています[*2] 。

“Reflections on Trusting Trust” 再読

正直に告白すると、Thompson のこの講演録は記事を書いていたときに手に入れていたのです。が、けっきょく本文には書かず、脚注に簡単に紹介するだけに止めました。もちろん、締め切り間際だったことがいちばんの理由だったのですけど、講演録の内容をどのようにまとめて紹介すればよいか? で悩んでしまったことも大きかったと記憶してます。というのも、講演では Thompson Hack そのものには触れず、そこで用いられたであろう 3 つのプログラミングテクニックについてのみ語られていたから

[*1] “Reflections on Trusting Trust”：
`https://www.ece.cmu.edu/~ganger/712.fall02/papers/p761-thompson.pdf`

[*2] ちなみに、Dennis Ritchie も “Reflections on Software Research” というタイトルで講演をしました。が、他の多くの文献でも語られている「Unix の開発」に関わるトピックが中心で、おそらくチューリング賞の受賞式典の空気には非常にマッチしたそつのない講演だったのではないかと思います。
“Reflections on Software Research”：
`http://www.valleytalk.org/wp-content/uploads/2011/10/p758-ritchie.pdf`

……。Thompson は講演をヒントに安易にトロイの木馬が作られることを避けたかったのでしょう。

1 つ目のテクニックは、今日では Quine と呼ばれている「それ自身のソースコードと完全に同じ文字列を出力するプログラム」の紹介です。Quine は Wikipedia（英語版、日本語版）にソース付きで紹介されていますので、そちらを見ていただいた方がわかりやすいと思います[*3] 。

2 つ目のテクニックは、C コンパイラのソースをコンパイルする際のトリックに関してです。拡張を施した新しいコンパイラのソースを古いコンパイラでコンパイルするためには「学習」が必要であることを説明しています。

3 つ目のテクニックは、コンパイラにトロイの木馬（Trojan horse）を仕込む方法についてです。ここで Thompson は `login` コマンドをコンパイルするときにのみ既知のパスワードを受け入れるコードを生成させる仕掛けを打ち明けています。

ここで Thompson が語った 3 つのテクニックだけでは彼が実装したとされるトロイの木馬を具体的にイメージするのは（天才的なハッキングセンスの持ち主以外は）難しいのですが、彼が指摘したかったのは C コンパイラに細工をするとソースコードを検査してもわからない形でトロイの木馬が実装できるという事実だったようです。

このテクニックを思いついたきっかけについて、Thompson は「Multics の初期実装のセキュリティに対する米空軍の論評で読んだ」と語っていますが、講演の後、このテクニックはコンピュータセキュリティの研究課題として深く掘り下げられることになりました。

そのあらましは、Wikipedia の「Backdoor (computing)」のページの Compiler backdoors の項[*4] によくまとまった説明があります。そこでの記述によれば、Thompson が読んだとされる米空軍の批評論文は Karger, Paul A.; Schell, Roger R. (June 1974).“Multics Security Evaluation: Vulnerability Analysis”[*5] だそうです。今日では、このテクニックは Diverse Double–Compiling (DDC) という名前で知られているそうですが、David Wheeler の論文 “Countering Trusting Trust through Diverse Double–Compiling”[*6] で、その方法が具体的に解説されています[*7] 。

ともあれ、コンピュータサイエンティストの間では最も注目されるチューリング賞の受賞記念講演の影響力は絶大で、一気に注目を集めることになったコンパイラを使ったバックドアの実装方法は今となっては研究され尽くした感があります。もし Wheeler の論文が 2002 年〜2003 年に発表されていれば、『Unix 考古学』で取り上げていたでしょう。

考古学的な考察

さて Thompson Hack のあらましを語り終えたところで、トロイの木馬からは離れて『Unix 考古学』的な話題に移りましょう。

Thompson の講演録を見ながら僕が感じた素朴な疑問の 1 つ目は、Thompson はなぜ記念講演の

[*3] Quine：`https://en.wikipedia.org/wiki/Quine_(computing)` [Wikipedia：英語版]
`https://ja.wikipedia.org/wiki/クワイン_(プログラミング)` [Wikipedia：日本語版]

[*4] Compiler backdoors：
`https://en.wikipedia.org/wiki/Backdoor_(computing)#Compiler_backdoors` [Wikipedia]

[*5] “Multics Security Evaluation: Vulnerability Analysis”：
`http://csrc.nist.gov/publications/history/karg74.pdf`

[*6] “Countering Trusting Trust through Diverse Double-Compiling”：
`https://www.acsac.org/2005/papers/47.pdf`

[*7] さらに詳細に解説した論文もあります。
“Fully Countering Trusting Trust through Diverse Double-Compiling”：
`https://www.dwheeler.com/trusting-trust/dissertation/html/wheeler-trusting-trust-ddc.html`

テーマとしてこの話題を選んだのか?　ということです。それは、彼がチューリング賞を受賞したのが1983年だったことと関係があると僕は考えてます。

1983年といえば、『Unix考古学』の第12章で紹介したようにARPANETがInternetに切り替わった年、言い換えればInternetが誕生した年です。さらに第14章で述べたように4.2BSDのリリースが開始された年でもありました。それまでのネットと無縁な社会が、今日のネット社会に突然切り替わったターニングポイントの年だったわけです。

僕と同世代の方々には記憶が残っているかもしれませんが、当時の世相の1つにハッカーを名乗るティーンエイジャーの横行がありました。1983年は映画『ウォー・ゲーム』[*8]が封切られた年でもありますが、この作品ではあどけない顔をした主人公が大陸間弾道弾（ICBM）を管理するNORADのシステムに興味本位で侵入するストーリが展開されます。それも、このシステムの開発者が密かに残したバックドアを探し当てて……。

映画自体はファミリー向けに仕立てられていたのであまりあおられた印象ではないのですが、当時のハッカーたちの実態はもっと深刻だったようです。Thompsonの講演のなかで登場する“the 414 gang, the Dalton gang, etc.”といった集団の悪行の数々はWikipediaの「The 414S」[*9]で紹介されていますが、彼らが驚異的な集中力で天才的なハッキングができることをThompsonはよく知っていたのでしょう。彼は突然切り替わったネット社会に合わせて頭を切り替えることに難渋している（彼と同世代の）大人たちにこれから起こるであろう新しいタイプの犯罪を具体的に示す意図があったのだと思います。

事実、この5年後の1988年には世界中のUnixマシンに侵入・繁殖したMorris wormの事件が発生します[*10]。この事件がきっかけで「`libc`の`gets`関数を使うな」と言われるようになったことを覚えている方はいらっしゃいますかね?　今日では（表立って語られることは少ないですが）日常的に発生する事件・事故はこの年以降に発生するようになりましたが、当時はあまり喜ばしくない「来たるべき世界」という認識だったのですね。

もう1つの疑問は「Thompson Hackは実際に存在し使われたのか?」です。これについては読者の方から、「しかし、最近読んでいる『Unix考古学』では普通にサポートに利用されていたらしく書かれてあったので、ちょっと気になった。」という指摘を受けています[*11]。

ここで指摘されているように、Thompsonはトロイの木馬を仕込んだCコンパイラが存在したことは認めたものの配布したことは認めていないこと、しかしJargon File（Hacker's Dictionaryの元ネタ）のBack Doorの項[*12]にはBBNで深夜に`kt`という名前でログインしたプロセスが動作している目撃情報があることから、存在し実際に使われた可能性は濃厚ですが、真偽を確かめるのは難しいようです（少なくともご本人から明確な言質を引き出すことは）。

ここからは僕の個人的な考察として理解してほしいのですが、僕は「実際に存在し使われていた」と考えています。なぜならThompsonはその当時そういう手段を必要としていたと推測されるからです。

まず、Thompson Hackが実装されたと推測される時期ですが、インスパイアされたKarger&Schellの報告書が発行されたのが1974年6月で、これは研究版Unix Version 5がリリースされた時期とおおむね重なります。翌1975年5月にはVersion 6がリリースがされますが、Version 5では大学など

[*8] 『ウォー・ゲーム』：`https://ja.wikipedia.org/wiki/ウォー・ゲーム_(映画)`
[*9] The 414S：`https://en.wikipedia.org/wiki/The_414s`[Wikipedia]
[*10] Morris worm：`https://en.wikipedia.org/wiki/Morris_worm`[Wikipedia]
[*11] てきとうなメモ：`http://boscono.hatenablog.com/entry/2016/06/05/165745`
[*12] Jargon FileのBack Door：`http://www.catb.org/jargon/html/B/back-door.html`

の教育組織へ、Version 6 ではさらに一般企業へと、当時は Unix の配布対象が拡大されていた時期で、それにより Thompson の日常はかなり大きな影響を受けたと推測しています。

というのも、『Unix 考古学』でも紹介したように当時の AT&T の配布は「ソース付き、サポートなし、無保証」のポリシーでしたが、Thompson や Ritche はこのポリシー（特にサポートなし）には密かに反感をもっていました。実際に当時の Unix は Thompson の支援なしにはインストールできないシステムだったことは Peter Salus の “A Quarter Century of Unix”[*13] の随所で述べられています。結果的に Unix を導入した組織（大学、企業）は付属する簡単なメモランダムによるインストールに行き詰まると、Thompson に電話をする羽目になったわけです。

一方、ベル研の立場では Thompson が自ら対応する限り「外部の組織に Unix のデバッグ作業を手伝ってもらっているという方便が成り立ちます。このような活動から 50 Bugs などが生まれるわけですが……。これが後の BSD でも継承される公式リリース前にベータ版を配布する慣習として定着しました。

Version 5 の配布当初は教育機関への配布のみだったので学会などで顔見知りの人たちからの相談に個別に対処すればよかったわけですが、Version 6 のリリースを経て配布対象がどんどん拡大していくなか、Unix をまったく触ったことのない人たちが、いきなりシステムのインストールをしなければならないケースもまた増えていく……。こういった状況が Back Door の必要性、つまり Unix のスキルがまったく期待できないサイトで、Thompson 自身がトラブルを解決できる手段が必要になったのではないかと僕は推測しています。

この状況はそれまでの AT&T の配布ポリシーに抵触する可能性のある法的にはグレーな活動でしたし、その後 AT&T が Unix のビジネス化を決定した以降は、ベル研のメンバーには禁止された活動だったので、その当時の水面下での活動に当事者の口が重くなるのも当然なのではないかと思います。

そもそも、1970 年代は Unix に限らず、コンピュータは限られた人たちだけが使える魔法の箱でしたし、当時の技術レベルや品質を考えると、所有者の経済的あるいは法的な損失に直結するような情報を扱える代物でもありませんでした。そしてまったくネットワーク化されてなかったことから、そこにバーチャルな社会が形成されるはずもなく、誰もコンピュータにそんなことができるとは信じていなかったわけですから、見ず知らずの人間がアクセスしてきても「変わった輩」だと判断される牧歌的な時代でした（もちろん、薄気味悪い事件ですけどね。笑）。

ですが……、すべては 1983 年に変わった。Thompson の “Reflections on Trusting Trust” はそのターニングポイントであることを物語る記憶としても重要なのではないかと僕は考えます。皆さん、いかがでしょうか?（笑） ∎

ニコニコ超会議 2017 ／ 超技術書典 特別企画
『Unix 考古学』2017 年春スペシャル
Ken Thompson のチューリング賞授賞記念講演
“Reflections on Trusting Trust”
2017 年 4 月 29 日 発行
著　者　藤田昭人
発　行　アスキードワンゴ編集部

[*13] “A Quarter Century of Unix” :
http://wiki.tuhs.org/lib/exe/fetch.php?media=publications:qcu.pdf