L-034

SELinux による組込みシステムのソフトウェア保護手法
Securing Embedded System Software Using SELinux

田中 大樹† 内匠 真也‡ 金井 遵§ 鄭 俊俊† 毛利 公一†
Daiki Tanaka Shinya Takumi Jun Kanai Junjun Zheng Koichi Mouri

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®，ARM®などのさまざまな CPU と，Windows®，Linux®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．本稿では，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．

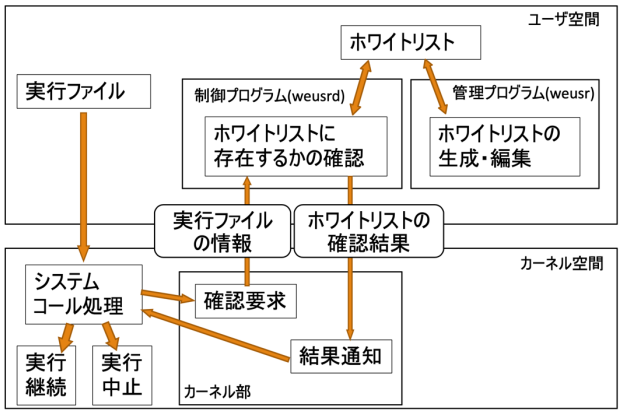

図 1　WhiteEgret の構成

本稿では，WhiteEgret の保護を検討するため，まずは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について述べる．また，これらのシステムコールの制御を実現するため，強制アクセス制御を行う LSM (Linux Security Module) を利用した SELinux (Security-Enhanced Linux) [2]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結果を述べる．以下，2 章で保護するソフトウェアの資産について述べ，3 章で制御対象のシステムコールについて述べる．4 章では保護資産を SELinux で保護するためのアクセスベクタの調査について述べ，5 章で関連研究について述べる．最後に 6 章でまとめる．

2. ソフトウェアの保護資産

本稿で保護するソフトウェアの例である WhiteEgret は，Linux 機器上で実行を許可するプログラムをあらかじめホワイトリストとして定義しておき，ホワイトリストにないプログラムの実行を拒否することで，機器・システムを保護する技術である．WhiteEgret による ホワイトリスト型実行制御の概要を図 1 に示す．ユーザ空間上に実装する WhiteEgret の判定プログラムは，用途に応じて任意に実装することが可能だが，サンプルとして，ホワイトリスト生成・編集処理を行うプログラム weusr とカーネルからの問い合わせに対してホワイトリストとのマッチングを行うデーモンプログラム weusrd が用意されている．これらのユーザ空間上の資産は，管理者権限によって削除や書換えなどが可能である．しかし，管理者による操作であれば良いが，攻撃者が管理者権限を得た場合，WhiteEgret を侵害される可能性がある．そのため，攻撃

† 立命館大学　Ritsumeikan University
‡ 東芝インフラシステムズ株式会社
Toshiba Infrastructure Systems & Solutions Corporation
§ 東芝　Toshiba

表 1　侵害する可能性のあるシステムコール

保存場所	権限	システムコール				
二次記憶	削除	unlink	unlinkat	creat	truncate	ftruncate
		open	openat	open_by_handle_at		
	書換え	write	writev	mmap	copy_file_range	splice
	読込み	read	readv	mmap		
	パス変更	mount	umount	pivot_root	rename	renameat
		renameat2	link	linkat	symlink	symlinkat
	DAC 権限変更	chmod	fchmod	chown	fchown	fchownat
メモリ	停止・削除	kill	tgkill	prlimit64	capset	quotactl
(プロセス)	書換え	write	writev	ptrace	process_vm_writev	
	読込み	read	readv	ptrace	process_vm_readv	
	実行	execve	execveat	mprotect		
その他	時間	settimeofday	adjtimex	clock_adjtime		
	カーネル書換え	kexec_load	kexec_file_load	init_module	finit_module	

者に管理者権限が奪われた場合の WhiteEgret の保護について考える．本章では，WhiteEgret が正常に動作し続けるために保護すべき資産とその資産が一般的なソフトウェアのどの資産に相当するかについて述べる．

実行ファイル

weusrd や weusr などの実行ファイルである．削除や書き換えが発生した場合，正常な動作や，次回起動時に起動しなくなる可能性が存在する．この資産は，一般的なソフトウェアの実行ファイルに相当する．

ホワイトリストファイル・設定ファイル

現在の実装では，起動時にのみ読込み，再読込みを行わない．そのため実行ファイルと同様に，削除や書換えにより現在起動しているシステムへ影響はないが，次回起動時に正しく動作しない可能性がある．この資産は，一般的なソフトウェアが参照する設定ファイルに相当する．

メモリ上のプログラム・ホワイトリスト・設定ファイル

WhiteEgret では実行ファイルやホワイトリストファイルを読込み，メモリに書き込む．そのデータを削除，改ざんされると，システムの停止や意図しない動作を引き起こす可能性がある．この資産は，一般のソフトウェアではメモリにマップされたファイルである．

ログファイル

ログファイルは，フォレンジック調査を行う可能性があるため，ログファイルには正確な時間が記録される必要がある．そのため，システムの時間を保護する必要がある．しかしログファイルは単なるテキストファイルであり，削除，書換えがシステムの動作に直接関係しないため，保護の優先度が低い資産とする．この資産は，一般ソフトウェアでもログファイルに相当する．

カーネル通信ドライバファイル

WhiteEgret では，ユーザ空間とカーネル空間間のデータのやり取りに，LSM 関連の情報のやり取りに利用される擬似ファイルである securityfs を利用している．このファイルの削除，書換えにより，意図しない動作やシステムの停止が起こる可能性がある．この資産は，ファイルとしての実態はなく一般ソフトウェアでは書込みが必要だが内容によっては侵害となる資産である．

デバイスファイル

WhiteEgret の資産は，メモリや SSD などのデバイスに保存されている．これらのデバイスへのアクセスは，デバイスファイルというインタフェースを利用する方法がある．デバイスファイルを読書きすることによって，デバイス内のデータを読書きすることが可能となる．メモリのユーザ空間のデバイスファイル「/dev/mem」と，カーネル空間のデバイスファイル「/dev/kmem」は OS によって保護されるため，保護が必要な資産としない．しかし，二次記憶のデバイスファイル「/dev/sda など」は，OS で保護されておらず，書換えが容易であるため保護が必要な資産とする．

ライブラリ

ライブラリは，通常の実行ファイルとは異なり，共有ファイルとして複数のプロセスからアクセスが可能となり，他プロセスからの侵害が比較的容易に起こりうる．一般のソフトウェアでもライブラリに相当する．

ファイルシステム

ファイルは，ファイルシステムによって管理されている．ファイルシステムは，攻撃者によってアンマウントされ，同様のディレクトリ構成をしたファイルシステムをマウントすることでファイルパスを偽装することが可能となる．一般のソフトウェアでも，資産を管理するファイルシステムに相当する．

3. 制御対象システムコールの調査

2 章で述べた保護資産を攻撃者が侵害するにはシステムコールの実行が必要である．したがって，資産を守るには，攻撃者によるこの資産を侵害するシステムコールの実行を禁止するように制御すればよい．そのため本章では，全システムコールの調査を行い，実行されると WhiteEgret が侵害されるシステムコールについて調査する．

3.1 調査方法

調査した環境は，Ubuntu17.10 の Linux カーネル 4.13.16 である．今回の調査では，システムコールテーブルを確

認することでシステムコール一覧を取得し，現環境に実装されているシステムコールのマニュアルを確認して侵害する可能性があるか判断した．

3.2 調査結果

調査した結果を表 1 に示す．保護すべき資産は，二次記憶に保存されているものと，メモリに保存されているもので大きく分けることが可能であり，それぞれについて侵害の種類ごとにどのシステムコールに WhiteEgret への侵害の可能性があるのか述べる．記述されていないシステムコールに関しては，WhiteEgret に対して直接侵害することが不可能なものである．

3.2.1 二次記憶

二次記憶に保存される資産への侵害としては，ファイルやディレクトリへの読込み，書込み，実行ファイルの実行が考えられる．また，現在のファイルシステムをアンマウントし，同じ名前の悪性ファイルシステムをマウントすることで，悪性ファイルを正常ファイルと同一であると見せかけることも可能である．

3.2.2 メモリ

メモリに保存される資産への侵害とは，保護対象のプロセスとは別のプロセスからメモリ内のデータに対して，読込み，書込み，実行されることが考えられる．また，プロセス自体もメモリ内の保護資産と定義し，プロセスの停止や削除も侵害可能性として考えられる．

4. SELinux による保護

WhiteEgret を保護するためには，2 章で述べた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が可能とした SELinux による制御を検討する．本章では，制御対象のシステムコールの実行に必要なアクセスの種類であるアクセスベクタについて調査した結果を述べる．

4.1 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きる．

4.2 調査方法

制御すべきシステムコールに対応したアクセスベクタを調査するために，システムコールを実行するテストプログラムを制限されたドメインで実行し，SELinux が実行を拒否した際に残るログを確認する．

今回のテストプログラムは，LTP (Linux Test Project) [3] に含まれるシステムコールの動作確認テスト用のプログラムを使用した．LTP は，Linux の安全性，信頼性，堅牢性などを検証するためのツールやプログラムを提供するものであり，Linux カーネルと関連機能をテストするためのツールやプログラムのソースコードを含む．LTP の中には，システムコールのテストを行うため，主要なシステムコールそれぞれの引数などに使われるフラグなどを変えた複数のテストプログラムが含まれる．ただし，本調査で実行しなければいけないシステムコールのテストプログラムが全て用意されてはいないため，テストプログラムの存在しないシステムコールは，システムコール を利用しているコマンドの改変や自作のテストコードを作成，実行することでアクセスベクタの観測を行なった．

なお， SELinux には，要求されたアクセスベクタとポリシーを比較し，システムコールの実行を停止した場合の拒否されたアクセスベクタをキャッシュする実装となっている．調査では，システムコール実行に必要なアクセスベクタを全て明らかにする必要があるが，このキャッシュにヒットした場合はログが残らないため，次のような手法を用いた．

システムコール実行の正確なログを観測するためには，システムコール実行の直前でキャッシュを削除する必要がある．キャッシュの削除は管理者権限で SELinux のモードを変更することで可能である．しかし本調査では，SELinux により管理者権限を制限するため，モード変更が禁止される．以上の問題を，ドメイン遷移により，テストプログラムを実行して生成されるプロセスのみ権限を低くする設定を行い，モード変更プロセスを高い権限で動作させることで解決する．また，システムコール直前でのモード変更は，システムコールの直前に sleep()を埋込み，プロセスが休止している間にモード変更を行うことで実現する．

4.3 調査結果

システムコール実行に必要なアクセスベクタを調査した結果を表 2，表 3 に示す．表 2 は LTP のテストプログラムによって正常にテストできたシステムコールの結果で，表 3 は LTP のテストプログラムが存在せず自作のプログラムもしくはコマンドの改変によってテストを行ったシステムコールの結果である．表 2，表 3 の 2 列目は，テストプログラムの番号である．

表 2，表 3 では，引数に与えるフラグによって観測されたアクセスベクタが異なり，それぞれ制御が必要であるシステムコールに(※)を付与している．mprotect システムコールは，現存のもしくは確保するメモリ空間に与える権限のマクロを引数に与え，その違いによって必要なアクセスベクタが変わる．PROT_READ は read，PROT_WRITE は write，PROT_EXEC は execmem アクセスベクタが必要となる．mmap システムコールは，mprotect と基本的には同様であるが，PROT_EXEC は execute アクセスベクタが必要となる点のみが異なる．open，openat，open_by_handle_at のファイルを開くシステムコールは，開くファイルへの権限をフラグで引数として与える．

表 3　LTP にテストプログラムが存在しないシステムコール

システムコール名	番号	アクセスベクタ			
kexec_load	1	sys_boot			
kexec_file_load	1	sys_boot			
init_module	1	sys_module	getattr		
finit_module	1	sys_module	getattr		
prlimit64	1	sys_resource	setrlimit		
pivot_root	1	sys_admin			
open_by_handle_at(※)	1	dac_read_search	open	read	write
	2	dac_read_search	open	read	
	3	dac_read_search	open	write	
	4	dac_read_search	open	read	append
	5	dac_read_search	open	read	write

O_RDONLY は read，O_WRONLY と O_TRUNC は write，O_RDWR は read と write，O_APPEND は append，O_CREAT とは create アクセスベクタが必要となる．creat システムコールは，ファイルを新規作成するシステムコールであり create アクセスベクタが必要となるが，すでに作成対象のファイルが存在した場合ファイルをオープンする．このときのファイルアクセス権限は open システムコールでいう O_WRONLY|O_TRUNC であり，open と write アクセスベクタが必要となる．kill と tgkill システムコールは，指定したプロセスに対して送るシグナルの種類によってアクセスベクタが異なる．SIGKILL シグナルは sigkill，SIGSTOP シグナルは sigstop，それ以外のシグナルは signal アクセスベクタが必要である．これは，シグナルは処理内容を変更することができるが SIGKILL と SIGSTOP は変更できないことが理由であることが考えられる．

表 3 の pivot_root と open_by_handle_at システムコール以外のシステムコールは，システムコールを実行するコマンドを改変して観測を行った．LTP のテストプログラムは，引数や返り値全てを網羅できるように作成されている．しかし，コマンドの改変では機能などを網羅できず，観測されたアクセスベクタで完全に制御できるとは限らない．そのため，アクセスベクタに対応したフックポイントがシステムコール実行中に必ず実行されるものであれば，想定外の機能が存在しても制御可能である．init_module と finit_module システムコールは，sys_module と getattr アクセスベクタが必要である．しかし，getattr は stat システムコールなど多くのシステムコールの実行に必要なアクセスベクタであり，getattr による制御は適切ではない．そのため，sys_module のフックポイントの場所のみ調査した．その結果，今回調査したフックポイントの位置はすべて必ず実行されるものであったため，制御可能であることが明らかになった．open_by_handle_at システムコールは，open 系のファイルを開くシステムコールであるため，LTP の open システムコールのテストプログラムを改変して観測する．pivot_root システムコールは，pivot_root を実行するコマンドの pivot_root コマンドの実行を試みたが，実行できなかったため，ソースコードによる調査を行った．

表 4　open 系で制御されるシステムコール

システムコール名	open のフラグ	アクセスベクタ
read,readv	O_RDONLY	read
write,writev	O_WRONLY	write
process_vm_readv	O_RDONLY	read
process_vm_writev	O_WRONLY	write
copy_file_range	O_WRONLY	write
splice	O_WRONLY	write

4.4 アクセスベクタが観測されないシステムコール

表 2 で観測結果をなしとしたシステムコールは，実行前にファイルを開く必要があり，open 系のファイルを開くシステムコールによって制御されていることが考えられる． 例えば read システムコールは，引数にファイルを読取り権限をフラグとして引数に渡した open 系のシステムコールの返り値のファイルディスクリプタを渡す必要がある．その場合，read システムコール自体ではアクセスベクタで制御できないため open 系システムコールの read アクセスベクタで制御することになる．以上のように，open 系システムコールで制御されるシステムコールと，実行に必要なフラグ，フラグに対応したアクセスベクタをまとめたものを表 4 に示す．

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ことで実現した．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た．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した [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

本稿では，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 の資産と資産に対して侵害する可能性のある制限すべきシステムコールについて述べ，SELinux で制限するために制限すべきシステムコールの実行継続に必要なアクセスベクタの調査について述べた．本調査では，重要なソフトウェアを侵害する可能性のあるシステムコールと，システムコールを SELinux で制御するためのアクセスベクタについて明らかにした．これらを制御する設定を適用させ，侵害可能なシステムコールを制御することでソフトウェアを保護することが可能であるが，セキュリティ管理者はシステムの実行や管理のためそれらのシステムコールの実行権限が必要となる場合がある．そのため，アクセス制御により他のシステムにどのような影響があるか調査する必要がある．今後は，以上の調査と実際に制御を行う場合に禁止すべきアクセスベクタと許可すべきアクセスベクタについて検討し，設定を適用させて検証を行う．

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™ 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] NSA, “ Security-Enhanced Linux ”, available from <http://www.nsa.gov/selinux/>, (accessed 2019-6-6).
[3] LTP developers, “ Linux Test Project ”, available from <http://linux-test-project.github.io/>, (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™ 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] V.Costan, L.Lebedev, S.Devadas, “ Intel SGX Explained ”, available from <https://eprint.iacr.org/2016/086.pdf> (accessed 2019-6-6).
[7] Intel, “Protect Application Code & Secrets from Attack”, available from <https://software.intel.com/sites/default/files/managed/50/8c/Intel-SGX-Product-Brief.pdf> (accessed 2019-6-6).

表 2　LTP のテストプログラムが存在するシステムコール

システムコール名	番号	アクセスベクタ				
symlinkat	1	create	write	add_name		
	2	create	write	add_name	search	
kill,tgkill(※)	1	signal	sigkill	sigstop		
capset	1	setcap				
settimeofday	1	sys_time				
adjtimex	1	sys_time				
clock_adjtime	1	sys_time				
quotactl	1	quotamod	quotaon	sys_admin		
	2	quotamod	sys_admin			
	3	quotamod				
chown	1	setattr				
	2	setattr	chown	fsetid		
	3	setattr	chown			
fchown	1	setattr				
	2	setattr	chown	fsetid		
	3	setattr	chown			
fchownat	1	setattr				
	2	setattr	chown			
chmod	1	setattr				
fchmod	1	setattr				
	2	setattr	fowner			
renameat2	1	write	remove_name	rename	add_name	unlink
	2	write	remove_name	rename	add_name	
mmap(※)	1	read	write			
	2	read				
	3	read	execute			
	4	ipc_lock				
read	1	なし				
readv	1	なし				
write	1	なし				
writev	1	なし				
process_vm_readv	1	なし				
process_vm_writev	1	なし				
copy_file_range	1	なし				
splice	1	なし				

システムコール名	番号	アクセスベクタ				
rename	1	rename	remove_name	write	add_name	
	2	rename	remove_name	write	add_name	unlink
renameat	1	rename	remove_name	write	add_name	
	2	rename	remove_name	write	add_name	search
unlink	1	unlink	remove_name	write		
unlinkat	1	unlink	write	remove_name		
	2	unlink	write	remove_name	search	
truncate	1	write	open			
ftruncate	1	write	open			
execve	1	execute	execute_no_trans	read	open	
execveat	1	execute	execute_no_trans	read	open	
	2	execute	execute_no_trans	read	open	search
mount	1	mount	mounton	sys_admin		
	2	mount_on	sys_admin			
umount	1	unmount	sys_admin			
mprotect(※)	1	read				
	2	read	write			
	3	execmem				
ptrace	1	ptrace				
open(※)	1	open	read	write	create	add_name
	2	read	open			
	3	read	write	open		
	4	write	open			
openat(※)	1	open	read	write		
	2	open	read	write	search	
	3	open	read	append		
creat(※)	1	create	open	write	add_name	
	2	open	write			
link	1	link	write	add_name		
linkat	1	write	add_name	link		
	2	write	add_name	link	search	
symlink	1	create	write	add_name		