外部入力の変更によるプロセスヒーリングシステム

富山高彰† 大山恵弘†

1. はじめに

近年，計算機の普及に伴いセキュリティの問題が深刻となっている．例えば，悪意のあるユーザがネットワークを介して外部の計算機のデータを盗んだり，改竄したりすることが挙げられる．有効な対処法のひとつとして，アクセスが制限された環境の中でプロセスを動作させるセキュリティシステムがある．このようなセキュリティシステムの多くにおいては，ポリシーに従って資源へのアクセスが制御される．しかし，それらの中には，ポリシーに違反した場合にプロセスを強制終了させるなどの単純な対策しか取れず，柔軟な制御ができないものが多い[1)2)]．

プロセスが強制終了すると，いくつか問題が生じる場合がある．例えば，獲得中の資源が解放できなかったり，データの一貫性がとれなかったりする．さらに，Web サーバなどクライアントにサービスを提供しているプロセスの場合，サービスの継続性の観点から終了させるのは不適切である．

そこで本研究では，ポリシーに従いアクセスが制限された環境でプロセスを動作させ，ポリシー違反時や異常発生時にプロセスを強制終了させるのではなく正常な状態に戻すセキュリティシステムを提案する．具体的には，本システムは上記の時にプロセスの状態をポリシー違反時や異常発生前の状態に自動的に巻き戻す．

2. 設計方針

本システムはプロセスとして動作し，保護の対象となるプロセスが発行するシステムコールの監視および制御を行う．以降では，本システムのプロセスをモニタプロセス，監視対象となるプロセスを対象プロセスと呼ぶ．システムコールの制御は本システムの管理者が与えるポリシーに従って行われる．ポリシーには，実行を禁止する処理としてシステムコールとその引数情報を記述する．例えば，“open 1 /etc/passwd” は，第一引数に /etc/passwd をとる open システムコールの実行を禁止する．

本システムは，対象プロセスがポリシー違反となるシステムコールを発行したり，対象プロセス内でセグメント違反などの異常が発生した時に，対象プロセスの状態を過去の状態に巻き戻す．状態の巻き戻しは，ポリシー違反や異常発生の原因となりうるイベントが対象プロセス内で発生した際に，対象プロセスのコピー (シャドウプロセス) を作成して停止させておくことにより実現する．そのイベントは，具体的には，recv などの外部入力を受け取るシステムコールの発行である．状態の巻き戻しの際には，モニタプロセスは対象プロセスを終了させ，シャドウプロセスを新たな対象プロセスとして再開させる．recv などの外部入力を受け取るシステムコールの実行では，モニタプロセスが受信データを記憶しておき，そのシステムコールが再実行された時に対象プロセスにそのデータを与える．外部入力を受け取るタイミングでシャドウプロセスを作成しているのは，ポリシー違反や異常は外部からの入力によって引き起こされることが多いためである．例として，バッファオーバーフロー攻撃が挙げられる．本システムでは，アプリケーションのソースコードやカーネルに修正を必要としない．

3. 実　　装

対象プロセスの監視には ptrace システムコールを用いる．モニタプロセスは ptrace によって，対象プロセスがシステムコールを実行する直前と直後で停止させる．そして，モニタプロセスは停止させたプロセスのレジスタやメモリを読み込み，ポリシーに違反しているかどうかを検査する．

シャドウプロセスの生成は，モニタプロセスが対象プロセスに強制的に clone システムコールを呼び出させることで実現する．シャドウプロセスは生成直後にモニタプロセスによって停止され，勝手に実行が進むことはない．ポリシー違反やセグメント違反などの異常を検出した時に，モニタプロセスは現在監視しているプロセスを終了させ，監視対象をシャドウプロセス

† 電気通信大学大学院情報理工学研究科総合情報学専攻

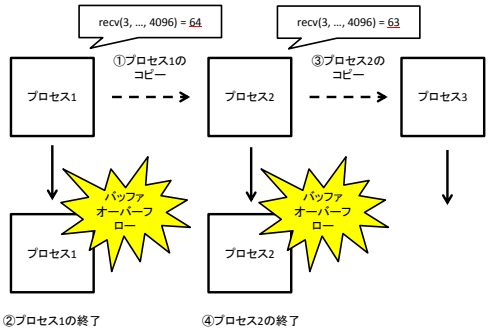

図 1 外部入力の変更例

に切り替えることにより，プロセスの状態を巻き戻す．再実行を開始する直前には，clone システムコールによりシャドウプロセスのコピーを作成し，再び再実行が発生したときに対応できるようにする．モニタプロセスは対象プロセスのシステムコール引数に含まれるプロセス ID を仮想化することにより，たとえプロセスが切り替わった時でも，複数の対象プロセスが互いに同期を取る際の参照関係を崩さないようにする．

プロセスの状態を単純に巻き戻すだけでは，同様の異常が再び発生する可能性が高い．そこで，対象プロセスがシャドウプロセスに切り替わる度に，外部入力の値を少しずつ変えていき，ポリシー違反や異常が発生しなくなるまで繰り返す (図 1)． また，外部入力の値を変えても同じ場所でポリシー違反や異常が発生する場合は，1 つ古い状態まで巻き戻って実行を再開する．

再実行されるプログラム部分に副作用を伴う処理が存在する場合，その処理が複数回実行されるという問題が生じる．シャドウプロセスを生成してから巻き戻るまでの間に変更があったファイルは，巻き戻しの際に元の状態に復元する．これにより，ファイルへの副作用を消すことができる．ファイルの復元は，Stackable ファイルシステムである Aufs により実現する．端末への出力処理については，対処のための手法を現在検討中である．現状では，例えば，シャドウプロセスを生成してから巻き戻るまでの間に printf() 関数を呼ぶようなプログラムの場合，再実行のたびに端末に文字列を表示してしまう．

4. 実　　験

本システムにより，ポリシー違反や異常を検出した後でもプロセスを正常に継続できるか確認するため予備実験を行った．実験には文字列を送信するだけのクライアントと，文字列を受信して端末に表示するサーバを使用した．ここで，サーバにはバッファオーバーフローの脆弱性を含ませた．実際に本システムが監視しているサーバへバッファオーバーフロー攻撃を含んだ文字列を送信したところ，攻撃を検知する度にプロセスは切り替わり，最後にはバッファオーバーフローを起こさずに文字列を表示したことを確認できた．

5. 関 連 研 究

Rx[3] は本研究と同様に巻き戻しの度にプロセスの環境を変更するシステムである．プロセスに異常が発生すると，バッファの位置をずらしたり，パディングを行うことで異常を乗り越えようと試みる．Rx ではメモリ管理やスケジューリングなどを変更して異常に対処するが，本システムでは外部入力を変更して対処する．

Systrace[4] はポリシーをシステムコールベースで指定するセキュリティシステムである．Systrace はポリシー違反時に対象プロセスを強制終了させるが，本システムではポリシー違反後も対象プロセスの実行を継続させる．

6. 現状と今後の課題

現状は，ポリシー違反時や異常発生時にプロセスの状態を，違反前の状態に自動で巻き戻すセキュリティシステムの提案をした．また，予備実験を行う程度の簡単な実装ができた．

今後は，シャドウプロセスを複数保持できるような多重ロールバックや，異常発生時の詳細なログの保存の実装を行いたいと考えている．そして，Apache (Web サーバ) や Squid (Proxy サーバ) などの実用的なプログラムに適用して評価を行いたい．

参　考　文　献

1) A. Acharya, M. Raje. MAPbox : Using Parameterized Behavior Classes to Confine Untrusted Applications, In *Proceedings of the 9th USENIX Security Symposium*, August 2000.
2) L. C. Lam et al. Automatic Extraction of Accurate Application-Specific Sandboxing Policy, In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection*, September 2004.
3) F. Qin, J. Tucek, J. Sundaresan, and Y. Zhou. Rx: Treating Bugs as Allergies - A Safe Method to Survive Software Failures, In *Proceedings of the 20th Symposium on Operating Systems Principles*, October 2005.
4) N. Provos. Improving Host Security with System Call Policies, In *Proceedings of the 12th USENIX Security Symposium*, August 2003.