JPCERT CC®

制御システム用ソフトウエアの脆弱性対策に有効なCERT C コーディングルールの調査

2014年6月2日

一般社団法人JPCERT コーディネーションセンター
情報流通対策グループ
脆弱性解析チーム

はじめに

近年、制御システム分野においても、一般の情報システムと同様、セキュリティ対策の必要性が強く認識され始めています。

セキュアなソフトウエア開発のためのコーディングガイドラインとしては、ソフトウエア開発者向けにまとめられた「CERT C コーディングスタンダード」があります。しかし、このスタンダードには 300 を超えるコーディングルールが含まれており、開発現場で活用するには、開発現場で取り組む際の優先度付けが求められていました。

制御システム用ソフトウエアの脆弱性については、米国 ICS-CERT が ICS-CERT Advisory として公開しています。ICS-CERT Advisory によって 2010 年から 2013 年末の約 4 年間に 445 件の制御システム用ソフトウエアの脆弱性が公開されていますが、大半は、プログラマがセキュアなコードを書く方法を知っていれば脆弱性を作り込まずに済んだ、実装上の問題です。

本レポートでは、ICS-CERT Advisory で公開された制御システム用ソフトウエアの脆弱性を調査し、それらの低減に役立つ22 個のルールを CERT C コーディングスタンダードの中から抽出しました。

制御システム開発者の皆様が「CERT C コーディングスタンダード」を活用してセキュアな製品開発に取り組む契機として本レポートをご活用ください。

概要

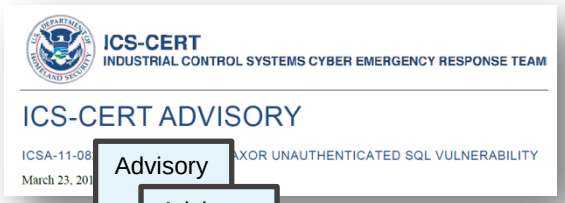
調査目的

- 制御システム用ソフトウエア製品の脆弱性対策に有効な CERT-C コーディグルールを特定し、提示すること

調査方法

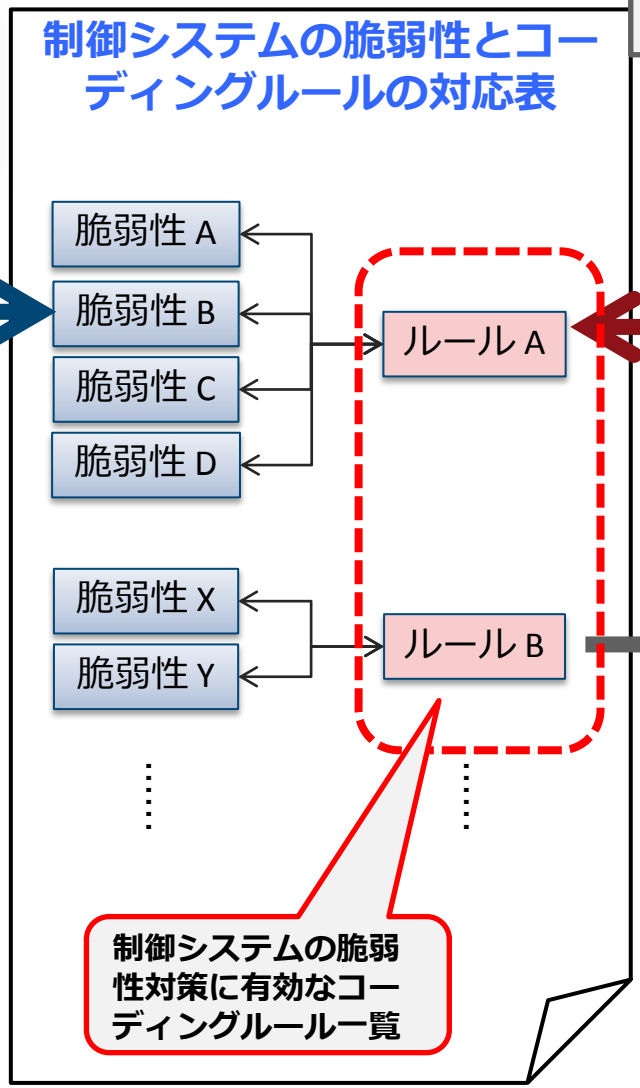
- 米国 ICS-CERT が公開したアドバイザリから脆弱性(CWE)を抽出する
- 抽出された各脆弱性の対策に有効な CERT-C コーディングルールをリストアップする

調査の流れ



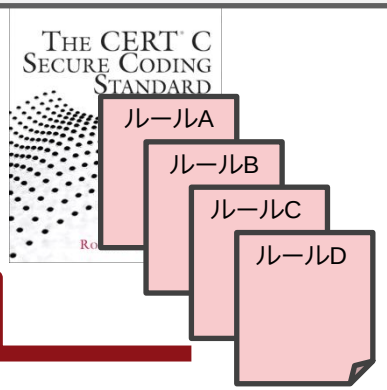
STEP 1. データ収集

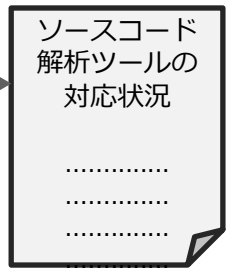
ICS-CERT アドバイザリから脆弱性情報(CWE等)を収集し、制御システム用ソフトウエアの脆弱性の状況を把握する



STEP 2. コーディングルールの抽出

脆弱性(CWE)とCERT-Cのルールとの対応関係を明らかにする





STEP 3. まとめと考察

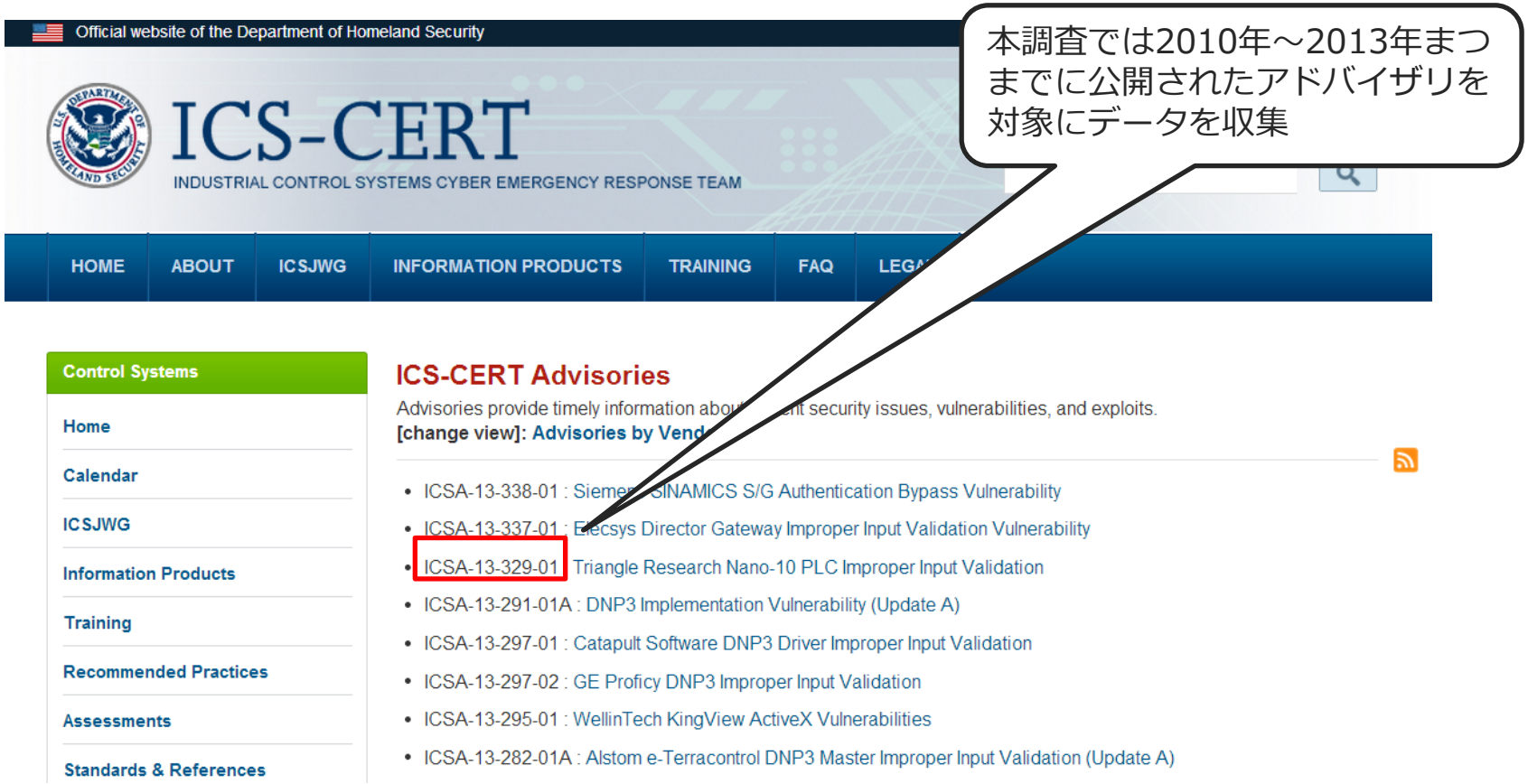
データの考察、まとめ

調査 Step 1. データ収集

制御システム用ソフトウエアの脆弱性

ICS-CERT アドバイザリとは

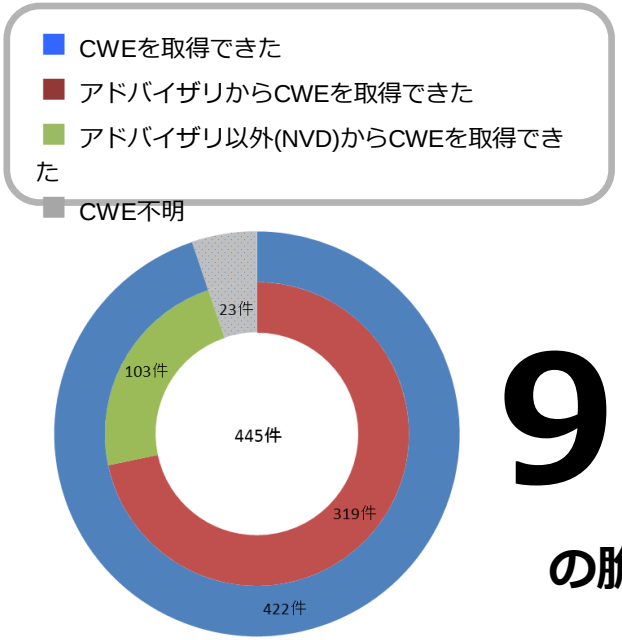
- 米国土安全保障省の産業制御システムセキュリティ担当機関が提供する制御製品の脆弱性情報
  - http://ics-cert.us-cert.gov/advisories



ICS-CERT アドバイザリの脆弱性

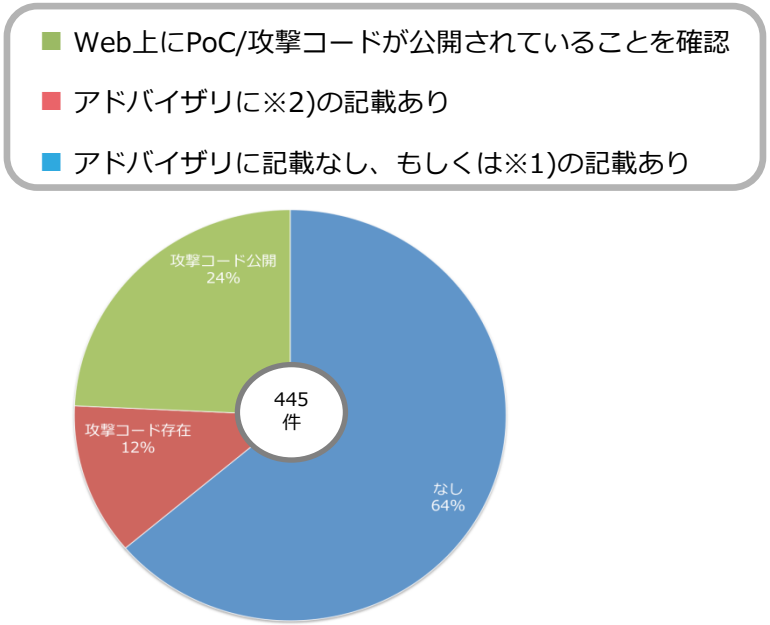
	件数
アドバイザリの件数	242
脆弱性の件数	**445**
1アドバイザリ当たりの脆弱性件数	1.8

期間: 2010年～2013年末



95%

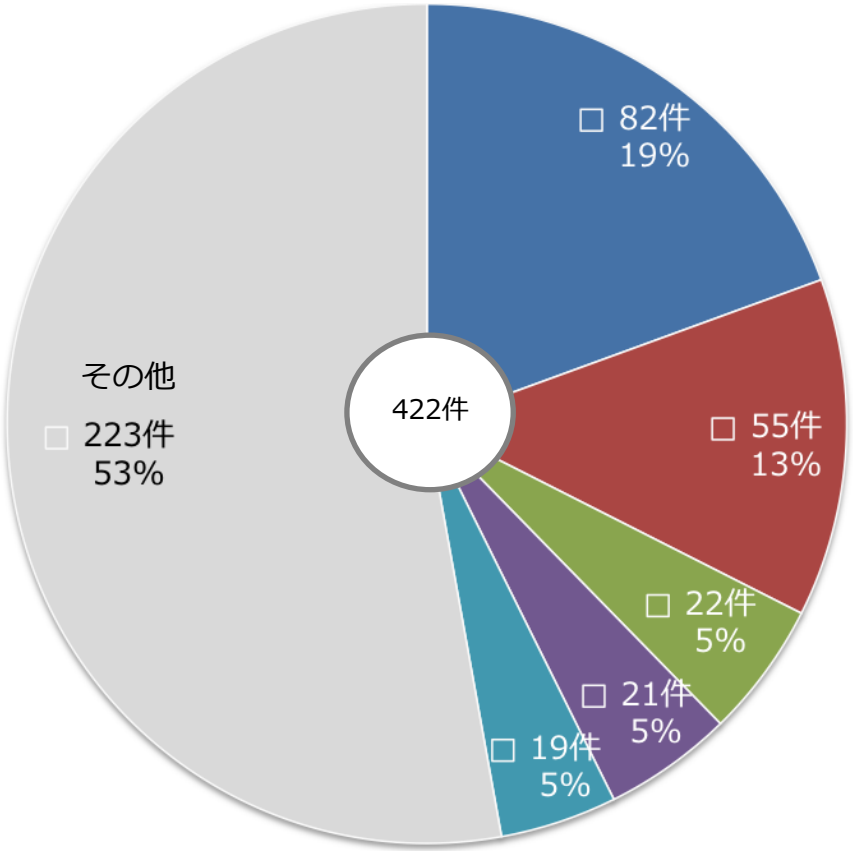
の脆弱性を特定



36%

に攻撃コードが存在

脆弱性ランキング



1位 **CWE-119**
バッファエラー

2位 **CWE-20**
不適切な入力確認

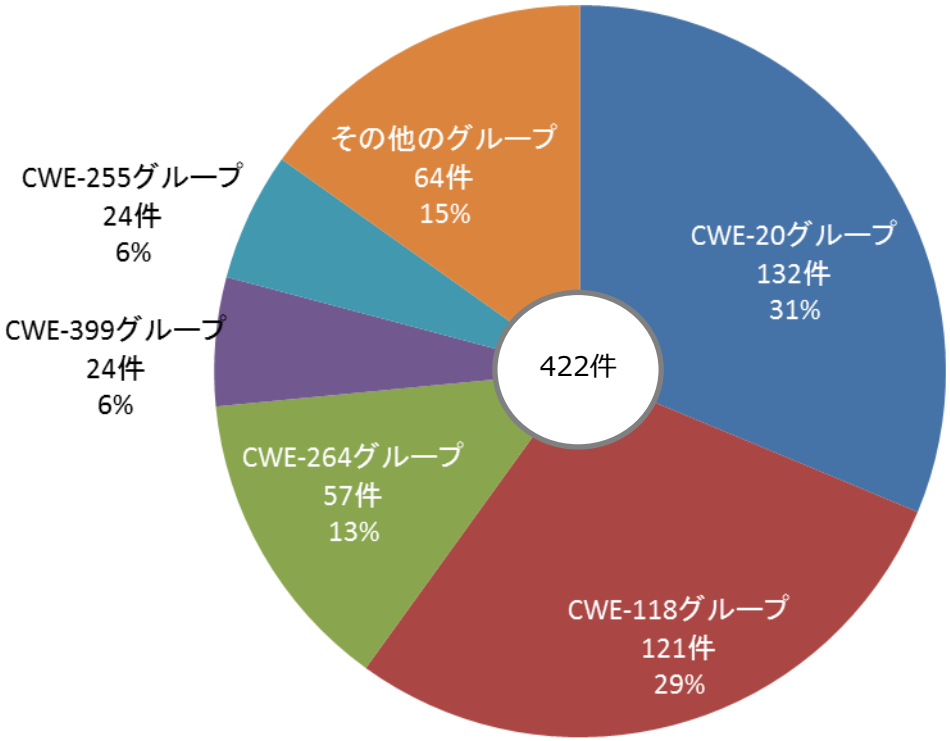
3位 **CWE-79**
クロスサイトスクリプティング

4位 **CWE-22**
パストラバーサル

5位 **CWE-121**
スタックバッファオーバーフロー

脆弱性の分類

CWEを上位の階層で集約した場合の脆弱性の割合



1位 **CWE-20**
不適切な入力確認

2位 **CWE-118**
範囲エラー

3位 **CWE-264**
認可・権限・アクセス制御

4位 **CWE-399**
リソース管理の問題

5位 **CWE-255**
証明書・パスワードの管理

脆弱性ランキング

順位	CWE	脆弱性のタイトル	件数
1	CWE-119	バッファエラー	82
2	CWE-20	不適切な入力確認	55
3	CWE-79	クロスサイトスクリプティング	22
4	CWE-22	パストラバーサル	21
5	CWE-121	スタックバッファオーバーフロー	19
6	CWE-287	不適切な認証	14
7	CWE-264 CWE-89	認可・権限・アクセス制御 SQL インジェクション	13
9	CWE-122 CWE-284	ヒープバッファオーバーフロー 不適切なアクセス制御	12
11	CWE-259	パスワードのハードコーディング	8
12	CWE-200 CWE-427	情報漏洩 ファイル検索パスの制御不備	7
14	CWE-399 CWE-190 CWE-255 CWE-331 CWE-400 CWE-476	リソース管理エラー 整数オーバーフローとラップアラウンド 証明書やパスワードの管理 不十分なエントロピー リソース枯渇 NULL ポインタ参照	5
20	CWE-23 CWE-798 CWE-311 CWE-352 CWE-592 CWE-912	相対パストラバーサル 認証情報のハードコーディング センシティブなデータを暗号化しない クロスサイトリクエストフォージェリー (CSRF) 認証バイパス 隠れた機能	4
26	CWE-94 CWE-125 CWE-134 CWE-189 CWE-306 CWE-416 CWE-425	コードインジェクション 境界外読取り 書式文字の制御に不備 数値処理の問題 重要な機能に対する認証の欠如 解放済みメモリ使用 リクエストの直接送信 ('Forced Browsing')	3

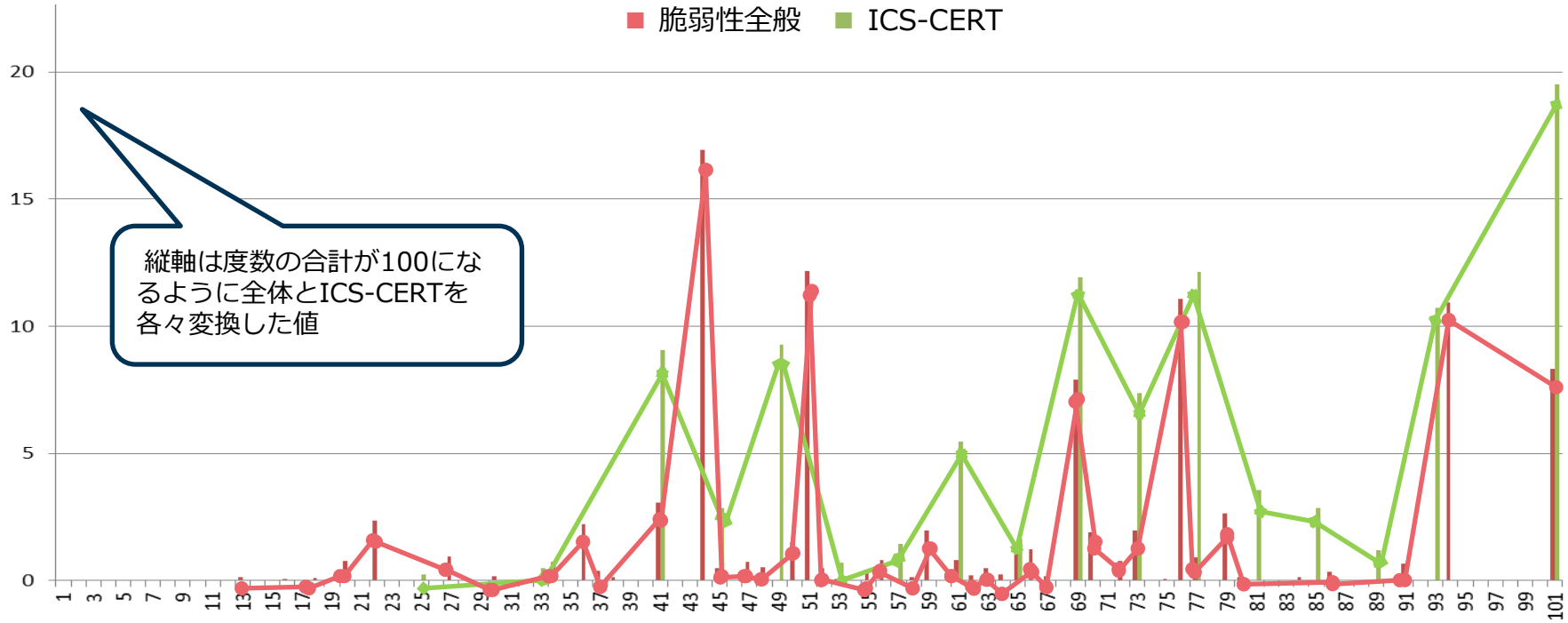
ICS-CERTアドバイザリで特定された脆弱性とその件数

33	CWE-78 CWE-120 CWE-91 CWE-261 CWE-269 CWE-285 CWE-310 CWE-330 CWE-388 CWE-538 CWE-618 CWE-471 CWE-321	OS コマンドインジェクション 古典的バッファオーバーフロー XML インジェクション パスワードの暗号化の強度不足 不適切な権限管理 不適切な認証 暗号に関する問題 強度の不十分な乱数の使用 エラー処理 ファイルやディレクトリ情報の漏洩 危険な ActiveX メソッドの公開 不変であるべきデータの改変 (MAID) 暗号化鍵のハードコーディング	2
46	CWE-114 CWE-601 CWE-123 CWE-28 CWE-319 CWE-21 CWE-40 CWE-73 CWE-113 CWE-118 CWE-219 CWE-248 CWE-250 CWE-262 CWE-294 CWE-307 CWE-315 CWE-320 CWE-326 CWE-338 CWE-362 CWE-389 CWE-404 CWE-410 CWE-415 CWE-521 CWE-522 CWE-611 CWE-680 CWE-703 CWE-732 CWE-749 CWE-755 CWE-763 CWE-788 CWE-835	プロセスコントロール オープンリダイレクト Write-what-where 条件 パストラバーサル: '..¥filedir' センシティブな情報の平文送信 パストラバーサルと同値エラー パストラバーサル: '¥¥UNC¥share¥name¥' (Windows UNC Share) ファイル名やパス名の外部制御 HTTP レスポンス分割 範囲エラー Web ルートにセンシティブなデータ キャチされない例外 必要の無い権限による実行 パスワードに有効期限を設定しない 認証バイパス (Capture-replay) 認証処理に対する回数制限や時間制限の未設定 センシティブな情報をクッキーに平文で保存する 鍵管理エラー 不適切な暗号強度 暗号学的に強度の足りない PRNG の使用 競合状態 エラー条件、戻り値、ステータスコード 不適切なリソースのシャットダウンやリリース 不十分なリソースプール だぶるフリー 弱いパスワードを設定できる 認証情報の不十分な保護 XML 外部要素参照 ('XXE') の不適切な制限 整数オーバーフローからバッファオーバーフローにつながる 例外条件の不適切なチェックや処理 重要なリソースに不適切なパーミッションを付与する 危険なメソッドや機能の公開 例外条件の不適切な処理 無効なポインタや参照の解放 バッファの終端外のメモリへのアクセス 無限ループ	1

脆弱性の深刻度 (ICS-CERT vs. 脆弱性全般)

脆弱全般 (CVEの19,012件)とICS-CERTアドバイザリで公開された脆弱性(422件)の**CVSS Base Score**を比較する (期間: 2010年～2013年末)

	全般	ICS-CERT	備考
平均値	6.3	**7.5**	合計値の累計÷度数の累計
中央値	5.0	**6.2**	データを小さい順に並べたとき、ちょうど中央にくる値
最頻値	4.3	**10.0**	度数が一番大きい値



調査 Step 1. 「データ収集」のまとめ

ICS-CERT アドバイザリの情報から次のことが分かる

- 脆弱性の95%から該当するCWEやCVSS値を取得できた
  → 特定できなかった5%は、原因不明の権限昇格、DoS等

- バッファオーバーフロー関連の脆弱性が全体の**30%**を占める
  → C/C++で書かれたプログラムであると推測される

- 12%の脆弱性がCWE-20 “Improper Input Validation”
  → 脆弱性の根本原因がはっきりしない

- PoCや攻撃コードが公開されている脆弱性は36%と高い割合

- IT全般の脆弱性よりも脅威度が高い傾向

調査 Step 2. コーディングルールの抽出

対策に有効な CERT C コーディングルール

CERT C コーディングスタンダード(1/3)

制御システムの脆弱性(CWE)の傾向はつかめた

これらの脆弱性を作り込まないために有効な
コーディングルールは何か

CERT C コーディングスタンダード※から抽出する

CERT C コーディングスタンダードは、**脆弱性(セキュリティホール)**につながる恐れのあるコーディング作法や未定義の動作を減らすことを目的に、**CERT/CC**を中心とするC言語のエキスパート、開発コミュニティが共同でまとめたコーディング規約。

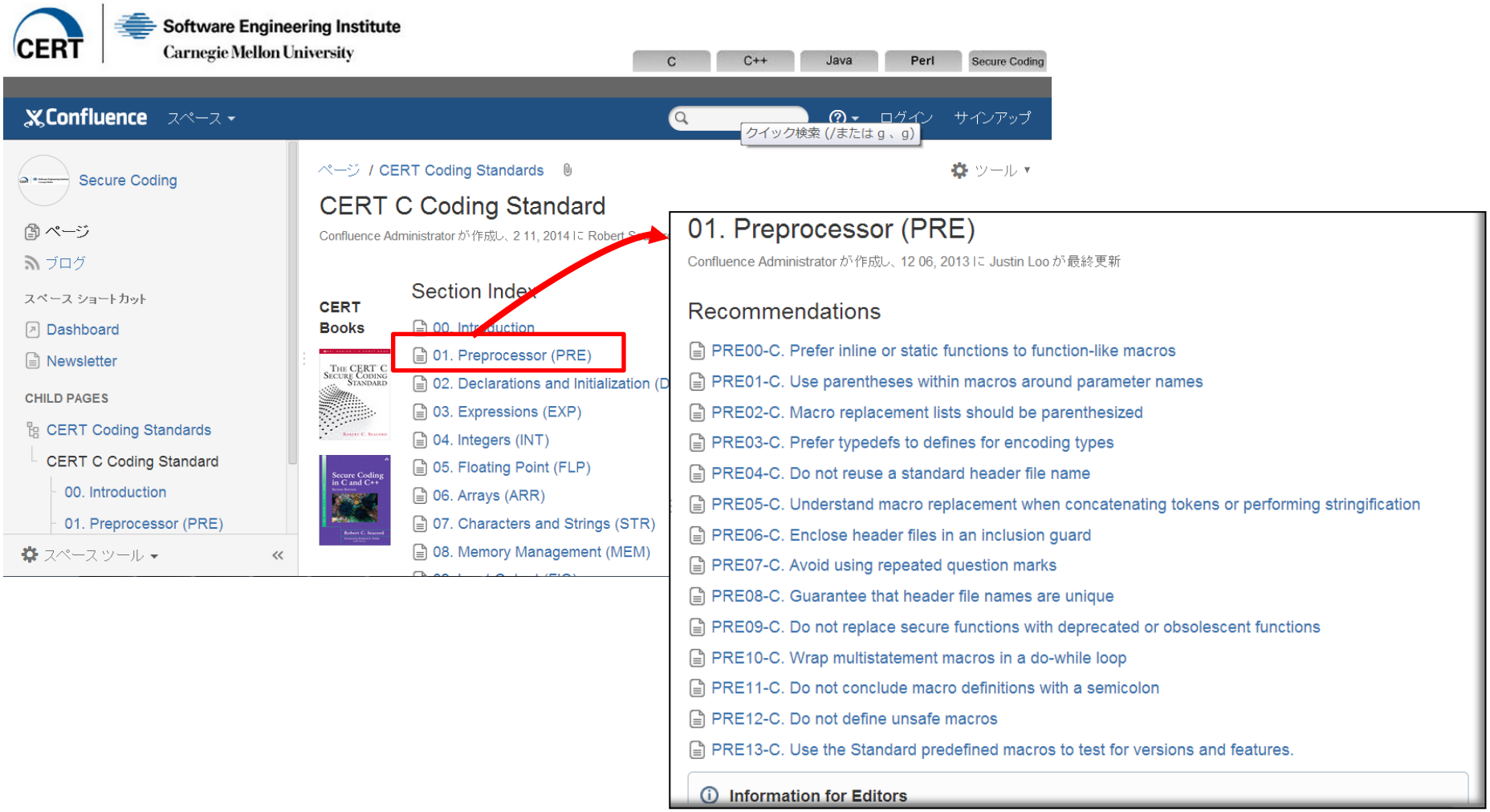
この規約は、**C言語**を使ってセキュアコーディングを行うためのルールやレコメンデーション(アドバイス)を定める。具体的には、文字列や整数、浮動小数点数の取扱いや入出力、メモリ管理等について、脆弱性を作り込まないコーディング作法を定める。

※ 旧称「CERT C セキュアコーディングスタンダード」

CERT C コーディングスタンダード(2/3)

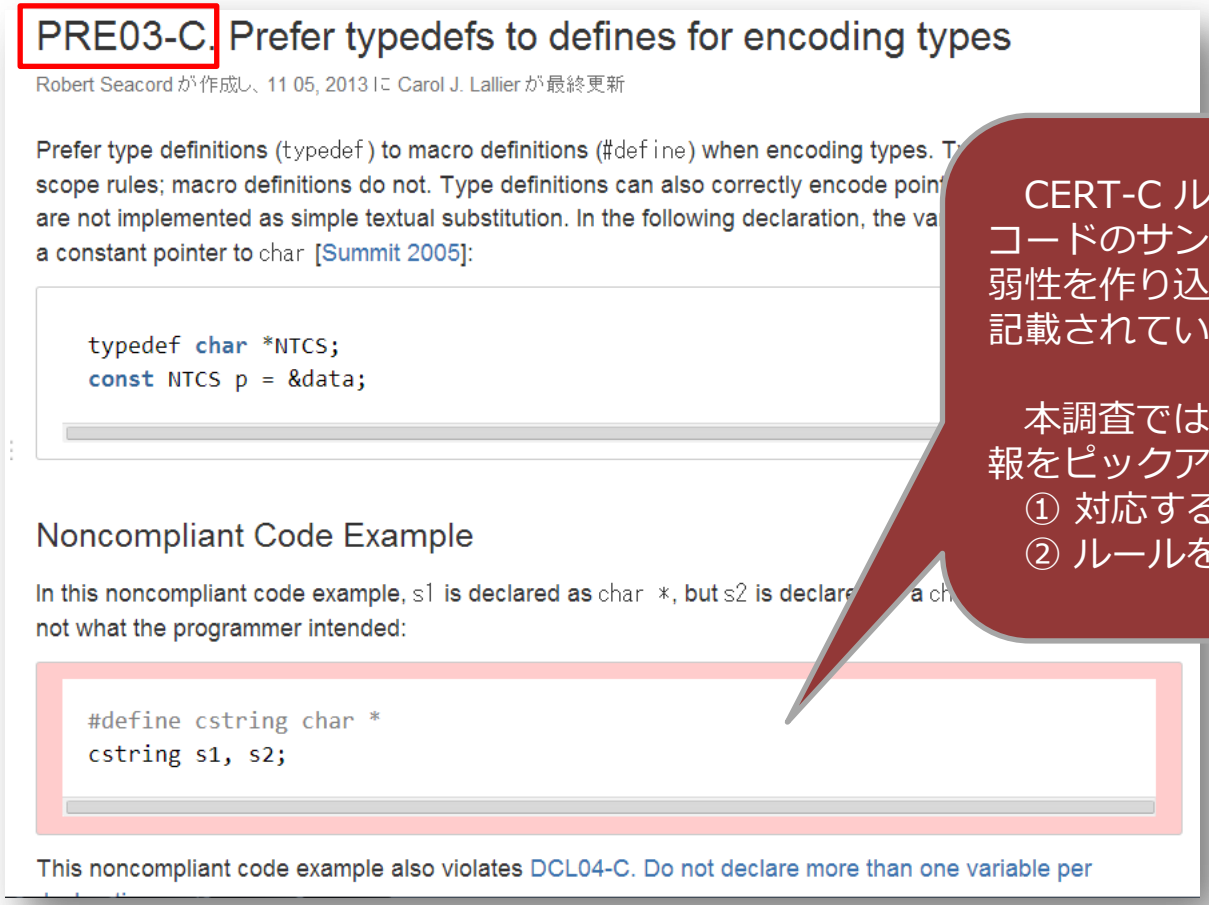
CERT C Coding Standard

– https://www.securecoding.cert.org/confluence/display/seccode/CERT+C+Coding+Standard



CERT C コーディングスタンダード(3/3)

CERT-C のルール



PRE03-C. Prefer typedefs to defines for encoding types

Robert Seacord が作成し、11 05, 2013 に Carol J. Lallier が最終更新

Prefer type definitions (typedef) to macro definitions (#define) when encoding types. T
scope rules; macro definitions do not. Type definitions can also correctly encode poin
are not implemented as simple textual substitution. In the following declaration, the va
a constant pointer to char [Summit 2005]:

```
typedef char *NTCS;
const NTCS p = &data;
```

Noncompliant Code Example

In this noncompliant code example, s1 is declared as char *, but s2 is declare a ch
not what the programmer intended:

```
#define cstring char *
cstring s1, s2;
```

This noncompliant code example also violates DCL04-C. Do not declare more than one variable per

CERT-C ルールには、違反コードや適合コードのサンプルやリスク評価など様々な脆弱性を作り込まないために遵守すべき内容が記載されている。

本調査では、各ルールから下記の2つの情報をピックアップした：

① 対応する脆弱性(CWE)

② ルールをサポートする静的解析ツール

脆弱性とコーディングルールの対応(1/2)

各CERT-Cルールの “Related Guidelines” セクションから関連するCWE(脆弱性)を抽出する。

Risk Assessment

Eliminating violations of syntax rules and other constraints can eliminate serious software vulnerabilities that can lead to the execution of arbitrary code with the permissions of the vulnerable process.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
MSC00-C	Medium	Probable	Medium	P8	L2

Related Vulnerabilities

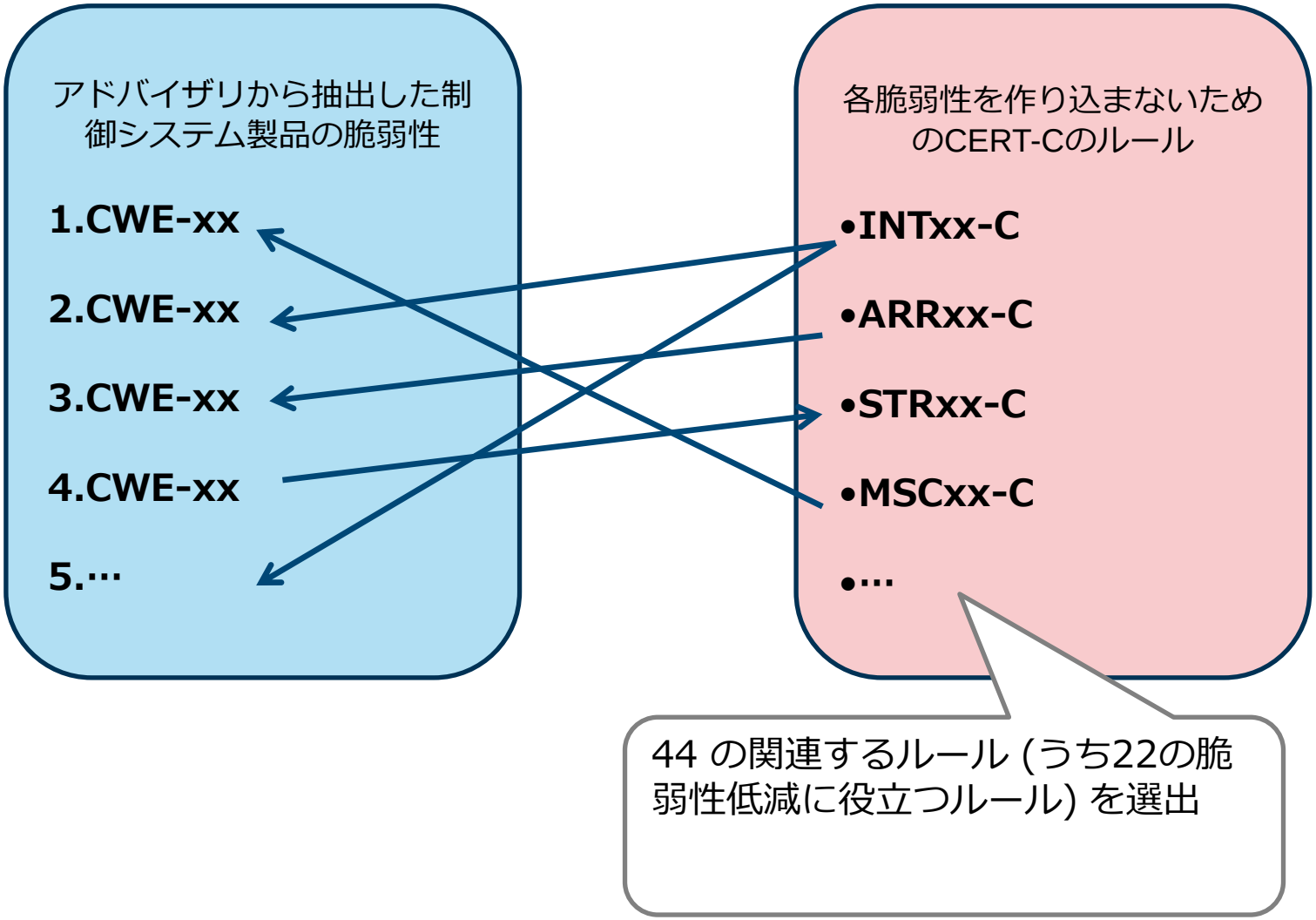
Search for vulnerabilities resulting from the violation of this rule on the CERT website.

Related Guidelines

CERT C++ Secure Coding Standard	MSC00-CPP. Compile cleanly at high warning levels
MITRE CWE	CWE-563, Unused variable CWE-570, Expression is always false CWE-571, Expression is always true

脆弱性とコーディングルールの対応(2/2)

CWEとCERT-Cの対応表を作成。



ルール抽出に関する留意点

- ICS-CERT アドバイザリで取り上げられた制御システム製品がすべてC言語を使って開発されているわけではない
  - たとえば、XSSの脆弱性がC言語のコーディングエラーで作り込まれる可能性は現実には極めて低い

- 製品の開発言語は多くの場合不明である。一方で、脆弱性の3割がC/C++の代表的コーディングエラー(CWE-119)であることが分かっている

  → 残りの7割やいかに？

- 本調査では、脆弱性がCのコーディングエラーが原因で作り込まれていたと仮定した場合にどのようなコーディングで防止防ぐことができたか、という観点に立ち、対策に有効な CERT-C コーディングルールを提示する

調査結果

制御システム用ソフトウエアの脆弱性に関連する

抽出された CERT C コーディングルール

調査した445件の制御システム用ソフトウエアの脆弱性に関連するルールは全部で44個存在する。

そのうち**22個**※1のルールが (脆弱性の低減に) 特に重要。

※1 ルールには、CERT-C への準拠に必須の rule と、それ以外の recommendation の2種類が存在する。太字の脆弱性低減に特に重要な22個のルールは、前者の rule のカテゴリに関するもの。

識別子	ルール
ARR38-C	**ライブラリ関数が無効なポインタを生成しないことを保証する**
MSC34-C	**廃止予定の関数や古い関数を使用しない**
ARR30-C	**境界外を指すポインタや配列添字を生成したり使用したりしない**
INT06-C	文字列トークンを整数に変換するには strtol() 系の関数を使う
MEM10-C	ポインタ検証関数を定義して使用する
ERR07-C	よりよいエラー検査を行える関数を使用する
STR31-C	**文字データと null 終端文字を格納するために十分な領域を確保する**
EXP33-C	**未初期化のメモリを参照しない**
EXP39-C	**適合しない型のポインタを使って変数にアクセスしない**

識別子	ルール
ARR00-C	配列の仕組みを理解する
STR32-C	**文字列は null 終端させる**
FIO37-C	**読み取られたデータが文字データであると思い込まない**
ENV01-C	環境変数のサイズについて勝手な想定をしない
FIO02-C	信頼できない情報源から取得したファイル名は正規化する
MSC18-C	プログラムコードの中でパスワードなどの機密情報を扱うときは注意する
MSC30-C	**疑似乱数の生成に rand() 関数を使用しない**
MSC32-C	**乱数生成器は適切なシード値を与えて使う**
INT35-C	**整数式をより大きなサイズの整数に対して比較や代入をする際には、事前に演算後のサイズで評価する**
INT30-C	**符号無し整数の演算結果がラップアラウンドしないようにする**
INT32-C	**符号付き整数演算がオーバーフローを引き起こさないことを保証する**
MEM07-C	calloc() の引数は乗算した結果がラップアラウンドしないようにする
MEM35-C	**オブジェクトに対して十分なメモリを割り当てる**
MSC11-C	診断テストはアサートを使って組み込む
EXP34-C	**NULL ポインタを参照しない**
ERR33-C	**標準ライブラリのエラーを検出し、対処する**
WIN04-C	Consider encrypting function pointers
MEM00-C	メモリの割り当てと解放は、同じ翻訳単位内の同一抽象レベルで行う
MEM01-C	free() した直後のポインタには新しい値を代入する

識別子	ルール
MEM30-C	**解放済みメモリにアクセスしない**
ENV03-C	外部プログラムを呼び出す際は環境を無害化する
FIO30-C	ユーザからの入力を使って書式指定文字列を組み立てない
STR02-C	複雑なサブシステムに渡すデータは無害化する
ENV33-C	**コマンドプロセッサが必要ない場合は system() を呼び出さない**
FIO01-C	ファイル名を使用してファイルを識別する関数の使用に注意する
POS02-C	最小権限の原則に従う
POS36-C	**権限は正しい順序で破棄する**
POS37-C	**権限の破棄は確実に行う**
WIN02-C	Restrict privileges when spawning child processes
FIO31-C	**同一のファイルを同時に複数回開かない**
CON08-C	Do not assume that a group of calls to independently atomic methods is atomic
POS01-C	ファイルを操作するときにはリンクが存在するかどうかを検査する
FIO22-C	Close files before spawning processes
FIO42-C	**不要になったファイルは正しくクローズする**
FIO06-C	適切なパーミッションを持つファイルを作成する

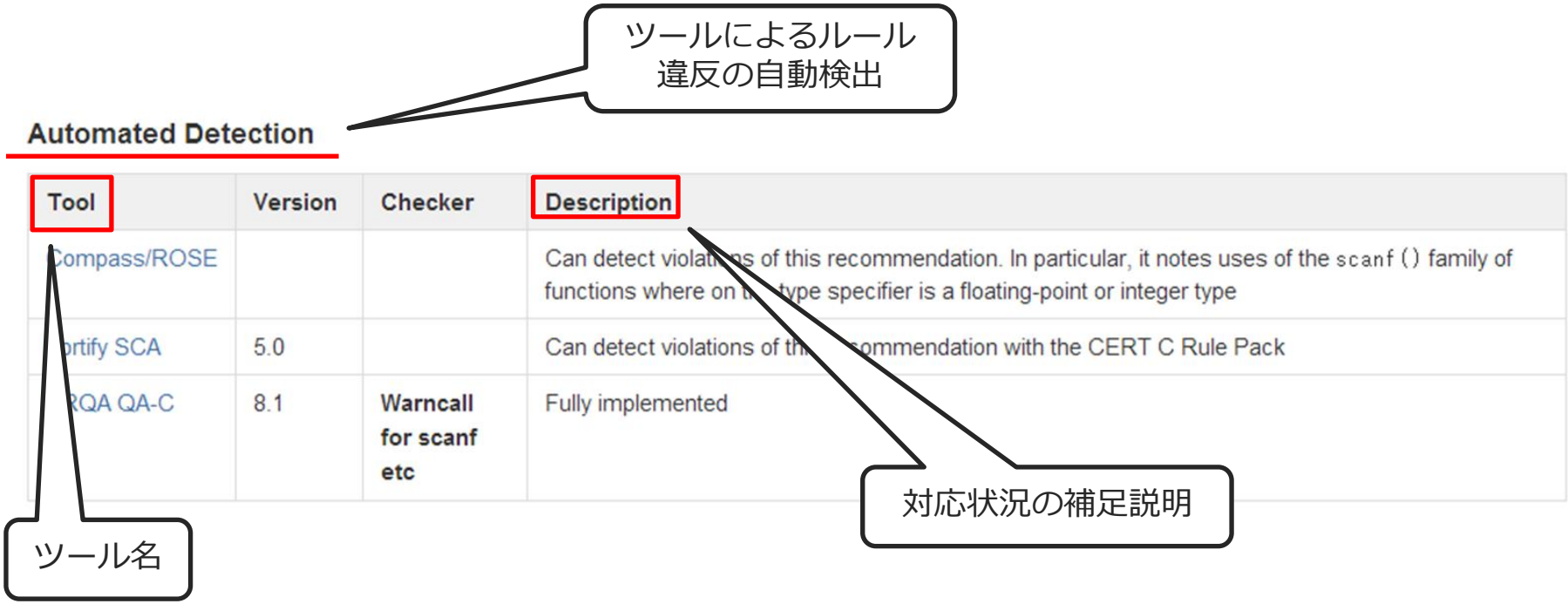
※ 本リストはJPCERT/CCのWebに公開しているのルール (2014年5月13日現在) を示す
※ ルールの識別子やタイトル、URLは変更されることがあります

コーディングルールへの準拠性をチェックできる

ソースコード解析ツールの対応状況

CERT-C をサポートするソースコード解析ツール(1)

各CERT-C ルールをサポートしているソースコード解析ツールは、
ルールの “Automated Detection” セクションにまとめられている



ツール毎のCERT-Cサポート率

ソースコード解析ツール	CERT-C全体	ICS-CERT
Compass/ROSE	40.3%	**59.1%**
LDRA tool suite	39.0%	**45.5%**
PRQA QA-C	36.7%	**36.4%**
ECLAIR	21.3%	**4.5%**
Coverity	15.7%	**40.9%**
Fortify SCA	15.7%	**40.9%**
Klocwork	12.5%	**38.6%**
Splint	8.5%	**15.9%**
GCC	7.2%	**4.5%**
EDG	1.0%	**0.0%**
ツールのサポート率	66.6% ※1	**81.8%** ※2

※1 CERT-C 全305ルール中203のルールがいずれかのツールで検出できる
※2 ICS-CERT アドバイザリで公表された脆弱性に関連する44ルール中、36ルールはツールで検出できる

調査 Step 2. 「コーディングルールの抽出」のまとめ

- 制御システム用ソフトウエアの脆弱性低減に役立つ**22のCERT-Cコーディングルール**を選出した
  - CERT Cルール全体の**7**% (22/305)
  - 多くはツールによる静的解析が可能

- 脆弱性の多くは実装段階にコーディングエラーが原因で作りこまれる問題であることから、コーディング規約や静的解析ツールを活用したセキュアコーディングが、脆弱性対策に有効であると考えられる

STEP 3.

まとめと推奨事項

まとめ

- ICS-CERT アドバイザリで取り上げられた制御システム用ソフトウエアの脆弱性は、IT全般の脆弱性よりも脅威度が高い傾向にある

- **22個のCERT-Cコーディングルール**を活用することで、これらの脆弱性を低減することができる

脆弱性低減のための推奨事項

- 本調査で特定したコーディングルールを社内のコーディング規約がカバーしているかを確認する
  → カバーしていないルールは採用を検討

- 発注時の要求事項として、ルール遵守を開発者に求める

- ソースコード解析ツールを活用している場合、ルールの違反をツールが検出可能であるかを確認する

- ソースコード解析ツールを新規導入する場合、ルールのサポート状況を考慮する

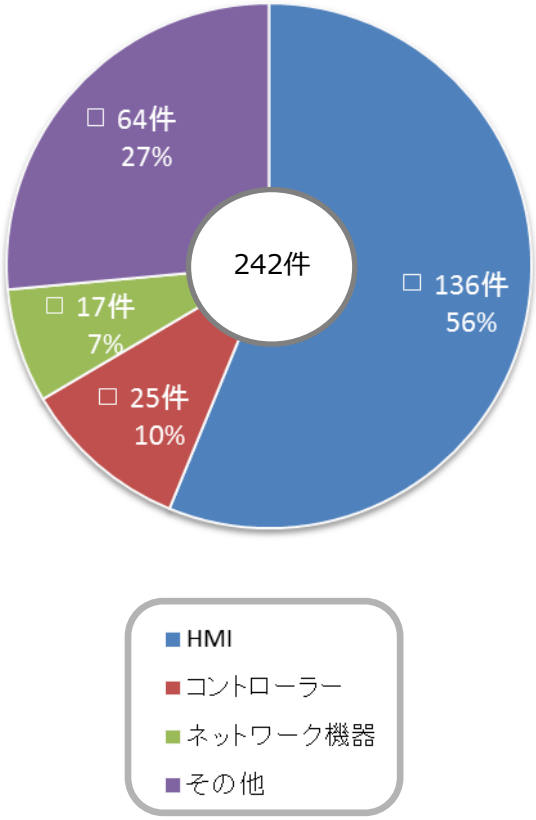
- ルールや脆弱性について学ぶ機会をプログラマに提供する

付録

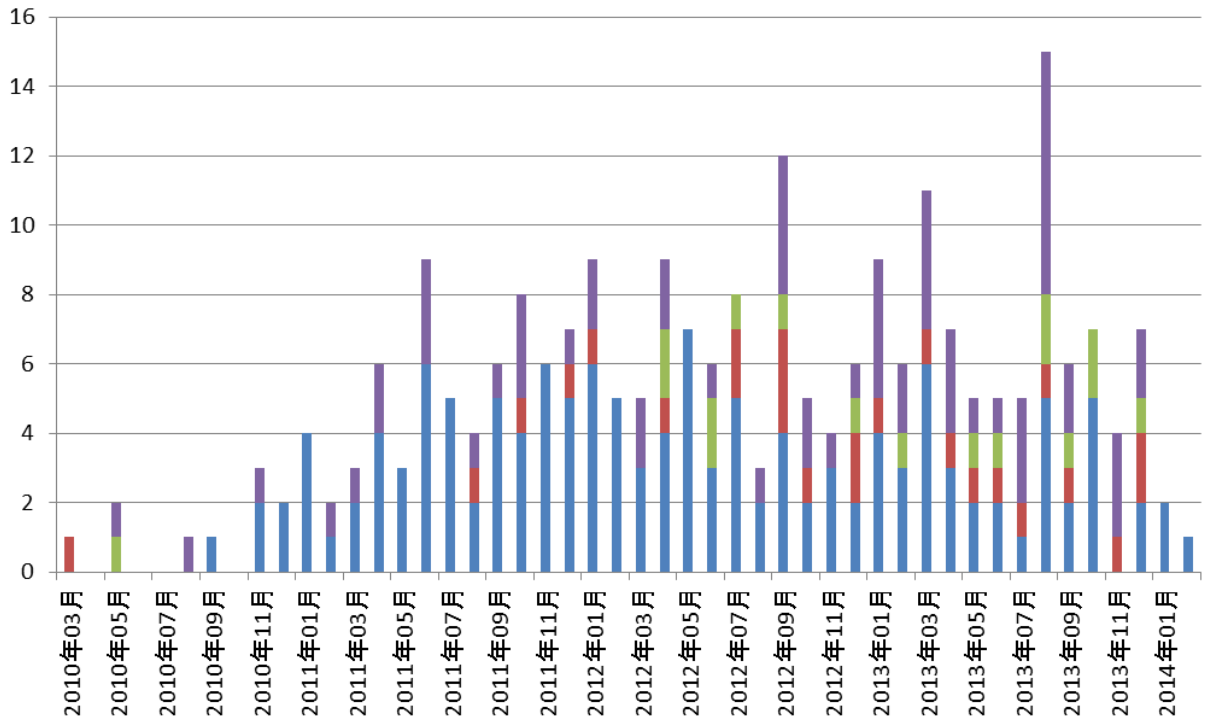
月別・種別の脆弱性公表件数

ICS-CERT アドバイザリの「BACKGROUND」などから製品種別をHMI、コントローラー、ネットワーク機器、その他の４分類とし、製品種別ごとの脆弱性の公開傾向を下図に示す

製品種別ごとの脆弱性の内訳



月別・製品種別ごとの脆弱性公開件数



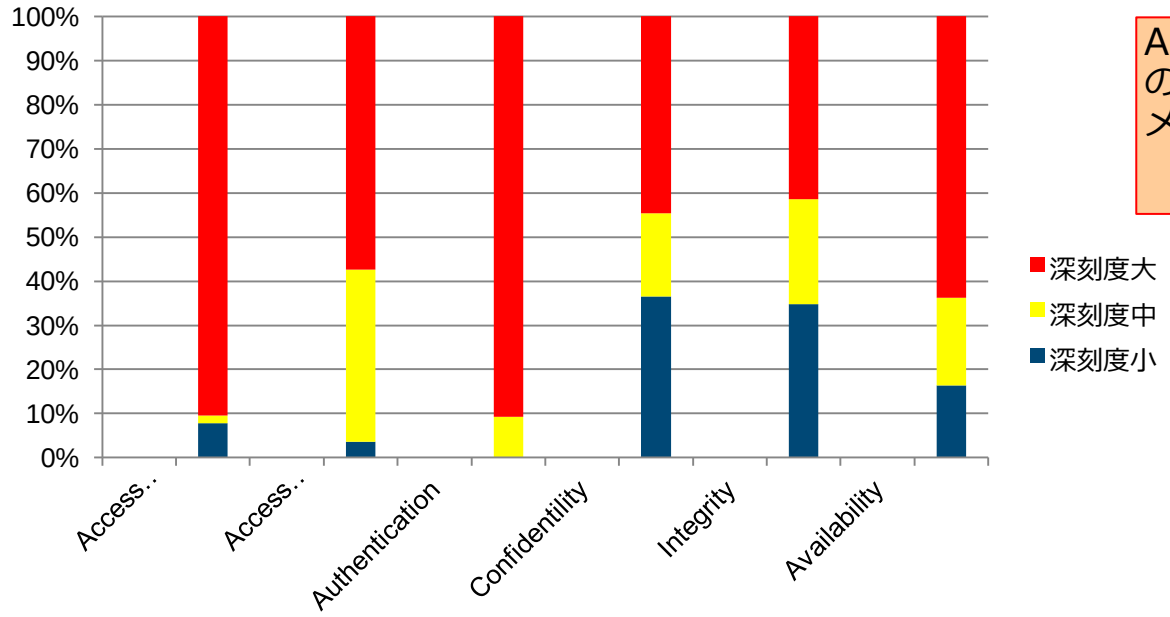
HMIの脆弱性が全体の約60%弱を占め、年月に関わらず万遍なく公表されている

ICS-CERT が公表した脆弱性の深刻度

CVSS Vector String (深刻度の度合い)

	Access Vector		Access Complexity		Authentication		Confidentility		Integrity		Availability	
深刻度小	Local	33	High	15	Multiple	0	None	154	None	147	None	69
深刻度中	Adjacent Network	7	Medium	166	Single	39	Partial	80	Partial	100	Partial	84
深刻度大	Network	384	Low	243	None	385	Complete	190	Complete	177	Complete	271

深刻度の割合



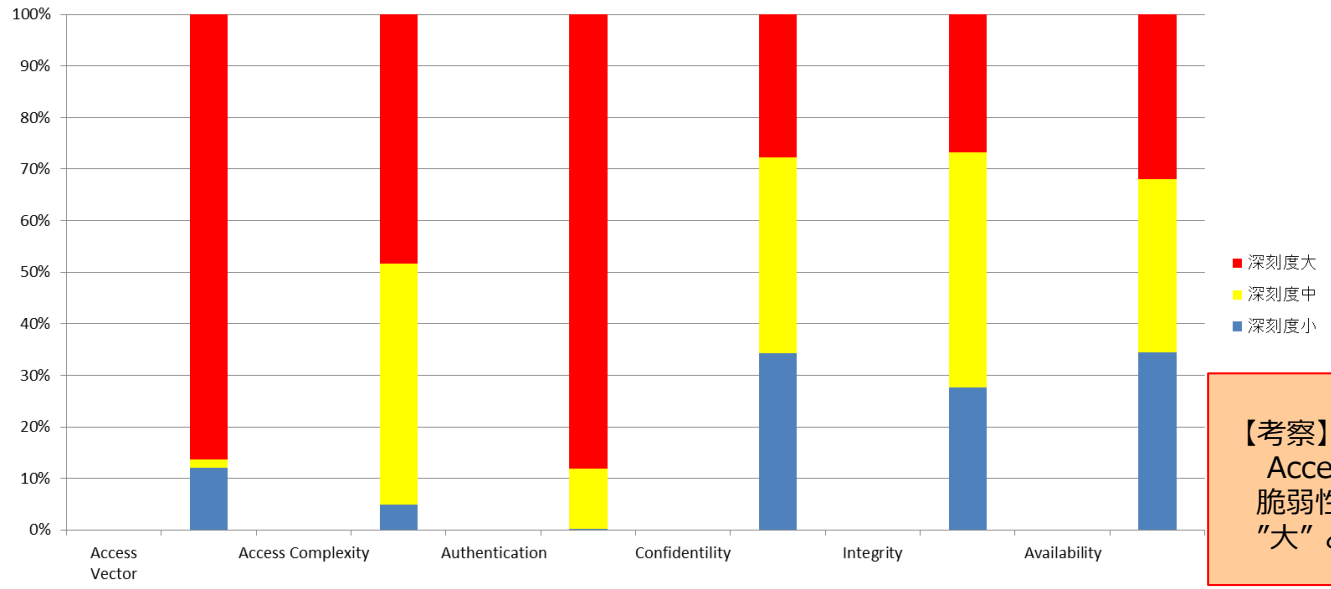
Access VectorとAuthenticationの脆弱性の深刻度が他のパラメータと比べて高い

脆弱性全般の深刻度

脆弱性全般のCVSS Vector String (深刻度の度合い)

	Access Vector		Access Complexity		Authentication		Confidentility		Integrity		Availability	
深刻度小	Local	2,288	High	949	Multiple	26	None	6,525	None	5,263	None	6,560
深刻度中	Adjacent Network	326	Medium	8,864	Single	2,230	Partial	7,207	Partial	8,651	Partial	6,376
深刻度大	Network	16,398	Low	9,199	None	16,756	Complete	5,280	Complete	5,098	Complete	6,076

深刻度の割合



【考察】
Access VectorとAuthenticationの脆弱性の深刻度が他と比較すると”大”と言える。

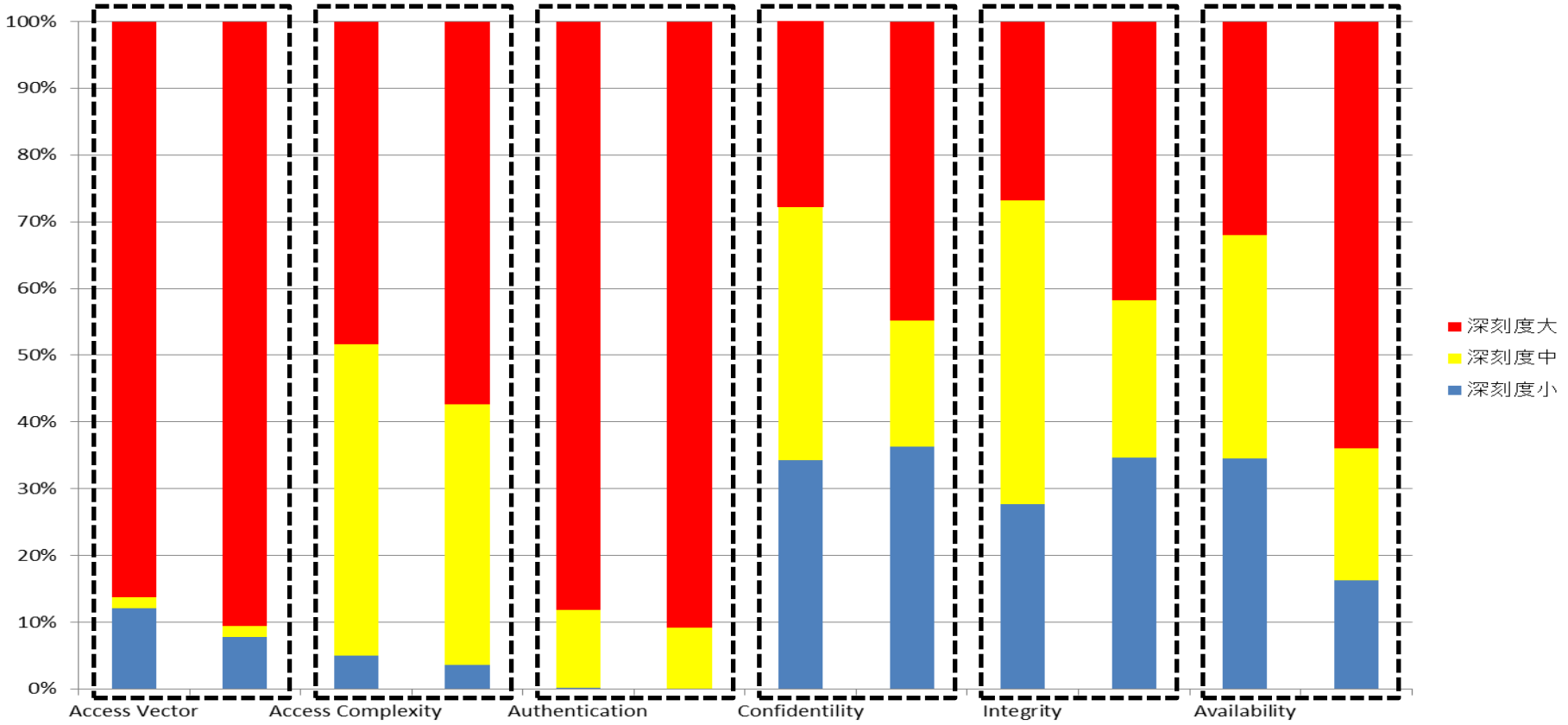
ICS製品とIT製品の脆弱性の深刻度

Vector Stringの比較

脆弱性全体 (19,012件)とICS-CERT(不明を除く422件)の**Vector String**を比較する

全体の深刻度の割合

※ 棒グラフの左側が脆弱性全般、右側がICS-CERTアドバイザリの脆弱性



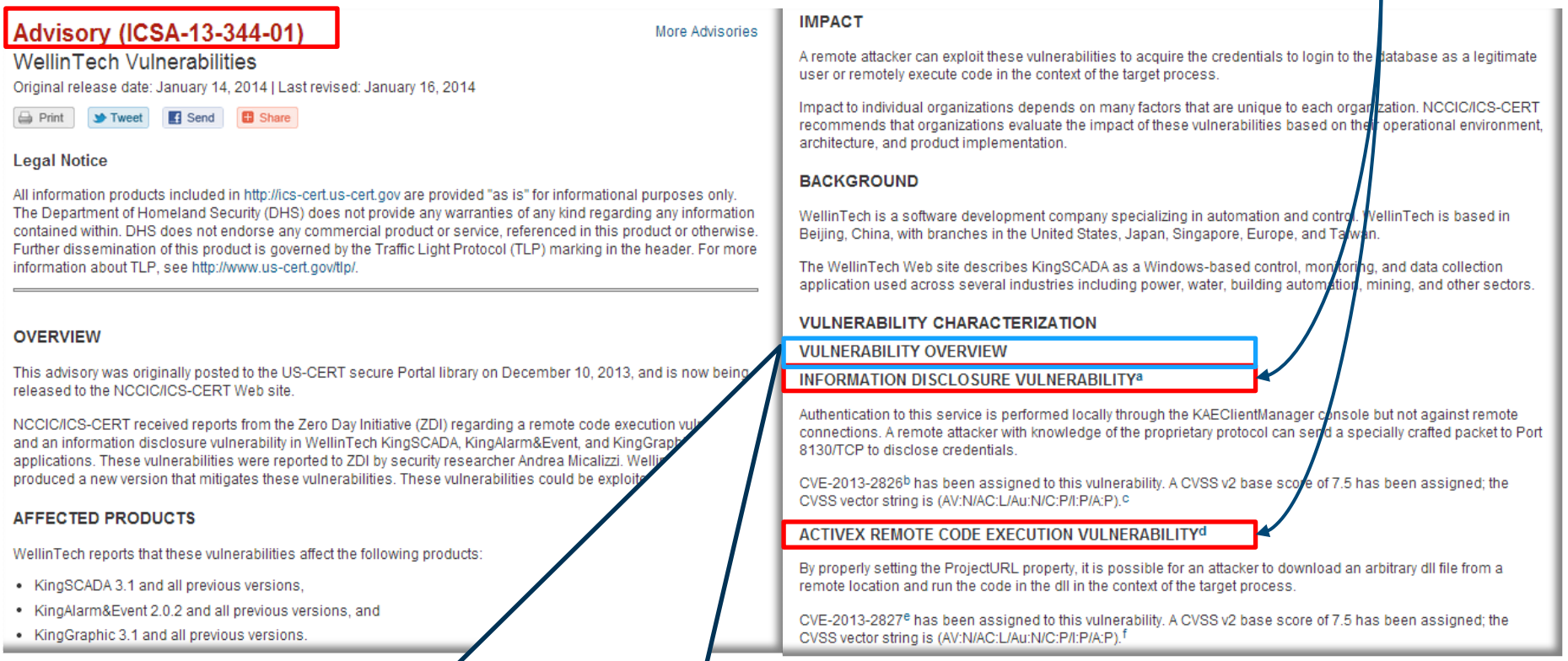
まとめ

- CVSS Base Scoreの平均値・中央値・最頻値ともにICS-CERTアドバイザリの脆弱性が脆弱性全般よりも高い傾向にある
  - ICS-CERTアドバイザリではバッファオーバーフローの脆弱性が指摘される率が高いのが原因
- CVSS Vector String は全体的にICS-CERTアドバイザリの脆弱性の方が脅威度が高い傾向にある
- 制御の分野では、一般的情報システムよりも深刻度の高い脆弱性が指摘される傾向にある

脆弱性の件数のカウント方法

アドバイザリ1件に対して、1件以上の脆弱性が存在する。
下図の例では、1件のアドバイザリに2件の脆弱性が存在している。

Vulnerability Type



Advisory (ICSA-13-344-01)

WellinTech Vulnerabilities

Original release date: January 14, 2014 | Last revised: January 16, 2014

More Advisories

Print Tweet Send Share

Legal Notice

All information products included in http://ics-cert.us-cert.gov are provided "as is" for informational purposes only. The Department of Homeland Security (DHS) does not provide any warranties of any kind regarding any information contained within. DHS does not endorse any commercial product or service, referenced in this product or otherwise. Further dissemination of this product is governed by the Traffic Light Protocol (TLP) marking in the header. For more information about TLP, see http://www.us-cert.gov/tlp/.

OVERVIEW

This advisory was originally posted to the US-CERT secure Portal library on December 10, 2013, and is now being released to the NCCIC/ICS-CERT Web site.

NCCIC/ICS-CERT received reports from the Zero Day Initiative (ZDI) regarding a remote code execution vul and an information disclosure vulnerability in WellinTech KingSCADA, KingAlarm&Event, and KingGraph applications. These vulnerabilities were reported to ZDI by security researcher Andrea Micalizzi. Wellin produced a new version that mitigates these vulnerabilities. These vulnerabilities could be exploite

AFFECTED PRODUCTS

WellinTech reports that these vulnerabilities affect the following products:

- KingSCADA 3.1 and all previous versions,
- KingAlarm&Event 2.0.2 and all previous versions, and
- KingGraphic 3.1 and all previous versions.

IMPACT

A remote attacker can exploit these vulnerabilities to acquire the credentials to login to the database as a legitimate user or remotely execute code in the context of the target process.

Impact to individual organizations depends on many factors that are unique to each organization. NCCIC/ICS-CERT recommends that organizations evaluate the impact of these vulnerabilities based on their operational environment, architecture, and product implementation.

BACKGROUND

WellinTech is a software development company specializing in automation and control. WellinTech is based in Beijing, China, with branches in the United States, Japan, Singapore, Europe, and Taiwan.

The WellinTech Web site describes KingSCADA as a Windows-based control, monitoring, and data collection application used across several industries including power, water, building automation, mining, and other sectors.

VULNERABILITY CHARACTERIZATION

VULNERABILITY OVERVIEW

INFORMATION DISCLOSURE VULNERABILITY[a]

Authentication to this service is performed locally through the KAEClientManager console but not against remote connections. A remote attacker with knowledge of the proprietary protocol can send a specially crafted packet to Port 8130/TCP to disclose credentials.

CVE-2013-2826[b] has been assigned to this vulnerability. A CVSS v2 base score of 7.5 has been assigned; the CVSS vector string is (AV:N/AC:L/Au:N/C:P/I:P/A:P).[c]

ACTIVEX REMOTE CODE EXECUTION VULNERABILITY[d]

By properly setting the ProjectURL property, it is possible for an attacker to download an arbitrary dll file from a remote location and run the code in the dll in the context of the target process.

CVE-2013-2827[e] has been assigned to this vulnerability. A CVSS v2 base score of 7.5 has been assigned; the CVSS vector string is (AV:N/AC:L/Au:N/C:P/I:P/A:P).[f]

昔のアドバイザリ(2012年以前）は現在ほどアドバイザリのフォーマットが形式化されておらず、”VULNERABILITY OVERVIEW” のセクションがしない。そのため、アドバイザリの全文を目視確認して脆弱性をカウントする必要があった。脆弱性は後述するCWEと紐付けた。

アドバイザリから抽出するデータ

ICS-CERT アドバイザリの脚注等に記載されているCWE,CVE,CVSSの情報を本調査では取得する

Advisoryの一部抜粋（脚注)

a. ICS-ALERT-13-256-01 WellinTech KingView ActiveX Vulnerabilities, http://ics-cert.us-cert.gov/alerts/ICS-ALERT-13-256-01, Web site last accessed October 22, 2013.

b. CWE-40: Path Traversal: '\\UNC\share\name\' (Windows UNC Share), http://cwe.mitre.org/data/definitions/40.html, Web site last accessed October 22, 2013.

c. NVD, http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-6127, NIST uses this advisory to create the CVE Web site report. This Web site will be active sometime after publication of this advisory.

d. CVSS Calculator, http://nvd.nist.gov/cvss.cfm?version=2&vector=AV:N/AC:M/Au:N/C:N/I:P/A:P, Web site last visited October 22, 2013.

e. CWE-28: Path Traversal: '..\filedir', http://cwe.mitre.org/data/definitions/28.html, Web site last accessed October 22, 2013.

f. NVD, http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-6128, NIST uses this advisory to create the CVE Web site report. This Web site will be active sometime after publication of this advisory.

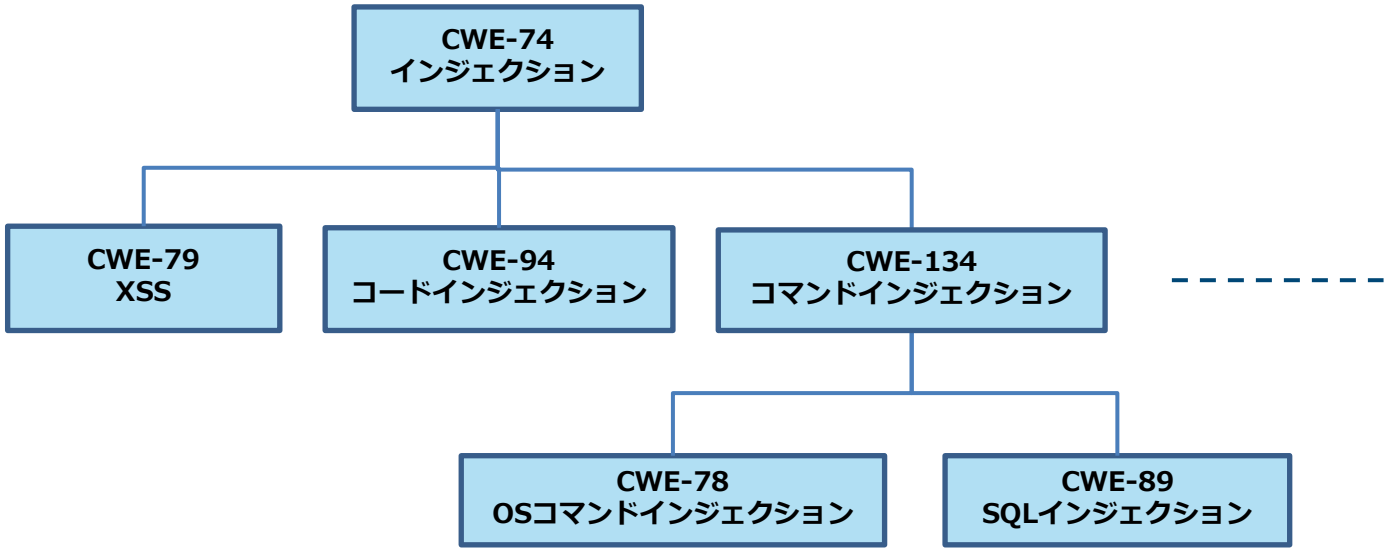
g. CVSS Calculator, http://nvd.nist.gov/cvss.cfm?version=2&vector=AV:N/AC:M/Au:N/C:N/I:P/A:P, Web site last visited October 22, 2013.

CWEとは？(1/2)

CWE (Common Weakness Enumeration) とは？

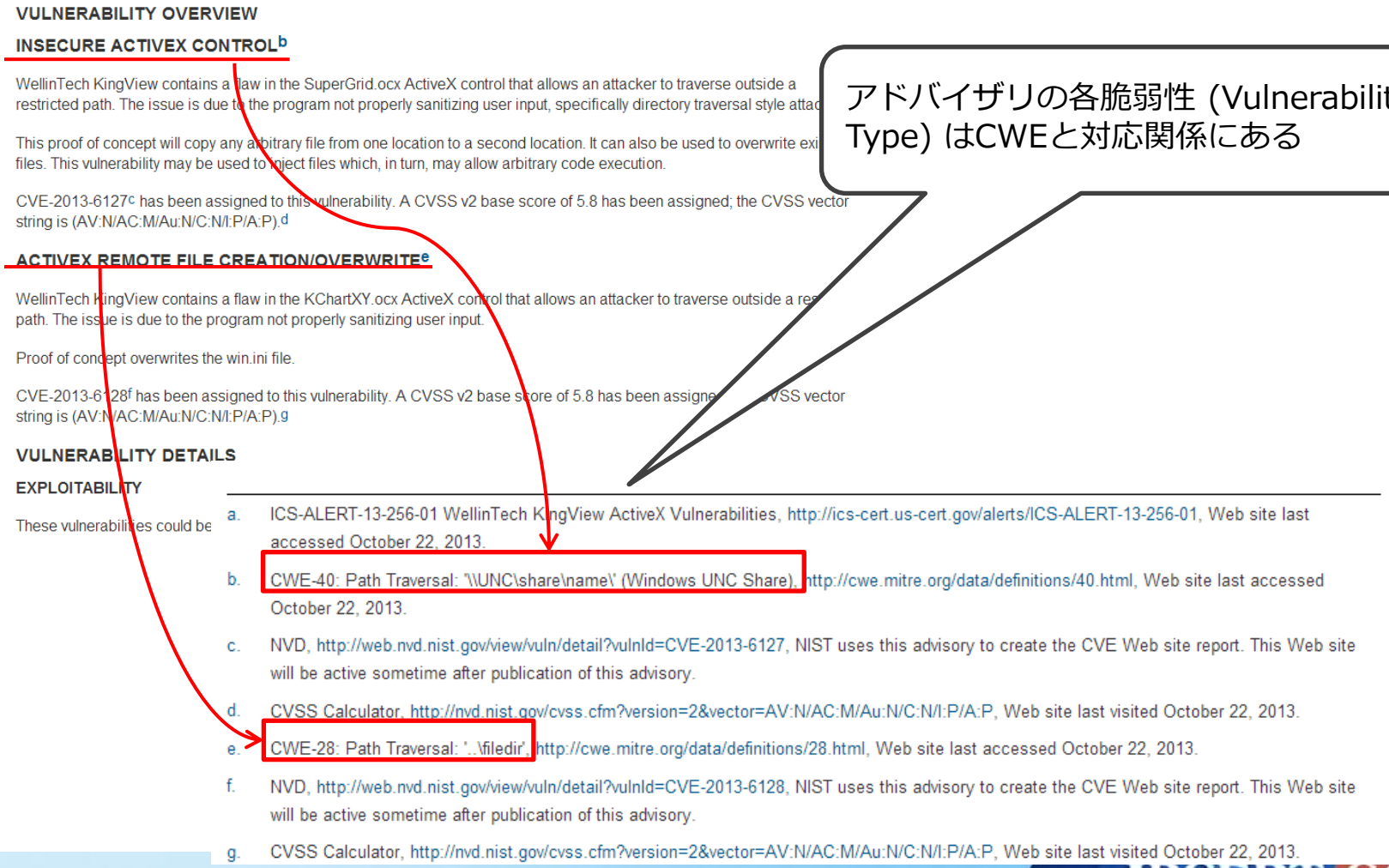
CWE(共通脆弱性タイプ一覧)とは、ソフトウエアにおけるセキュリティ上の弱点（脆弱性）の種類を体系化し、共通の脆弱性を表現するためのラベルを提供する体系。

下図はインジェクションの階層構造の例



CWEとは？(2/2)

アドバイザリの”VULNERABILITY OVERVIEW”に記載されている各脆弱性(Vulnerability Type)はCWEに関連付けられているため、どのような種類の脆弱性が製品に作り込まれたのか分かる。



VULNERABILITY OVERVIEW

INSECURE ACTIVEX CONTROL[b]

WellinTech KingView contains a flaw in the SuperGrid.ocx ActiveX control that allows an attacker to traverse outside a restricted path. The issue is due to the program not properly sanitizing user input, specifically directory traversal style attac

This proof of concept will copy any arbitrary file from one location to a second location. It can also be used to overwrite exi files. This vulnerability may be used to inject files which, in turn, may allow arbitrary code execution.

CVE-2013-6127[c] has been assigned to this vulnerability. A CVSS v2 base score of 5.8 has been assigned; the CVSS vector string is (AV:N/AC:M/Au:N/C:N/I:P/A:P).[d]

ACTIVEX REMOTE FILE CREATION/OVERWRITE[e]

WellinTech KingView contains a flaw in the KChartXY.ocx ActiveX control that allows an attacker to traverse outside a re path. The issue is due to the program not properly sanitizing user input.

Proof of concept overwrites the win.ini file.

CVE-2013-6128[f] has been assigned to this vulnerability. A CVSS v2 base score of 5.8 has been assigne VSS vector string is (AV:N/AC:M/Au:N/C:N/I:P/A:P).[g]

VULNERABILITY DETAILS

EXPLOITABILITY

These vulnerabilities could be

アドバイザリの各脆弱性 (Vulnerability Type) はCWEと対応関係にある

a. ICS-ALERT-13-256-01 WellinTech KingView ActiveX Vulnerabilities, http://ics-cert.us-cert.gov/alerts/ICS-ALERT-13-256-01, Web site last accessed October 22, 2013.

b. CWE-40: Path Traversal: '\\UNC\share\name\' (Windows UNC Share), http://cwe.mitre.org/data/definitions/40.html, Web site last accessed October 22, 2013.

c. NVD, http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-6127, NIST uses this advisory to create the CVE Web site report. This Web site will be active sometime after publication of this advisory.

d. CVSS Calculator, http://nvd.nist.gov/cvss.cfm?version=2&vector=AV:N/AC:M/Au:N/C:N/I:P/A:P, Web site last visited October 22, 2013.

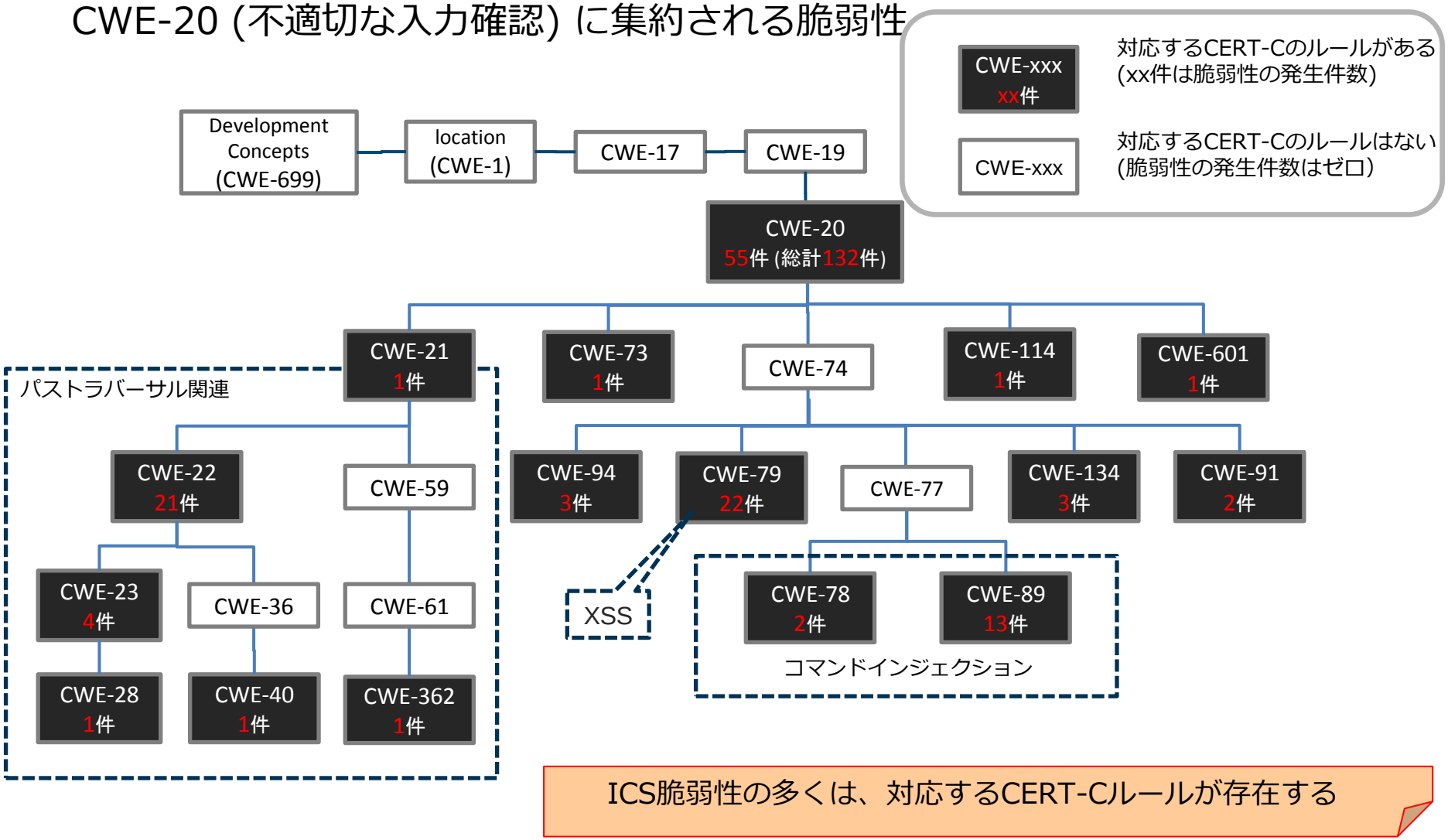
e. CWE-28: Path Traversal: '..\filedir', http://cwe.mitre.org/data/definitions/28.html, Web site last accessed October 22, 2013.

f. NVD, http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-6128, NIST uses this advisory to create the CVE Web site report. This Web site will be active sometime after publication of this advisory.

g. CVSS Calculator, http://nvd.nist.gov/cvss.cfm?version=2&vector=AV:N/AC:M/Au:N/C:N/I:P/A:P, Web site last visited October 22, 2013.

CWE-20 に属する脆弱性

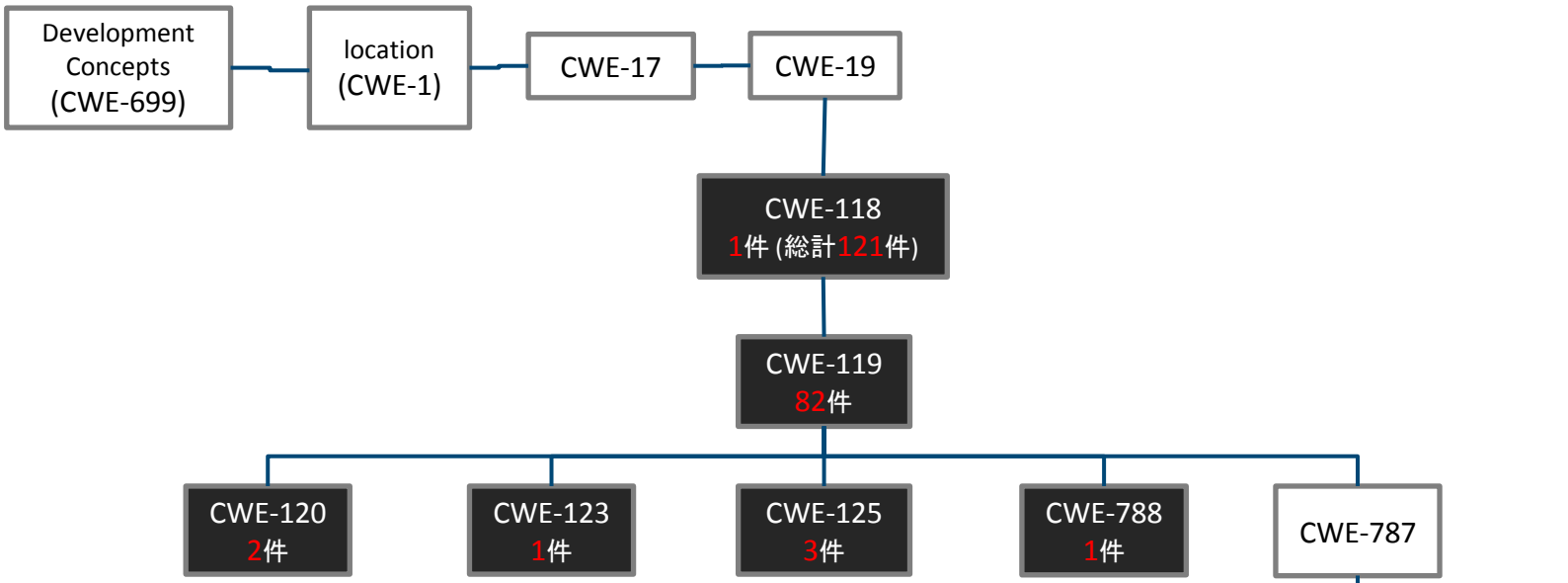
CWE-20 (不適切な入力確認) に集約される脆弱性



ICS脆弱性の多くは、対応するCERT-Cルールが存在する

CWE-118に属する脆弱性

CWE-118 (範囲エラー) に集約される脆弱性

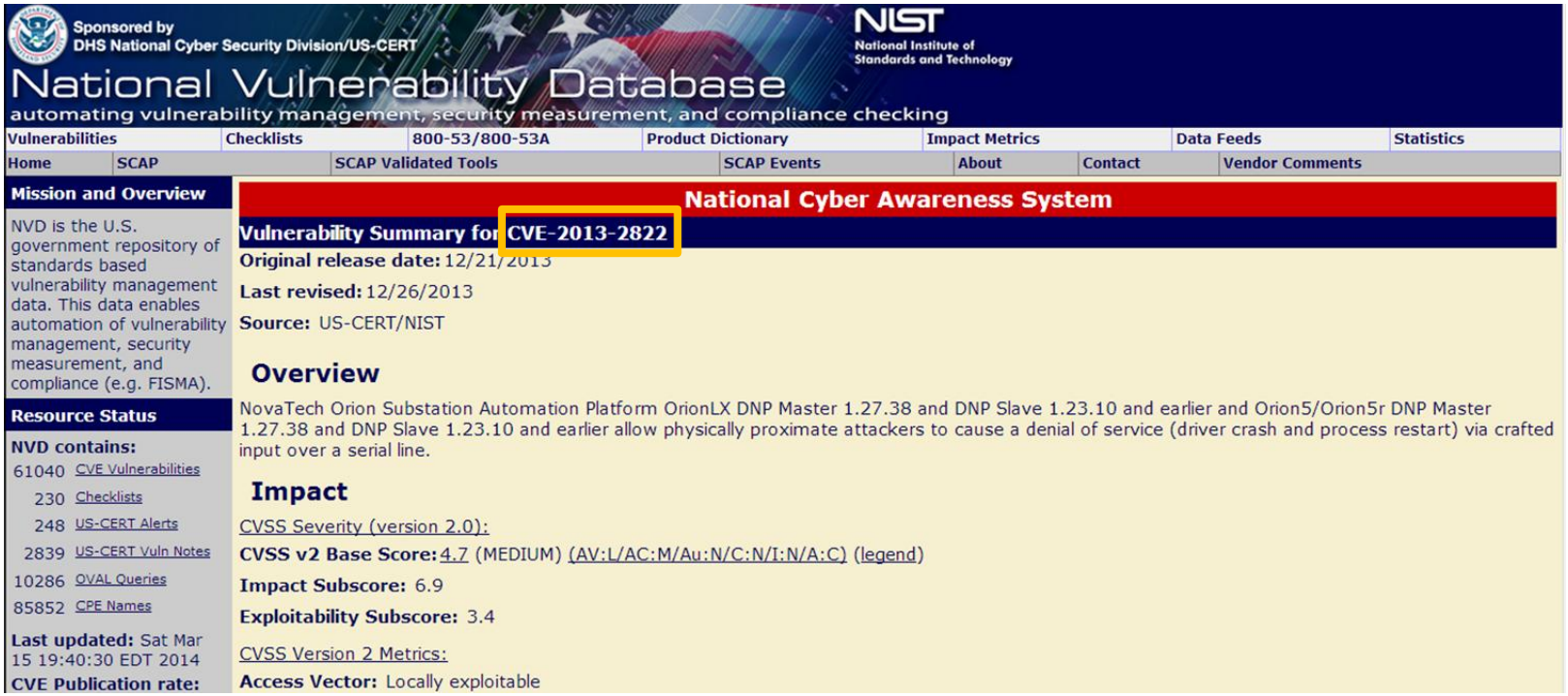


CWE-118 範囲エラー
CWE-119 メモリバッファの範囲内での処理が適切に制限されない
CWE-120 古典的バッファオーバーフロー
CWE-121 スタックバッファオーバーフロー
CWE-122 ヒープバッファオーバーフロー
CWE-123 任意の値を任意のメモリに書込める条件
CWE-125 領域外の読取り
CWE-787 領域外への書込み
CWE-788 バッファの終端を超えたメモリへのアクセス

CVEとは？

CVE (Common Vulnerabilities and Exposures)**とは？**

CVE(共通脆弱性識別子)は、個別製品に作り込まれた脆弱性を一意に特定するための識別子。米国政府の支援を受ける非営利団体MITRE社が管理している。このCVE識別番号は「CVE-西暦-連番」で構成される。
ICS-CERT アドバイザリ中にCWE,CVE,CVSS等の記載がない場合であっても、NVDからこれらの情報を取得できる場合もある。NVD(National Vulnerability Database) はNIST(米国国立標準技術研究所)が管理する脆弱性情報データベースである。下図はNVDの「CVE-2013-282」の情報。



CVSSとは？(1/2)

CVSSとは？

CVSS(Common Vulnerability Scoring System)は情報システムの脆弱性を定量的に評価する仕組みの1つであり、特定のベンダーに依存しない共通の評価方法として広く利用されている。2007年8月20日にCVSS v2がされ、NVDやICS-CERTアドバイザリ等ではCVSS v2のスコアが掲載されている。ICS-CERT アドバイザリには、CVSSの一部であるBase ScoreとVector Stringが脆弱性の特徴を記述するための指標として用いられている。

– Base Score
情報システムに求められる3つのセキュリティ特性、「機密性（ Confidentiality Impact ）」、「完全性(Integrity Impact ）」、「可用性(Availability Impact ）」に対する影響を、ネットワークから攻撃可能かどうかといった基準で評価し、CVSS基本値(Base Score)が算出される。
Base Scoreは脆弱性に固有であり、時間の経過や利用環境の影響を受けない。
0.0~10.0の数値で表現され、高い数値ほどより深刻度が高い。
ベンダーや脆弱性を公表する組織などは、脆弱性固有の深刻度を表すためにこの値を用いることが多い。

Base Scoreと深刻度の関係は下図の通り。

深刻度	Base Score
レベルⅢ(危険)	7.0~10.0
レベルⅡ (警告)	4.0~6.9
レベルⅠ (注意)	0.0~3.9

CVSSとは？(2/2)

CVSSとは？

– Vector String

下表の6つのBase Metricsから構成され、これら6つのパラメータから Base Score が算出される

Base Metrics		説明
AV	攻撃元区分 (Access Vector)	脆弱性のあるシステムをどこから攻撃可能であるかを評価
AC	攻撃条件の複雑さ (Access Complexity)	脆弱性のあるシステムを攻撃する際に必要な条件の複雑さを評価
Au	攻撃前の認証要否 (Authentication)	脆弱性を攻撃するために対象システムの認証が必要であるかを評価
C	機密性への影響 （情報漏えいの可能性、Confidentiality Impact）	脆弱性を攻撃された際に、対象システム内の機密情報が漏えいする可能性を評価
I	完全性への影響 （情報改ざんの可能性、Integrity Impact ）	脆弱性を攻撃された際に、対象システム内の情報が改ざんされる可能性を評価
A	可用性への影響 （業務停止の可能性、Availability Impact ）	脆弱性を攻撃された際に、対象システム内の業務が遅延・停止する可能性を評価

参考URL

資料	URL
CERT C Coding Standard	https://www.securecoding.cert.org/confluence/display/seccode/CERT+C+Coding+Standard
CERT C Coding Standard 日本語版	https://www.jpcert.or.jp/sc-rules/
ICS-CERT Advisories	http://ics-cert.us-cert.gov/advisories
CWE Development Concepts	http://cwe.mitre.org/data/graphs/699.html
CVSS v2	http://www.first.org/cvss/cvss-guide
CVE	http://cve.mitre.org/
NVD - Data Feeds	http://nvd.nist.gov/download.cfm#CVE_FEED

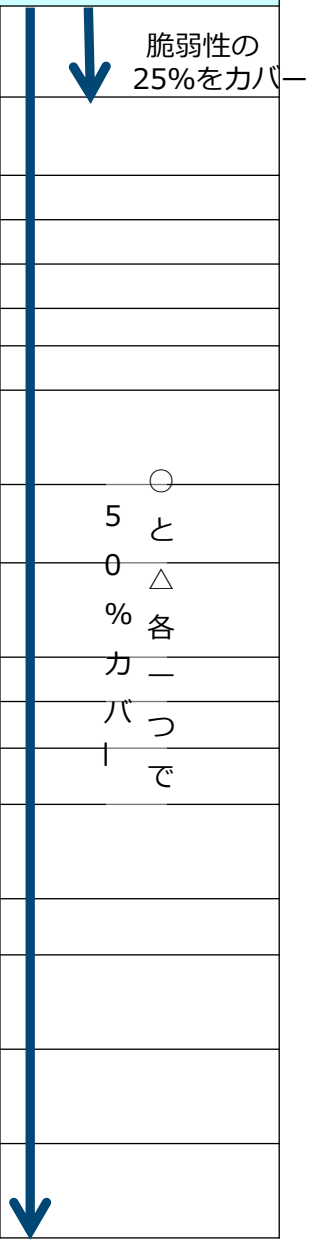
CERT C コーディングルールールーと脆弱性の対応一覧

脆弱性の半数に2つのルールが関連

ARR38-C と **MSC24-C** の2つのルールが50%の脆弱性に関連づけられる

ルール	対応する脆弱性	
ARR38-C　ライブラリ関数が無効なポインタを生成しないことを保証する		
	CWE-119	バッファの境界エラー
	CWE-121	スタックバッファオーバーフロー
	CWE-123	Write-what-where Condition
	CWE-125	境界外読み込み
MSC24-C　廃止予定の関数や古い関数を使用しない		
	CWE-20	不適切な入力確認
	CWE-73	ファイル名やパス名の外部制御
	CWE-79	クロスサイトスクリプティング
	CWE-89	SQL インジェクション
	CWE-91	XML インジェクション (Blind XPath Injection)
	CWE-94	コードインジェクション
	CWE-114	プロセスの制御 (Process Control)
	CWE-120	古典的バッファオーバーフロー
	CWE-601	オープンリダイレクト

順位	CERT-C のルール	ルールのタイトル	脆弱性の件数	対応する脆弱性	カバー率
1	ARR38-C	Guarantee that library functions do not form invalid pointers	86	CWE-119,CWE-123,CWE-125	○
			19	CWE-121	
2	MSC24-C	Do not use deprecated or obsolescent functions	97	CWE-20,CWE-79,CWE-89,CWE-91,CWE-94,CWE-114,CWE-601	△
			2	CWE-120	
			1	CWE-73	
3	ARR30-C	Do not form or use out-of-bounds pointers or array subscripts	86	CWE-119,CWE-123,CWE-125	○
			12	CWE-122	
			1	CWE-788	
4	INT06-C	Use strtol() or a related function to convert a string token to an integer	97	CWE-20,CWE-79,CWE-89,CWE-91,CWE-94,CWE-114,CWE-601	△
	MEM10-C	Define and use a pointer validation function	97	CWE-20,CWE-79,CWE-89,CWE-91,CWE-94,CWE-114,CWE-601	△
	ERR07-C	Prefer functions that support error checking over equivalent functions that don't	97	CWE-20,CWE-79,CWE-89,CWE-91,CWE-94,CWE-114,CWE-601	△
7	STR31-C	Guarantee that storage for strings has sufficient space for character data and the null terminator	86	CWE-119,CWE-123,CWE-125	○
			2	CWE-120	
8	EXP33-C	Do not read uninitialized memory	86	CWE-119,CWE-123,CWE-125	
	EXP39-C	Do not access a variable through a pointer of an incompatible type	86	CWE-119,CWE-123,CWE-125	○
	ARR00-C	Understand how arrays work	86	CWE-119,CWE-123,CWE-125	○
	STR32-C	Do not pass a non-null-terminated character sequence to a library function that expects a string	86	CWE-119,CWE-123,CWE-125	○
	FIO37-C	Do not assume that fgets() or fgetws() returns a nonempty string when successful	86	CWE-119,CWE-123,CWE-125	○
	ENV01-C	Do not make assumptions about the size of an environment variable	86	CWE-119,CWE-123,CWE-125	○



順位	CERT-C のルール	ルールのタイトル	脆弱性の件数	脆弱性の種類	カバー率	
14	FIO02-C	Canonicalize path names originating from tainted sources	26	CWE-22,CWE-23,CWE-28	▲	
			1	CWE-40		
			1	CWE-73		
15	MSC18-C	Be careful while handling sensitive data, such as passwords, in program code	14	CWE-259,CWE-321,CWE-798	▲	
			5	CWE-311		
			5	CWE-311,CWE-319		
			2	CWE-261		
			1	CWE-326		
16	MSC30-C	Do not use the rand() function for generating pseudorandom numbers	5	CWE-331		
			2	CWE-330		
			1	CWE-338		
	MSC32-C	Properly seed pseudorandom number generators	5	CWE-331		
			2	CWE-330		
			1	CWE-338		
18	INT18-C	Evaluate integer expressions in a larger size before comparing or assigning to that size	5	CWE-190		
	INT30-C	Ensure that unsigned integer operations do not wrap	5	CWE-190		
	INT32-C	Ensure that operations on signed integers do not result in overflow	5	CWE-190		
	MEM07-C	Ensure that the arguments to calloc(), when multiplied, do not wrap	5	CWE-190		
	MEM35-C	Allocate sufficient memory for an object	5	CWE-190		
	MSC11-C	Incorporate diagnostic tests using assertions	5	CWE-190		
	EXP34-C	Do not dereference null pointers	5	CWE-476		
	ERR33-C	Detect and handle standard library errors	5	CWE-476		
	WIN04-C	Consider encrypting function pointers	5	CWE-311,CWE-319		

ここまで含めると、脆弱性の60%をカバーできる

順位	CERT-C のルール	ルールのタイトル	脆弱性の件数	対応する脆弱性	カバー率	
27	MEM00-C	Allocate and free memory in the same module, at the same level of abstraction	3	CWE-416		
			1	CWE-415		
	MEM01-C	Store a new value in pointers immediately after free()	3	CWE-416		
			1	CWE-415		
	MEM30-C	Do not access freed memory	3	CWE-416		
			1	CWE-415		
	ENV03-C	Sanitize the environment when invoking external programs	2	CWE-78		
			2	CWE-426,CWE-471		
31	FIO30-C	Exclude user input from format strings	3	CWE-134		
32	STR02-C	Sanitize data passed to complex subsystems	2	CWE-78		
	ENV33-C	Do not call system()	2	CWE-78		
34	FIO01-C	Be careful using functions that use file names for identification	1	CWE-73		
	POS02-C	Follow the principle of least privilege	1	CWE-250		
	POS36-C	Observe correct revocation order while relinquishing privileges	1	CWE-250		
	POS37-C	Ensure that privilege relinquishment is successful	1	CWE-250		
	WIN02-C	Restrict privileges when spawning child processes	1	CWE-250		
	FIO31-C	Do not open a file that is already open	1	CWE-362		
	CON08-C	Do not assume that a group of calls to independently atomic methods is atomic	1	CWE-362		
	POS01-C	Check for the existence of links when dealing with files	1	CWE-362		
	FIO22-C	Close files before spawning processes	1	CWE-404		
	FIO42-C	Close files when they are no longer needed	1	CWE-404		
	FIO06-C	Create files with appropriate access permissions	1	CWE-732		