SELinuxによる組込みシステムのソフトウェア保護手法と検証

田中 大樹[1,a)]　内匠 真也[2]　金井 遵[3]　鄭 俊俊[1]　毛利 公一[1]

概要：さまざまなセンサーやデバイスをクラウドに接続するIoT化が進んでいる．IoTシステムには，脆弱性が対策されないなどセキュリティ面に課題がある．そのため，IoTシステムには侵入され管理者権限を奪われた場合でも重要なソフトウェアを保護する必要がある．そこで，我々は，ソフトウェアの改変や削除，設定変更などの侵害はシステムコールによってなされることに着目し，該当するシステムコールを制御するために，重要ソフトウェアをSELinuxによって保護する手法を検討した．具体的には，重要ソフトウェアの資産を侵害するシステムコールおよび，それらをSELinuxによって制御するために必要なアクセスベクタについての調査を行なった．本論文では，重要ソフトウェアの資産それぞれに対して必要な制御について検討し，アクセスベクタを調査した結果から制御すべきアクセスベクタを決定した．さらに，上記に従ったポリシーを生成，適用し，重要ソフトウェアが保護できることを検証した．検証の結果，重要ソフトウェアが保護できることを確認した．

キーワード：組込み機器，セキュリティ，SELinux，オペレーティングシステム，システムコール

Software protection method and its verification for embedded systems by SELinux

Daiki Tanaka[1,a)]　Shinya Takumi[2]　Jun Kanai[3]　Junjun Zheng[1]　Koichi Mouri[1]

Abstract: Various sensors and devices, connecting Internet of Things (IoT) to the cloud, construct IoT systems. The fast growth of IoT systems led to many security issues, such as vulnerability not being fixed. It is necessary to protect important software even if the system is intruded and deprived of administrator authority. As a solution, Tanaka et al. (2019) proposed a method to protect important software based on SELinux. More specifically, these system calls that infringe important software assets and their corresponding required access vectors with SELinux were investigated. In this paper, we discuss the control required for each important software asset and determine the access vector needed to be controlled according to the investigation results by Tanaka et al. (2019) on the access vector. The security policy is further generated and applied the to verify whether the important software can be protected. The result shows that SELinux can protect important software well.

Keywords: Embedded device，Security，SELinux，Operating system，System call

1. はじめに

近年，さまざまなセンサーやデバイスをクラウドに接続するIoT(Internet of Things)化が進んでいる．IoT機器の普及に伴い，IoTマルウェアMiraiやIDDoS攻撃のようなIoT機器を狙ったサイバー攻撃が増加するなど，IoT機器

[1] 立命館大学
Ritsumeikan University
[2] 東芝インフラシステムズ株式会社
Toshiba Infrastructure Systems & Solutions Corporation
[3] 株式会社 東芝
Toshiba
[a)] dtanaka@asl.cs.ritsumei.ac.jp

の課題の一つとしてセキュリティがある．IoT システムでは，IoT 機器の高性能化に伴い，Intel®[1]，ARM®[2] などのさまざまな CPU と，Windows®[3]，Linux®[4] といった汎用 OS，TCP/IP に代表される汎用プロトコルが採用されることが増えている．しかし，IoT システムでは，計算リソースが限られること，更新が容易でない環境で利用されることなど，ソフトウェアや OS の脆弱性が導入時から対処されないままで放置されることが多く，この脆弱性を悪用したエクスプロイトによってシステムが容易に侵害されてしまうことがある．

このように，エクスプロイトによって管理者権限を奪取された場合でも，システム内のソフトウェアへの侵害によって仕様外の動作を行わないように，攻撃者が持つ管理者権限から重要なソフトウェアを保護する必要がある．これを実現するために，ソフトウェアの実行ファイルなどの重要資産への侵害には，システムコールを経由する必要があることに着目する．そして，どのようなシステムコールが発行されることで侵害が行われるのか調査し，保護する資産や使用する保護機能など，ソフトウェア保護の検討を行う．本論文では，重要なソフトウェアの例として，ホワイトリスト型の実行制御機構 WhiteEgret™ [1] の保護を検討する．

WhiteEgret の各保護資産は適切にパーミッション制御されているため，エクスプロイトにより一般ユーザの権限が奪取された場合には WhitEgret の動作は阻害されない．また，多くのエクスプロイトは対象機器に侵入後，マルウェアをダウンロードして実行する (ドライブバイダウンロード型攻撃)．root 権限で動作するプログラムやシェルであってもホワイトリストに記載されない実行ファイルは起動できないため，エクスプロイトによりマルウェアをダウンロードして root 権限で実行するようなことはできない．さらに，保護資産への不正アクセスに利用されうる Linux コマンドはホワイトリストに記載しない，root 権限で動作するプログラムを最小限にする，ケーパビリティ制御を併用する，パッチを迅速に適用するといった対策で root 権限の奪取を防ぐとともに，root 権限奪取時の影響を最小化している．しかし，エンドポイントの IoT デバイスではこれらの対策がすべて採れるとは限らない．よって，WhiteEgret の応用範囲を IoT デバイスまで広げようとした場合には，エクスプロイトによる root 権限奪取時にも保護資産へのアクセスを防ぐ仕組みを設けることが望ましい．

先行研究 [2] では，WhiteEgret の保護を検討するため，全てのシステムコールについて調査することで，WhiteEgret を侵害するうえで実行される可能性があるシステムコールを明らかにした．また，これらのシステムコールの制御を実現するため，SELinux(Security-Enhanced Linux)[3] での制御について，侵害可能性のあるシステムコール実行に必要なアクセスベクタに関する調査を行なった．

本論文では，WhiteEgret の保護すべき資産と資産に必要な権限と与えてはいけない権限について述べる．そして，先行研究で明らかにしたアクセスベクタを元に，資産に設定すべき権限に沿って SELinux のポリシーを生成し，適用することで実際に制御できているか検証した結果について述べる．

2. ソフトウェア資産に設定すべき権限

先行研究では，重要ソフトウェアの例として本論文で保護する WhiteEgret の資産について述べた．先行研究では，資産が保存される場所として資産を二次記憶，メモリで分類した．しかし，SELinux で制御するうえで資産の保存される場所は重要ではなく，資産がどのような形態として OS と SELinux から認識されるかが重要である．そのため本論文では，ファイルシステムによって管理されファイル，ディレクトリとして存在する資産と，プロセスとプロセスのメモリ空間に存在する資産に大きく分類する．

資産それぞれに対して与える必要のある権限と，与えてはいけない権限を表 1，表 2 に示す．表 1 はファイルとディレクトリの資産，表 2 はプロセスの資産に関する表である．表の「○」は，重要ソフトウェアを正常に動作させるために資産に対して与える必要のある権限で，「×」は，資産を侵害する可能性があり，資産に対して与えてはいけない権限である．空欄は，その権限の有無がソフトウェアの動作に影響しない権限であり，「－」は，実行ファイル以外のファイルへの実行権限など，与えても意味のない権限である．資産それぞれに与えるべき権限と与えるべきでない権限について以下で述べる．

プログラム

プログラムは，WhiteEgret のデーモンプログラムの実行ファイルである．このファイルは，システムの構築時に用意するものであり WhiteEgret 運用後は攻撃者だけでなく管理者からも削除，書換え，権限変更は許可しない．また，攻撃者による実行は，多重実行による想定外の動作を防ぐために許可しない．

ホワイトリスト，ログ・設定ファイル

これらのファイルは，設定の変更やログの書込みなど書換え，読込みの権限が管理者には必要となる．また，ログファイルと設定ファイルの有無は WhiteEgret の動作に関係しないため，削除，権限変更は管理者，攻撃者とも禁止しない．

マウントポイント

マウントポイントは，WhiteEgret の資産が保存するファ

---

1) Intel は，アメリカ合衆国およびその他の国における Intel Corporation またはその子会社の商標または登録商標です．
2) Arm は Arm Limited（またはその子会社）の US またはその他の国における登録商標または商標です．
3) Windows は，マイクロソフト企業グループの商標です．
4) Linux は，Linus Torvalds 氏の商標です．

表 1　ファイルとディレクトリの資産

	管理者						攻撃者					
資産	削除	書換	読込	実行	パス変更	権限変更	削除	書換	読込	実行	パス変更	権限変更
プログラム	×	×		○		×	×	×		×	×	×
ホワイトリスト	×	○	○	−		×	×	×		−	×	×
ログ・設定ファイル		○	○	−						−		
マウントポイント	×	×		−	○	×	×	×		−	×	×
デバイスファイル		×		−				×		−		
ドライバファイル	×	○	○	−		×	×	×		−	×	×

表 2　プロセスの資産

	管理者						攻撃者					
資産	削除	書換	読込	実行	停止	リソース制限	削除	書換	読込	実行	停止	リソース制限
プロセス	○	×			○	×	×	×		×	×	×
プログラム	×	×			−	−	×	×		×	−	−
ホワイトリスト	×	×		−	−	−	×	×		−	−	−
ライブラリ	×	×			−	−	×	×		×	−	−

イルシステムがマウントされるマウントポイントである．起動時，ファイルシステムをマウントする必要があるため，管理者はマウントポイントへのパス変更は許可する必要がある．

デバイスファイル

デバイスファイルは，二次記憶のデバイスファイルであり，書換えられた場合，二次記憶のデバイスの中身が書き換えられる．そのため，デバイスファイルへの書込みは許可しない．一方で，デバイスファイルはユーザが操作するためのインタフェースであるため，デバイスファイルを操作する必要はない．

ドライバファイル

ドライバファイルは，ユーザ空間とカーネル空間の WhiteEgret のプログラムの間でデータのやり取りを行う擬似ファイルである．そのため，管理者によるファイルへの書込み権限と読込み権限が必要である．一方で，削除された場合正常な動作をしなくなるため削除は許可しない．

プロセス

WhiteEgret の資産には，WhiteEgret を実行するためにメモリにロードされたプログラムやホワイトリストなどがある．これらはプロセスがプロセスのメモリ空間にロードされる．また，SELinux ではこれらの資産に対する制御はプロセスへの制御としてまとめて行われる．管理者は，設定の変更などのためプロセスの削除，停止の権限は必要となる．

システム全体

重要ソフトウェアは，ソフトウェア以外のシステムへの侵害によって間接的に侵害される可能性がある．一つ目は，システム時間の変更である．時間の変更により，ログファイルに記述されるタイムスタンプが改竄される可能性がある．二つ目は，カーネルの変更である．Linux には，カーネルに機能を追加するためのものとして LKM(Loadable Kernel Module) がある．悪性コードが LKM によってカーネルにロードされた場合，悪性コードがカーネルと同じ権限で動作する．また kexec は，次回コンピュータを起動したときにロードするカーネルを変更する Linux のメカニズムである．kexec によって悪性コードが含まれたカーネルがロードされた場合侵害される．これらに関連するシステムコールは，システム起動時や環境構築時に実行される可能性があり，管理者による実行は許可する．

3. SELinux による保護

WhiteEgret を保護するためには，攻撃者などによる WhiteEgret を侵害する可能性のあるシステムコールを制御する必要がある．Linux には，WhiteEgret も利用しているシステムコールの制御によって強制アクセス制御を行う LSM が存在し，LSM を利用することで厳格で複雑な制御を可能とした SELinux による制御を検討する．

3.1 SELinux の概要

SELinux では，プロセスにドメインというラベルを付与し，ファイルにタイプというラベルを付与することができる．さらに，ドメインがタイプに対してどのような操作を実行許可するのかを決定するアクセスベクタを設定できる．また，ファイルやソケットなどのリソースを約 30 種のオブジェクトクラスとして定義し，オブジェクトクラスごとに細かくアクセスベクタを設定することで，詳細なアクセス制御を可能としている．

また，SELinux はドメイン遷移という概念を有する．通常 SELinux のドメインは，親プロセスのドメインが子プロセスに継承されるが，この規則が不都合な場合もある．そのため，あらかじめ起動元ドメインと実行ファイルのタイプの組み合わせによって起動先のドメインを決定することができ，必要最低限のアクセス許可のみ与えることがで

表 3　管理者から WhiteEgret の資産に対してのアクセスベクタに設定すべき制御

	資産	アクセスベクタ					
禁止	プログラム	rename	unlink	write	setattr	create	link
	ホワイトリスト	rename	unlink	write			
	ログ，設定ファイル	なし					
	ファイルシステム	mounton	unmount	sys_admin			
	デバイスファイル	link	setattr	write	create		
	ドライバファイル	rename	unlink	write			
	プロセス	ptrace					
許可	プログラム	execute	execute_no_trans	read	open	search	getattr
	ホワイトリスト	read	open	write			
	ログ，設定ファイル	read	open	write			
	ファイルシステム	なし					
	デバイスファイル	なし					
	ドライバファイル	read	open	write			
	プロセス	sigkill	sigstop	signal	setcap		

表 4　攻撃者から WhiteEgret の資産に対してのアクセスベクタに設定すべき制御

	資産	アクセスベクタ					
禁止	プログラム	rename	unlink	write	setattr	create	link
		open	execute	execute_no_trans			
	ホワイトリスト	rename	unlink	write	open	setattr	create
		link					
	ログ，設定ファイル	なし					
	ファイルシステム	mounton	unmount	sys_admin			
	デバイスファイル	link	setattr	write	create		
	ドライバファイル	rename	unlink	write	open	setattr	create
		link					
	プロセス	ptrace	sigkill	sigstop	signal	setrlimit	setcap

きる．

3.2　制御すべきアクセスベクタ

WhiteEgret の資産を侵害する可能性のあるシステムコールを SELinux によって制御するためには，侵害するプロセスから保護対象へのアクセスベクタを設定する必要がある．先行研究で調査したシステムコールとアクセスベクタを元に，制御すべきアクセスベクタを決定する．管理者から WhiteEgret の資産へのアクセスを制御するために設定すべきアクセスベクタを表 3 に示す．また，攻撃者から WhiteEgret の資産へのアクセスを制御するために設定すべきアクセスベクタを表 4 に示す．攻撃者に対して許可する権限は存在しない．そのため，表 4 は禁止する権限のみである．

4.　検証

3 章で述べた設定すべき制御を WhiteEgret に模した環境に適用し，実際に資産を侵害することで SELinux によって資産が保護されることを検証する．WhiteEgret はオープンソースだが，ユーザ空間で動作し，実行の可否を判断するデーモンプログラムはオープンソースではないため，検証ではそれに模したプログラムを利用する．

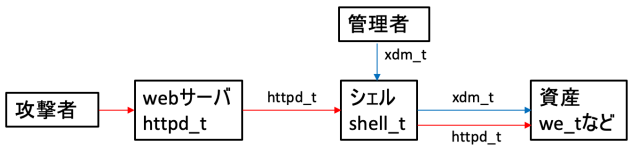

図 1　検証環境

4.1　想定状況

検証では，攻撃者が web サーバを経由し，管理者権限で任意のコマンドが実行できるものとする．また，管理者は USB のキーボードなどを物理的に接続してアクセスするものとする．この二つのアクセス経路を SELinux のドメインで区別することで制御の違いを実現する．本論文の検証において上記の状況として作成する環境を図 1 に示す．攻撃者のアクセス経路の web サーバのデーモン httpd プロセスは，httpd_t ドメインを付与する．一方で管理者のドメインは，xdm_t を付与する．xdm_t は，SELinux にデフォルトでシステム管理者用に設定されているユーザの sysadm_u が実行するシェルに付与されるドメインである．

4.2　環境構築

検証を行うために構築した環境について述べる．検証では，攻撃者による web サーバの侵害によって管理者権限を

表 5　資産と資産に付与するラベル

保護資産	名前	ラベル
weusrd 実行ファイル	weexec	we_exec_t
ホワイトリスト	wl.txt	we_wl_t
ログファイル	log.txt	we_log_t
設定ファイル	config.txt	we_conf_t
ファイルシステム	mountpoint	we_mnt_p_t
デバイスファイル	/dev/sda	devfile_t
プロセス	weexec 実行で生成されるプロセス	we_t
ドライバファイル	/sys/kernel/security/whiteegret/wecom	we_driver_t

奪われることを想定するため，脆弱な web サーバの構築をする必要がある．使用する web サーバは Apache であり，管理者権限を持ったユーザで動作させる．この web サーバ上で任意のコマンドを実行できる脆弱な web ページを動作させることで攻撃者による管理者権限でのコマンド実行による侵害を再現する．検証のため WhiteEgret の資産を模したファイルなどとそれらに付与する SELinux のタイプ，ドメインを表 5 に示す．今回の検証環境は，CentOS7，Linux-4.13.16 である．

4.3　検証項目

保護されているか確認するため，以下の項目について検証を行う．

- 管理者による禁止すべきシステムコールの実行を制限できているか
- 管理者による許可すべきシステムコールの実行を許可できているか
- 攻撃者による禁止すべきシステムコールの実行を制限できているか

4.4　検証手順

今回検証する手順を以下に示す．

(1) 資産へのラベルの付与と管理者，攻撃者のシェルを異なる制限されたドメインで動作させるポリシーを作成して適用
(2) 資産を侵害
(3) SELinux のポリシー自動生成機能により侵害を許可するポリシーを作成
(4) 表 3，表 4，表 5 をもとにポリシーを変更，適用
(5) 実際に侵害し，適切に制御できているか確認

侵害する可能性のあるシステムコールの実行を制御した結果 WhiteEgret を保護できることを検証するため，侵害する可能性のあるシステムコール以外の要因によりアクセスが拒否されることを防ぐ必要がある．そのためにまずは検証する侵害行為を permissive モードで行い，拒否されたログからポリシ自動生成機能である audit2allow を利用してポリシーを生成する．さらに生成したポリシーを表 3 と表 4 に即した設定に変更する．

侵害については，先行研究 [2] においてソフトウェアを侵害可能なシステムコールを調査している．また，それらのシステムコールを利用するコマンドも示されているため，それを実行することで侵害行為とする．表 1，表 2 に示す権限それぞれ一つずつのコマンドの実行で検証し，コマンドが存在しないシステムコールは，自作のプログラムを作成し利用する．

4.5　検証結果

検証した結果の例として，ホワイトリストファイルへの書換えとしてパイプによるファイルの書換え，WhiteEgret のプロセスの削除として kill コマンドについて述べる．ホワイトリストの書換えの場合資産に必要な権限は，管理者による書換えは許可，攻撃者による書換えは拒否である．今回はホワイトリストを模した wl.txt へのパイプによる上書きを適切に制御できているかを検証する．管理者によるホワイトリストの上書きの実行結果を図 2 に，攻撃者による上書きの実行結果を図 3 に示す．図 2 では正常に実行され，図 3 では拒否されたため，正常に制御できることが確認できた．

次にプロセスの削除について述べる．プロセスの削除は，kill コマンドによって SIGKILL シグナルを WhiteEgret に模したプロセスに送ることによって行う．管理者によるプロセスの削除は許可し，攻撃者には許可しない．検証した結果を管理者によるものを図 4 に，攻撃者によるものを図 5 に示す．図 4 では正常に実行され，図 5 では拒否されたため，正常に保護できることが確認できた．以上と同様に他の侵害行為も検証を行った．検証結果を表 6 に示す．この結果は，2 章で示した設定すべき権限に即したものである．よって，検討したアクセスベクタを制御することで重要なソフトウェアを保護することができたと言える．

5.　関連研究

5.1　システムコール処理による権限の変化に着目した権限昇格攻撃の防止手法

赤尾らの研究 [4] では，権限昇格攻撃の防止を，特定のシステムコール前後の権限情報を比較し，そのシステムコールが変化させることがない権限が変化したことを検知する

```
[seadmin@dhcp-212 mountpoint]$ cat wl.txt
abc
[seadmin@dhcp-212 mountpoint]$ sudo echo def | sudo tee wl.txt
def
[seadmin@dhcp-212 mountpoint]$ cat wl.txt
def
```

図 2　管理者によるホワイトリスト上書きの実行結果

```
type=AVC msg=audit(1565101502.662:153): avc:  denied  { write } for
  pid=8956 comm="tee" name="wl.txt" dev="dm-0" ino=67605914 scontex
t=system_u:system_r:httpd_t:s0 tcontext=staff_u:object_r:we_wl_t:s0
 tclass=file permissive=0
```

Submit

sudo echo abc | sudo tee /mountpoint/wl.txtstring(4) "abc "

図 3　攻撃者によるホワイトリスト上書きの実行結果

```
0x7ffd85249f10=bbbbbb

0x7ffd85249f10=bbbbbb

強制終了
```

```
[seadmin@dhcp-212 ~]$ sudo kill -9 9522
[seadmin@dhcp-212 ~]$
```

図 4　管理者による kill コマンドの実行結果

```
type=USER_ACCT msg=audit(1562062324.807:184): pid=9196 uid=1000 auid=4294967295 ses=4294967295 subj=system_u:system_r:httpd_t:s0
msg='op=PAM:accounting grantors=pam_unix,pam_localuser acct="user" exe="/usr/bin/sudo" hostname=? addr=? terminal=? res=success'
type=USER_CMD msg=audit(1562062324.807:185): pid=9196 uid=1000 auid=4294967295 ses=4294967295 subj=system_u:system_r:httpd_t:s0 m
sg='cwd="/var/www/html" cmd=6B696C6C202D392038343635 terminal=? res=success'
type=CRED_REFR msg=audit(1562062324.807:186): pid=9196 uid=0 auid=4294967295 ses=4294967295 subj=system_u:system_r:httpd_t:s0 msg
='op=PAM:setcred grantors=pam_env,pam_fprintd acct="root" exe="/usr/bin/sudo" hostname=? addr=? terminal=? res=success'
type=USER_START msg=audit(1562062324.813:187): pid=9196 uid=0 auid=4294967295 ses=4294967295 subj=system_u:system_r:httpd_t:s0 ms
g='op=PAM:session_open grantors=pam_keyinit,pam_limits,pam_keyinit,pam_limits,pam_unix acct="root" exe="/usr/bin/sudo" hostname=?
 addr=? terminal=? res=success'
type=AVC msg=audit(1562062324.816:188): avc:  denied  { sigkill } for  pid=9198 comm="kill" scontext=system_u:system_r:httpd_t:s0
 tcontext=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0 tclass=process permissive=0
type=SYSCALL msg=audit(1562062324.816:188): arch=c000003e syscall=62 success=no exit=-13 a0=2111 a1=9 a2=9 a3=7ffc6a388de0 items=
0 ppid=9196 pid=9198 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=(none) ses=4294967295 comm="kill
" exe="/usr/bin/kill" subj=system_u:system_r:httpd_t:s0 key=(null)
type=PROCTITLE msg=audit(1562062324.816:188): proctitle=6B696C6C002D3900383435
type=USER_END msg=audit(1562062324.817:189): pid=9196 uid=0 auid=4294967295 ses=4294967295 subj=system_u:system_r:httpd_t:s0 msg=
'op=PAM:session_close grantors=pam_keyinit,pam_limits,pam_keyinit,pam_limits,pam_unix acct="root" exe="/usr/bin/sudo" hostname=?
addr=? terminal=? res=success'
type=CRED_DISP msg=audit(1562062324.817:190): pid=9196 uid=0 auid=4294967295 ses=4294967295 subj=system_u:system_r:httpd_t:s0 msg
='op=PAM:setcred grantors=pam_env,pam_fprintd acct="root" exe="/usr/bin/sudo" hostname=? addr=? terminal=? res=success'
```

Hello - Mozilla Fi

Hello

localhost/?com=sudo+k

Submit

sudo kill -9 8465NULL

図 5　攻撃者による kill コマンドの実行結果

表 6　検証結果

資産	侵害行為	コマンド	実行結果	
			管理者	攻撃者
ファイル，ディレクトリ	削除	rm	×	×
	書換え	パイプ	○	×
	読込み	cat	○	×
	DAC 権限変更	chown	×	×
	実行	実行	○	×
	パス変更	mount,umount	×	×
プロセス	削除	kill	○	×
	書換え	自作	×	×
	読込み	自作	×	×
	実行	自作	×	×
その他	時間変更	date	○	×
	カーネル書換え	insmod	○	×

ことで実現している．この論文では，プロセス権限に関するデータがカーネル空間に保存されることを前提としている．そのため，設定の変更には，変更をするためのインタフェースとして新たなシステムコールの実装や，カーネルのビルドなどをする必要があり，大きなコストがかかる．本手法は，アクセス制御の設定をユーザ空間に保存した場合を想定した保護手法であるため，設定の変更のコストを低くすることが可能である．

5.2　Intel SGX による保護方式の検討

内匠らの研究 [5] では，WhiteEgret を保護する手法として，Intel SGX を利用した方法が検討されている．Intel SGX は，メモリ空間隔離によりプログラムのコードやデータの保護を実現する [6][7]．Intel SGX により，保護領域内の完全性と機密性を実現できる．しかし，Intel SGX では，保護領域でシステムコールが発行不可能であるため，非保護領域を経由する必要がある．攻撃者によって weusrd と OS 間の通信を改ざんされると，マルウェア等の意図しないアプリケーションの実行を許す恐れがある．そのため，完全性を担保することが難しい．

また，Intel SGX は，非保護領域の処理が必要である．例えば，Intel SGX の設定処理である．そのため，攻撃者

は Intel SGX の設定処理を書き換えることで，保護領域の起動を妨げることができる．前述のように，非保護領域の書き換えが可能であり，非保護領域を利用するプロセスの動作を妨害できる．

一方で，この論文では，難読化や暗号化を用い，攻撃者に読まれても内容がわからない実装となっている．鍵の管理などのコストが発生するが，機密性は確保できたと言える．

本手法は，WhiteEgret の資産を侵害する可能性のあるシステムコールを制御することにより，資産への改ざんや削除を防ぎ，完全性，可用性を確保することが可能である．

6. おわりに

本論文では，組込みシステムにおける重要ソフトウェアの例として，WhiteEgret の資産に対して与えるべき権限と与えるべきではない権限と，侵害するシステムコールを SELinux によって制限することで保護できることを検証した結果について述べた．本検証では，先行研究で明らかにした侵害可能なシステムコールに必要なアクセスベクタを制御することで組込み機器の重要ソフトウェアの資産を保護できることが明らかとなった．しかし，本研究で検証する対象とした WhiteEgret は，ネットワークを利用しないものである．そのため，ネットワークを利用するソフトウェアを保護するための SELinux のアクセスベクタの調査を今後する必要がある．

参考文献

[1] 小池正修, 小椋直樹, 内匠真也, 花谷嘉一, 春木洋美: Linux 上でのホワイトリスト型実行制御機能 WhiteEgret^{TM} の開発，コンピュータセキュリティシンポジウム 2017 論文集，Vol. 2017, No. 2, pp. 1317–1323 (2017).

[2] 田中大樹, 内匠真也, 金井　遵, 鄭　俊俊, 毛利公一: SELinux による組込みシステムのソフトウェア保護手法 (in press).

[3] NSA: Security-Enhanced Linux, available from "`http://www.nsa.gov/selinux/`" (accessed 2019-6-6).

[4] 赤尾洋平，山内利宏: システムコール処理による権限の変化に着目した権限昇格攻撃の防止手法，コンピュータセキュリティシンポジウム 2016 論文集，Vol. 2016, No. 2, pp. 542–549 (2016).

[5] 内匠真也, 藤松由里恵, 小池正修, 金井　遵, 花谷嘉一: ホワイトリスト型実行制御機構 WhiteEgret^{TM} における Intel SGX による保護方式の検討，研究報告システムソフトウェアとオペレーティング・システム (OS)，Vol. 2018-OS-144, No. 7, pp. 1–7 (2018).

[6] Costan, V.: Intel SGX Explained, available from "`https://software.intel.com/sites/default/files/managed/50/8c/Intel-SGX-Product-Brief.pdf`" (accessed 2019-6-6).

[7] Intel: Intel SGX:Protect Application Code & Secrets from Attack, available from "`https://eprint.iacr.org/2016/086.pdf`" (accessed 2019-6-6).