脅威カード補足資料

CAPEC ID	タイトル	関連する項目	概要
	CAPECとは？		効果的なサイバーセキュリティを実現するためには、攻撃者がどのように行動するのかを理解することが不可欠です。CAPEC™は、アナリスト、開発者、テスター、および教育者が理解を深め、サイバー防御を強化するために使用できる、既知の攻撃の包括的な辞書および分類です。
	既存機能の悪用		攻撃者はアプリケーションの1つまたは複数の機能を悪用します。これにより、アプリケーションが意図しない悪意のある動作を実行します。または、標的の機能に影響を与えるまでリソースを枯渇させます。攻撃者が意図しない方法で機能を悪用し、アプリケーションの動作や情報の完全性に影響を及ぼす、幅広い種類の攻撃です。攻撃の結果は、情報漏洩、破壊、サービス拒否、ターゲットマシン上での任意のコードの実行などがあります。
113	APIの操作	既存機能の悪用	攻撃者は、アプリケーションプログラミングインターフェイス（API）の機能を悪用し、APIを実装しているシステムを攻撃します。意図しない機能が実行され、APIを実装するシステムを侵害します。APIの操作には予期または意図しない使用など、さまざまな形があります。例えば、攻撃者は不正なデータ検証を行う非標準APIを利用するアプリケーションに要求を行います。メタキャラクタや代替エンコーディングを入力することで操作できるかもしれません。その結果、SQLインジェクション、クロスサイトスクリプティング、コマンド実行など、さまざまな攻撃が可能となります。もう1つの例は、本番アプリケーションで無効にすべきAPIが無効になっていないため、攻撃者が本番環境でセキュリティ侵害を引き起こす危険な機能を実行することです。
113 121	テストAPIの悪用	APIの操作	攻撃者はデフォルトでは安全ではなく、本番システムに常駐するべきではないサンプル、デモンストレーション、テストAPIを悪用します。一部のアプリケーションには、管理者がドメインをテストして調整できるようにするためのAPIが含まれています。通常、システムの実稼働環境ではこれらのAPIを無効にする必要があります。テストAPIには、管理者を支援するための多くの診断情報が公開されている可能性があります。攻撃者が攻撃の洗練化のため、これらの情報を使用することもできます。テストAPIは本番環境での使用を目的としたものではないため、適切なセキュリティ制御が存在しない可能性があります。このため、攻撃者がセキュリティ侵害可能な多くの欠陥や脆弱性が含まれる可能性があります。
113 133	すべての共通スイッチを試す	APIの操作	攻撃者はターゲットの弱点を発見するため、ターゲットアプリケーションのすべての共通スイッチとオプションを呼び出そうとします。たとえば、一部のアプリケーションでは--debugスイッチを追加するとデバッグ情報が表示されます。場合によっては重要な情報の処理や設定情報が攻撃者に表示されることがあります。この攻撃は、攻撃者が知っているオプションのみを対象とするものではありません。1つ以上のオプションが機能することを期待して、盲目的にオプションを呼び出そうとしている点で、他のAPI悪用とは異なります。攻撃者が公開アプリケーションの公開オプションに精通している場合も、この攻撃方法により未公開機能を発見できる可能性があるため、依然として有益な攻撃手法です。
113 160	スクリプトベースのAPIを悪用する	APIの操作	いくつかのAPIはスクリプト命令を引数としてサポートしています。スクリプト命令（またはスクリプト命令への参照）を用いる方法は、非常に柔軟で強力です。ただし、攻撃者がこれらのメソッドを機能させるスクリプトを指定できる場合、多くの機能にアクセスが可能となります。たとえば、HTMLページでは、スクリプト言語をページに埋め込み、受信側のWebブラウザで解釈できるようにする<script>タグをサポートしています。コンテンツプロバイダに悪意がある場合、これらのスクリプトはクライアントアプリケーションを危険にさらす可能性があります。一部のアプリケーションでは、スクリプトを提供しているユーザのIDではなく、独自のIDでスクリプトを実行することがあり、攻撃者は他の対策で拒否されている活動も実行できる可能性があります。
113 36	未公開APIの使用	APIの操作	攻撃者は、ターゲットシステム設計者が利用されることを意図していないAPIを探して呼び出します。APIがリクエストを認証しなかった場合、攻撃者はこれらの機能を呼び出すことができるかもしれません。
125	フラッディング	既存機能の悪用	攻撃者はターゲットとのインタラクションの数を急激に増やすことで、ターゲットのリソースを消費します。このタイプの攻撃はレート制限またはフローの弱点を攻撃します。攻撃が成功すると正当なユーザがサービスにアクセスできなくなり、ターゲットがクラッシュする可能性があります。この攻撃は、ターゲットへのリクエストの量に依存せず、ターゲットの操作に焦点を当てているリークまたはアロケーションによるリソースの枯渇とは異なります。フラッド攻撃の成功要因は一定期間内に攻撃者が行うことができる要求の数です。この数値が大きいほど、特定のターゲットに対して攻撃が成功する可能性が高くなります。
125 482	TCP Flood	フラッディング	攻撃者は、正当なユーザがサービスにアクセスできないように、TCPプロトコルを用いてフラッディング攻撃を実行します。これらの攻撃は、TCPプロトコル内の弱点（サーバが保持する必要があるコネクションに関する状態情報）を悪用します。

125 486	UDP Flood	フラッディング	攻撃者はUDPプロトコルを用いたフラッディング攻撃により利用可能なネットワーク帯域幅を消費し、正規のユーザがサービスにアクセスできないようにします。また、ファイアウォールはサービスへのUDP接続ごとにポートを開いているため、本質的に接続状態を保持しています。UDPフラッドの大量のパケット特性は、ファイアウォールに割り当てられたリソースを圧倒する可能性があります。UDPフラッド攻撃は、UDPを利用するDNSやVoIPなどのサービスをターゲットにすることもできます。さらに、UDPプロトコルはセッションレスの通信であるため、パケットの送信元が簡単に偽装でき、攻撃元を見つけることが困難になります。
125 487	ICMP Flood	フラッディング	攻撃者はICMPプロトコルを用いたフラッディング攻撃により、利用可能なネットワーク帯域幅を消費することで、正当なユーザがサービスにアクセスできないようにします。典型的な攻撃例は、被害者サーバが広範囲の送信元アドレスから急激にICMPパケットを受信することです。さらに、ICMPプロトコルのセッションレスな性質のため、パケットの送信元が簡単に偽装でき、攻撃元を見つけることが困難になります。
125 488	HTTP Flood	フラッディング	攻撃者はHTTPプロトコルを用いたフラッディング攻撃により、Webサービスやインフラストラクチャなどのアプリケーション層のリソースを消費させ、正当なユーザがサービスにアクセスできないようにします。この攻撃は大量のサーバリソースを消費させるため、正当なセッションに基づくHTTP GET要求を使用します。正当なセッション用いるため、この攻撃を検出することは非常に困難です。
125 489	SSL Flood	フラッディング	攻撃者はSSLプロトコルを用いたフラッディング攻撃により、サーバで利用可能なリソースをすべて消費することで、正当なユーザがサービスにアクセスできないようにします。この攻撃は、クライアントが使用する処理能力とサーバがセキュアな接続を作成するために使用する処理能力との間の非対称的な関係を利用します。攻撃者はプロビジョニングされていないマシンで多数のHTTPS要求を作成します。サーバ上で不均衡な大量のリソースを消費することができます。クライアントは継続してSSL接続の再ネゴシエーションを行います。多数の攻撃マシンが掛け合わされることで、正当なユーザのクラッシュまたはサービス不能を引き起こす可能性があります。
125 490	増幅攻撃	フラッディング	攻撃者は、応答のサイズがそれを生成する要求のサイズよりもはるかに大きくなるように増幅することができます。この攻撃の目的は比較的少数のリソースを使用してターゲットサーバに対して大量のトラフィックを作成することです。この攻撃を実行するため、攻撃者はソースアドレスをターゲットサーバのものになりすました状態でサードパーティサービスに要求を送信します。要求を受信したサードパーティサービスによって生成された大きなレスポンスがターゲットサーバに送信されます。攻撃者は多数の要求を送信することにより、ターゲットに向けられた膨大な量のトラフィックを生成することができます。最初の要求と最終的なターゲットへのペイロードのサイズの不一致が大きければ大きいほど、この攻撃の有効性が増します。
125 528	XML Flood	フラッディング	攻撃者はXMLメッセージを用いたフラッディング攻撃により、正当なユーザがウェブサービスにアクセスできないようにします。この攻撃は、多数のXMLに基づく要求を送信し、サービスがそれぞれの要求を解析しようとすることで実現されます。
130	過度な割り当て	既存機能の悪用	攻撃者は正当なサービスに利用可能なリソースの低減、サービスの悪用・拒否のため、ターゲットに過度のリソースを割り当てるように要求します。この攻撃はメモリ割り当てに重点を置いていますが、ターゲット上の有限のリソースは、帯域幅、処理サイクル、その他のリソースも含む攻撃対象となる可能性があります。この攻撃は、多数の要求（リソース枯渇やフラッディング）により強制的に割り当てを試みるものではありません。慎重にフォーマットされた1つまたは少数の要求を使用して、ターゲットに要求を処理するための過剰なリソースを割り当てさせます。多くの場合、ターゲット内のバグを利用して、ターゲットが通常の要求に必要とされるリソースを大幅に超えてリソースを割り当てるようにします。
130 230	ペイロードをネストされたXML	過度な割り当て	アプリケーションはXMLパーサを使用してXML形式のデータを変換する必要があります。攻撃者がXMLパーサの処理時に、XMLパーサに悪影響を与えるデータを注入する可能性があります。XMLデータをネストし、データを継続的に自己参照することにより、攻撃者はXMLパーサの処理に多くのリソースを消費させます。また、過剰なメモリ消費とCPU使用を招く可能性があります。攻撃者の目標は、パーサの失敗が自分の利益になることです。多くの場合、ターゲット内のバグを利用して、ターゲットが通常の要求に必要とされるリソースを大幅に超えてリソースを割り当てるようにします。
130 231	XML大規模ペイロード	過度な割り当て	アプリケーションはXMLパーサを使用してXML形式のデータを変換する必要があります。攻撃者はXMLパーサの処理時に、XMLパーサに悪影響を与えるデータを注入する可能性があります。攻撃者は、XMLパーサによって処理される入力ベクトルに大きすぎるペイロードを与えることで、処理中のXMLパーサにより多くのリソースを消費させます。過剰なメモリ消費とCPU使用を招き、潜在的に任意のコードを実行できる可能性があります。攻撃者の目標は、パーサの失敗が自分の利益になることです。多くの場合、この攻撃は、アプリケーションが不安定になったり、フリーズしたり、クラッシュしたりするため、サービス拒否につながります。また、クラッシュを引き起こし、任意のコードが実行され、データプレーンからコントロールプレーンにジャンプする可能性があります。

130 492	正規表現の指数関数的拡張	過度な割り当て	攻撃者は、不適切な正規表現（Regex）を実装しているプログラムに極端な状況になるような正規表現を入力することにより攻撃を実行します。極端な状況は、入力サイズと比較して指数関数的な大きさで動作します。これは、Regexアルゴリズムを構築するためにほとんどのマシンが非決定性有限オートマトン（NFA）状態を使って実装していることが原因です。このアルゴリズムは有限状態のマシンを構築し、入力の終わりに達するまで、すべての状態を通る入力遷移に基づいています。NFAエンジンは、バックトラック中に入力文字列の各文字を複数回評価することがあります。アルゴリズムは一致が見つかるまで、NFAを通過する各パスを1つずつ試行します。悪意のある入力の場合、すべてのパスが試行された後、失敗します。正規表現を利用するとプログラムが完了するのに非常に長い時間がかかります。この攻撃は、正規表現が使用されるため、インターネットのさまざまな層をターゲットにする可能性があります。
130 493	SOAP配列拡張	過度な割り当て	攻撃者は、通信中のSOAPメッセージを使用するWebサービスに対して攻撃を実行することがあります。 非常に大きなSOAP配列宣言をWebサービスに送信することで、XMLパーサによる解析の前に、Webサービスに配列要素の領域を割り当てるよう強制します。 通常、攻撃者のメッセージは小さなサイズと大きな配列宣言を含んでいます。例えば1,000,000要素の大きな配列宣言と2つの配列要素です。 この攻撃はWebサービスのメモリリソースの枯渇を目標としています。
130 494	TCPフラグメンテーション	過度な割り当て	攻撃者はフィルタリングルールを回避する目的で、ターゲットに対してTCPフラグメンテーション攻撃を実行します。 IP断片化は、IPデータグラムの大きさが通過しなければならない経路のMTUより大きい場合に発生します。攻撃者は、通常はヘッダフラグフィールドがフィルタリングされていない2つ目のフラグメントにプッシュされるように、TCPパケットを断片化しようと試みます。この動作は、通常、最初のパケットのヘッダーでFLAGSをチェックするIPSおよびファイアウォールフィルタを無効（最初のパケットを破棄することで、次のフラグメントが処理されて組み立てられるのを防ぐ）にします。別の方法は、重なり合ったフラグメントであり、無害な最初のセグメントがフィルタを通過し、2つ目のセグメントが本質的に悪意のある真のペイロードでTCPヘッダデータを上書きします。適切に作成された悪意のあるペイロードは、リソース消費やカーネルクラッシュによるDoSにつながる可能性があります。さらに、タイムアウトよりも少し遅い速度でフラグメントを送信するでフラグメンテーションを連鎖させることもできます。パケットを結合するリソースにタスクを完了するための過度の時間を待たせることによってDoS条件を引き起こします。断片化識別番号はIPv4では16ビットしか存在しないので、65536パケットだけが必要となるため、非常に簡単に複製できます。
130 495	UDPフラグメンテーション	過度な割り当て	攻撃者は帯域幅やCPUなどのリソースを消費するため、ターゲットサーバに対してUDPフラグメンテーション攻撃を実行します。 IP断片化はIPデータグラムが通過しなければならない経路のMTUより大きい場合に発生します。 通常、攻撃者は1500バイト以上の大きなUDPパケットを使用します。これは、イーサネットMTUが1500バイトであり、フラグメンテーションを強制できるためです。 この攻撃は、UDPフラッド攻撃の変化形ですが、より少ないパケットでより多くのネットワーク帯域幅を消費することができます。 さらに、サーバのCPUリソースを消費し、断片化されたパケットの処理と再組み立てに関連するメモリバッファを埋める可能性もあります。
130 496	ICMPフラグメンテーション	過度な割り当て	攻撃者はリソース消費やクラッシュを引き起こすため、ターゲットに対してICMP断片化攻撃を実行します。 攻撃者は断片化されたICMPメッセージを含む多数の断片化された同一IPパケットを作ります。 攻撃者がこれらのメッセージをターゲットホストに送信すると、ホストが応答しなくなります。 他の攻撃動機として、ハングアップを引き起こすヘッダー内の不正なサイズを持つターゲットホストに断片化ICMPメッセージを送信する可能性があります。
131	リソースのリーク	既存機能の悪用	攻撃者は、ターゲット上のリソースリークを利用して、正当な要求を処理するために利用可能なリソースの量を枯渇させます。リソースリークは、メモリが割り当てられ、使用後に解放されない場合、メモリリークの形で生じることが最も多いです。理論的には、予約可能な他のリソースにおいても起こる可能性があります。攻撃者は、どのようなアクティビティがリソースを漏洩させたのかを判断し、ターゲットのアクティビティをトリガします。リークによっては規模が小さく、攻撃者による多数の要求を必要とする場合があります。この攻撃は、要求の割合が一般的には重要ではないという点で、フラッド攻撃とは異なります。これは、ターゲットがリセット（通常は再起動）されるまで、リークに起因する失われたリソースが累積するためです。したがって、ターゲットをあふれさせることができない、資源の乏しい攻撃者は、この攻撃を継続して用いる必要があります。攻撃者がリークされたアロケーションのサイズを制御できない場合があり、リークがターゲットのパフォーマンスに影響を及ぼすのに十分な大きさになるまで蓄積する必要があるという点で、リークによるリソースの消耗はリソースの枯渇とは異なります。
212	機能の悪用	既存機能の悪用	攻撃者はアプリケーションの正当な能力を悪用します。 システムの機能は変更もされていませんが、意図されていない方法で使用されます。 これは、特定の機能を過度に使用したり、不正な機密データへのアクセスが可能な設計上の欠陥がある機能を活用することによって実現されます。

2122	アカウントロックアウトの誘導	機能の悪用	攻撃者は攻撃を阻止しようとするシステムのセキュリティ機能を利用して、正当なシステムユーザへのサービス拒否攻撃を実行します。たとえば、多くのシステムでは、一定回数の不正なログイン試行後にアカウントをロックするパスワード調整メカニズムを実装しています。攻撃者は、この調整メカニズムを利用して正当なユーザを自分のアカウントからロックすることができます。攻撃者によって悪用されている弱点は、攻撃に対抗するための非常に高いセキュリティ機能です。
21250	パスワードリカバリの悪用	機能の悪用	攻撃者はアプリケーション機能を利用して、元のユーザと同じ権限でシステムにアクセスするために、忘れたパスワードをユーザが回復できるようにする機能を利用します。一般に、パスワード回復スキームは脆弱で不安定な傾向があります。それらのほとんどは1つの質問のみを使用します。たとえば、母親の旧姓はかなり一般的に使用されます。多くの場合、特に攻撃者が正当なユーザを知っている場合は、この情報を知るのは難しくありません。これらの一般的なセキュリティに関する質問は、多くのアプリケーションで再利用されているため、さらに脆弱にしています。例えば、職場の同僚が銀行の代理人と話しており、検証プロセスのために母親の名前を伝えている会話を聞きます。攻撃者は被害者のアカウントの1つにログインするため、パスワードを忘れたをクリックするとセキュリティに関する質問が母親の旧姓になっている可能性があります。脆弱なパスワード回復スキームは、強力なパスワードスキームの有効性を完全に損ないます。
212620	暗号化レベルを低下させる	機能の悪用	攻撃者は暗号化レベルを下げるよう強制し、暗号化されたデータに対する攻撃を成功させることができます。
216	通信チャネルの操作	既存機能の悪用	攻撃者は、通信チャネル上の設定やパラメータを操作してセキュリティを侵害します。これは、情報漏洩、通信ストリームからの情報の挿入/削除、システムの侵害をもたらす可能性があります。
21612	メッセージ識別子の選択	通信チャネルの操作	この攻撃は、他のクライアントに割り当てられたパラメータ値を判断し、他のクライアントのためのマルチキャストまたはパブリック情報チャネルを介して配信されるメッセージを選択するものです。攻撃者が配信手段を介して特権を持つ情報にアクセスし、偽装することで他の攻撃を実行する可能性があります。操作されているチャネル/メッセージが(コマンドバスなどの)システムの出力メカニズムではなく入力である場合、この攻撃手法を使用して、攻撃者の識別子をより特権的な識別子に変更することができます。
216217	誤ったSSL設定の悪用	通信チャネルの操作	攻撃者は誤って設定されたSSL通信を利用して、暗号化されるはずのデータにアクセスできるようにします。攻撃者は、この攻撃により暗号化されたストリームにコマンドやその他のトラフィックを注入して、クライアントまたはサーバのいずれかを侵害する可能性があります。
227	継続的なクライアントによる攻撃	既存機能の悪用	攻撃者は、特定のリソースを継続的に関与することによって、正当なユーザによるリソースへのアクセスを拒否しようとします。攻撃者の主な目標は、ターゲットをクラッシュまたはフラディングさせることではなく、防御側に警告することです。与えられたリソースが縛られて正当なユーザが利用できないように、アルゴリズム上の欠陥を繰り返す、または、悪用します。一見良心的な要求でリソースを抱かせる要求を注意深く作成することによって、正当なユーザはリソースへのアクセスが制限されるか、完全に拒否されます。攻撃が成功する度合いは、正当なユーザによる通常の使用量を超える量のリソース要求に時間をかけてリソース要求を維持する攻撃者の能力、および標的の負荷をシフトする能力や追加のリソースを獲得する能力などのその他の緩和的な状況によって異なります。枯渇に対処する。この攻撃は、大量の要求に完全に依存しないため、フラッディング攻撃と異なり、リソースが機能するために必要な周囲の環境を悪用する傾向のあるリソースリークエクスポージャーとは異なります。サステイナビリティ攻撃の重要な要素は、通常より処理に時間がかかる繰り返し要求です。
227469	HTTP DoS	既存機能の悪用	攻撃者はHTTPレベルでフラッディングを実行し、TCP / IP接続でリッスンする特定のWebアプリケーションのみを停止させます。このサービス拒否攻撃では、送信するパケット数が大幅に少なく、DoSを検出するのが困難になります。これは、HTTPのSYNフラッドに相当します。HTTPセッションを無期限に保ち、それを何百回も繰り返します。この攻撃は、Webサーバソフトウェアのリソースが枯渇するという弱点を標的にしています。Webサーバは、接続スレッドが使い尽くされている間、通信を開始するためのHTTPセッションは攻撃者の応答を待つことになります。
272	プロトコル操作	既存機能の悪用	攻撃者は通信プロトコルを悪用します。この攻撃手法により、攻撃者が他人を偽装、機密情報の発見、セッションの結果を制御等を実行する可能性があります。この攻撃は、プロトコルの実装上の固有な前提、プロトコルの不正な実装、プロトコル自体の脆弱性を対象としています。

272 220	クライアント－サーバプロトコルの操作	プロトコル操作	攻撃者は、通信プロトコルの弱点を利用し、クライアントとサーバが予期しない動作を実行させます。クライアントとサーバのアプリケーション間でメッセージを転送するには、通信プロトコルが必要です。様々な種類のインタラクションに異なるプロトコルを使用することができます。たとえば、認証プロトコルを使用してサーバとクライアントのIDを確立し、別のメッセージングプロトコルを使用してデータを交換することができます。クライアントとサーバが使用するプロトコルに弱点がある場合、攻撃者はこれを利用してさまざまな種類の攻撃を実行できます。たとえば、攻撃者が認証プロトコルを操作できる場合、他のクライアントやサーバを偽装する可能性があります。攻撃者がメッセージングプロトコルを操作できる場合、機密情報を読み取りやメッセージの内容を変更する可能性があります。この攻撃は、サーバが多くのクライアントに同様の役割を果たすため、複数のプロトコルをサポートしていることにより簡単に行われます。例えば、サーバはレガシークライアントを含む広範囲のクライアントをサポートするために、いくつかの異なる認証プロトコルをサポートすることができます。古いプロトコルの中には、攻撃者がクライアントとサーバとのやりとりを操作できる脆弱性が存在する可能性があります。
272 105	HTTPリクエスト分割攻撃 HTTPリクエストスマグリング攻撃	プロトコル操作	HTTPリクエスト分割は、攻撃者が元の（エンベロープ）HTTP要求に追加のHTTP要求を挿入しようとする攻撃手法です。この要求は、ブラウザは1つの要求として解釈しますが、サーバはそれを2つの要求と解釈します。 HTTPリクエスト分割攻撃を実行する方法はいくつかあります。 1つ目の方法は、リクエストに2つのContent-Lengthヘッダーを含めることで、要求を解析するデバイスがそれぞれ異なるヘッダーを使用する可能性があるという事実を利用するものです。2つ目の方法は、次のパースエンティティによって別のHTTPリクエストとみなされるHTTPリクエスト内のペイロードを許可するために、setRequestHeaderで設定されたリクエストヘッダーにチャンクされたTransfer EncodingでHTTPリクエストを送信することです。 3つ目の方法は、HTTPヘッダー技術でDouble CRを使用する方法です。また、特定のWebサーバの特定の脆弱性を対象とした一般的ではない技術もいくつかあります。
272 276	コンポーネント間のプロトコル操作	プロトコル操作	コンポーネント間プロトコルは、単一のコンピュータ内の異なるソフトウェアモジュールとハードウェアモジュール間で通信するために使用されます。 例として、割り込み信号とデータパイプがあります。 プロトコルを無効にすることで、攻撃者は他人の偽装、機密情報の発見、セッションの結果を制御することができます。 このタイプの攻撃は、プロトコルの実装に内在する無効な前提、プロトコルの不正な実装、プロトコル自体の脆弱性を対象としています。
272 277	データ交換プロトコル操作	プロトコル操作	データ交換プロトコルは、エンティティ間で構造化データを送信するために使用されます。これらのプロトコルは、多くの場合、特定のドメイン（B2B：発注書、請求書、輸送ロジスティクス、運送状、医療記録）においては固有のものが使用されています。 プロトコルを無効にすることで、攻撃者が他人に偽装、機密情報の発見、セッションの結果を制御などを行うことができます。 このタイプの攻撃は、プロトコルの実装に内在する無効な仮定、プロトコルの不正な実装、プロトコル自体の脆弱性を対象としています。
272 278	Webサービスプロトコル操作	プロトコル操作	攻撃者は、Webアプリケーションに関連するプロトコルを操作し、Webアプリケーションまたはサービスに意図したものとは異なる動作を発生させます。 これは、予期しない値を含めるために呼び出しパラメータを操作するか、呼び出された関数を制限するものに変更することによって実行できます。 この攻撃により、攻撃者は通常制限されているデータやリソースにアクセスしたり、アプリケーションやサービスをクラッシュさせたりすることができます。
554	機能のバイパス	既存機能の悪用	攻撃者は、システムを保護するために意図された機能の一部または全部をバイパスすることによってシステムを攻撃します。 多くの場合、システムユーザは保護が適切であると考えていますが、その保護の背後にある機能を攻撃者が無効にしています。
554 465	透過的なプロキシの悪用	機能のバイパス	透過的なプロキシは、クライアントとインターネットとの中間的な役割を果たします。クライアントから発信されたすべての要求を傍受し、正しい場所に転送します。プロキシは、クライアントへのすべての応答を傍受し、クライアントに転送します。これらはクライアントにとって透過的な方法で行われます。プロキシは、企業やISPによってよく使用されています。クライアントから発信された要求に対して、プロキシはクライアントのデータパケットの最終的な宛先を把握する必要があります。そのためには、レイヤー3（ネットワーク）IPアドレスを調べる方法と、レイヤー7（アプリケーション）HTTPヘッダーの宛先を調べる方法の2つの方法があります。ブラウザにはポリシーがあり、通常、あるドメインからのスクリプトが他のWebサイトへの要求を開始しないようにします。しかし、これを回避するために、ユーザのブラウザで実行中の悪意のあるFlashやアプレットにより、クライアントからリモートドメインへのクロスドメインソケット接続の作成を試みる可能性があります。プロキシは要求のHTTPヘッダーを調べ、それをリモートサイトに送ることで、ブラウザの同じオリジンポリシーを部分的にバイパスします。これは、トランスペアレントプロキシがネットワークレイヤのIPアドレス情報ではなくアドレス指定にHTTPホストヘッダー情報を使用する場合に発生します。この攻撃により、標的ブラウザ内の悪意のあるスクリプトは、トランスペアレントプロキシにアクセス可能なすべてのホストに対してクロスドメイン要求を発行できます。

554 179	マイクロサービスの直接呼び出し	機能のバイパス	攻撃者は、Web上でマイクロサービスを検出して参照することができます。マイクロサービスの実装方法と機能に関する情報を収集することによって、悪用することができます。Webページのマイクロサービスは、ページの一部をサーバに接続し、ページ全体を更新する必要なくコンテンツを更新できるようにします。これにより、ユーザは、ページの一部を他の機能を停止させることなく、より迅速に変更することができます。ただし、これらのマイクロサービスは、他の形式のコンテンツと同じレベルのセキュリティレビューを受けていない可能性があります。たとえば、サーバにリクエストを送信してSQLクエリを変換するマイクロサービスは、SQLインジェクション攻撃から適切に保護されない可能性があります。結果として、マイクロサービスは、一連の攻撃に対して別の攻撃ベクトルを提供する可能性があります。マイクロサービスの存在が必ずしもサイトの攻撃に対して脆弱になるとは限りませんが、Webページの複雑さが増すため、脆弱性が含まれる可能性が高くなります。
554 464	エバークッキー	機能のバイパス	攻撃者は永続的なクッキーを作成し、ユーザはそれが削除されたと考えた後でも存在し続けるようにします。クッキーは、以下を含む10か所以上の被害者のマシンに保存されます。標準のHTTP Cookie、ローカル共有オブジェクト（Flash Cookie）、Silverlightアイソレートストレージ、自動生成・強制キャッシュされ、RGB値で保存されるクッキー、HTML5キャンバスタグを使用してピクセル（クッキー）を読み込むPNG、ウェブ履歴に保存されるクッキー、HTTP ETagsに格納されるクッキー、Webキャッシュに保存されるクッキー、window.nameキャッシュ、Internet Explorerユーザデータストレージ、HTML5セッションストレージ、HTML5ローカルストレージ、HTML5グローバルストレージ、HTML5データベースストレージ（SQLite経由）などです。標的がブラウザ内の従来の方法でCookieキャッシュをクリアすると、その操作によって特定の場所からCookieが削除されますが、他の場所は削除されません。その後、悪意のあるコードは、削除されなかったすべての場所から、すべての可能な保存場所にCookieを再度複製します。したがって、標的は元の保管場所のすべてにクッキーを再度保持します。 つまり、1つの場所でもCookieを削除できないと、どこでもCookieが復活します。特定の保存場所（たとえば、ローカル共有オブジェクト）が異なるブラウザ間で共有されているため、エバークッキーは異なるブラウザ間でも存続します。
	情報の収集と分析		この攻撃は、攻撃者による情報の収集、盗難に焦点を当てています。攻撃者は、能動的問い合わせおよび受動的観察を含む様々な方法を通じて情報を収集することができます。標的とその通信の設計または構成における弱点を悪用することにより、攻撃者は標的よりも多くの情報を明らかにすることができます。取得された情報は、攻撃者の目的を達成するための潜在的な弱点、脆弱性、技術について推論する際に攻撃者を助けるかもしれなません。情報には、ターゲットの構成・機能に関する詳細、アクティビティのタイミング・性質、それ以外の機密情報に関する手がかりが含まれる場合があります。多くの場合、このような種類の攻撃は、他の種類の攻撃に備えて行われますが、情報の収集自体が場合によっては攻撃者の最終目標になる場合もあります。
116	発掘	情報の収集と分析	攻撃者は、目的達成に必要な情報を取得するため、ターゲットを積極的に調査します。これは、ターゲットの情報を収集するために、通常のやりとりでターゲットを探索すること、または、所望のデータを含む応答を生成するため、構文的に無効・非標準のデータを送信することによって実現されます。これにより、攻撃者がそのセキュリティ、設定、潜在的な脆弱性について推論するのを助けるターゲットからの情報を得ることができます。ターゲットと典型的なメッセージを交換することで、スタックトレース、構成情報、パス情報、データベース設計などの情報を明らかにする未処理の例外または冗長なエラーメッセージをトリガーする可能性があります。このタイプの攻撃には、無効なSQLクエリを生成するためのURI内のクエリ文字列の操作や、サーバが有用な情報を返すことを期待して代替パスの値を試すことも含まれます。
116 150	共有なリソースの場所からデータを収集する	発掘	攻撃者は、標的のセキュリティを侵害するため、よく知られているリソースの場所を悪用します。 多くの場合、ファイルとリソースはデフォルトのツリー構造で編成されています。上記の場合，攻撃者は攻撃に必要なリソースやファイルの場所をよく知っているため、攻撃者にとって有益です。ターゲットリソースの正確な場所がわからない場合でも、命名規則は、リソースが通常配置されているターゲットマシンのファイルツリーの小さな領域を示す場合があります。たとえば、設定ファイルは通常Unixシステムの/ etcディレクターに格納されます。 攻撃者はこれを利用して他のタイプの攻撃を行うことができます。
116 406	ダンプスターダイビング スカベンジング スキャベンジング ゴミ箱あさり	発掘	攻撃者は、ごみ箱、ゴミ捨て場、または攻撃者に有用な会社の情報が誤って廃棄された可能性のある場所を検索します。危険なアイテムや情報には、医療記録、履歴書、個人的な写真や電子メール、銀行口座の明細、アカウントの詳細、ソフトウェアに関する情報、技術サポートログなどがあります。これらの情報を収集することによって、攻撃者は標的とする個人または組織に関する重要な事実を知ることができます。
116 54	情報のためにシステムに問い合わせ	発掘	アプリケーションの場所を知っている（アプリケーションの使用を認可されている）攻撃者は、アプリケーションの構造を調べ、要求を送信し、応答を調べることによって、アプリケーションの堅牢性を評価します。多くの場合、これは予想されるクエリの変異体を送信することによって実現されます。これらの変更されたクエリにより予想されるクエリが返答する情報以上の情報を返答することを期待しています。

116 545	システムリソースからデータを引き出す	発掘	既知のシステムリソースを検索する権限を持っている、または能力を持っている攻撃者が、有用な情報を収集する意図を持って攻撃します。システムリソースには、ファイル、メモリ、およびターゲットシステムの他のリソースが含まれます。この攻撃のパターンでは、攻撃者はデータを引き出すときに何を見つけようとしているのかを必ずしも把握していません。
116 569	ユーザが提供するデータを収集する	発掘	攻撃者は、ツール、デバイス、またはプログラムを利用して、ターゲットシステムのユーザが扱う特定の情報を取得します。この情報は、攻撃者が後に攻撃を開始するためによく利用されます。この攻撃がソーシャルエンジニアリングとは異なる点は攻撃者がユーザを騙さない点です。代わりに、攻撃者は、ユーザが合法的にシステムに入力した情報を取得するメカニズムを利用します。例として，キーロガーの配備、UACプロンプトの実行、Windowsのデフォルト資格プロバイダのラッピングがあります。
117	傍受	情報の収集と分析	攻撃者は、情報を収集するためにターゲットへの、または、ターゲットからのデータの流れを監視します。この攻撃は、後の攻撃をサポートするための情報を収集するために実施されるか、または収集されたデータが攻撃の最終目標になる可能性があります。この攻撃には、ネットワークトラフィックを傍受することが含まれますが、無線などの他のタイプのデータの流れを監視することも含まれます。この攻撃のほとんどの種類で、攻撃者は受動的であり、定期的な通信を観察するだけですが、攻撃者はデータストリームの確立を開始したり、送信されるデータの性質に影響を与えることがあります。ただし、攻撃者はデータストリームの意図された受信者ではありません。他のデータ漏洩攻撃とは異なり、攻撃者は明示的なデータチャネル（ネットワークトラフィックなど）を観察してコンテンツを読み取っています。これは、通信量やデータストリームを介して明示的に通信されていないその他の情報など、より定性的な情報を収集する攻撃とは異なります。
117 157	スニッフィング攻撃	傍受	攻撃者は、ネットワークの論理ノードまたは物理ノード間で送信される情報を監視します。攻撃者はコンテンツの受信や変更を実行せず、トラフィックを観察して読み取るだけです。攻撃者は、観察されたトランザクションの内容に間接的に影響を与える可能性がありますが、攻撃者は情報の意図された受信者ではありません。攻撃者が送信者と受信者の間の内容を傍受することができるならば、理論的にはあらゆる伝送媒体が盗聴される可能性があります。
117 499	インテントの傍受	傍受	攻撃者は、以前にインストールされた悪意のあるアプリケーションを通じて、信頼できないAndroidベースのアプリケーションからのメッセージを傍受し、サービス拒否、情報漏洩、データ注入などのさまざまな目的を達成しようとします。信頼できるアプリケーションから送信された潜在的なインテントは、適切なインテントフィルタを宣言したアプリケーションで受信できます。インテントが悪意のあるアプリケーションからパーミッションによって保護されていない場合、攻撃者はそのインテントに含まれるデータにアクセスできます。さらに、インテントはインテントされた送信先に到達するのを妨害、または変更された送信先に潜在的に転送される可能性があります。
117 651	盗聴	傍受	攻撃者は、ソフトウェア（例えば、マイクロホンおよびオーディオ録音アプリケーション）、ハードウェア（例えば、録音機器）、または物理的手段（例えば、物理的接近）により通信内容（例えばテキスト、オーディオ、ビデオ）を傍受します。 盗聴の目的は、通常、金融、個人、政治、またはその他の利益のために、ターゲットに関する機密情報への不正アクセスを取得することです。 盗聴は、ネットワークベースの通信チャネル（例えば、IPトラフィック）上で行われないので、スニファリング攻撃とは異なります。 代わりに、2人以上の関係者間の会話の生のオーディオソースを聞く必要があります。
117 634	オーディオとビデオ周辺機器で盗聴	傍受	攻撃者は、マルウェアやスケジュールされたタスクによりターゲットシステムのオーディオとビデオの機能を利用します。 ペリフェラルデバイス（例えば、マイクやウェブカメラ）またはオーディオとビデオ機能を備えたアプリケーションを使用して2者間の通信データを収集することによって目標（財務、個人、政治、またはその他の利益のためのターゲットに関する機密情報をキャプチャ）を達成します。
169	フットプリンティング	情報の収集と分析	攻撃者は探査活動により、対象の構成要素と特性を特定します。フットプリントは、さまざまな情報収集手法を記述する際の一般的な用語であり、攻撃の準備のために、よく利用されます。ターゲットとするアプリケーション、システム、またはネットワークの構成およびセキュリティのメカニズムについて、できるだけ多くのことを把握するためにツールを使用します。収集される可能性のある情報には、オープンポート、アプリケーションとそのバージョン、ネットワークトポロジ、および同様の情報が含まれます。フットプリンティングは害を及ぼすものではありませんが（ネットワークスキャンなどの特定のアクティビティでは、偶発的に脆弱なアプリケーションへの中断が生じることがあります）、以降の攻撃により深刻になる可能性があります。

169 292	ホストの発見	フットプリンティング	攻撃者はIPアドレスに信号を送信し、ホストが動作しているかどうかを判断します。ホスト発見は、ネットワーク偵察の初期段階の1つです。攻撃者は、通常、ターゲットネットワークに属するIPアドレス範囲の探索から開始し、さまざまな方法を使用して、そのIPアドレスにホストが存在するかどうかを判断します。通常、ホスト発見は「Ping」スキャンと呼ばれます。攻撃者の目標は、IPアドレスにパケットを送信し、ホストからの応答を求めることです。攻撃者がその応答に基づいて動作しているホストを特定できたならば、「ping」はどんな細工されたパケットでもかまいません。攻撃は通常、特定の種類のpingがIPアドレスの範囲に送信される「ping sweep」で実行されます。
169 300	ポートスキャン	フットプリンティング	攻撃者は、複数の手法を組み合わせて、リモートターゲットのポートの状態を確認します。TCPまたはUDPネットワーキングで使用可能なサービスまたはアプリケーションは、ネットワーク上の通信用にポートを解放しています。一般的なサービスではポート番号が割り当てられていますが、サービスとアプリケーションは任意のポートで実行できます。さらに、ポートスキャンは、任意のマシンが最大65535個の可能なUDPまたはTCPサービスを持つ可能性があるため、複雑です。ポートスキャンの目的は、開いているポートを特定するだけでなく、攻撃者がファイアウォールの構成に関する情報を知ることができる等、もっと広い場合があります。使用されるスキャン方法に応じて、スキャンプロセスを隠蔽するか、より目立たせることができます。後者は関連するパケットの量、異常なパケット特性、またはシステムロギングにより検出しやすくなります。一般的なポートスキャンでは、ポート範囲にプローブを送信し、応答を監視します。ポートスキャンで通常検出されるポートの状態には、4つのタイプがあります。攻撃者の戦略的では、フィルタによって保護されているオープンポートと、フィルタで保護されていないクローズドポートを区別することが有益です。特定のスキャンタイプでは、これらのきめ細かな区別を行うことは不可能です。たとえば、TCP接続スキャンでは、アクティブなサービスを持つファイアウォールでブロックされたポートと、ファイアウォールでブロックされていない閉じたポートを区別できません。スキャンタイプによっては閉じたポートだけを検出できますが、別のスキャンタイプではポートの状態を全く検出できず、フィアウォールによるフィルタの有無のみが検出されます。ポートスキャンによりどのポートを攻撃者が直接攻撃できるか判断できます。フラグメンテーション、ソースポートスキャンなどのフィルタ回避技術が必要か、どのポートがネットワークサービスを使用していないが保護されていない（すなわち、ファイアウォールされていない）かを判断できます。攻撃者は、ホストのおけるファイアウォールのフィルタリングメカニズムを完全に把握するために、さまざまな手法を組み合わせることがあります。
169 309	ネットワークトポロジマッピング	フットプリンティング	攻撃者は、ネットワークノード、ホスト、デバイス、および経路をマッピングするためにスキャンを行います。攻撃者は通常、外部ネットワークに対する攻撃の初期段階でこのようなネットワーク偵察を実行します。ICMPツール、ネットワーク可視化ツール、ポートスキャナ、およびtracerouteなどのルートテストユーティリティを含む、多くの種類のスキャンユーティリティが利用されています。
169 529	マルウェアによる内部偵察	フットプリンティング	攻撃者は、組織境界内にインストールされたマルウェアまたは同様に制御されるアプリケーションを使用して、対象となるアプリケーション、システムまたはネットワークの構成およびセキュリティメカニズムに関する情報を収集します。
169 580	アプリケーションフットプリンティング	フットプリンティング	攻撃者は、リモートターゲットにインストールされているアプリケーションのタイプまたはバージョンを特定するために、アクティブな探索を行います。これは、アプリケーションの出力を調査するため受動的なフィンガープリンティングとは異なります。
188	リバースエンジニアリング	情報の収集と分析	攻撃者は、分析されたエンティティがどのように構築され動作するかを効果的に判定するために、様々な分析技術を使用して、オブジェクト、リソース、システムの構造、機能、および構成を探索します。リバースエンジニアリングの目的は、機能の一部を複製することとです。リバースエンジニアリング技術は、機械的オブジェクト、電子デバイス、またはソフトウェアに適用することができますが、それぞれのタイプで分析に必要な手法は大きく異なります。
188 167	ホワイトボックスリバースエンジニアリング	リバースエンジニアリング	攻撃者は、ホワイトボックス解析技術により、ある種のコンピュータソフトウェアの構造、機能、構成を発見します。 ホワイトボックス技術は、実行可能ファイルまたは他のコンパイルされたオブジェクトを直接解析でき、実行時に観察される機械命令の少なくとも一部を明らかにする場合にソフトウェアの一部に適用できる方法も含みます。
188 189	ブラックボックスリバースエンジニアリング	リバースエンジニアリング	攻撃者は、ブラックボックス解析技術により、ある種のコンピュータソフトウェアの構造、機能、構成を発見します。「ブラックボックス」技術は、実行可能オブジェクトへの直接アクセスがない場合に間接的にソフトウェアと対話することを含みます。通常、このような分析には入出力のベクトル、ライブラリ、APIなどのより大きな実行環境を有するソフトウェアインタフェースの境界上のソフトウェアと対話する必要があります。

192	プロトコル解析 解読	情報の収集と分析	攻撃者は、パケット交換データネットワーク上の相互接続されたノードまたはシステム間で情報を送信するために使用されるネットワークまたはアプリケーション通信プロトコルのプロトコル情報を解読および/または復号する活動に従事します。この種の分析は、本質的にネットワーキングプロトコルの分析を伴いますが、実際の物理的なネットワークの存在を必要としません。プロトコル解析のための特定の技術は、通信コンポーネント間で「オンザワイヤ」相互作用を操作することで利益を得ます。また、ネットワークインターフェイスドライバなどのデバイスドライバだけでなく、実行可能ファイルに適用される静的または動的解析手法も使用できます。使用される方法に応じて、プロセスは、ホスト間で発生する実際の通信を観察し、対話し、変更することを含みます。プロトコル分析の目的は、プロトコルによって使用されるパケットまたはコンテンツ区切り文字を含む意味のあるコンテンツを抽出するだけでなく、データ送信構文を導出することです。この種の分析は、クローズド仕様のプロトコルまたはプロプライエタリなプロトコルで実行されることが多いですが、公開されている仕様を分析して、特定の実装が公開された仕様からどのように逸脱しているかを判断するのにも役立ちます。
192 97	暗号解読	情報の収集と分析	暗号解読は、暗号アルゴリズムの弱点を見つけ出し、これらの弱点を使用して秘密鍵を知ることなく暗号文を解読するプロセスです。暗号アルゴリズム自体の弱点がなくても、暗号解析が成功する方法があります。ほとんどの場合、暗号解読が成功しても、攻撃者は平文に関するいくつかの情報を推測することができないかもしれません。しかし、状況によっては、攻撃者にとっては十分かもしれません。
192 608	セルラ暗号の暗号解読	情報の収集と分析	暗号解読技術を使用して暗号鍵を導出するか、セルラー暗号化を無効にしてトラフィックの内容を明らかにする。A5 / 1やA5 / 2（GSM用に指定されている）などの一部のセルラー暗号化アルゴリズムは、このような攻撃に対して脆弱であることが知られており、これらの攻撃を実行し、リアルタイムで携帯電話の会話を解読する商用ツールが利用できます。 UMTSおよびLTEで使用されている新しい暗号化アルゴリズムはより強固で、現在のところ、この種の攻撃に対して脆弱ではないと考えられています。 ただし、Cellular Rogue Base Stationを使用する攻撃者は、新しいモバイルデバイスでも弱いセルラ暗号の使用を強制できます。
192 463	パディングオラクル攻撃	情報の収集と分析	攻撃者は、ターゲットシステムが暗号文の復号中にパディングエラーが発生したかどうかのデータを漏洩した場合に、復号鍵を知らずにデータを効率的に解読することができます。このタイプの情報を漏洩するターゲット・システムはパッディング・オラクルとなり、攻撃者はそのパッディング・オラクルへの平均128 * bコールを発行することによって復号キーを知らずにそのオラクルを効率的に解読することができます。解読を実行することに加えて、パディング・オラクルを使用することで、攻撃者は暗号鍵を知らずに、有効な暗号文を生成する（すなわち、暗号化を実行する）こともできます。暗号化されたメッセージが復号される前にそれらの正当性の保証を認証されず、パッディングエラーに関する情報が攻撃者に漏れると、どの暗号システムもパッディングオラクル攻撃に対して脆弱になる可能性があります。この攻撃技法は、例えば、CAPTCHAシステムを壊すか、またはクライアント側のオブジェクト（例えば、隠されたフィールドまたはクッキー）に格納された状態情報を解読/変更するために使用することができます。この攻撃手法は、不適切に実装された復号ルーチンからのデータ漏洩を使用して暗号システムを完全に破壊する、暗号システムに対するサイドチャネル攻撃です。どのような形式であっても、復号中のパディングエラーが発生したかどうかを攻撃者に知らせる1ビットの情報は、暗号システムを破壊するのに十分です。その情報のビットは、パディングエラー、返された空白のページ、または応答時間が長いサーバ（タイミング攻撃）に関する明示的なエラーメッセージの形で得られる可能性があります。この攻撃は、攻撃者がクロスドメイン情報漏洩により、標的が通信しているターゲットシステム/サービスからのパディングオラクルからの情報のビットを得ることができるクロスドメインを開始することができます。攻撃者は暗号文を含む要求をターゲットシステムに送信します。ブラウザの同じ発信元ポリシーのため、攻撃者は応答を直接見確認できませんが、クロスドメイン情報漏洩技術を使用して必要な情報（すなわち、パディングエラーが発生したか否かに関する情報）を得ることができます。たとえば、これは ″img″ タグとonerror()/ onload()イベントを使用して行うことができます。攻撃者のJavaScriptは、ターゲットサイトにイメージをロードし、イメージがロードされているかどうかを知るためにWebブラウザを作成することができます。これはパッディング・オラクル攻撃が機能するために必要な1ビットの情報です。イメージがロードされている場合は有効なパディングです。そうでない場合は有効ではありません。
224	フィンガープリント	情報の収集と分析	攻撃者は、ターゲットシステムからの出力とターゲットに関する特定の情報を一意に識別する既知のインジケータと比較します。通常、フィンガープリントは標的にとって有害ではありません。しかし、フィンガープリントによって収集された情報は、しばしば攻撃者が標的の既存の弱点を発見することを可能にします。

224 312	アクティブなOS フィンガープリンティング	フィンガープリント	攻撃者は、環境内のオペレーティングシステムまたはファームウェアに関する情報を明らかにするために設計されたプローブを使用して、デバイス、サーバ、またはプラットフォームを調べることによって、リモートターゲットのオペレーティングシステムまたはファームウェアバージョンを検出します。オペレーティングシステムの検出は、一般的なプロトコル（IPやTCPなど）は異なる方法で実装されているため可能です。実装の違いはプロトコルとの互換性を「破る」には十分ではありませんが、その違いは検出することは可能です。ターゲットは、プロトコルにおけるパケット構成の論理的ルールを破る特定の探索活動に独自の方法で応答するためです。異なるオペレーティングシステムには、異常な入力に対して固有の応答があり、OSの動作にフィンガープリントを割り当てる根拠が存在します。このタイプのOSフィンガープリンティングは、オペレーティングシステムの種類とバージョンを区別できます。
224 313	パッシブなOSフィンガープリンティング	フィンガープリント	攻撃者は、デバイス、ノード、またはアプリケーション間の通信を受動的に監視することによって、環境内のOSソフトウェアのバージョンまたはタイプを検出します。オペレーティングシステム検出のための受動的技術は、プローブ信号をターゲットに送信しません。既知のシグネチャまたは値のデータベースと比較して観察された挙動に基づいてオペレーティングシステムを識別するために、ノード間のネットワークまたはクライアント－サーバ通信を監視します。受動的なOSフィンガープリンティングは通常、アクティブな方法ほど信頼性がありませんが、一般的に標的からの検出を避けることができます。
224 541	アプリケーション フィンガープリンティング	フィンガープリント	攻撃者は、リモートターゲットにインストールされているアプリケーションのタイプまたはバージョンを判別するためにフィンガープリントを調査します。
410	情報の抽出	情報の収集と分析	攻撃者は情報を抽出する目的で、ソーシャルエンジニアリング手法の任意の組み合わせを使用して個人を関与させます。ターゲット企業や個人に関する重要な情報を知るなど、正確なコンテキストキューや環境キューは、攻撃の成功と収集される情報の品質を大幅に向上させる可能性があります。詳細な知識と組み合わせた信憑性のある偽装は、誘発攻撃の成功を増加させます。
410 407	プリテキスティング	人間の行動を操作する	攻撃者は、ターゲットからの情報を要求したり、ターゲットを操作して攻撃者の利益に役立つ何らかのアクションを実行させるため、プレテクスティングを行います。攻撃者はターゲットとする被害者に情報を公開したり、何らかの措置を講ずるように識別情報や役割を仮定するシナリオを考案します。嘘をつくだけではなく、場合によっては、受信者を操作するために完全な新しい識別情報を作成し、使用するかもしれません。プレテクスティングは、攻撃者が経験したことがない特定の仕事や役割の人を偽装するのにも使用できます。この攻撃で簡単にターゲットに関する情報を知ることができます。より複雑なやりとりとしては、攻撃者が組織の弱点の悪用や安全な施設・システムへのアクセスを得るための何らかのアクションをターゲットに実行するように要求するかもしれません。良い情報収集技術は、良い口実を作り、標的を破壊することができます。確かな口実は、信頼を築くために不可欠な部分です。攻撃者のエイリアス・ストーリー・アイデンティティに弱点がある、信頼性がない信ぴょう性がない場合でも、ターゲットは被害を受ける場合があります。
223	確率論を利用する		攻撃者は、数学的確率に基づきターゲットのセキュリティ特性を探索し、克服する確率論的手法を利用します。攻撃者は、セキュリティプロパティを保持していない非常にまれな条件を特定して悪用することができます。
223 112	ブルートフォース	確率論を利用する	この攻撃では、一部の資産（情報、機能、アイデンティティなど）が秘密の値で保護されています。攻撃者は、アセットをロック解除する秘密（または機能的に同等の値）を見つけるために、ブルートフォースにより資産にアクセスして、すべての可能な秘密値を徹底的に探索します。秘密の例には、パスワード、暗号化キー、データベースルックアップキー、および一方向関数への初期値などがありますが、これらに限定されません。この攻撃を成功する主の要素は、可能な秘密空間を迅速に探索する攻撃者の能力です。これは、秘密空間のサイズと攻撃者が問題に耐えられる計算能力に依存します。攻撃者が小さなリソースしかを持たず、秘密スペースが大きい場合、攻撃の成功する可能性が小さくなります。防御側は攻撃者が利用できるリソースを制御できませんが、秘密空間のサイズは制御できます。大きな秘密空間を作成するには、できるだけ多くの選択できるフィールドから秘密を選択し、攻撃者が利用可能な手がかりまたは解読法を使用してこのフィールドのサイズを縮小できないようにする必要があります。秘密空間を小さくすることは非常に難しいです。コンピュータのような決定論的なマシンを使用することは、パターンの排除（潜在的な秘密空間を減らすのに役立つ攻撃者の手がかりを提供すること）が困難なためです。有限の秘密スペースではブルートフォース攻撃は最終的に成功します。防御側は、攻撃成功までに必要な時間とリソースが情報の価値を超えていることを確認する必要があります。たとえば、探索に何百年もかかる秘密空間は、ブルートフォース攻撃から安全である可能性が高いです。
223 20	暗号化ブルートフォース	確率論を利用するブルートフォース	暗号テキストと使用した暗号化アルゴリズムを保持している攻撃者は、暗号文を解読して平文を得るための鍵を特定するために、鍵空間について徹底的な（ブルートフォース）検索を実行します。

22349	パスワードブルートフォース	確率論を利用するブルートフォース	この攻撃では、攻撃者は成功するまでパスワードの可能な値をすべて試します。ブルートフォース攻撃は、使用されるアルファベット（小文字、大文字、数字、記号など）とパスワードの最大長に関係なく、基本的にすべてのパスワードを確認するので、常に成功します。ユーザが選択したパスワードが適切なパスワードポリシーに準拠した強力なパスワードであることを保証するための適切な強制メカニズムがない場合、システムはこの種の攻撃に対して特に脆弱です。実際には、パスワードが弱いと思われる場合を除いて、パスワードに対する純粋なブルートフォース攻撃はめったに使用されません。はるかに効果的な他のパスワードクラッキング方法（例えば、辞書攻撃、レインボーテーブルなど）が存在します。
22355	レインボーテーブルパスワードクラッキング	確率論を利用するブルートフォース	攻撃者はパスワードのハッシュが格納されているデータベーステーブルにアクセスします。次に、事前計算されたハッシュチェーンのレインボーテーブルを使用して、元のパスワードを探索します。ハッシュに対応する元のパスワードが取得されると、攻撃者は元のパスワードを使用してシステムにアクセスします。パスワードレインボーテーブルには、さまざまなパスワードのハッシュチェーンが格納されています。元のパスワードPから、圧縮関数Rおよびハッシュ関数Hを介して、パスワード連鎖が計算されます。以下の場合、漸化関係が存在します。Xi + 1 = R(H(Xi))、X0 = PX1、X2、X3、…、Xn−2、Xn−1、Xn、H(Xn)は、元のパスワードPの長さnのハッシュチェーンを形成することができます。PとH(Xn)は、レインボーテーブルに一緒に格納されます。レインボーテーブルの作成には非常に時間がかかり、計算コストが大きくなります。様々なハッシュアルゴリズム（例えばSHA1、MD5など）のために別個のテーブルを構築する必要があります。しかし、レンボーテーブルが計算されると、ソルトを使わずにハッシュ化されたパスワードの解読には非常に効果的です。
28	ファジング	確率論を利用する	この攻撃では、攻撃者はファジングを利用してシステムの弱点を特定しようとします。ファジングはシステムにランダムに構成された入力を供給し、その入力に応答して障害が発生したという指示を探す、ソフトウェアセキュリティおよび機能テスト方法です。ファジングはシステムをブラックボックスとして扱い、システムに関する先入観や前提は意味をなしません。攻撃者がシステムのユーザ入力に関する特定の前提を発見するのに役立ちます。ファジングは、システムの内部について何も知らないにもかかわらず、攻撃者にこれらの前提の一部を潜在的に明らかにする簡単な方法です。これらの前提条件は、攻撃者が目標を達成できるように特別に作成したユーザ入力により、システムを無効にすることができます。
156	人を欺くやりとりを行う		このカテゴリ内の攻撃パターンは、標的を欺き、納得させるために、ターゲットとの悪意のある相互作用に重点を置いています。他のプリンシパルと対話し、標的と他のプリンシパルとの間に存在する信頼レベルに基づいてアクションを実行します。これらのタイプの攻撃は、ある種のコンテンツまたは機能がアイデンティティに関連付けられ、この関連性のためにコンテンツ/機能が標的によって信頼されていることを前提としています。「なりすまし」という言葉によって識別されることが多いこの種の攻撃は、ターゲットがコンテンツの正当性を間違って信じるような方法でコンテンツ・アイデンティティの改ざんを行なっています。例えば、攻撃者は、2つの当事者間の金融取引を取引量が増加したように変更することができます。受信者が変更を検出できない場合、変更されたメッセージが元の送信者のものと誤って判断する可能性があります。これらのタイプの攻撃には、コンテンツをゼロから作成するか、正当なコンテンツを取得し修正する攻撃者が含まれます。
148	コンテンツスプーフィング	人を欺くやりとりを行う	攻撃者は、コンテンツの見かけのソースを変更せずに元のコンテンツ制作者が意図したものとは異なるものを含むようにコンテンツを変更します。コンテンツスプーフィングという用語は、所有者のコンテンツの代わりに攻撃者のコンテンツを表示するために標的に表示するウェブページを変更することを記述する際に最もよく使用されます。ただし、電子メールメッセージの内容、ファイル転送、または他のネットワーク通信プロトコルの内容を含む、どんなコンテンツでも偽装することができます。コンテンツは、ソース（例えば、ウェブページのソースファイルを変更する）または転送中（例えば、送信者と受信者との間のメッセージのインターセプト・変更）に改ざんすることができます。通常、攻撃者はコンテンツが変更されたという事実を隠そうとしますが、ウェブサイトの改ざんなどの場合は、隠す必要はありません。コンテンツスプーフィングは、マルウェアの感染、財務詐欺（コンテンツが金融取引を管理する場合）、プライバシー違反、およびその他の望ましくない結果につながります。
148 502	インテントの偽装	コンテンツスプーフィング	攻撃者は、データの変更、情報漏洩、およびデータ注入を含む様々な異なる目的を達成するために、事前にインストールされた悪意のあるアプリケーションを介して、特定の信頼できるアプリケーションのコンポーネントに向けられたインテントを発行します。誤って出力され、公開されたコンポーネントは、この種の攻撃の対象となります。コンポーネントが無差別にインテントの動作を信頼する場合、ターゲットアプリケーションは攻撃者の要求で機能を実行し、攻撃者の目的を達成するのを助けてしまいます。

148 627	GPS信号の偽装	コンテンツスプーフィング	攻撃者は、通常のGPS信号に類似するように構成された偽造GPS信号を送信することによってGPS受信機を欺むこうとします。これらのスプーフィングされた信号により、実際の場所以外のどこかに、または攻撃者が決定した別の場所に位置しているようにレシーバに表示させることができます。
151	識別情報のなりすまし	人を欺くやりとりを行う	識別情報のなりすましとは、ある他のエンティティ（人間または人間以外）の識別情報を仮定し、その識別情報を用いて目標を達成する動作を言います。攻撃者は、異なる関係者から送られたように見えるメッセージを作成したり、盗まれた/偽装された認証資格を使用することがあります。あるいは、正当な送信者からのメッセージを傍受し、その内容を変更せずにメッセージが正当な送信者から送られてきたようにすることもできます。この攻撃は、正当なユーザの資格情報を乗っ取り使用する場合もあります。識別情報のなりすまし攻撃は、送信されるメッセージに限定されません。識別情報に関連付けられたすべてのリソース（たとえば、署名付きファイル）は、攻撃者が見た目の同一性を偽装する攻撃の対象になる可能性があります。この攻撃は、攻撃者がメッセージの識別情報を変更せず、メッセージの内容のみを変更したいコンテンツスプーフィングとは異なります。識別情報のなりすまし攻撃では、攻撃者がコンテンツの識別情報を変更しようとしています。
151 194	データ元の偽造	識別情報のなりすまし	攻撃者は、改ざんした識別情報の下でデータを提供します。偽造された識別情報を使用する目的は、提供されたデータの追跡を防止するためかもしれません。また、攻撃者が別の識別情報に与えられた権利を受け取ろうととしているかもしれません。この攻撃の最も単純な方法の1つは、メッセージが実際の送信者以外から送信されたように見せかけるため、Fromフィールドを変更した電子メールメッセージを作成することです。攻撃の結果は攻撃の詳細によって異なりますが、特権昇格、他の攻撃の難読化、データの破損/操作が含まれています。
151 275	DNSリバインディング	識別情報のなりすまし	攻撃者は、自らが制御するDNSサーバによってIPアドレスが解決されるコンテンツを提供します。ウェブブラウザ（または同様のクライアント）による最初の接続の後、攻撃者は、その名前解決するIPアドレスを公開されていないターゲットブラウザの組織内のアドレスに変更します。これにより、Webブラウザが攻撃者のために内部アドレスを調べることができます。 Webブラウザは、ゾーン間の情報漏洩を防ぐために、DNS名に基づいてセキュリティゾーンを適用します。 DNSバインディングでは、攻撃者が攻撃することで、自分のサーバ上のコンテンツを自分の名前とDNSサーバで公開します。ターゲットが攻撃者のコンテンツに初めてアクセスするとき、攻撃者のドメイン名はIPアドレスに解決されなければなりません。攻撃者のDNSサーバは、この解決を実行し、ターゲットが値をキャッシュしないように短いTTL（Time-To-Live）を提供します。ターゲットが攻撃者のコンテンツに後続の要求を行うと、攻撃者のDNSサーバが再度照会されなければならなりませんが、今度はDNSサーバがターゲットの組織の内部で外部のソースからアクセスできないアドレスを返します。同じ名前がこれらの両方のIPアドレスに解決されるため、ブラウザは両方のIPアドレスを同じセキュリティゾーンに配置し、アドレス間で情報を流すことができます。攻撃者は元のメッセージから取り出されたターゲットのコンテンツにスクリプトを使用して、指定された内部アドレスからデータを抽出することができます。これにより、攻撃者は企業の内部ネットワークに関する機密情報を発見することができます。標的のブラウザを持つコンピュータと攻撃者が識別する内部のマシンとの間に信頼関係がある場合、追加の攻撃が可能です。この攻撃は、悪意のあるDNSサーバの正当な所有者であり、外部DNSサービスの動作を侵害する必要がないという点で、ファーミング攻撃とは異なります。
151 195	プリンシパル・スプーフィング	識別情報のなりすまし	プリンシパル・スプーフィングは識別情報のなりすましの一種で、攻撃者が相互のやりとりの中で他人のふりをするものです。これは、よく攻撃者以外から来たように見えるメッセージ（書面、口頭、または視覚のいずれか）を作成することによって達成されます。フィッシング攻撃とファーミング攻撃は、機密情報を収集しようとする試みが正当な情報源から来たように見せるかけるために、行われることがよくあります。プリンシパル・スプーフィングは、盗まれた、または、偽装された認証資格情報を使用せず、代わりに識別情報をみせかけるメッセージの外観と内容を使用します。プリンシパル・スプーフィングの考えられる結果は、識別情報のなりすましと同様（例えば、特権の昇格）です。同様に、識別情報のなりすまし（メッセージの作成・メッセージの中断と使用・メッセージの変更）のほとんどの技法は、プリンシパル・スプーフィング攻撃で使用できます。プリンシパル・スプーフィングは人を偽装するために使用され、ソーシャル・エンジニアリングにも使える攻撃手法です。
151 587	クロスフレームスクリプティング（XFS）	識別情報のなりすまし	この攻撃は、悪意のあるJavascriptと隠れたiframeを読み込まれた正当なWebページに結合しています。悪意のあるJavascriptは、ユーザに気づかない方法で正当なウェブページとインタラクションすることができます。この攻撃は通常、ソーシャルエンジニアリングのいくつかの要素を活用します。攻撃者は、操作しているWebページに訪問するようにユーザを誘導する必要があります。
151 473	署名の偽装	識別情報のなりすまし	攻撃者は、受信者が信用するようにメッセージまたはデータブロックを生成します。生成されたメッセージまたはデータブロックは、信頼・信用できるソースによって暗号化署名され、被害者のオペレーティングシステムを誤解させて悪意のある行為を実行させます。

151 89	ファーミング	署名の偽装	ファーミング攻撃は、被害者がオンラインバンクサイトや取引プラットフォームなどの信頼できる場所であるとだまされて機密データを入力した場合に発生します。攻撃者は、これらの信頼できるサイトを偽装し、被害者が意図したサイトではなく自分のサイトに誘導させることができます。ファーミング攻撃が成功するためにはスクリプトの注入や悪質なリンクのクリックを必要としません。
151 98	フィッシング	署名の偽装	フィッシングは、攻撃者が後で使用する機密情報（認証資格情報等）を取得するため、被害者がビジネスを行う正当な画面等を偽装するソーシャルエンジニアリング技術です。
154	リソースの位置をなりすまし	人を欺くやりとりを行う	攻撃者は、アプリケーションやユーザを欺いて、意図しない場所からリソースを要求するように誘導します。リソースの場所をスプーフィングすることによって、攻撃者は代わりのリソースを使用させることができます。多くの場合、攻撃者が制御し、悪意のある目標を達成するために使用されます。
154 159	ライブラリへのアクセスをリダイレクトする	リソースの位置をなりすまし	攻撃者は、外部ライブラリへの呼び出し実行フローを悪用して、攻撃者が提供するライブラリまたはコードを実行させます。不正なコードの実行によって攻撃者がアプリケーションやサーバを侵害する可能性があります。アプリケーションは通常、アプリケーションの外部のライブラリの一部である関数を呼び出します。これらのライブラリは、オペレーティングシステムの一部でも、サードパーティライブラリでもかまいません。攻撃者がアプリケーションによるこれらのライブラリへのアクセスを攻撃者が提供する他のライブラリにリダイレクトできる場合、ターゲットアプリケーションに任意のコードを実行させることを強制できます。これは、対象となるアプリケーションの特権が与えられている場合に特に危険です。アクセスは、シンボリックリンクの使用、検索パスの変更、相対パスの操作など、さまざまな方法でリダイレクトできます。
154 616	不正な位置を確立する	リソースの位置をなりすまし	攻撃者は、正当なリソースがある場所の類似した場所にリソースの悪意のあるバージョンを配置します。悪意のある場所を確立した後、攻撃者は犠牲者がその場所を閲覧し、悪質なリソースにアクセスするのを待ちます。
173	アクションスプーフィング	人を欺くやりとりを行う	攻撃者は、ある行動を別の行動に変えることができます。ユーザが異なる動作を開始しようとするときに、あるタイプの別の動作を開始し、ユーザを騙します。たとえば、ユーザがボタンをクリックするとクエリが送信されると考えられるかもしれませんが、実際にはソフトウェアをダウンロードします。攻撃者は、犠牲者にアクションを実行させるか、ユーザの性癖を利用するなど、社会的な手段でこの攻撃を実行することがあります。またはユーザが1つのインターフェースを見ているつもりでも、実際には第2の目に見えないインターフェースとやり取りしているクリックジャック攻撃のような技術的手段を介して行われます。
173 103	クリックジャッキング	アクションスプーフィング	クリックジャック攻撃では、被害者は、まったく異なるシステムとUIによってやりとりしながら、知らないうちに別のシステムで何らかのアクションを開始するように誘導されます。ターゲットシステムにログインしている間に、被害者は、やりとりしたいUIを表示している攻撃者の悪意のあるサイトにアクセスします。クリックジャックされたページには、実際に閲覧しているUIの上に透明なレイヤーがあり、攻撃者が被害者に実行させたいアクションをコントロールします。被害者は、ページ上に表示されるボタンまたは他のUI要素をクリックすると、透明な重なった層内のアクションコントロールを起動します。そのアクションコントロールにより、被害者は認証されたターゲットシステム内で潜在的に特権（および最も望ましくない）機能を実行するように騙すかもしれません。
173 506	タップジャック攻撃	アクションスプーフィング	攻撃者は、事前にインストールされた悪意のあるアプリケーションを介してユーザを誤認させるインターフェースを表示し、被害者が攻撃者が望む画面上の場所をタップするように促します。これは、1つの画面を別の画面の上に重ねて表示しながら、1つの画面を表示することによって実現されることがよくあります。これを達成するために使用される主な技術は2つあります。1つ目は、透明なプロパティを活用して、画面上のタップを可視アプリケーションからバックグラウンドで実行されているアプリケーションに渡すことです。2つ目は、可視的な画面の上に小さなオブジェクト（例えば、ボタンまたはテキストフィールド）を配置し、基礎をなすアプリケーションの一部であるように見せることです。 どちらの場合も、ユーザは画面をタップしていると考えていますが、実際にやりとりしているアプリケーションは認識していません。
416	人間の行動を操作する	人を欺くやりとりを行う	攻撃者は、ターゲットとなる個人またはグループに影響を与えて情報を求めたり、ターゲットを操作して攻撃者の利益に役立つアクションを実行するため、人間の心理的性質を利用します。多くのソーシャルエンジニアリング技術は完全な欺瞞を伴いません。多くの場合、障壁を取り除くために標的を操作します。ターゲットが快適に感じられるようにし、ターゲットが情報を直接共有するようにするか、重要な情報を意図せずに取得します。熟練な攻撃者は、所望の結果を得るのに適切な場合、これらの技術を使用します。操作の技法は、攻撃者の発言や思考パターンに誘惑させる上司やヘルプディスクの偽装などさまざまです。

416 407	プリテキスティング	人間の行動を操作する	攻撃者は、ターゲットからの情報を要求したり、ターゲットを操作して攻撃者の利益に役立つ何らかのアクションを実行させるため、プレテクスティングを行います。攻撃者はターゲットとする被害者に情報を公開したり、何らかの措置を講ずるように識別情報や役割を仮定するシナリオを考案します。嘘をつくだけではなく、場合によっては、受信者を操作するために完全な新しい識別情報を作成し、使用するかもしれません。プレテクスティングは、攻撃者が経験したことがない特定の仕事や役割の人を偽装するのにも使用できます。この攻撃で簡単にターゲットに関する情報を知ることができます。より複雑なやりとりとしては、攻撃者が組織の弱点の悪用や安全な施設・システムへのアクセスを得るための何らかのアクションをターゲットに実行するように要求するかもしれません。良い情報収集技術は、良い口実を作り、標的を破壊することができます。確かな口実は、信頼を築くために不可欠な部分です。 攻撃者のエイリアス・ストーリー・アイデンティティに弱点がある、信頼性がない信ぴょう性がない場合でも、ターゲットは被害を受ける場合があります。
416 417	認知の影響	人間の行動を操作する	攻撃者はソーシャルエンジニアリングを使用して、攻撃者とターゲットとの関係におけるターゲットの認知を悪用します。 この攻撃の目標は、無意識のうちに攻撃者に有利な情報を漏らしたりするようにターゲットを誘導することです。
416 425	フレーミングによるターゲットの影響	人間の行動を操作する	攻撃者はフレーミング技法を使用して対話を文脈化し、ターゲットが攻撃者から影響を受けやすいようにします。フレーミングは、私たちがしなければならない決心の反応方法を変える情報と経験です。この攻撃は、ターゲットからの否定的または拒否的な反応を回避しつつ、被害者がデータおよびその意義を認知するように調整する方法を利用します。特定のテクニックではなく、フレーミングは会話の方法論であり、ゆっくりとターゲットが攻撃者の視点に立つように誘導します。フレーミングの1つの技法は、「いいえ」という言葉の使用を避け、肯定的な方法で応答を文脈化することです。巧みに実行されると、ターゲットは情報を自ら提供したり、攻撃者に有利な行動を起こす可能性が非常に高くなります。
416 426	インセンティブによる影響	人間の行動を操作する	攻撃者は、何かの影響を操作することで、ターゲットの行動を誘発します。一般的に財政的、社会的、またはイデオロギー的インセンティブに関連しています。例としては、金銭的な詐欺、仲間の圧力、ターゲットの倫理、倫理を犠牲にすることなどがあります。1つのターゲットに対する最も効果的なインセンティブは、他のターゲットには効果的でない可能性があります。攻撃者はターゲットの脆弱性に関する情報を収集し、効果的なインセンティブを特定する必要があります。
416 427	心理学的原則による影響	人間の行動を操作する	攻撃者は、人間のインタラクションや学習の方法に焦点を当て、認知社会心理学などの要素を活用して、ターゲットの行動を操作します。様々な方法で、ターゲットに行動を起こさせるか、または行動を起こすように影響を与えることができます。どのように人間が特定のシナリオやきっかけに反応するのかについての研究を活用します。
416 152	予期しない項目を挿入する		この攻撃は、データ入力用のインターフェイスを介して送信される細工されたデータ、またはターゲットシステムへの悪質なコードのインストールと実行により、ターゲットの動作を制御または中断する能力に焦点を当てています。前者は、攻撃者がアプリケーションによって解釈される入力に攻撃コード等を追加して、対象のアプリケーションに管理者が意図しないステップを実行させたり、アプリケーションを不安定な状態にします。後者の攻撃は、(バッファオーバーフローの場合)入力の解釈を有効にして、ユーザ提供のデータを保持する基盤となる構造を破壊するという攻撃を行います。(整数型攻撃の場合)ターゲットアプリケーションが正しく処理できない値が発生する可能性があります。インジェクション攻撃では、入力はアプリケーションによって解釈されますが、攻撃者はターゲットアプリケーションが従う解釈関数に攻撃命令を含めます。
137	パラメータの注入	予期しない項目を挿入する	攻撃者は、入力の妥当性の弱点を悪用し、要求されるパラメータの内容を操作することで、ターゲットのセキュリティを侵害します。いくつかのパラメータエンコーディングは、分離文字としてテキスト文字を使用します。たとえば、HTTP GETメッセージのパラメータは、アンパサンド(&)で区切られた名前と値のペアとしてエンコードされます。攻撃者がこれらのパラメータを埋めるために使用されるテキスト文字列を提供できる場合、エンコーディングスキームで使用される特殊文字を挿入してパラメータを追加または変更できます。たとえば、ユーザの入力がHTTP GET要求に直接入力され、ユーザが値 "myInput&new_param=myValue"を指定した場合、入力パラメータはmyInputに設定されますが、新しいパラメータ(new_param)もmyValueの値に追加されます。これにより、サーバによって処理される入力の意味が大幅に変更される可能性があります。パラメータが識別され、テキスト文字で区切られたコード体系は、この攻撃に対して潜在的に脆弱です。上で使用されたHTTP GETエンコーディングは単なる例にすぎません。

137 134	メールヘッダーインジェクション	パラメータの注入	攻撃者は、プロトコル固有の区切り文字を使用してデータを注入することによって、電子メールメッセージのヘッダーと内容を操作します。多くのアプリケーションでは、ユーザはフィールドに記入して電子メールメッセージを送信できます。たとえば、Webサイトには、サイトを友人と共有するためのリンクがあります。このリンクには、ユーザが受信者の電子メールアドレスを入力し、Webアプリケーションによって、件名や本文などの他のすべてのフィールドが入力されます。このパターンでは、攻撃者は、メールメッセージのヘッダーを構成する入力フィールドに追加のコンテンツを挿入することによって、ヘッダーと本文の情報を電子メールメッセージに追加します。この攻撃は、RFC 822がメールメッセージのヘッダーを改行で区切る必要があるという事実を悪用しています。その結果、攻撃者は、区切りの復帰改行を追加し、新しい見出しと本文情報を入力するだけで、新しいヘッダーやコンテンツを挿入することができます。本文の復帰改行が通常の文字として扱われるため、ユーザがメッセージ本文のみを提供できる場合、この攻撃は機能しません。
137 135	フォーマット文字列のインジェクション 書式文字列攻撃	パラメータの注入	攻撃者は、ターゲットアプリケーションの文字列入力フィールドに書式文字列を含めます。ほとんどのアプリケーションでは、ユーザが静的テキストを入力し、フォーマット文字の存在に予期せず応答することが想定されます。たとえば、printfなどのCプログラミング言語の特定の関数では、書式設定文字%sはこの場所に文字列を識別するメモリ位置の内容を出力し、書式設定文字%nはメモリに書き込まれたDWORD数を出力します 。攻撃者はこれを使用してメモリの場所やファイルを読み書きしたり、予期しない方法でテキストの値を操作することができます。攻撃者がプログラムスタックに書き込むことができる場合、メモリを読み書きによりプログラムがクラッシュし、メモリの書き込みにより任意のコードが実行される可能性があります。
137 138	反射インジェクション	パラメータの注入	攻撃者はターゲットアプリケーションに値を入力します。この値はリフレクションメソッドにより、クラス・メソッド・フィールドを識別するために使用されます。例えば、Javaプログラミング言語ではリフレクションライブラリは、アプリケーションがクラスとそのコンポーネントを名前で検査、ロード、呼び出しできるようにします。攻撃者がクラス/メソッド/フィールドの名前やメソッドに渡されるパラメータなど、これらのメソッドへの入力を制御できる場合、ターゲットアプリケーションが不正なメソッドを呼び出す、ランダムなフィールドを読み込む、攻撃者が作成した悪質なクラスを読み込んで利用する可能性があります。これにより、機密情報を取得するアプリケーション、不正確な結果を返すアプリケーション、攻撃者が制御するアプリケーションにつながる可能性があります。
137 15	コマンド区切り文字	パラメータの注入	この攻撃はプログラムの脆弱性を悪用し、攻撃者のコマンドをファイルシステムやデータベースなどの他のリソースを対象とした目的で正当なコマンドに連結します。ホワイトリスト検証とは対照的に、フィルタまたはブラックリストの入力検証を使用するシステムは、フィルタまたはブラックリストに存在しない区切り文字（または区切り文字の組み合わせ）を予測する攻撃者にとって脆弱です。他のインジェクション攻撃と同様に、攻撃者はコマンドデリミタペイロードをエントリポイントとして使用してアプリケーションをトンネリングし、SQLクエリ、シェルコマンド、ネットワークスキャンなどによる追加の攻撃を有効にします。
137 182	フラッシュインジェクション	パラメータの注入	攻撃者は、被害者を騙して攻撃者が指定したフラッシュコールやコマンドを実行する悪質なフラッシュコンテンツを実行します。この攻撃の一例はクロスサイトフラッシング（XSF）であり、攻撃者が指定したコンテンツからリファレンスコールへの攻撃者制御のパラメータが読み込まれます。
137 6	引数の挿入	パラメータの注入	攻撃者は、公開されたサービスやメソッドの検証されていない引数やフィルタされていない引数をターゲットに使用して、データやコマンド構文を注入することによって、対象となるアプリケーションの動作や状態を変更します。
175	コードの包含	予期しない項目を挿入する	攻撃者は標的上の弱点を利用して、任意のコードをローカルまたは遠隔地から取得して実行させます。コードインジェクションがコードの直接の包含を伴うのに対し、コードの包含はコードファイルへの参照の追加または置換を伴い、コードファイルはターゲットによってロードされ、あるアプリケーションのコードの一部として使用されるという点で、コードインジェクションとは異なります。
175 251	ローカルコードのインジェクション	コードの包含	攻撃者は、アプリケーションがローカルマシンから任意のコードファイルをロードするように強制します。攻撃者はこれを使用して、既知の脆弱性を持つライブラリファイルの古いバージョンをロードしたり、攻撃者が以前の攻撃時にローカルマシンに置いたファイルをロードしたり、予期しない方法でターゲットアプリケーションの機能を変更することができます。
175 253	リモートコードのインジェクション	コードの包含	攻撃者はアプリケーションにリモートから任意のコードファイルをロードさせます。攻撃者はこれを使用して、既知の脆弱性を持つライブラリファイルの古いバージョンをロードしたり、攻撃者が以前の攻撃時にリモートマシンに置いたファイルをロードしたり、予期しない方法でターゲットアプリケーションの機能を変更できます。
240	リソースインジェクション	予期しない項目を挿入する	攻撃者はリソースの意図しない変更または指定を可能にするリソース識別子を操作することによって、入力検証の弱点を悪用します。
240 610	セルラーデータのインジェクション		攻撃者は、モバイル技術のトラフィック（データフローまたはシグナリングデータ）にデータを注入して、通信を中断させたり、追加の監視操作を実行したりします。

242	コードのインジェクション	予期しない項目を挿入する	攻撃者はターゲット上の入力検証の弱点を利用して、現在実行中のコードに新しいコードを注入します。これは「コードの包含」とは異なります。コードを包含することには、コードファイルへの参照の追加または置換が含まれ、その後、ターゲットによってロードされ、アプリケーションのコードの一部として使用されます。
242 19	スクリプト内でのスクリプトの埋め込み	コードのインジェクション	この攻撃は、リモートホストがスクリプトを実行できるようにすることにより生起するプログラムの脆弱性を悪用します。攻撃者は、この機能を利用して、ターゲットソフトウェアが実行する可能性のある他のスクリプトに自分のスクリプトを埋め込みます。攻撃者は、実行される可能性が高いスクリプトにスクリプトを注入する能力を備えていなければなりません。これにより、攻撃者は潜在的にWebサーバのローカル環境、DMZ、Webサーバが通信できるバックエンドリソース、その他のホストに対してさまざまな探索と攻撃を開始できます。WAFl、ネットワークデバイス、JVMやWebサーバを持つプリンタなどの中間媒体が増えているため、攻撃者が悪意のあるスクリプトを挿入できる場所が数多くあります。
242 23	ファイル内容のインジェクション	コードのインジェクション	この攻撃は、バイナリファイルを含むリモートコンテンツの実行に対するホストの信頼を利用します。ファイルは攻撃者によって悪意のあるペイロード（標的ソフトウェアがアクセス可能なファイルシステムを標的とする）で被害を与えます。電子メールやPDF・マルチメディアファイルなどの標準的なWebコンテンツなどの標準チャネルを経由する可能性があります。攻撃者は、既知の脆弱性を悪用したり、ターゲットプロセスのルーチンを処理します。この種の脆弱性は、Microsoft OfficeからAdobe AcrobatおよびApple Safari Webブラウザに至るまで、さまざまな商用アプリケーションで検出されています。 攻撃者が標準的な処理ルーチンを認識し、脆弱性やエントリポイントを特定できる場合、正常に見えるコンテンツによって悪用される可能性があります。攻撃が実行されると、攻撃者のプログラムは、C:¥Program Filesや他の標準システムディレクトリなどの相対ディレクトリにアクセスして、さらなる攻撃を開始できます。最悪のシナリオは、これらのプログラムは他の伝播ロジックと組み合わされ、ウイルスとして動作します。
242 468	一般的なクロスブラウザ、クロスドメイン窃取	コードのインジェクション	攻撃者は、カスケーディングスタイルシート（CSS）注入を利用して、被害者のブラウザからデータのクロスドメインを盗みます。この攻撃は、CSSの読み込みに関連する標準を悪用します。1.任意のCSS負荷（クロスドメインを含む）でCookieを送信します。2.返されたCSSを解析し、CSSパーサによって有効なCSS記述子が見つかる前に意味をなさないデータがすべて無視されます。被害者のドメイン内のテキストを制御することにより、攻撃者は一見有効なCSS文字列を注入することができます。このCSS文字列の前に他のデータがあるかどうかは関係ありません。CSSパーサはCSS文字列を探します。攻撃者が2つのインジェクションポイントを制御できる場合、攻撃者が制御するサイトでレンダリングを実行している間にインジェクションされたCSSを参照されたとき、攻撃者はこの攻撃を使用して2つのCSSインジェクションポイント間のすべてのデータを盗み出すことができます。レンダリング時に、CSSパーサは有効なCSS文字列を検出して、意味のないデータを解析して無視します。データは単純にレンダリングされます。 データは実際には攻撃者がクロスドメインを盗んだだけのデータです。盗まれたデータには、CSRF保護トークンなどの機密情報が含まれている場合があります。
242 63	クロスサイトスクリプティング（XSS）	コードのインジェクション	攻撃者は、Webブラウザーに配信されるコンテンツに悪質なスクリプトを埋め込みます。この攻撃の目的は、ターゲットソフトウェア（クライアント側のブラウザ）がユーザの権限レベルでスクリプトを実行することです。この種の攻撃は、リモートホストがコードやスクリプトを実行できるようにすることでもたらされるプログラムの脆弱性を悪用します。たとえば、Webブラウザには単純なセキュリティコントロールが用意されていますが、リモートの攻撃者がスクリプトを（掲示板のようなユーザ生成のコンテンツに注入して）実行できる場合、これらのコントロールはバイパスされる可能性があります。さらに、エンドユーザがこれらの攻撃を検出することは非常に困難です。
248	コマンドインジェクション	予期しない項目を挿入する	攻撃者が任意の命令を実行しようと際、既存のコマンドに新しい項目を挿入し、意図されたものから解釈を変更します。このコンテキストのコマンドは、多くの場合、ダウンストリームコンポーネントによって解釈され、特定の応答を引き起こす独立した文字列です。このタイプの攻撃は、信頼できない値を使用してこれらのコマンド文字列を作成する場合に可能です。入力検証やコマンド構築の弱点は、攻撃を可能にし、攻撃を成功させることにつながります。
248 136	LDAPインジェクション	コマンドインジェクション	攻撃者はLDAPクエリを操作または作成し、ターゲットのセキュリティを侵害します。一部のアプリケーションでは、ユーザ入力を使用してLDAPサーバによって処理されるLDAPクエリを作成します。たとえば、ユーザは認証時にユーザ名を入力し、ユーザ名は認証プロセス中にLDAPクエリに挿入されます。攻撃者はこの入力を使用して、機密情報を開示する可能性のあるLDAPクエリに追加のコマンドを注入することができます。たとえば、前述のクエリに*を入力すると、システム上のすべてのユーザに関する情報が返されることがあります。この攻撃は、追加の情報を収集するか、特定の戻り値を強制するためにクエリを操作する点で、SQLインジェクション攻撃とよく似ています。

248 183	IMAP / SMTPコマンドインジェクション	コマンドインジェクション	攻撃者はIMAP / SMTPサーバの入力検証の弱点を利用して、サーバ上でコマンドを実行します。Webメールサーバは、インターネットとIMAPまたはSMTPメールサーバの間に置かれることがよくあります。ユーザ要求はWebメールサーバによって受信され、Webサーバは要求された情報をバックエンドメールサーバに照会し、この応答をユーザに返します。IMAP / SMTPコマンドインジェクション攻撃では、メールサーバコマンドは、Webメールサーバに送信された要求の一部に埋め込まれます。Webメールサーバがこれらの要求を適切にサニタイズできない場合、これらのコマンドはWebメールサーバによって照会されたときにバックエンドメールサーバに送信され、そこでコマンドが実行されます。この攻撃は、管理者がバックエンドサーバが直接インターネットアクセスから保護されていると思い込み、悪意のあるコマンドの実行に対して適切に保護されない可能性があるため、特に危険です。
248 40	書き込み可能なターミナルデバイスの操作	コマンドインジェクション	攻撃者は他のユーザによって書き込みが可能なターミナルデバイスを悪用します。攻撃者は、標的のユーザがenterを押して悪意のあるコマンドを特権で実行することを望むコマンド文字列を標的ターミナル装置に送ります。攻撃者は、(/etc/passwdをコピーするなどの)結果を既知のディレクトリに送信します。攻撃が成功すると所望の情報を収集することができます。
248 250	XMLインジェクション	コマンドインジェクション	攻撃者は、SQLインジェクションと同様の手法により、細工されたXMLのユーザ制御可能な入力を使用して、XMLデータベースにデータをプローブ・攻撃・インジェクションします。ユーザが制御可能な入力により、認証やフロントエンドアプリケーションをバイパスして直接XMLデータベースにアクセスし、場合によってはデータベース情報を変更するなど、権限のないデータの閲覧が可能になります。
248 66	SQLインジェクション	コマンドインジェクション	この攻撃は、ユーザの入力に基づいてSQL文を構成するターゲットソフトウェアを悪用します。ターゲットソフトウェアが入力に基づいてSQL文を構成しているとき、攻撃者は、結果として得られるSQL文がアプリケーションが意図した以外のアクションを実行するように、入力文字列を作成します。SQLインジェクションは、アプリケーションが入力を適切に検証するのに失敗した場合に発生します。SQL構文の一部として適切に検証されていない、特別に細工されたユーザ制御の入力がSQL構文で構成されている場合、アプリケーション設計時に想定されていない方法でデータベースから情報を収集できます。データベースとアプリケーションの設計に応じて、インジェクションを活用して、データベースに攻撃者が選択したシステム関連のコマンドを実行させることも可能です。SQLインジェクションにより、攻撃者はデータベースと直接やりとりでき、アプリケーションを完全にバイパスできます。インジェクションが成功すると、情報漏洩とデータベース内のデータの追加または変更が可能になります。
248 88	OSコマンドインジェクション	コマンドインジェクション	この攻撃は、攻撃者がオペレーティングシステムコマンドを既存のアプリケーション機能に注入します。信頼できない入力でコマンド文字列を作成するアプリケーションは脆弱です。攻撃者はアプリケーションのOSコマンドインジェクションを用いて特権を昇格します。任意のコマンドを実行し、オペレーティングシステムに侵入することができます。
549	コードのローカル実行 標的型マルウェア	予期しない項目を挿入する	攻撃者は、組織の情報テクノロジ環境における既知の脆弱性を利用する標的型マルウェアを開発します。これらの攻撃のために作られたマルウェアは、特に、テクノロジ環境について収集された情報に基づいています。マルウェアを正常に実行されることにより、さまざまな負の影響を与えます。
624	フォルトインジェクション	予期しない項目を挿入する	攻撃者は破壊的な信号またはイベント(例えば、電磁パルス、レーザパルス、クロックグリッチなど)を使用して、電子デバイスへの誤動作を引き起こします。暗号操作を実行するデバイスで制御された方法で実行されると、この誤動作を利用して秘密鍵情報を導き出すことができます。
624 625	モバイルデバイスへのフォルトインジェクション	フォルトインジェクション	モバイルデバイスに対するフォルトインジェクション攻撃は、障害のある動作を引き起こすために破壊的な信号またはイベント(例えば、電磁パルス、レーザパルス、クロックグリッチなど)を使用します。暗号操作を実行するデバイスで制御された方法で実行されると、この誤動作を利用して秘密鍵情報を導き出すことができます。この攻撃は通常、モバイルデバイスの物理的な制御を必要としますが、非破壊であり、秘密鍵が侵害されたという兆候なしに攻撃後にデバイスを使用することができます。
594	トラフィックインジェクション 接続リセット	予期しない項目を挿入する	この攻撃では、攻撃者はターゲットの接続の一方または両方に接続リセットパケットを注入します。したがって、攻撃者はターゲットサーバおよび/または宛先サーバの間でトラフィックを直接フィルタリングする必要なく、接続を切断することができます。
594 596	TCP RST インジェクション	トラフィックインジェクション 接続リセット	攻撃者は、ターゲットがHTTP GET要求を行った後、1つ以上のTCP RSTパケットをターゲットに注入します。この攻撃の目的は、ターゲットおよび/または宛先WebサーバにTCP接続を終了させることです。

586	オブジェクトインジェクション	予期しない項目を挿入する	攻撃者はシリアル化されたオブジェクトの処理中に追加の悪質なコンテンツを注入することによって、アプリケーションを悪用しようとします。開発者はデータや状態を静的なバイナリ形式に変換してディスクに保存したり、ネットワーク経由で転送したりするためにシリアル化を利用します。これらのオブジェクトは、データ/状態を回復するために必要なときにデシリアライズされます。脆弱なアプリケーションに不正な形式のオブジェクトを挿入することにより、攻撃者は、デシリアライズ処理を操作することによってアプリケーションを潜在的に危険にさらす可能性があります。これにより、リモートでコードが実行されるなど、さまざまな望ましくない結果が生じる可能性があります。
255	データ構造を操作する		この攻撃は、構造の意図された使用法および保護に違反するため、システムデータ構造の特性を操作・悪用します。これは、システムがデータ構造をどのように処理し、管理するかの脆弱性のため、関連するシステムデータへの不適切なアクセスまたはシステム自体のセキュリティプロパティの違反のいずれかをもたらすような方法で行われます。これらのデータ構造の脆弱性および悪用可能性は、その設計および規定された処理におけるあいまいさ、および、思い込みのために生じます。
123	バッファの操作	データ構造を操作する	攻撃者は、アクセス権を与えられていないデータを読んだり変更するため、アプリケーションとバッファとのやりとりを操作します。バッファ攻撃は、バッファの内容を解釈するコードではなく、攻撃の対象となるバッファ空間自体である点で区別されます。ほぼすべてのバッファ攻撃では、バッファに格納されるコンテンツは重要ではありません。代わりに、ほとんどのバッファ攻撃は、割り当てられたバッファに格納できる以上の入力を取得または送信し、意図しない他のプログラムメモリを読み書きします。
123 100	バッファオーバーフロー	バッファの操作	バッファオーバーフロー攻撃は、通常、攻撃者によって注入された入力によってトリガされるバッファ操作上の不適切または欠落した境界チェックを対象とします。結果として、攻撃者はメモリ内に割り当てられたバッファ領域の境界を越えて書き込みを行うことができ、攻撃者の選択に応じてプログラムのクラッシュや潜在的な実行のリダイレクトを引き起こす可能性があります。
123 540	バッファオーバーリード	バッファの操作	攻撃者は、アプリケーションに定義されたバッファの境界を超えて読み取られる入力を送信することによって、ターゲットを攻撃します。これは読み取りを開始または停止する場所に影響を与える値が、バッファの有効メモリ位置外の位置を参照するように設定されている場合に生起します。この攻撃は、機密情報の漏洩、システムクラッシュ、任意のコードの実行につながる可能性があります。
124	共有データの操作	データ構造を操作する	攻撃者は、複数のアプリケーションまたはアプリケーションプール間で共有されるデータ構造を利用して、アプリケーションの動作に影響を与えます。データは、複数のアプリケーション間、または単一のアプリケーションの複数のスレッド間で共有することができます。データ共有は、通常、単一のメモリロケーションへの相互アクセスによって実行されます。攻撃者がこの共有データを操作できる場合（通常、アプリケーションまたはスレッドの1つを選択することによって）、共有データを使用する他のアプリケーションまたはスレッドは、共有データの妥当性を信頼し続け、計算に使用します。これにより、信頼の前提が無効になり、共有データの他のユーザの通常の操作による追加データの破損、または共有アプリケーションのクラッシュなどの原因となる可能性があります。
129	ポインタの操作	データ構造を操作する	標的アプリケーション内のポインタが攻撃者の操作を含み、その結果、アプリケーションは意図しないメモリ位置にアクセスします。これにより、アプリケーションがクラッシュしたり、特定のポインタ値に対して通常は不可能なデータにアクセスしたり、任意のコードが実行されたりする可能性があります。ポインタは単に整数型の変数なので、ポインタ攻撃では整数型の攻撃がよく使われることがあります。
153	入力データの操作	データ構造を操作する	攻撃者は、入力処理インターフェースへのデータのフォーマット、構造、および構成を制御することで、入力値の検証の弱点を悪用します。非標準または予期しない形式の入力を実行することにより、攻撃者はターゲットのセキュリティに悪影響を与える可能性があります。たとえば、異なる文字エンコーディングを使用することで危険なテキストが安全なテキストとして扱われる可能性があります。または、攻撃者はファイル拡張子などの特定のフラグを使用して、ターゲットアプリケーションを「提供されたデータは、データが実際には適切なタイプでないときに、特定のインタプリタを使用して処理されるべきである。」と信じさせるかもしれません。これによりは、保護メカニズムを回避し、ターゲットが入力処理に特定のコンポーネントを使用するようにしたり、ユーザのデータを他の方法で処理することがあります。
153 126	パストラバーサル ディレクトリトラバーサル	入力データの操作	攻撃者は、パス操作により、ターゲットの入力検証が不十分であることを悪用して、通常の整形式要求では検索できないデータにアクセスします。典型的な攻撃には、ドット・スラッシュ文字と一緒に目的のファイルへのパスを指定します。その結果、ファイルアクセスAPIまたは関数が意図されたディレクトリ構造からルートファイルシステムに移動します。期待される経路情報を置換または変更することにより、アクセス機能またはAPIは、攻撃者が望むファイルを検索します。これらの攻撃では、攻撃者が目的のファイルへの完全なパスを入力するか、制御文字（たとえば、パス区切り文字（/）やドット（.））を使用して目的のディレクトリまたはファイルにアクセスします。

153 128	整数型攻撃 整数オーバーフロー攻撃	入力データの操作	攻撃者は整数変数の構造を悪用して、これらの変数がアプリケーションによって期待されない値になるようにします。たとえば、符号付き整数変数の最大の正の整数に1を加算すると、負の数になります。負の数はアプリケーションでは不正である可能性があり、攻撃者がアプリケーションから直接それらを提供するのを防ぐ可能性があります。しかし、アプリケーションによっては、2つの正の数を加算し、整数の格納形式の構造に負の数を加えられることを考慮している場合があります。
153 267	代替エンコーディングの利用	入力データの操作	エンコーディング標準を検証すると、入力フィルタリングが難しくなるので、エンコードを検証していない・効果的ではないアプリケーションに有害な入力をエンコードし、送る可能性を利用します。
262	システムリソースを操作する		この攻撃パターンは、望ましい結果を達成するために1つまたは複数のリソースを操作する攻撃者の能力に焦点を当てています。攻撃者がリソースの状態の一部を変更し、システムの動作（可用性）や情報の整合性に影響を及ぼすことができる幅広い種類の攻撃です。リソースの例には、ファイル、アプリケーション、ライブラリ、インフラストラクチャ、および構成情報が含まれます。この攻撃による結果は、破壊行為からサービスの停止、ターゲットマシン上の任意のコードの実行までさまざまです。
161	インフラストラクチャの操作	システムリソースを操作する	攻撃者は、ネットワークエンティティのインフラストラクチャの特性を悪用します。ネットワークオブジェクト上の攻撃や情報収集を実行し、ネットワークオブジェクト間の通常の情報フローの変化を生じさせます。ほとんどの場合、これにはネットワークメッセージのルーティングが含まれているため、適切な宛先に到着せず、攻撃者が選択したエンティティ（通常は攻撃者によって制御されるサーバ）に誘導されます。被害者は、自分のメッセージが正しく処理されていないことに気付くことが困難です。たとえば、標的となったクライアントは自分の銀行に接続していると思っていても、実際には実際の銀行口座を乗っ取るためにユーザのログイン情報を収集する攻撃者が管理するフィッシングサイトに接続している可能性があります。
161 141	キャッシュポイズニング	インフラストラクチャの操作	攻撃者は、キャッシュ技術の機能を悪用して、攻撃の目的達成に役立つ特定のデータをキャッシュします。これは、攻撃者が不正または有害なデータをキャッシュに置く攻撃です。標的となるキャッシュは、アプリケーションのキャッシュ（例えば、ウェブブラウザキャッシュ）またはパブリックキャッシュ（例えば、DNSまたはARPキャッシュ）です。キャッシュがリフレッシュされるまで、ほとんどのアプリケーションまたはクライアントは、破損したキャッシュ値を有効として扱います。これは、マルウェアをインストールするサイトへのWebブラウザのリダイレクトや不正確な値に基づく誤った計算など、さまざまな悪用につながります。
161 268	監査ログの操作	インフラストラクチャの操作	攻撃者は、悪意のあるログエントリをログファイルに挿入、操作、削除、または偽造し、ログファイルの監査を誤解させたり、攻撃の痕跡を隠そうとしています。ログファイルのアクセス制御やロギングメカニズムが不十分であるため、攻撃者は上記の操作を実行できます。
161 571	セントラルリポジトリへのログ記録をブロックする	インフラストラクチャの操作	攻撃者は、インジケータがホストマシンから送信されるのを阻止しようとするかもしれません。インジケータを報告するネットワークの場合、攻撃者は、監視部署のよる分析を妨ぐために、報告に関連するトラフィックをブロックするかもしれません。これは、特定のサーバへのトラフィックをブロックするため、ホストベースのファイアウォールルールを作成してローカルプロセスを停止するなどの多くの手段で実行できます。
165	ファイルの操作	システムリソースを操作する	攻撃者は、アプリケーションが誤って処理してしまうようにファイルの内容や属性（拡張子や名前など）を変更します。攻撃者は、この攻撃により、アプリケーションを不安定な状態にする、機密情報を上書きする、アプリケーションの特権で任意のコードを実行することができます。この攻撃手法は、ファイル操作によりバッファオーバーフローや誤ったインタプリタの使用など、意図しない動作をファイル処理により発生する構成情報への攻撃とは異なります。構成情報への攻撃は、有害な構成情報を挿入するため、ファイルを正しく解釈するアプリケーションに依存します。また、リソースロケーション攻撃は、アプリケーションがファイルを保存する能力を制御することに依存します。ファイル操作攻撃では、上記2つのクラスの攻撃が組み合わせていたとしても、アプリケーションがデフォルト以外の場所を参照しません。
165 17	悪意のあるファイルの利用	ファイルの操作	攻撃者がシェルアクセスなどの実行可能ファイルに直接アクセスできるようにするシステムの設定を悪用します。 または、攻撃者はファイルをアップロードし、実行できるようにします。 多くの統合ポイントを持つWebサーバ、FTPサーバ、およびメッセージ指向のミドルウェアシステムは、プログラマーと管理者の両方が各インターフェースの正しい特権について同期しなければならないため、特に脆弱です。

165 177	より高い分類で保護されたファイルと同じ名前でファイルを作成する	ファイルの操作	攻撃者は、保護されたファイルまたは特権のあるファイルと同じ名前のファイルを作成することでオペレーティングシステムまたはアプリケーションのファイル保存場所でアルゴリズムを悪用します。攻撃者の作成したファイルが、オペレーティングシステムまたは元のファイルをロードしようとするアプリケーションコンポーネントによって信頼されている場合、攻撃者はシステムを操作できます。アプリケーションでは、ライブラリや設定ファイルなどの外部ファイルを読み込んだり、組み込んだりすることがあります。これらのファイルは、悪意のある操作から保護される必要があります。 ただし、アプリケーションがファイルの場所を特定するときにファイル名のみを使用する場合、攻撃者は同じ名前のファイルを作成して、正当なファイルを持つディレクトリが検索される前にアプリケーションが検索されるディレクトリに配置できます。攻撃者のファイルは最初に検出されるため、ターゲットアプリケーションによって使用されます。この攻撃は、参照ファイルが実行可能であり、かつ/または特定の名前を持つことにのみに基づいて特別な権限が与えられている場合、非常に破壊的である可能性があります。
165 635	偽のファイル名による代替実行	ファイルの操作	ファイル名の拡張子は、さまざまなコンテキストで頻繁に使用され、アプリケーションを実行するために使用されます。 攻撃者が別のアプリケーションを使用させる可能性がある場合、悪質なコードの実行、サービス拒否、機密情報を漏洩の可能性があります。
165 11	Webサーバの誤った分類を起こす	ファイルの操作	このタイプの攻撃は、ファイル名またはファイル拡張子に基づいてアクションを実行するWebサーバの処理を悪用します。異なるファイルタイプは異なるサーバプロセスによって処理されるため、誤って分類すると、Webサーバが予期しない動作をする可能性があります。これにより、サーバがリソースを使い果たしたり、デバッガやシステムデータを攻撃者に提供したり、攻撃者をリモートプロセスにバインドする可能性があります。この種の脆弱性は、IIS、Lotus Domino、Orionなど、広く使用されている多くのサーバで検出されています。攻撃者は、標準の通信プロトコルと通信方法が使用し、正当な要求の末尾に悪質な情報が付加します。攻撃のペイロードはさまざまで、ピリオドなどの特殊文字や、javaアプリケーションサーバのjspのようなサーバ側での操作に特別な意味を持つタグを単に追加したものもあります。この攻撃は、攻撃者がサーバがリクエストの名前に基づき機能を実行するように欺いています。
165 636	ファイル内に悪意のあるデータやコードを隠す	ファイルの操作	さまざまなオペレーティングシステム上のファイルは、内容に加えて他のデータの格納を可能にする複雑な形式を持つことができます。 多くの場合、画像ファイルのキャッシュサムネイルなど、ファイルに関するメタデータです。 ユーティリティが特定の方法で呼び出されない限り、このデータはファイルの通常の使用中は表示されません。 攻撃者は、これらの機能を使用して悪意のあるデータやコードを格納することは可能ですが、これを発見するのは困難です。
165 168	Windows :: 代替データストリーム	ファイルの操作	攻撃者は、Microsoft NTFSの代替データストリーム（ADS）の機能を利用して、システムのセキュリティを侵害します。 ADSは、複数のファイルを「ファイル名：ストリーム名」という形で1つのディレクトリエントリに格納することができます。 1つまたは複数の代替データストリームを任意のファイルまたはディレクトリに格納することができます。 通常のMicrosoftユーティリティは、ファイルに添付されたADSストリームの存在を表示しません。 ADSの追加領域は、表示されるファイルサイズには記録されません。 ADSの追加領域は、ボリューム上の使用済み領域で含まれます。 ADSはどんな種類のファイルでもよく、標準のMicrosoftユーティリティによってNTFSボリューム間でコピーされます。 ADSは、攻撃者がツール、スクリプト、データを通常のシステムユーティリティによる検出から逃れるために使用できます。 多くのアンチウィルスプログラムはADSのチェック・スキャンをしません。 Windows Vista等には、代替ストリームを表示するコマンドラインDIRコマンドにスイッチ（-R）があります。

165 35	非実行ファイルで実行可能コードを利用する	ファイルの操作	この攻撃は、システムが構成ファイルとリソースファイルを信頼していることを悪用します。実行可能ファイルがリソース（イメージファイルやコンフィギュレーションファイルなど）をロードするとき、攻撃者は悪質なコードを直接実行するようにファイルを修正するか悪意のある構成パラメータに基づいて実行するターゲットプロセス（例えば、アプリケーションサーバ）を操作することができます。システムは相互に関連しているため、ローカルおよびリモートソースからのリソースをマッシュアップ（加工・編集）するので、この攻撃が発生する可能性は高いです。マイクロソフトセキュリティ情報MS04-028のように、特別に細工されたJPEGファイルがブラウザにロードされるとバッファオーバーランを引き起こす可能性があるように、無害のように見えるイメージファイルのロードによりバッファオーバーランが発生するなど、クライアントシステムを攻撃することができます。別の例は、クライアントがPDFファイルを読み込む場合です。この場合、攻撃者は正当なpdfのURLの最後にjavascriptを追加するだけです。（http://www.gnucitizen.org/blog/danger-danger-danger/） http://path/to/pdf/file.pdf#whatever_name_you_want=javascript:your_code_here クライアントは、pdfを読んでいると思っていますが、攻撃者はリソースを修正し、実行可能なjavascriptをクライアントのブラウザプロセスにロードしています。この攻撃は、サーバプロセスをターゲットにすることもできます。攻撃者は、リソースファイルまたは構成ファイルを編集します。たとえば、J2EEアプリケーションサーバのセキュリティアクセス許可を設定するために使用されるweb.xmlファイルにロール名 publicを追加すると、パブリックロールを持つすべてのユーザが管理機能を使用できるようになります。サーバは設定ファイルを正しいものとして信頼しますが、操作されることで攻撃者は完全に制御します。
165 73	ユーザに制御されたファイル名	ファイルの操作	このタイプの攻撃には、悪意のあるキャラクタ（XSSリダイレクトなど）をファイル名に直接的または間接的に挿入し、ターゲットソフトウェアがHTMLテキストやその他の潜在的に実行可能なコンテンツを生成するために使用されます。 多くのWebサイトは、ユーザが作成したコンテンツに依存し、ユーザが提供するデータから直接ファイル、ファイル名、URLリンクなどのリソースを動的に構築します。 攻撃者はクライアントブラウザで実行できるコードをアップロード、クライアントブラウザを攻撃者が所有するサイトにリダイレクトします。 すべてのXSS攻撃のペイロードバリアントを使用して、これらの脆弱性を悪用することができます。
176	構成/環境の操作	システムリソースを操作する	攻撃者は、対象アプリケーションの外部にあるファイルや設定を操作して、そのアプリケーションの動作に影響を与えます。たとえば、多くのアプリケーションは外部構成ファイルとライブラリを使用します。これらのエンティティを変更したり、アプリケーションのそれを使用する能力に影響を与えることは、構成/環境の操作攻撃になります。
176 203	アプリケーションレジストリ値を操作する	構成/環境の操作	攻撃者は、さまざまな可能性のある攻撃を実行するためにアプリケーションによって使用されるレジストリ値を操作します。 多くのアプリケーションは、レジストリを使用して構成情報とサービス情報を格納しています。したがって、レジストリを操作する攻撃は、個々のサービスや対象となるアプリケーションの全体的な構成に影響を与える可能性があります。この攻撃はMicrosoft Windowsレジストリを参照するだけでなく、アプリケーションによって使用されるすべてのレジストリに適用されます。たとえば、Java RMIとSOAPは、使用可能なサービスを追跡するためにレジストリを使用します。レジストリ値の変更は、別の攻撃の一部として行われることがあります。たとえば、パスのトラバーサル（相対パス修飾子の挿入）またはバッファオーバーフロー（レジストリ値をアプリケーションの格納能力を超えて拡大する）です。しかし、多くのレジストリ値は長期的に使用されることを考慮すれば、レジストリ操作は独自の目的である可能性があります。
176 271	スキーマポイズニング	構成/環境の操作	攻撃者はスキーマの内容を破損・変更し、ターゲットのセキュリティを侵害します。スキーマは、アプリケーションで使用されるリソースの構造とコンテンツの定義を提供します。スキーマを置換または変更することにより、攻撃者はアプリケーションがリソースを処理または解釈する方法に影響を与えることができます。サービスの拒否、予期しない状態への移行、不完全なデータの記録につながることがよくあります。
176 578	セキュリティソフトウェアを無効にする	構成/環境の操作	攻撃者は、検出が行われないようにセキュリティツールを無効にすることができるかもしれません。これは、プロセスの強制終了、実行時にツールが起動しないようにレジストリキーを削除、ログファイルの削除等の形で行うことができるかもしれません。
176 75	書き込み可能な構成ファイルを操作する	構成/環境の操作	一般的に、これらは手動で編集されたファイルで、システム管理者のプレビューやこれらのファイルを変更するための攻撃者の能力には含まれません。たとえばCVSリポジトリでは、許可されたユーザと同じように、アプリケーションに直接不正なアクセスを行います。
184	ソフトウェアの完全性を攻撃する	システムリソースを操作する	攻撃者は、ユーザ、プログラム、サーバ、またはデバイスにアクションを実行する一連のイベントを開始します。ソフトウェアコード、デバイスデータ構造、またはデバイスファームウェアの完全性を損なうことで、ターゲットの整合性を変更して安全性の低い状態にします。
184 185	悪意のあるソフトウェアのダウンロード	ソフトウェアの完全性を攻撃する	攻撃者は不正な方法を使用して、ユーザまたは自動化されたプロセスに、攻撃者が制御する攻撃元から発信する危険なコードをダウンロード・インストールさせます。この攻撃戦略にはいくつかのバリエーションがあります。

184 186	悪意のあるソフトウェアの更新	ソフトウェアの完全性を攻撃する	攻撃者は不正な方法を使用して、ユーザまたは自動プロセスに攻撃者によって制御される情報源からの有効な更新を見せかけた危険なコードをダウンロード・インストールさせます。この攻撃戦略にはいくつかのバリエーションがありますが、すべての攻撃方法は、攻撃者が悪意のあるコンテンツを配置して偽装する能力に依存しているという点で同じです。つまり、プログラムによって処理される正当なソフトウェアアップデートの装い、アプリケーションの整合性を損います。この攻撃は、アップデートやそのソースを隠すための心理的または技術的メカニズムによって強化された「なりすまし」技術を拡張しています。事実上すべてのソフトウェアは頻繁な更新やパッチを必要とするため、攻撃を構成する際に攻撃者に多くの機会を与えます。バラマキ型の攻撃は、スパム、フィッシング、トロイの木馬、ボットネットなどの大量配信システムにより、膨大な数のユーザに電子メールやその他のメッセージを配信します。標的型攻撃は、特定の人物または組織を対象とします。企業のFacebookまたはMyspaceのページを用いれば、メールアドレスの収集やスパムに頼ることなく特定の会社または機関のユーザを簡単にターゲットにすることができます。この攻撃によるフィッシングを助ける1つの方法は、ソフトウェアアップデートのように見えるものを使用することです。組織の実際の電子メールアドレスを取得したり、一般的に使用されている電子メールアドレスを生成したりして、不正なソフトウェアアップデートを手動でダウンロードしてインストールするように要求することができます。このタイプの攻撃は、インスタントメッセージングのウイルスペイロードを使用して行われています。ダウンロードした更新プログラムを適用させるためにユーザの連絡先リストから名前を取得し、取得したユーザにインスタントメッセージを送信します。どちらの方法も攻撃をサポートする高度な自動メカニズムが必要ですが、IMクライアントまたはWebアプリケーション内のリンクをクリックすると更新が開始される可能性があります。悪意のあるソフトウェアの更新を含む自動化された攻撃では、ユーザが誘導する活動はほとんど必要ないため、攻撃者にとって非常に有効です。手動攻撃の複雑な予備設定段階を避けるため、スパムフィルタやWebセキュリティフィルタなどの対策を避けながら、効果的にユーザを誘導する必要があります。
184 438	製造時の変更	システムリソースを操作する	攻撃者は、サプライチェーンのライフサイクルに関与する一部のエンティティに対する攻撃を行う目的で、製造段階の技術、製品、またはコンポーネントを変更します。攻撃者がソフトウェア構成、ハードウェアの設計と組み立て、ファームウェア、または基本的な設計の仕組みに潜在的に侵入する可能性があるため、製造者が製造を開始したときに、攻撃者が技術を変更できる方法は無制限です。さらに、主要構成部品の製造は、一次製造業者によって組み立てられた最終製品を外注することが多いです。しかし、最大のリスクは、悪意のあるハードウェアやデバイスを製造するための設計仕様書の意図的な操作です。1つの集積回路には何十億ものトランジスタがあり、悪意のある機能を作り出すために必要なトランジスタ数は10個以下であることが研究によって示されています。
184	開発の変更	製造時の変更	攻撃者は、開発中に技術、製品、またはコンポーネントを変更します。 次に、製品は攻撃者の目的が達成されるため、ユーザに配送されます。
184	デザインの変更	製造時の変更	攻撃者は、開発中に技術、製品、またはコンポーネントのデザインを変更します。 次に、製品は攻撃者の目的が達成されるため、ユーザに配送されます。
439	配布中の操作	システムリソースを操作する	攻撃者は、流通経路のある段階で製品、ソフトウェア、または技術の完全性を攻撃します。流通中の変更や操作の主な脅威は、最終的に資産が提供される際に複数のサプライヤや企業を経由する可能性があるため、多くの流通段階から生起します。
439 522	悪意のあるハードウェアコンポーネントに交換	配布中の操作	攻撃者は、システム内の正当なハードウェアを、サプライチェーンの流通経路に偽造品や改ざんされたハードウェアで置き換えます。システムの展開時に悪意のある中断や追加を行います。この攻撃目的は悪意のある中断を引き起こすこと、またはシステムの導入時に追加の侵害を生じさせることです。
439 523	悪意のあるソフトウェアの埋め込み	配布中の操作	攻撃者は、システムが開発されたときに悪意のある中断を引き起こすか、または追加の侵害を発生させる目的で、悪意のあるソフトウェアをサプライチェーンの流通経路のシステムに埋め込みます。
439 524	不正な統合手順	配布中の操作	攻撃者は、悪意をもって変更されたコンポーネントをシステムに挿入するため、統合した機能の中に不正なプロセスを変更または確立します。その後、攻撃者は悪質なコンポーネントを提供します。 これにより、システムの導入時に悪質な攻撃や追加的な侵入が可能になります。
440	ハードウェア完全性攻撃	システムリソースを操作する	攻撃者は展開され、使用中に被害者の場所で攻撃を実施する目的でテクノロジ、製品、コンポーネント、またはサブコンポーネントを変更します。

440 401	ハードウェアをハッキングする	ハードウェア完全性攻撃	攻撃者は、攻撃を行う目的でシステムの整合性を損なうようにハードウェアコンポーネントを変更または交換します。ハードウェアコンポーネントの配備後、ハードウェアや交換可能な各種部品のアップグレードや交換が行われることは珍しくありません。これらのアップグレードおよび交換は、欠陥を修正し、追加機能を提供し、壊れた部品または摩耗した部品を交換するために実施されます。しかし、欠陥のあるコンポーネントまたは破損したコンポーネントとの良好なコンポーネントの置換を強制またはなりすますことによって、攻撃者は悪意のある影響を与えるための既知の欠陥を悪用するかもしれません。
440 534	悪意のあるハードウェアの更新	ハードウェア完全性攻撃	攻撃者は、更新または交換作業中に悪質なハードウェアを導入し、標的に侵入または運用の中断を企図します。
441	悪意のあるロジックの挿入	システムリソースを操作する	攻撃者は悪意のあるロジックをシステムの一見正しいコンポーネントにインストールまたは追加します。このロジックは、しばしばシステムのユーザから隠され、画面の裏で悪影響を与える動作を実行します。
441 442	製品ソフトウェアに悪意のあるロジックを挿入する	悪意のあるロジックの挿入	攻撃者は、通常、従来のウイルスやトロイの木馬のバックドアの形で悪意のあるロジックをソフトウェアに挿入します。
441 452	製品ハードウェアに悪意のあるロジックを挿入する	悪意のあるロジックの挿入	攻撃者は悪意のあるロジックをハードウェアに挿入し、悪意のあるロジックを利用するかもしれません。
441 456	製品メモリに悪意のあるロジックを挿入する	悪意のあるロジックの挿入	攻撃者は悪意のあるロジックをメモリに挿入し、悪意のあるロジックを悪用するかもしれません。
607	妨害	システムリソースを操作する	攻撃者は、システムコンポーネント間のやり取りを妨害します。システム間の相互作用を中断または無効にすることにより、攻撃者はシステムを劣化状態にしたり、さらには障害を引き起こすことがあります。
607 547	デバイスまたはコンポーネントの物理的破壊	妨害	攻撃者はデバイスやコンポーネントを物理的に攻撃し、設計された機能が動作しなくなるように破壊します。
607 582	経路の無効化	妨害	攻撃者は、2つのターゲット間のネットワークルートを無効にします。目標は、2つのエンティティ間の通信チャネルを完全に切断することです。これは、重要なインフラストラクチャを管理している人が重大なミスを犯したり、インターネットキルスイッチを使用したことが原因かもしれません。この攻撃パターンは、経路上を通過するデータではなく、経路自体を標的としているため、他のほとんどの妨害パターンとは異なります。
607 601	ジャミング	妨害	攻撃者は、通信を混乱させるために無線ノイズまたは信号を使用します。不正なトラフィックでシステムリソースを意図的に覆い隠すことにより、正規のユーザの正当なトラフィックがサービス拒否されます。
607 603	封鎖	妨害	攻撃者が重要なシステムリソースの配信をブロックし、システムの動作を停止させます。
548	汚染された資源	システムリソースを操作する	攻撃者は認可されていない分類/機密性の情報を取り扱わせることによって、組織の情報システム（デバイスやネットワークを含む）を汚染します。 情報は、そのような情報へのアクセスが許可されていない個人に公開されます。情報システム、デバイス、またはネットワークは、流出されている間は利用できません。
225	アクセス制御を破棄する		攻撃者は、IDと認証、そのリソースへの認可機能へのアクセスを管理するためにターゲットが使用するメカニズムの弱点、制限、前提条件の活用を積極的に悪用します。この攻撃は、ターゲットシステムの信頼を完全に破壊する可能性があります。このようなシステムは、相互作用する任意のエンティティの識別情報を持っており、データまたは機能に対して有するすべての制御を完全に破壊できるかもしれません。認証メカニズムの破壊によって標的とされる弱点は、実装された認証メカニズムの強さや厳しさの思い込みや過信に起因します。権限管理の標的となる弱点は、以下の3つの主要因によるものです。1）効果的な認証メカニズムへの根本的な依存。2）様々なエンティティ間の特権の分離に対する効果的な制御の欠如。3）実装された認証メカニズムの強さまたは厳格さに関する思い込みおよび過信。

21	信頼できる資格情報の悪用	アクセス制御を破棄する	セッションIDとリソースIDへの攻撃は、一部のソフトウェアが信頼性を確認せずにユーザの入力を受け入れるという事実を利用しています。たとえば、サービスリクエスタがオープンチャネル（匿名FTPなど）を介してキューにメッセージをポストできるようにするメッセージキューイングシステムでは、ポストされたメッセージに含まれるグループまたは役割のメンバシップをチェックすることによって認可が行われます。しかし、メッセージ自体やメッセージ内の情報（グループまたは役割のメンバシップ）、またはキューにメッセージを書き込んだプロセスが正しいもので、正当な権限があることを示すことができません。多くのサーバ側のプロセスは、これらの攻撃に対して脆弱です。サーバ間通信はセキュリティの観点から分析されていないか、プロセスがファイアウォールの中にあるため、他のシステムを信頼するためです。同様に、簡単に推測可能、または、デジタル識別情報を表現するためのスプーフィング可能なスキームを使用するサーバも脆弱かもしれません。このようなシステムでは、暗号化やデジタル署名のない（または暗号化が破られた）スキームが頻繁に使用されます。セッションIDは、不十分なランダム性、脆弱な保護（平文の利用）、完全性の欠如（署名なし）、またはアクセス制御ポリシーの設定との不適切な相互関係によって推測される可能性があります。システムパスワード、データベース接続文字列などを含む公開された構成ファイルやプロパティファイルは、攻撃者にこれらの識別子を識別するきっかけを与える可能性があります。その結果、なりすましが可能になり、攻撃者がシステム上の認証、承認、および監査コントロールを破る可能性があります。
21 196	セッションクレデンシャルの改ざん	信頼できる資格情報の悪用	攻撃者は、サービスへのアクセスを取得または侵害するために、偽の機能的なセッションクレデンシャルを作成します。セッションクレデンシャルを使用すると、ユーザは、すべてのメッセージに認証情報（通常はユーザ名とパスワード）を再送する必要なく、最初の認証後にサービスに自分自身を識別させることができます。攻撃者が有効なセッションクレデンシャルを偽造できる場合、認証をバイパスするか、または他の認証されたユーザのセッションを後追いすることができます。この攻撃は、セッションIDの再利用及びセッションのサイドジャック攻撃とは異なります。セッション再利用攻撃では、攻撃者は以前のまたは既存の資格情報を変更せずに使用します。この攻撃は、以前に観察された認証情報に基づいている可能性がありますが、攻撃者は独自の認証情報を作成する必要があります。
21 510	SaaSユーザリクエスト偽造	信頼できる資格情報の悪用	攻撃者は、以前にインストールされた悪意のあるアプリケーションを介して、信頼できるユーザのセッションに加えられた永続で暗黙の信頼を利用して、サードパーティのSaaSアプリケーションに対して悪質な操作を実行します。この攻撃は、信頼できるユーザがクラウドサービスで認証され、認証されたセッションでピギーバックした後に実行されます。クラウドサービスは信頼できるユーザとのみ通信できていると思い込んでいます。この攻撃が成功した場合、悪意のあるアプリケーションに埋め込まれたアクションは、対象となるSaaSアプリケーションによって処理、受容され、信頼されたユーザの特権レベルで実行されます。
21 560	既知のドメイン資格情報の使用	信頼できる資格情報の悪用	攻撃者は、盗んだ認証情報（例えば、ユーザIDおよびパスワード）を使用して、ローカルネットワーク上の同じ認証フレームワークの下で管理されるシステムにアクセスします。多くの場合、ユーザは同じパスワードを使用して接続されている別のマシンにログインすることができます。攻撃者はあるマシンでパスワードを発見すると、それらのマシンから侵入拡大が可能になります。
21 593	セッションハイジャック	信頼できる資格情報の悪用	この攻撃には、認証の実行時におけるアプリケーションのセッション使用の弱点を悪用する攻撃も含まれます。攻撃者は、アクティブなセッションを盗み、操作することで、アプリケーションへの無差別なアクセスを取得できます。
21 62	クロスサイトリクエストフォージェリ	信頼できる資格情報の悪用	攻撃者は悪意のあるWebリンクを作成し、各種方法（Webページ、電子メールなど）でそれらを配布し、ユーザがリンクをクリックすることで第三者のアプリケーションに対して悪意のある行為を実行します。成功した場合、悪意のあるリンクに埋め込まれたアクションは、処理され、ユーザの特権レベルでターゲットアプリケーションによって起動されます。この攻撃は、多くのWebアプリケーションによりユーザセッションCookieが配置され、永続で暗黙な信頼を悪用します。 このようなアーキテクチャでは、ユーザがアプリケーションに認証されると、ユーザのシステム上にセッションCookieが作成されます。最初のセッションの続くすべてのトランザクションは、攻撃者によって開始された潜在的なアクションを含むCookieを使用して認証されます。そして、単に既存のセッションクッキーは使用されるだけです。
22	クライアントにおける信頼の悪用	アクセス制御を破棄する	この攻撃は、クライアント/サーバ通信チャネル認証とデータ整合性の脆弱性を悪用します。これは、サーバがクライアントの間接的な信頼を認識すること、または、サーバがクライアントであると考える間接的な信頼を悪用します。攻撃者は、クライアントとサーバの間の通信チャネルに自分自身を認識させることによって、この種の攻撃を実行します。サーバが有効なクライアントとだけ通信していると思い込んでいる場合に、サーバに直接通信することが可能となります。この種の攻撃手法には多数のバリエーションがあります。

22 202	悪意のあるクライアントを作成する	クライアントにおける信頼の悪用	攻撃者は、クライアントがサービスによりクライアントに強いられる取り決めに違反するため、ターゲットサービスとのインターフェースを違反するクライアントアプリケーションを作成します。(IMAPやPOPクライアントなどの一般的なクライアントアプリケーションを使用するサービスとは異なり)クライアントアプリケーションを指定したサービスでは、クライアントが特定の手順に従うと取り決めされていることがあります。たとえば、サーバは、クライアントが値(価格など)を正確に計算し、正しく構造化されたメッセージを送信し、サーバとの効率的なやりとりを保証していると想定しています。リバースエンジニアリングにより独自のバージョンのクライアントを作成することで、攻撃者はこれらの取り決めを利用し、サービス機能を悪用することができます。たとえば、購買サービスは単価をそのクライアントに送信し、クライアントが購入の総コストを正しく計算することを期待するかもしれません。しかし、攻撃者が悪意のあるクライアントを使用する場合、サーバの入力を無視して総額を不正に改ざんすることができます。同様に、攻撃者はサーバパフォーマンスを低下させるため、ネットワークまたは他のサーバリソースを必要以上に長期間、使用するようにクライアントを構成することもできます。一般的なクライアントのサービスであっても、特定のクライアントの動作を想定すると、この攻撃の影響を受けやすくなります。しかし、そのようなサービスは最初にクライアントの行動についての取り決めを少なくすることができます。したがって、攻撃者が悪用できると取り決めはより少なくなります。
22 207	重要なクライアント機能の削除	クライアントにおける信頼の悪用	攻撃者は、サーバが存在し、信頼できると判断したクライアント上の機能を削除または無効にします。攻撃者は、場合によっては、機密性の高い機能やデータを保護するためにロジックの裏をかくことができますかもしれません。クライアントアプリケーションには、サーバが正確で安全な操作のために必要とする機能が含まれている場合があります。この機能には、サーバに危険なコンテンツが送信されるのを防止するフィルター、価格計算などの論理機能、および許可されたユーザのみがクライアントを利用するための認証ロジックが含まれますが、これに限定されません。攻撃者がクライアント上でこの機能を無効にできる場合、攻撃者は、サーバが禁止されていると考えられる操作を実行できます。これにより、サーバによる取り決めに違反するクライアントの動作が発生し、さまざまな攻撃が行われる可能性があります。上記の例では、危険なコンテンツ(スクリプトなど)をサーバに送信したり、不正な価格計算やサーバリソースへの不正アクセスを行うことができます。
22 39	不透明なクライアントベースのデータトークンの操作	クライアントにおける信頼の悪用	アプリケーションが重要なデータをトークン(クッキー、URL、データファイルなど)でクライアント側に保持する場合、そのデータは操作できます。クライアントまたはサーバ側のアプリケーションコンポーネントがそのデータを認証トークンまたはデータとして再解釈する場合(アイテムの価格設定やウォレット情報の保存など)、そのデータをただ操作しても、攻撃者にとっては有益かもしれません。このパターンでは、攻撃者は、クライアント側のトークンが暗号化または難読化の使用によって改ざんから適切に保護されているという前提を損なうことになります。
22 77	ユーザが制御する変数の操作	クライアントにおける信頼の悪用	この攻撃は、ユーザが制御する変数(DEBUG = 1、PHP Globalsなど)を対象としています。攻撃者は、データサニタイズを行わずにアプリケーションサーバ上で直接使用される、ユーザが提供する信頼できないクエリ変数を利用して環境変数を上書きできます。極端な場合、攻撃者はアプリケーションのビジネスロジックを制御する変数を変更できます。例えば、PHPのような言語では、不適切な多くのデフォルト設定によって、ユーザが変数を無効にすることができます。
22 94	中間者攻撃	クライアントにおける信頼の悪用	このタイプの攻撃は、2つのコンポーネント(通常はクライアントとサーバ)間の通信を対象としています。攻撃者は、2つのコンポーネント間の通信チャネルに自分自身を配置します。1つのコンポーネントが他のコンポーネント(データフロー、認証チャレンジなど)との通信を試みるたびに、データは最初に攻撃者に送信され、攻撃者はそのデータを観察または変更し、あたかも決して傍受されていなかったかのように、他のコンポーネントに送ります。この介入は透過的であり、侵害された2つのコンポーネントは通信の潜在的な破損または漏洩を認識しません。潜在的な中間者(Man-in-the-Middle)攻撃は、コミュニケーションへの信頼の間接的な欠如、または2つのコンポーネント間の識別の欠如です。
22 384	Man-in-the-Middle経由のアプリケーションAPIメッセージ操作	アクセス制御を破棄する	攻撃者は、交換されているメッセージまたはアイテムの内容を変更するために、アプリケーションフレームワーク内でイベントまたはトランザクションに関与します。この攻撃を実行することで、攻撃者は正しく見えるメッセージやコンテンツを生成するような方法でコンテンツを操作することができます。しかし、詐欺的なリンクを含んでいたり、1つのアイテムを別のアイテムに置き換えたり、既存のアイテムをスプーフィングしたり、誤った交換を行ったり、交換されているものの金額や身元を変更したりすることがありえます。この手法では、攻撃者がさまざまなアプリケーションの内容を変更するため、Webブラウザとリモートシステム間の中間者通信を可能にする特殊なソフトウェアの使用が必要です。多くの場合、ゲームで交換されるアイテムは、コインやバーチャルドルなどの販売を通じて収益化することができます。

384 385	アプリケーションAPI操作によるトランザクションまたはイベントの改ざん	アクセス制御を破棄する	攻撃者は、交換中のメッセージまたはアイテムの内容を変更するため、アプリケーションフレームワーク内でイベントまたはトランザクションに関与します。この攻撃で、攻撃者は真正に見えるメッセージやコンテンツを生成し、コンテンツを操作できます。不正なリンクが含まれている、1つのアイテムを別のアイテムに置き換えている、既存のアイテムをスプーフィングしている、誤った交換を行っている、交換しているものの量やアイデンティティを変更している可能性があります。この手法では、攻撃者がさまざまなアプリケーションの内容を変更するためにWebブラウザとリモートシステム間の中間者通信を可能にする特殊なソフトウェアの使用が必要です。多くの場合、ゲームで交換されるアイテムはコイン、バーチャルドルなどの販売を通じて収益化することができます。攻撃の目的は、エクスチェンジに関係するデータパケットをトラップし、転送プロセスの整合性を変更することによって犠牲者を詐欺することです。
384 389	アプリケーションAPI操作によるコンテンツスプーフィング	アクセス制御を破棄する	攻撃者はメッセージの内容を変更するため、アプリケーションフレームワーク内のクライアントからの出力データまたは入力データのいずれかを操作します。この攻撃を実行することで、攻撃者は、真正に見えるメッセージやコンテンツを生成し、コンテンツを操作できます。これには詐欺的なリンク、スパムのようなコンテンツ、または攻撃者のコードへのリンクが含まれる可能性があります。一般に、この攻撃では攻撃者の意図に基づいてさまざまな種類の攻撃を行うことができます。攻撃者がWebブラウザとリモートシステム間の中間者通信を可能にする特殊なソフトウェアを使用する必要があります。
22 466	同じ発信元ポリシーを迂回する中間者攻撃を活用する	アクセス制御を破棄する	攻撃者は、被害者のブラウザで同じ発信元ポリシーの保護をバイパスするため、中間者攻撃を利用します。例えば、標的が公共WIFIホットスポットに接続中など、中間攻撃を誰でも（中間者）立ち上げることができます。攻撃者は、被害者のブラウザとTLSを使用しないいくつかの重要でないWebサイトとの間で、要求と応答を傍受することができます。たとえば、標的はフライトまたは天気の情報をチェックしている可能性があります。攻撃者が標的にバインドされたレスポンスを傍受すると、攻撃者は機密性の高い機能を持つドメインを参照するレスポンスにiFrame（大体は認識できない）を追加し、標的にレスポンスを転送します。標的のブラウザは機密性の高い機能を持つサイトへの不正なリクエストを自動的に開始します。 同一の発信元ポリシーを使用すると、Java Scriptが送信されたサイト以外のサイトにこれらのリクエストを送信できなくなります。しかし、攻撃者はこれらの自動要求を傍受して機密性の高い機能を持つドメイン/サービスにリダイレクトするため、中間者を使用します。
114	認証の悪用	アクセス制御を破棄する	攻撃者は、認証メカニズムの固有の弱点を知っているか、認証スキームの実装の欠陥を悪用して、アプリケーション、サービス、またはデバイスに不正にアクセスします。そのような攻撃では、認証メカニズムが機能していますが、慎重に制御された一連のイベントによって、メカニズムが攻撃者にアクセスを許可します。この攻撃は、信頼関係に関する前提や秘密値の生成に関する前提など、ターゲットの認証手順によって行われた前提を悪用する可能性があります。この攻撃は認証バイパス攻撃とは異なり、認証の悪用によって攻撃者が正当なユーザとして認証され、保護されたマテリアルにアクセスできます。標的がブラウザに持っている永続的なCookieは、これらの不正なリクエストに使用されます。攻撃者は機密機能を持つサイトに標的を積極的に誘導します。機密性の高い機能を備えたサイトが被害者の要求に応答すると、中間の攻撃者がこれらの応答を傍受し、自身の悪質なJava Scriptをこれらの応答に注入し、被害者のブラウザに転送します。標的のブラウザでは、そのJavaスクリプトは機密性の高い機能を持つサイトの制限の下で実行され、本質的に機密サイトと対話するために使用することができます。したがって、攻撃者が望むすべてのドメイン上で、標的のブラウザ内でスクリプトを実行できます。攻撃者はこの技術を使用して、攻撃者が望むあらゆるサイトに対して、被害者のブラウザからCookieを盗み出すことができます。これは、従来のXSS攻撃とは異なり、永続CookieとHTTPのみのCookieの両方に適用されます。攻撃者はこの手法を使用して、ログインフォームを暗号化するだけのサイトの認証資格情報を盗み出すこともできます。さらに、攻撃者は任意のオートコンプリート情報を盗むこともできます。この攻撃パターンは、セッション固定とキャッシュポイズニング攻撃を有効にする際にも使用できます。追加の攻撃を有効にすることもできます。
114 629	デバイスリソースの不正使用		特定のデバイスリソースへの不正アクセスを以前に得た攻撃者は、そのアクセスにより、デバイスのロケーションやネットワーク情報などの情報を取得します。
114 90	認証プロトコルへのリフレクション攻撃		攻撃者は、リフレクション攻撃の影響を受けやすい認証プロトコルを悪用して、攻撃する可能性があります。これにより、攻撃者は必要な資格情報を保持せずにターゲットシステムに不正にアクセスすることができます。リフレクション攻撃は、チャレンジハンドシェイクまたは同様のメカニズムに依存する認証プロトコルにとって大きな問題です。攻撃者は正当なユーザになりすますことができ、認証中に反射攻撃を正常にマウントすることでシステムへの不正なアクセスを得ることができます。

115	認証のバイパス	アクセス制御を破棄する	攻撃者は、認証メカニズムを回避することによって、許可されたユーザまたは特権ユーザの権限でアプリケーション、サービス、またはデバイスにアクセスします。したがって、攻撃者は認証が行われていなくても保護されたデータにアクセスできます。これは、攻撃者が認証手続きを経ることなく、認証されたユーザと同等のアクセス権を取得することを意味します。これは通常、攻撃者が予期しないアクセス手順を使用した結果、認証が行われる適切なチェックポイントを通過しないためです。例えば、ウェブサイトは、安全なマテリアルを取得し、リンクをクリックするすべてのユーザを認証するため、すべてのユーザが所定のリンクをクリックすると仮定しているかもしれません。しかし、攻撃者は、認証リンクをクリックするのではなく、コンテンツへのパスを明示的に入力することで、セキュリティで保護されたWebコンテンツにアクセスできる可能性があります。この攻撃パターンは、認証メカニズムの欠陥を悪用して認証を偽ったり、正当なユーザから資格情報を盗むのではなく、認証を完全に回避するという点で、他の認証攻撃とは異なります。
115 461	ハッシュ関数拡張機能の弱点を活用するWebサービスAPI署名偽造	認証のバイパス	Webサービスで呼び出し先を認証する必要がある場合、呼び出し元がWebサービス呼び出しに署名するために使用するトークン/秘密を呼び出し元に発行することがあります。そのようなスキームの1つには、リクエストを構築するときの呼び出し側は、ウェブサービスに渡されたすべてのパラメータを提供された認証トークンと連結し、連結された文字列（例えば、MD5、SHA1など）のハッシュを生成するものがあります。その後、そのハッシュは、メッセージの発信元の真正性および完全性を検証するためにサーバ側で使用されるWebサービスに渡される署名を作成します。ハッシュ関数の拡張/パディングの弱点を利用するこの種の認証スキームに対する攻撃があります。この弱点を利用して、秘密トークンを知らない攻撃者は、独自の呼び出しを生成することによってWebサービスに渡されたパラメータを変更し、正当な署名ハッシュを生成することができます。たとえば、Webサービスに渡すメッセージがM（このメッセージには、秘密トークン/キーバイトで連結されたWebサービスに渡されるパラメータが含まれています）とみなされたとします。メッセージMはハッシュ化され、そのハッシュはWebサービスに渡され、認証に使用されます。攻撃者はMはわかりませんが、ハッシュ（M）と長さ（M）を見ることができる。攻撃者は任意のM 'に対してハッシュ（M ||パディング（M）II M'）を計算することができます。攻撃者はメッセージMの全体と秘密のバイトをわかりませんが、それは問題ではありません。攻撃者は依然として自分のメッセージM 'に署名することができます。また、呼び出されたWebサービスがエラーのないメッセージの整合性を検証するようにできます。ハッシュ関数の反復設計のために、メッセージのハッシュとその長さだけから、最初のメッセージで始まるより長いメッセージのハッシュを計算し、最初のメッセージが512ビットの倍数に達するのに必要なパディングを含めることが可能です。この攻撃はMD5に限定されず、SHA1のような別のハッシュ関数も同様に機能することに注意することが重要です。
115 87	強制的ブラウズ	認証のバイパス	攻撃者は強制的ブラウズを使用して、直接URL入力によって到達できないWebサイトの一部にアクセスします。通常、フロントコントローラまたは類似のデザインパターンを使用して、Webアプリケーションの一部へのアクセスを保護しています。強制的ブラウズにより、攻撃者は情報にアクセスしたり、特権操作を実行したり、不適切に保護されたWebアプリケーションのセクションにアクセスすることができます。
122	特権悪用	アクセス制御を破棄する	攻撃者は、特権を持つユーザや管理者のために実行されるべきであるが、より低いまたは特権のないアカウントによって使用できるターゲットの機能を利用するかもしれません。権限のあるユーザだけがこれらのリソースにアクセスできるように、機密情報と機能へのアクセスを制御する必要があります。アクセス制御メカニズムが存在しないか、誤って構成されている場合、ユーザは上位レベルのユーザのみを対象としたリソースにアクセスできます。攻撃者はこれを利用して信頼性の低いアカウントで情報を取得し、より信頼できるアカウントに割り当てるべきアクティビティを実行することができます。この攻撃は、攻撃者が実際に特権を昇格させることはないという点で、特権昇格やその他の特権窃取攻撃とは異なります。しかし、代わりに、より低いレベルの特権を使用して、より高い特権のアカウントのために割り当てるべき（しかしそうなっていない）リソースにアクセスすることができます。同様に、攻撃者は信頼やシステム破壊を悪用しません。すべての制御機能は設定どおりに機能していますが、適切なレベルの重要リソースを適切に保護されていません。
122 1	ACLにより正しく制限されていない機能へのアクセス	特権悪用	アプリケーション、特にWebアプリケーションでは、機能へのアクセスは認証フレームワークによって対策されています。このフレームワークは、アクセス制御リスト（ACL）をアプリケーションの機能の要素にマッピングします。特に、Webアプリケーション用のURLです。管理者が特定の要素に対するACLの指定に失敗した場合、攻撃者は無断でACLにアクセスできる可能性があります。ACLによって適切に制約されていない機能にアクセスする能力を持つ攻撃者は、機密情報を取得し、アプリケーション全体を危険にさらす可能性があります。このような攻撃者は、より高い特権レベルのユーザにしか利用できないリソースにアクセスしたり、アプリケーションの管理セクションにアクセスすることができます。

122 17	悪意のあるファイルの利用	特権悪用	攻撃者がシェルアクセスなどの実行可能ファイルに直接アクセスできるようにするシステムの設定を悪用します。 または、攻撃者はファイルをアップロードし、実行できるようにします。 多くの統合ポイントを持つWebサーバ、FTPサーバ、およびメッセージ指向のミドルウェアシステムは、プログラマーと管理者の両方が各インターフェースの正しい特権について同期しなければならないため、特に脆弱です。
122 180	誤って設定されたアクセス制御セキュリティレベルの悪用	特権悪用	攻撃者はアクセス制御の構成の弱点を悪用し、保護を回避し、システムまたはネットワークへの不正アクセスすることができます。重要な機能は、常にアクセス制御で保護する必要があります。ただし、最も単純なアクセス制御システム以外のすべてを設定することは非常に複雑になり、間違いの可能性が高くなります。攻撃者が誤って設定されたアクセスセキュリティ設定を知ることができた場合、攻撃でこれを悪用する可能性があります。一般的には、攻撃者はアクセスを拒否されるべき行動を実行するため、重要な活動の保護が不適切な拒否コントロールを利用します。状況によっては、攻撃者が過度に制限的なアクセス制御ポリシーを利用できる可能性があります。アクセス制御ポリシーによりサービスの拒否を開始する（アプリケーションが許可されたアクセスの予期しない失敗のためにロックした場合）、またはセキュリティのために他の正当なアクションが失敗してしまいます。しかし、後者の攻撃は、不適切なセキュリティ制限に基づいた攻撃よりも、通常はあまり重大ではなく検出が容易です。
122 221	XML外部エンティティ	特権悪用	この攻撃は、置換の値がURIであるXMLのエンティティ置換プロパティを利用します。細工されたXML文書は、サービス拒否条件を作成するためにエンティティに大量のリソースを消費するURIを参照させる可能性があります。 これにより、URIに応じてシステムがフリーズ、クラッシュ、または任意のコードを実行する可能性があります。
122 503	WebViewの公開	特権悪用	攻撃者は、悪意のあるWebページを介して、WebViewのaddJavascriptInterface APIによって登録されたインターフェイスを利用して、アプリケーション固有の機能にアクセスします。インターフェースがaddJavascriptInterfaceを介してWebViewに登録されると、そのインターフェースはグローバルになり、WebViewに読み込まれたすべてのページがこのインターフェースを呼び出すことができます。
233	特権昇格	アクセス制御を破棄する	攻撃者は弱点を悪用してシステムの権限を昇格させ、実行権限のない処理を実行します。
233 104	クロスゾーンスクリプティング	特権昇格	攻撃者は、被害者にWebブラウザにコンテンツを読み込ませことができます。これにより、セキュリティゾーンのコントロールをバイパスし、スクリプトコードや署名されていないActiveXコントロールやアプレットなどのその他のWebオブジェクトを実行するための特権を昇格することができます。これは、ゾーンベースのWebブラウザセキュリティを対象とした特権昇格攻撃です。ゾーンベースのモデルでは、ページはそのページに割り当てられた特権レベルに対応する一連のゾーンの1つに属します。信頼できないゾーンのページは、システムへのアクセスレベルが低く、実行が許可された実行可能なコンテンツの種類が制限されます。クロスゾーンスクリプト攻撃では、権限の低いゾーンに割り当てるべきページに、より信頼できるゾーンの権限が与えられます。これは、攻撃者のコンテンツがより信頼できるページからのものとして扱われるクロスサイトスクリプティング攻撃とてブラウザのバグを悪用したり、ゾーンコントロールの不正な設定を悪用します。または、信頼ゾーンと信頼性の低いゾーンの両方からアクセス可能なシステム機能を活用することによって実行されます。クロスゾーンスクリプティング攻撃はブラウザで実装されたセキュリティゾーンの概念を攻撃します。
233 234	特権プロセスを乗っ取る	特権昇格	攻撃者は、特権で任意のコードを実行するために昇格された特権が割り当てられたプロセスを制御できます。一部のプロセスには、通常は特定のユーザ、グループ、または役割との関連付けによって、オペレーティングシステムで付与された特権が割り当てられます。攻撃者がこのプロセスを乗っ取ることができる場合、自分のコードを実行するために特権レベルを使用することができます。プロセスは、不適切なユーザ入力（バッファオーバーフローや特定のタイプのインジェクション攻撃など）や不適切に保護されたプロセス制御をサポートするシステムユーティリティを利用してハイジャックされる可能性があります。
233 30	特権実行スレッドのハイジャック	特権昇格	攻撃者は時には同期（誤って返す特権関数を呼び出す）または非同期（コールバック、シグナルハンドラなど）の手段を介して特権スレッドをシステムから乗っ取ることができます。これにより、攻撃者はシステムの設計者が意図していなかった機能にアクセスできるようになりますが、他のユーザに不可欠なサービスが検出できない、または拒否される可能性があります。
233 68	コード署名機能を破壊する	特権昇格	言語はコード署名機能を使用してコードの識別情報を確認します。コードを環境内の割り当てられた特権に結び付けるため、このメカニズムを無効にすることは、攻撃者が特権を昇格するのに役立ちます。バーチャルマシンがコード署名を強制する方法を覆す手段は、この攻撃スタイルに分類されます。この攻撃パターンには、署名鍵が盗まれた状況は含まれません。

233 206	製品環境からの署名鍵を抽出し悪質なコードに署名する	特権昇格	攻撃者は、コード署名に使用された認証情報を製品環境から抽出し、これらの認証情報を使用して悪質なコンテンツを開発者の鍵で署名します。多くの開発者は、コードまたはコードのハッシュに署名するための署名キーを使用します。ユーザまたはアプリケーションは、署名が正確であることを確認すると、コードが署名キーの所有者から送られてきたと考えられます。また、署名が適用されてからコードが変更されていないと考えます。攻撃者が署名の証明書を抽出した場合、その証明書を使用して自分のコードに署名することができます。コードに添付されている署名を検証するユーザまたはツールは、コードが正当な開発者からのものであると思い込み、コードをインストールまたは実行します。これにより、効果的に攻撃者が被害者のコンピュータ上で任意のコードを実行できるようにします。
233 237	サンドボックス内の別の言語から署名されたコードを呼び出し、利用する。	特権昇格	攻撃者は、特権へのアクセス権を得るために、別の言語の悪質な署名コードを使用することがあります。それは意図的にサンドボックスによって公開されていないため、サンドボックスから間接的に表現したものです。たとえば、Javaコードでは、バイトコードベリファイアとJVMによって制限されるため、任意のメモリ位置の変更など、安全でない操作は実行できません。許可された場合、Javaコードはネイティブ Cコードを直接呼び出すことができます。ネイティブCコードは、システムコールを呼び出すなどの安全でない操作を実行したり、動作上の任意のメモリ位置を変更することがあります。分離するために、Javaはネイティブ Cコードへの無制限なアクセスを信頼できないコードに与えることはありません。しかし、サンドボックス化されたコードは、通常、標準ライブラリの一部である既存のネイティブコードのサブセットを呼び出すことができます。
507	物理的な窃盗	アクセス制御を破棄する	攻撃者は物品の窃盗によってシステムまたはデバイスに物理的にアクセスできます。システムまたはデバイスの所持は、多数のユニークな攻撃を実行可能にし、攻撃者に攻撃を行うために長期の時間を与えます。 機密情報の保護は、攻撃者が物理的なアクセスと十分な時間を持っている場合には無効になります。
390	物理的セキュリティをバイパスする	アクセス制御を破棄する	施設では、伝統的なロックなどの物理的なセキュリティのための階層化されたモデル、電子ベースのカード入力システム、物理的なアラームを使用することがよくあります。ハードウェアのセキュリティメカニズムには、コンピュータケースとケーブルロックの使用、およびコンピュータ資産を追跡するためのRFIDタグがあります。この階層化されたアプローチでは、気付かれないでランダムな物理的なセキュリティ違反を行うことは難しくなりますが、意図的で慎重に計画された侵入を止めるにはあまり効果的ではありません。検出を避けるには、建物のセキュリティと監視を回避すること、およびエントリポイントを保護する電子的または物理的なロックをバイパスする方法から始まります。
395	電子ロックとアクセスコントロールをバイパスする	物理的セキュリティをバイパスする	攻撃者は、電子ロック等のアクセス制御をバイパスするために、セキュリティへの思い込みを利用します。電子アクセス制御に対するほとんどの攻撃は、同様の方法を用いますが、異なるツールも使用します。いくつかの電子ロックは磁気ストリップカードを利用し、他のものはカードまたはバッジ内に埋め込まれたRFIDタグを使用します。または声紋、指紋、または網膜バイオメトリクスなどのより洗練された保護を含みます。磁気ストリップとRFID技術は、コスト効率が高く、他の電子セキュリティ対策と容易に統合できるため、最も普及しています。これらの技術は、正当なカードやバッジをコピーしたり、リバースエンジニアリングされたアルゴリズムを使用して新しいカードを生成することによって、攻撃者がメカニズムによって保護されている施設にアクセスするために悪用できる共通の弱点を持っています。
395 396	カードまたはバッジベースのシステムをバイパスする	物理的セキュリティをバイパスする	攻撃者は、アクセスカードの複製やブルートフォース技術の使用などで、カードベースのシステムのセキュリティをバイパスします。カードベースのシステムは、ビジネス、政府機関、サプライチェーンの管理全体に広がっています。カードベースのシステムに対する攻撃は、攻撃者の目標に基づいて大きく異なりますが、通常、カードの不正な複製、有効なカードの値を総当たり、カードデータの読み取りや処理を行うシステムに対する攻撃が含まれます。カードやバッジのセキュリティには固有の弱点があるため、高度なセキュリティ環境ではセキュリティメカニズムとしてカードやバッジだけに依存することはほとんどありません。一般的なカードベースのシステムは、金融取引、ユーザ識別、およびアクセス制御に使用されます。
395 397	磁気ストリップカードのクローニング	物理的セキュリティをバイパスする	攻撃者は、磁気ストリップカード（すなわち、「スワイプカード」または「磁気ストライプ」）上のデータを複製して、物理的な場所または個人の個人情報への不正アクセスを行います。磁気ストライプカードは、長方形のカードに沿ってストライプ状に配列された鉄系磁性粒子のバンド上のデータを符号化しています。ほとんどの磁気ストライプカードのデータフォーマットは、ISO規格7810,7811,7813,8853、および4909に準拠しています。磁気ストライプ技術の主な利点は、エンコードとポータビリティの容易さです。しかし、これにより磁気ストリップカードを未許可な複数の影響を受けやすくしています。磁気ストライプカードをアクセス制御に使用する場合、攻撃者が必要とするのは、カードのコピーを作成するのに十分な長さの有効なカードを取得し、カードを元の場所（つまり、同僚の机）に戻すことだけです。磁気ストライプリーダ/ライタは、カード上に符号化されたデータを分析するためのソフトウェアと同様に広く利用可能です。有効なカードを機械に通すと、オリジナルとして機能する複製をいくつでも作成するのは簡単です。

395 398	磁気ストリップカードへのブルートフォース攻撃	物理的セキュリティをバイパスする	攻撃者は、2つ以上の磁気ストリップカード上のデータを分析し、不正アクセス・偽装を可能にする有効なシーケンスを含む新しいカードを生成することができます。磁気ストリップ符号化方法は共通のフォーマットを使用しています。単一のカードにより、複数の子会社が共有する本社複合施設へのアクセスを可能にすることができかもしれません。データがカードにどのように格納されているかを分析することにより、ブルートフォース攻撃によって有効なカードを作成することも可能です。たとえば、1枚のカードで建物、階、特別室へのアクセスを許可することができます。複数のカードのデータを読み込んで分析し、3つの異なるカードにエンコードされたデータの差分分析を行います。建物内の制限されたエリアやその建物内の特別室等へのアクセスを許可する有効なカードを生成する方法についての手がかりを得ることができます。磁気ストライプカードに保存されているデータは暗号化されていないことが多いため、2つ以上のカードが解析されたときにどのデータが変わっているかを比較することで、カードデータの構造を判断するのに役立ちます。簡単な例は、データトラック上の一般的なシステムデータフォーマットです。どのバイナリが、ビルの特別室のドアを開くカードをエンコードしたものかを見つけます。異なるバイナリ符号化セグメントを有する複数のカードを作成することにより、許可されていない領域に入ったり、他人の電子IDに与えられたチェックポイントを通過することが可能になります。
395 399	RFIDカードまたはチップのクローニング	物理的セキュリティをバイパスする	攻撃者は、RFIDチップから応答されたデータを分析し、この情報を使用して、ターゲットチップと同じように応答するRFID信号を複製します。RFIDチップは、アクセス制御、従業員の識別、サプライチェーンにより納入される製品のマーカーとして使用される場合があります。一部の組織では、RFIDタグをコンピュータ資産に埋め込んで、特定の部屋、ゾーン、建物から移動した場合に警報を発します。磁気ストリップカードと同様に、RFIDカードは複製（複製）および再利用の影響を受けやすいです。RFID（Radio Frequency Identification）は、RF信号を処理するための集積回路とアンテナとからなる受動素子です。RFIDデバイスはオンボード電源がないという点で受動的です。大部分のRFIDチップは、13.56MHzまたは135KHzの周波数で動作します。チップは、チップ上のアンテナによって信号が受信されたときに給電され、応答メッセージを送信するのに十分長くチップに電力を供給します。攻撃者は、応答するチップを電気的に刺激するか、遠隔送信機に応答を送信するときにチップに近接することによって、RFIDデータをキャプチャし分析することができます。これにより、攻撃者は信号を複製し、建物への不正アクセスやユーザの身元を偽装するなどの攻撃を行うことができます。
395 400	RFIDチップの無効化または破壊	物理的セキュリティをバイパスする	攻撃者は、応答しないタグを含むタグ、バッジ、カード、またはオブジェクトを応答不能にする目的で、パッシブRFIDタグを無効にする方法を使用します。RFIDタグは、主にアクセス制御、在庫管理、または盗難防止装置に使用されます。RFIDチップを攻撃する目的は、RFIDチップを収容する物体に損傷を与えることなく、チップを無効にするかまたはRFIDのみを破壊することです。正しく実行されると、RFIDチップは、チップを含む商品またはデバイスにも目に見える損傷またはマーキングなしに、無効化または破壊することができます。チップを直接攻撃することにより、セキュリティデバイスまたは対策をバイパスすることができ、デバイス自体を直接損なうことはありません。警報システムまたはコンピュータシステムなどのRFIDタグを損傷または非活動化するための様々な方法が存在します。例えば、一般的なRFIDチップは、チップ自体の近くに小さな電磁パルスを生成することによって破壊することができます。方法の一つは、使い捨てカメラの改造を必要とします。フラッシュバルブを外し、銅コイルをコンデンサにはんだ付けします。RFIDチップベースのデバイスの近くでこの構成でカメラを発射すると、改ざんの証拠を残さずにチップを破壊するのに十分なEMPパルスが生成します。これまでのところ、この攻撃は13.56MHzのRFIDチップに対して機能することが実証されています。
391	物理的なロックをバイパスする	アクセス制御を破棄する	攻撃者は、建物や施設の物理的なセキュリティ対策を回避するための手法や方法を使用します。 物理的なロックは、従来のロックとキーのメカニズム、ラップトップやサーバを保護するために使用されるケーブルロック、サーバケースのロックなどのさまざまなデバイスにまで及ぶ可能性があります。 ロックバンピング、スナップガンによるロック強制、ロックピッキングなどの技法を使用して、これらのロックをバイパスし、保護されている施設またはデバイスにアクセスすることができますが、ステルス、改ざんの証拠、 使用される方法を決定することが考慮事項となります。 物理的なロックは、ロック機構の複雑さによって制限されます。 ロックによっては、耐衝撃性フォームのような保護を提供して突き当てやロック強制方法を防止することができますが、多くの一般的に使用されるロックはそのような対策を提供していません。

391 393	ピッキング	物理的セキュリティをバイパスする	攻撃者は、ロックピッキングツールとテクニックを使用して、建物や施設のロックをバイパスします。ロックピッキングは、ロック内のピンを操作するための特別なツールを使用します。ロックの種類ごとに異なるツールセットが必要です。ロックピッキング攻撃は、正しく実行されるとロックが壊れないという点で非侵襲的であるという利点があります。標準的なロック・ピン・アンド・タンブラー・ロックは、タンブラー・デバイスが回転しないようにする一組の内部ピンによって固定されています。スプリング式のドライバーピンがキーピンを押し下げて回転を防止し、ボルトがロックされた位置に留まるようにします。正しいキーが挿入されると、キーの隆起部がキーピンを押し上げてドライバピンを押し付け、正しい位置合わせを引き起こし、ロックシリンダを回転させます。米国の家庭用ロックなどの最も一般的なロックは、標準の2つのツール（つまり、トーションレンチとフックピック）を使用してピッキングできます。
391 394	ピックガンを使用してロックを強制する	物理的セキュリティをバイパスする	攻撃者は、建物や施設にロックを強制するために、スナップガン（Pick Gun）を使用します。Pick Gunはロックバンピングと同様の原理で動作する特殊なタイプのロックピッキング装置です。スナップガンは、金属ピックが取り付けられたハンドヘルドデバイスです。金属ピックは、ロック内のピンに当たって、キーピンからドライバピンに動きを伝達し、ロックの瞬間的な位置合わせに強制します。標準のロックは、デバイスが回転しないようにする一連の内部ピンによって保護されています。バネ式のドライバピンがキーピンを押し下げます。正しいキーが挿入されると、キーの隆起部がキーピンを押し上げてドライバピンに押し付け、正しい位置合わせとなり、ロックシリンダを回転させます。スナップガンは金属ピンを使用してすべてのキーピンを一度に叩き、ドライバピンをロック解除位置に移動させます。バンプキーまたはロックピックとは異なり、スナップガンはロックを容易に損傷させるため、ロックがバイパスされた証拠を残します。
392	バンピング	アクセス制御を破棄する	攻撃者は、バンプキーを使用してビルや施設にロックをかけたり、入室したりします。ロックバンピングとは、ロック内のピンを一時的な位置合わせに落として、ロックを開くことを可能にするためにタップまたはバンプすることができる特殊なタイプのキーを使用することです。ロックバンピングにより、攻撃者は正しい鍵を持たずにロックを開くことができます。標準のロックは、デバイスが回転しないようにする一連の内部ピンによって保護されています。バネ式のドライバピンがキーピンを押し下げます。正しいキーが挿入されると、キーの隆起部がキーピンを押し上げてドライバピンに押し付け、正しい位置合わせを引き起こし、ロックシリンダを回転させます。バンプキーは、このデザインを利用する特別に構成されたキーです。バンプキーが強く叩かれると、その歯がタップの力をキーピンに伝え、ロックが一時的に機構の適切な位置合わせに移行します。
	タイミングと状態を操作する		攻撃者は、タイミングや状態を維持する機能の弱点を利用してターゲットコードとプロセスの実行フローによって防止されるアクションを実行します。状態攻撃の例は、アプリケーションの情報を操作して閲覧可能な資格情報または同様の情報を変更することです。通常はアクセスできないマテリアルにアプリケーションがアクセスできるようにします。タイミング攻撃の一般的な例は、テストアクションの競合状態です。いくつかの状態情報がテストされ、うまくいくと、アクションが実行されます。攻撃者が、アプリケーションがテストを実行する時間とアクションが実行される時間との間で状態を変更できる場合、悪意のある目的への行動の結果を操作できる可能性があります。
25	デッドロックを強制	タイミングと状態を操作する	攻撃者はターゲットソフトウェアのデッドロック状態によりサービス拒否を引き起こします。デッドロックは、2つ以上の競合するアクションがお互いに完了するのを待っているときに発生する可能性があります。デッドロック状態を検出するのは困難です。
26	レースコンディションを悪用	タイミングと状態を操作する	攻撃者は複数のプロセスが同じリソースに同時にアクセスして操作するときに発生する競合状態をターゲットとします。実行の結果はアクセスが行われる特定の順序に依存します。攻撃者は「競合の実行」、リソースの変更、および通常の実行フローの変更によって競合状態を利用することができます。たとえば、ファイルにアクセスする際に競合状態が発生する可能性があります。攻撃者は元のファイルのバージョンに置き換えて、システムに悪意のあるファイルを読み取らせます。
26 27	シンボリックリンクによる競合条件の活用		この攻撃は、機密ファイルに書き込むためにシンボリックリンク（Symlink）の使用を利用します。攻撃者は、アクセスできないターゲットファイルへのSymlinkを作成することができます。特権プログラムがSymlinkリンクと同じ名前の一時ファイルを作成しようとすると、攻撃者のSymlinkリンクが指すターゲットファイルに書き込みを行います。攻撃者が一時ファイルに悪意のあるコンテンツを挿入する可能性がある場合、Symlinkを使用して機密ファイルに書き込みます。レース（競合）は、システムが一時ファイルが存在するかどうかをチェックしてからファイルを作成するために発生します。攻撃者は通常、チェックと一時ファイルの作成の間にSymlinkを作成します。
26 29	チェック・タイムと使用時間（TOCTOU）競合条件の活用		この攻撃は、リソースのチェック（状態）とリソースの使用時間の間に発生する競合状態を対象としています。典型的な例はファイルアクセスです。ターゲットプログラムがファイルに初めてアクセスするときとターゲットプログラムがファイルを使用するときの間にリソースを変更することを意味する "レースの実行"によって、ファイルアクセス競合状態を悪用することができます。この間、敵がファイルを置き換えたり変更したりすることで、アプリケーションが予期せぬ動作をする可能性があります。

74	ユーザの状態を操作	タイミングと状態を操作する	攻撃者は標的のソフトウェアによって維持される状態情報を、ユーザがアクセス可能な場所に変更します。成功すると、標的のソフトウェアはこの汚染された状態情報を使用し、意図しない方法で実行されます。状態管理はアプリケーション内の重要な機能です。アプリケーションによって維持されるユーザ状態には、ユーザー名、支払い情報、ブラウズ履歴、およびショッピングカート内のアイテムなどのアプリケーション固有のコンテンツが含まれます。ユーザの状態を操作することは、攻撃者が特権を昇格させたり、不正な取引を行ったり、アプリケーションの流れを変更したりして特定の利益を得るために使用することができます。
74 140	複数のフォームセット内の中間フォームのバイパス		一部のWebアプリケーションでは、ユーザがWebフォームの順序付けられた順序で情報を提出する必要があります。これは、大量の情報が収集されている場合や、以前のフォームの情報を使用してフィールドを事前入力したり、アプリケーションが収集する必要のある追加情報を判断したりする場合によく行われます。シーケンス内のさまざまなフォームの名前を知っている攻撃者は、後のフォームの名前に明示的に入力し、前のフォームを先に進むことなくそのフォームに移動できます。これにより、情報の不完全な収集、攻撃者が提出した情報に関する誤った仮定、またはアプリケーションの機能を損なう可能性のあるその他の問題が発生する可能性があります。