仮想環境における真乱数生成器の評価

2017SE047:松下朋央 2017SE064:長村和希
指導教員：宮澤元

1 はじめに

Internet of Things(IoT) 機器の普及に伴い，IoT 機器が扱う情報の重要性が高くなってきている．これに伴い IoT 機器を標的とする攻撃が増加しており，IoT 機器のセキュリティ問題に注目が集まっている．一般的に組み込み系の IoT 機器は小型かつ低コストなものが求められており，非力で機能の少ない OS と CPU を搭載していることが多い．そのため，セキュリティ能力が脆弱で一般的な PC 等に比べソフトウェアの耐タンパー性能が低いことから，攻撃の対象になりやすい．このように，小型化等の利便性が向上する一方で，IoT 機器のセキュリティ面での問題に対しての危機感が広がっている．

IoT 機器のセキュリティ問題を Unikernel の技術を用いることで解決しようという提案がある [1]．Unikernel とは，対象となるアプリケーションをそれに対して特化させた OS とあわせて一つのバイナリとして，ハイパーバイザ上で仮想マシンとして動作させるものである．使用リソースの最適化を図りつつ，マルウェアによる攻撃のきっかけとなりやすいシェルやシステムコールなどの機能を搭載していないことでセキュリティを向上できるとされている．これに対し，複数の Unikernel 実装で Address Space Layout Randomization(ASLR) といったセキュリティ機能が想定通りに動作していないという報告がある [2]．原因として，Unikernel が動作する仮想環境では，エントロピー低下が原因で ASLR が前提としている乱数生成に問題がある可能性が指摘されている．Unikernel は，ハイパーバイザ上にて動作させるマシンイメージとして実現されるので，使用環境は通常仮想環境になる．

真乱数の生成においては機械的な動作などの物理的な要因に基づき十分なエントロピーが得られることを前提としている．しかし，仮想環境でソフトウェア的に実現された仮想ハードウェアではランダムな要因が少なく，真乱数の生成に問題が生じる可能性がある．例えば，ASLR 用にランダムに生成された使い捨ての値 (ナンス値) として，再起動前の値が維持されてしまい，ASLR によるランダム配置のアドレスが特定される危険性がありうる. この問題は Unikernel に限らず，すべての仮想環境上での真乱数を使用した機能の問題になる．

本研究では，仮想環境における真乱数の生成について調査するとともに，その問題点について考察する.Linux の擬似デバイスファイル/dev/random を使用して真乱数を生成し，乱数生成の速度や生成された乱数の質について調査を行う．また，乱数生成時に実際にナンス値が再利用されている可能性についても考察し，再利用されていた場合の改善案や，再利用されていなかった場合の問題点について検討する．

2 研究の背景

本節では IoT セキュリティと乱数の検定について述べる．

2.1 IoT セキュリティ

IoT とは，従来インターネットに接続されていなかったモノ (家電，車など) 自体がインターネットに接続されることによって，クラウド上のサーバなどとの間で相互に情報交換を行う仕組みを指す．IoT 機器の主な機能として，インターネットを介したデータの収集がある．IoT システムの特性として IoT 機器の周辺状況のモニタリングがある．IoT 機器にカメラなどのセンサを搭載することによって，モノ周辺で現在何が起きていてどのような状態にあるのかを，音・熱・振動などさまざまな形式で取得できる．こうした情報をインターネットを通じで受け渡してデータとして集積し，必要に応じて分析すれば，ユーザにとって何が必要とされているのか調べるなど様々に活用できる．このように，IoT では集めたデータを扱うことを前提とした仕組みであり，データの重要性が IoT 機器の普及に伴い増してきている [3].

データの重要性に比例して，IoT 機器のデータを対象とする攻撃が増えており，現在 IoT のセキュリティ問題について危惧されている．しかし，セキュリティ問題が危惧されている現状とは別に，一般的に組み込み系の IoT 機器は，小型で低コストなものをユーザーが求めているので，一般的な PC より非力で機能が少ない OS や CPU が搭載されていることが多い．そのため，多くの CPU に標準的に搭載されているさまざまなセキュリティ能力を内部に組み込むことが難しく，セキュリティ能力が脆弱でソフトウェアの耐タンパー性が低い．それに加え，IoT 機器は大規模な企業が管理するシステムとは異なり，一つ一つのデバイスが専門の管理者に管理されることが少ない．ユーザーが一つの IoT 機器を使い続ける期間が長く，デバイスが最新の状態に維持されることが少ないため，攻撃の標的となりやすいという傾向がある．IoT 機器が攻撃されることによるリスクとして，IoT 機器を通してユーザーの個人情報が盗み見られ悪用されたり，マルウェアによる感染から，IoT 機器による事故が発生したりといった危険がある．現在，社会全体で推進されている自動車の自動運転システムでは，自動車に多数搭載されている組込み系の IoT 機器を利用することになる．これらの IoT 機器がセキュリティ

的な攻撃を受けた場合，非常に深刻な影響が起こりうる．最悪の場合，自動車のハンドル制御システムを乗っ取られて自動車の操作が効かなくなってしまい，多くの人の命に関わるリスクが発生することも考えられる．

2.2 NIST 乱数検定

乱数生成器のセキュリティ的な安全性を判断するための検定法として，米国商務証標準技術局によって公開されている NIST Special Publication 800-22（NIST 乱数検定) という統計的検定法が存在する．

NIST 乱数検定は，16 種類の検定,189 個の試験から構成された検定ツールであり，乱数生成器の出力である 2 元乱数系列を統計的手法で検定することによって，著しく偏った乱数を発生させている乱数生成器を特定することが可能となる．NIST 乱数検定では，理想となる乱数の統計的動きを理論的に想定したうえで，与えられた乱数列と比較することによって p-value 値 (以下 p 値と称する) が得られる．p 値は NIST 乱数検定に含まれる各検定毎に得ることができる．p 値 > 0.01 の時，良い乱数と判断される．NIST 乱数検定による検定の対象は，0 と 1 から構成されている乱数のみである．

3 仮想環境における真乱数生成の調査

本研究では，仮想環境における真乱数の生成について調査するとともに，その問題点について考察する．OS カーネルに実装されている真乱数生成器を用いて真乱数を生成し，乱数生成の速度や生成された乱数について調査を行う．また，乱数生成時に実際にナンス値が再利用されている可能性についても考察し，再利用されていた場合の問題点について検討する．

4 実験

複数の環境で真乱数の生成を行い，乱数の生成速度を確認する実験を行った．真乱数の生成には Linux の 擬似デバイスファイル/dev/random を使用した．生成された乱数を利用して NIST 乱数検定にかけることで，生成された乱数の品質についても検証する．今回の実験では正常に動作していないと考えられていた Lempel-Ziv 検定を除いた 15 の検定を使用した [4]．乱数の検定で得られた p 値は，小数点以下 5 桁目を四捨五入して小数点以下 4 桁までを結果として示す．

4.1 実験環境

実験は表 1 に示す物理環境及び仮想環境で行う．

表 1 実験環境

	物理環境	仮想環境
CPU	IntelCorei7-2600	IntelCorei5-7200U
メモリ量	8GiB	8GiB
OS	Ubuntu20.04.3LTS	20.04.3 LTS(Virtualbox)
仮想環境割り当てメモリ量		3GB

4.2 仮想環境における一般的な真乱数の質の確認実験

仮想環境において生成される真乱数の品質を調べる実験を行った．仮想環境を一度再起動した後，dd コマンドを利用して /dev/random から真乱数を生成する．マウス等の入力装置の動作によって得られるエントロピーで乱数の品質が向上するかを確認するために，入力装置を積極的に利用するか否かの以下の 2 つのパターンで乱数を生成した．

入力装置動作あり 乱数の生成中にできるだけマウスやキーボードを動かしている状態にする

入力装置動作なし 乱数の生成中はマウスやキーボードを必要最低限しか動かさない

実験結果を表 2 に示す．

表 2 仮想環境における一般的な真乱数の検定

入力装置動作	乱数生成時間 (日)	検定合格数	不合格時の最低 p 値
あり	2	14	0.0090
なし	10	14	0.0070

まず，データの収集速度に関しては，入力装置動作ありでは実稼働時間 2 日程度でデータを収集できたが，入力装置動作なしでは 10 日もかかってしまった．約 5.0 倍も収集速度が変わっているという結果である．このことから，意図的に動作を加えることにより目に見えて，乱数を生み出すことができることが分かった．しかし，乱数の質の面ではどちらも 15 個ある検定の内 14 個合格という形になっており，大きな違いはなく，意図的に動きを加えるだけでは質を良くすることはできないことが分かった．

4.3 仮想環境における再起動直後の真乱数の質の確認実験

再起動直後に生成される真乱数の品質を調べる実験を行った．仮想環境，物理環境のそれぞれで環境の再起動後に dd コマンドを利用して /dev/random から真乱数を生成する．dd コマンドを使用して真乱数を生成すると再起動で得られたエントロピープールが枯渇するので，再度再起動を行って真乱数の生成を継続する．再起動は，真乱数が検定に必要な量 (130KB) 生成されるまで繰り返した．仮想環境と物理環境のそれぞれで以下のように真乱数を生成して，得られた乱数の品質を調べた．

仮想環境 エントロピープールが枯渇したときに再起動を行う

物理環境 真乱数が 130 バイト分生成されたときに再起動を行う

物理環境では，真乱数が 130 バイト分生成される前に再起動時に得られたエントロピープールは枯渇するので，いずれの環境でもエントロピープールが枯渇している状態で再起動を行っている．なお，マウス，キーボード等の入力装置の動作は必要最低限のみ行った．実験結果を表 3 に示す．

表 3 再起動を繰り返しながら生成した真乱数の検定

環境	乱数生成時間 (日)	検定合格数	不合格時の最低 p 値
仮想環境	3.5	9	2.7399×10^{-85}
物理環境	2.5	14	0.007

仮想環境の結果では 15 個の検定中 9 個しか合格しておらず，表 1 に示した一般的な仮想環境における乱数に比べて品質の点で明らかに劣る乱数が生成されていた．それに対して物理環境の結果では，一般的な仮想環境における乱数と同じく 15 個の検定中 14 個が合格していた．これらの結果から，仮想マシンにおいてエントロピー枯渇後の再起動で得られたエントロピーのみに基づく真乱数の生成には異常があることがわかる．また，真乱数生成の速度については，仮想環境では 3.5 日，物理環境ではが 2.5 日程度を要した．これは単純に物理環境と仮想環境でそれぞれ使用しているマシンの再起動に要する時間に違いがあることが原因だと考えられる. 再起動 1 回あたりに要する時間は，物理環境では 3 分ほど，仮想環境では 5 分ほどであった．

1 回の再起動後の真乱数生成に要する時間に注目すると，物理環境では 130 バイトの真乱数生成に 10 分ほど要することが稀にあった．これは，再起動時に得られたエントロピープールを乱数の生成に使い切ってしまった後，エントロピープールが明らかにたまりづらいタイミングが存在していることを示している．再起動直後に得られたエントロピープールを用いると，仮想環境と物理環境のいずれにおいても 100 バイト程度の真乱数を生成できることが確認できている．入力装置の動作を最小限にしているので，本来エントロピープールが仮想環境より貯まりやすいはずである物理環境でも，残りの 30 バイト程度の乱数を生成するためのエントロピーの獲得に時間がかかる場合があることになる．今回，仮想環境では再起動直後に得られるエントロピーのみを利用して生成した乱数の品質に問題があるので，この期間に仮想環境で得られるエントロピーに問題があると考えられる．

4.4 擬似乱数との比較実験

仮想環境において再起動直後に得られたエントロピーのみを使って生成された真乱数の品質を擬似乱数の品質と比較する実験を行った．仮想環境において dd コマンドを用いて以下の 2 通りの方法でそれぞれ 10 サンプルの乱数を生成した．

/dev/urandom　/dev/urandom から擬似乱数を生成する

haveged　CPU の内部状態からエントロピーを取り出す haveged を利用して /dev/random から真乱数を生成する

実験結果を表 4 に示す．検定合格数については 10 サンプルの平均を示した．

表 4 擬似乱数の検定

乱数生成方法	平均合格数	不合格時の最低 p 値
/dev/urandom	14.6	0.0008
haveged	14.3	9.4969 × 10~-5

乱数検定の合格数はどちらも 14 個，15 個のケースが多く，今まで集めていた真乱数と大きな変化がなかった．しかし，以下のような理由で，良い乱数を安定して生成し続けられているかどうかには疑問があり，合格数が同じでも乱数の質に差がある可能性がある．

- サンプルによっては検定の合格数が 13 と 2 つの検定が不合格のものがある
- サンプルによっては NIST 乱数検定の中でも合格率が高いはずの検定に不合格のものがある．これまでの実験で比較的品質の良い真乱数を使用した結果では，ランダムウォークに関する 2 つの検定しか不合格にはなっていなかった．これらの検定は NIST 乱数検定の中でも他の検定に比べて合格率が低い．
- 不合格時の p 値がこれまでの実験で比較的品質の良い真乱数を使用した結果と比べて悪い値である

4.5 仮想環境再起動時の乱数の質の悪さ

これまでの実験から，仮想環境においてエントロピープールが枯渇した状態で再起動を行い，再起動直後に得られたエントロピーのみを用いて真乱数を生成すると，擬似乱数と比べても明らかに検定の結果が良くない乱数が生成されることが分かった．本節では，この結果の各テストの p 値や各テストがどのような基準で乱数検定を行っているかに注目して，原因を調査する．そこで，この実験における不合格だったテストとその時の p 値を表 5 にまとめた．

表 5 仮想環境再起動時の不合格検定

	p 値
ブロック単位の頻度検定	2.7399×10^{-85}
連の検定	0.0017
Maurer のユニバーサル統計検定	2.9768×10^{-11}
系列検定	2.2487×10^{-15}
近似エントロピー検定	4.4280×10^{-15}
累積和検定	0.0013

表 5 の通り，不合格の検定の p 値はほとんど 10^{-10} より少ない値になっており，他のパターンの不合格と比べ圧倒的に小さいこと，擬似乱数などを対象とした実験結果と比較しても，たまたま不合格になったとは考えにくい．仮想環境において再起動直後のエントロピーのみを用いて生成された乱数は，検定の不合格数が多く， p 値も低いことから，品質が低い乱数だと言える．

NIST 乱数検定は参考文献をもとに分類分けをおこなうと，ブロック単位の頻度に関する検定が 7 個，一様性・圧縮可能性に関する検定が 3 個，周期性に関する検定が 2 個，

状態推移・ランダムウォークに関する検定が 3 個の計 15 個で構成されている．その中で一様性・圧縮可能性に関する検定にパターン 3 は全て不合格となっており，原因究明のために注目するべき内容だと考えられる．また，ブロック単位の頻度検定の p 値は他の不合格検定に比べ，圧倒的に低いことについても注目すべきである [5].

4.6 考察

仮想環境で再起動直後のエントロピーのみを用いて生成された乱数の品質の低さの原因について考察する．

まず原因として考えられるのはナンス値の再利用である．この乱数が不合格だった 3 つの一様性・圧縮可能性に関する検定は，指定したブロックの長さのケース数が一様に出現しているかを調べる検定である．これらに不合格ということは，ナンス値の再利用により同じケースが頻発していることが推定できる．まず，ナンス値の再利用によって起こりうる可能性が非常に高い理由は 3 つの一様性・圧縮可能性に関する検定は指定したブロックの長さのケース数が一様に出現しているかを調べる検定だからである．次に考えられるのは，仮想環境の再起動時に得られるエントロピー自体の品質の低さである．ブロック単位の頻度検定は指定した各ブロック毎の数字の偏りを検定するものであるが，この乱数に対する検定結果の p 値は非常に低く，数字の偏りが著しく大きいと考えられる．この原因としてナンス値の再利用以外に考えられるのは，そもそものナンス値に偏りがあることであり，これがエントロピー自体の品質の低さを示している．

5 議論

今回の実験により，実際にエントロピーを枯渇させた状態の再起動後した時の真乱数が質が悪いこと，またその原因がエントロピー低下だけではなく，物理環境との再起動直後の乱数取得方法の差がある可能性を知ることができた．このことから，実際に仮想環境でより良い乱数を生成し，セキュリティ機能を活かすにはどうするべきかを議論する．

5.1 仮想環境における真乱数生成に関する議論

現状では IoT 機器に対して仮想環境における真乱数を利用したセキュリティ機能を作成するのは難しいと考えられる．実験結果から，仮想環境における再起動直後の真乱数の質が低くナンス値が再利用されている可能性が高いことが確認できたからである．

仮想環境における一般的な真乱数の質が問題ないことは確認できたが，マルウェアによる攻撃では，たとえ攻撃による乗っ取りが失敗したとしても，クラッシュして再起動するケースがある．その場合，仮想マシンでは，仮想環境再起動の実験のように質の悪い乱数を生成し，セキュリティ機能に使用してしまい，次の攻撃に対して脆弱な状態になってしまう．また，IoT 機器はそれに加えて，攻撃を検知する能力が低いものも多いため，迅速に対処することも難しい．

5.2 再起動時の乱数に関する議論

この仮想環境における質の悪い真乱数の問題を解決するには今後物理環境の真乱数生成についても注目する必要があると考えられる．再起動時の真乱数収集において，仮想環境では乱数の質に問題が出たが，物理環境でも乱数の収集速度に問題が出ており，再起動時における乱数生成問題は物理環境でも状態に差異はあれど，起こっているからである．より詳しく，仮想環境と物理環境の両方の真乱数について調べることで，再起動時の乱数の問題の原因を突き止めることができると考えられる．

6 おわりに

実験により，仮想環境における再起動直後のエントロピーのみを用いて生成された乱数の品質が悪いことを確認した．これにより，このような状況の乱数生成ではナンス値の再利用が起こる可能性があることがわかった．また，実験の物理環境の乱数の収集速度にも問題があったことを踏まえ，今後は仮想環境だけではなく，物理環境にも注目することで，真乱数生成の問題の解決を目指す．

参考文献

[1] Bob Duncan, et al., “Enterprise IoT Security and Scalability: How Unikernels can Improve the Status Quo, ‘ ’ in *Proceedings of the 9th International Conference on Utility and Cloud Computing (UCC ’ 16)*, pp.292-–297, 2016. DOI:https://doi.org/10.1145/2996890.3007875

[2] Tian Wang, Guangxue Zhang, Anfeng Liu, Md Zakirul Alam Bhuiyan, and Qun Jin: “ A Secure IoT Service Architecture With an Efficient Balance Dynamics Based on Cloud and Edge Computing,” IEEE Internet of Things Journal. Vol. 6. No. 3. pp.4831–4843, https://doi.org/10.1109/JIOT.2018.2870288 , June.2019.

[3] IPA 独立行政法人情報処理推進機構．:“IoT のセキュリティ．,‘ ’ https://www.ipa.go.jp/security/iot/, March.2021.

[4] DavidJohnston．, ” sp800_22_tests ” https://github.com/dj-on-github/sp800_22_tests, June.2021.

[5] 情報処理振興事業協会セキュリティセンター．: “電子政府情報セキュリティ技術開発事業 擬似乱数検証ツールの調査開発 調査報告書．,” https://www.ipa.go.jp/files/000013665.pdf , February.2003.