FRAGILIDADES NOS SISTEMAS DE AUTENTICAÇÃO EM USO NA INDÚSTRIA FINANCEIRA

Porquê um nome de utilizador, uma senha, e um código por sms podem não chegar para protegê-lo?

Bruno Ronda
vic.brunoronda@gmail.com
BlueSky @vicvio.bsky
https://brunoronda.me

18 de Julho, 2025

Garantias e Isenções de Responsabilidade

As informações contidas neste documento são fornecidas “tal como estão” para fins educativos, sem qualquer tipo de garantia, expressa ou implícita, incluindo, sem limitação, garantias implícitas de comercialização, adequação a um fim específico ou não violação de direitos.

O autor não declara nem garante que as informações acessíveis através deste documento sejam precisas, completas ou atualizadas.

Em nenhuma circunstância o autor poderá ser responsabilizado por quaisquer danos especiais, acidentais, indiretos ou consequenciais de qualquer natureza, incluindo, sem limitação, perdas de dados, receitas ou lucros, independentemente da teoria jurídica invocada, resultantes do uso ou desempenho deste documento por qualquer pessoa individual ou coletiva.

Este documento pode conter imprecisões técnicas ou erros tipográficos. As informações aqui contidas são atualizadas periodicamente; essas atualizações serão incorporadas em novas edições do documento. O autor reserva-se o direito de efectuar melhorias e/ou alterações nos produtos, documentos e/ou programas aqui descritos, a qualquer momento.

Abstracto

Autenticação — o ato de validar que alguém ou algo é, de fato, quem ou o que afirma ser — é um conceito simples e intuitivo. Trata-se de um princípio fundamental de segurança em qualquer sistema ou aplicativo, pois regula o acesso a áreas e funções restritas, reservadas apenas a usuários devidamente autorizados.

À medida que a web evoluiu, os mecanismos de autenticação também se transformaram, buscando maior robustez e confiabilidade. O objetivo é claro: garantir aos proprietários de aplicativos e seus usuários que aquilo protegido pela barreira de autenticação está, de fato, seguro. Curiosamente, essa evolução não foi fruto de antecipação por parte dos programadores, mas sim de reações a falhas de segurança exploradas em ataques bem documentados ao longo dos anos — especialmente até a consolidação da Web 2.0. [1]

Nos últimos dois meses, como parte dos programas de *bug hunting* da Intigriti e Bugcrowd, dediquei-me à análise e subversão de sistemas de autenticação amplamente utilizados na indústria financeira. Por sua natureza sensível, esse setor é alvo de exigências regulatórias rigorosas e de esforços deliberados por parte dos seus *stakeholders* para proteger ativos informáticos. No entanto, as vulnerabilidades encontradas não se restringem ao setor financeiro: estão presentes em aplicativos de diversos segmentos — do retalho ao entretenimento — e muitas delas são surpreendentemente recorrentes.

Introdução

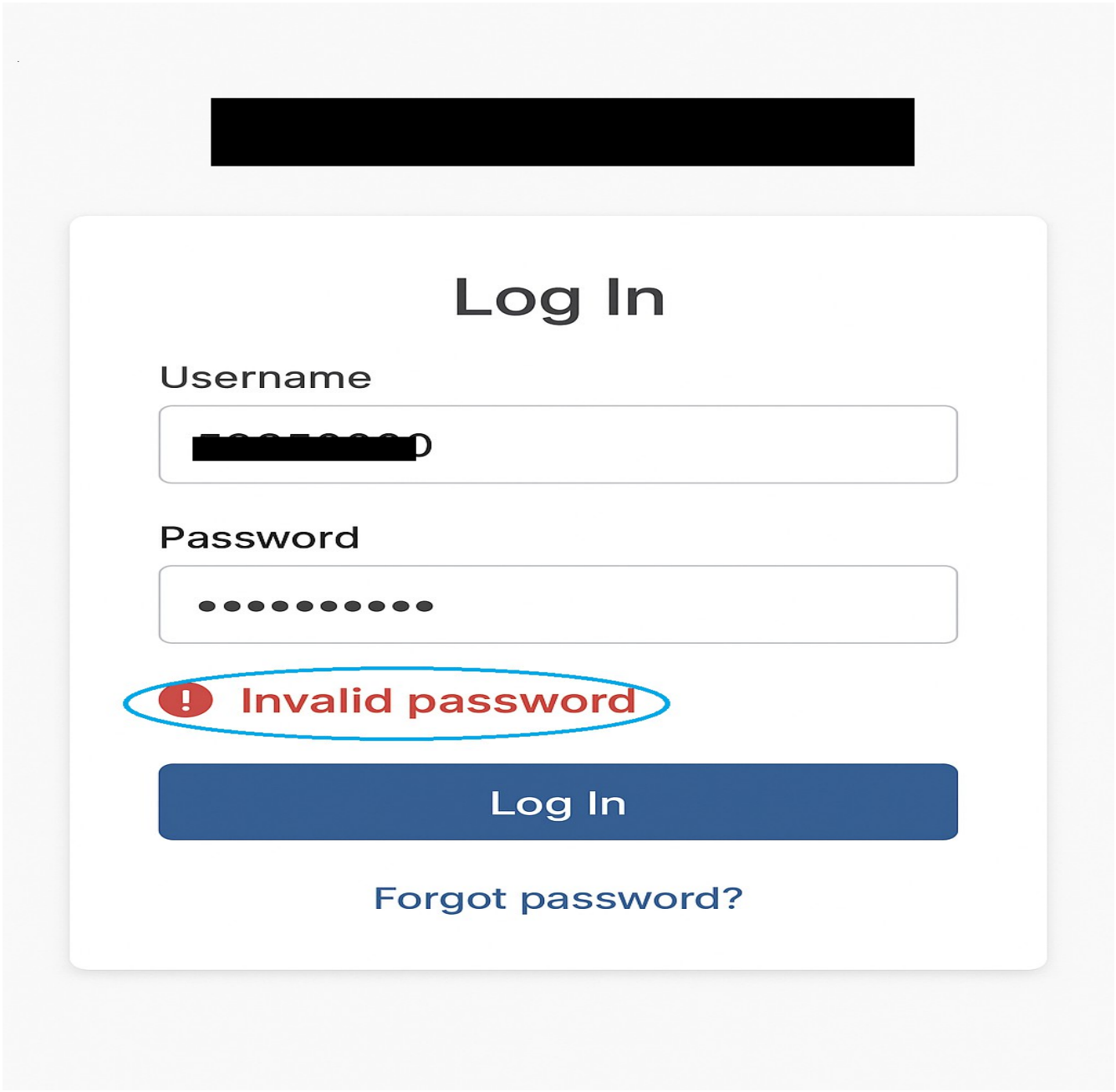
A maioria das vulnerabilidades encontradas em mecanismos de autenticação não decorre da tecnologia em si, mas sim de falhas na sua implementação. Essas falhas têm origem em fatores como o conhecimento limitado dos programadores sobre ameaças de segurança, a fraca integração de requisitos de segurança durante a fase de concepção do aplicativo e, em muitos casos, a ausência de um modelo de ameaças — ou seja, uma definição clara do que se pretende proteger, por que motivo, contra quem e com quais táticas potenciais. Em regra, não é possível desenvolver aplicativos verdadeiramente seguros sem considerar como serão atacados e por quem. [2]

Nas páginas seguintes, compartilho vulnerabilidades reais que identifiquei em aplicativos durante os últimos dois meses, no contexto de programas de *bug hunting*. Explico o impacto negativo dessas falhas tanto para os usuários quanto para as organizações proprietárias dos sistemas, e proponho medidas de mitigação que visam fortalecer os mecanismos de autenticação, de modo que cumpram efetivamente o papel a que se propõem.

1. Habilidade de Enumerar Usuários – A Falha Que Ninguém Leva a Sério

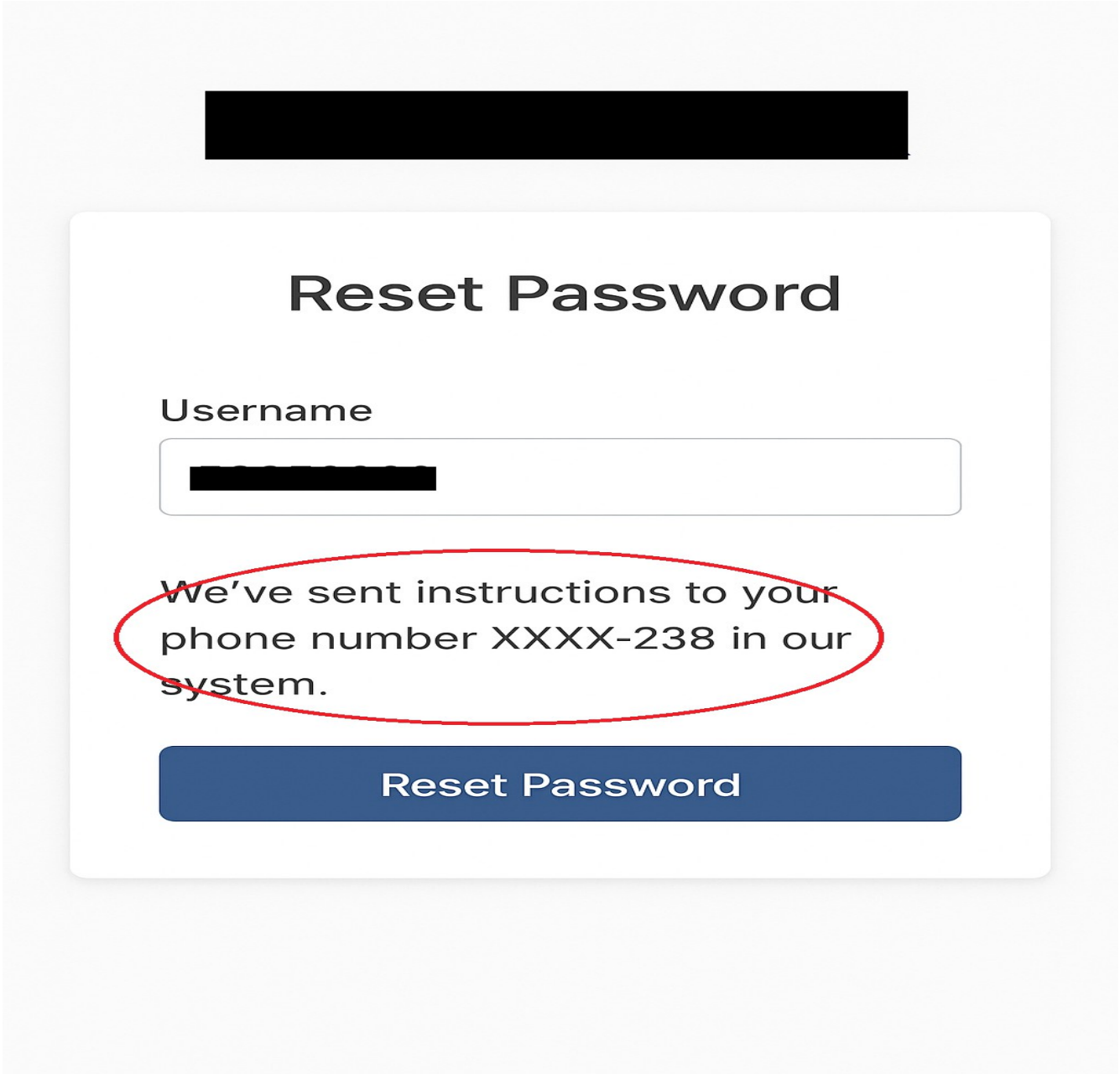
Não é exagero colocar as coisas dessa forma. Em praticamente todos os programas de caça a vulnerabilidades que já participei, a capacidade de enumerar usuários é sistematicamente excluída do escopo de falhas consideradas reportáveis.

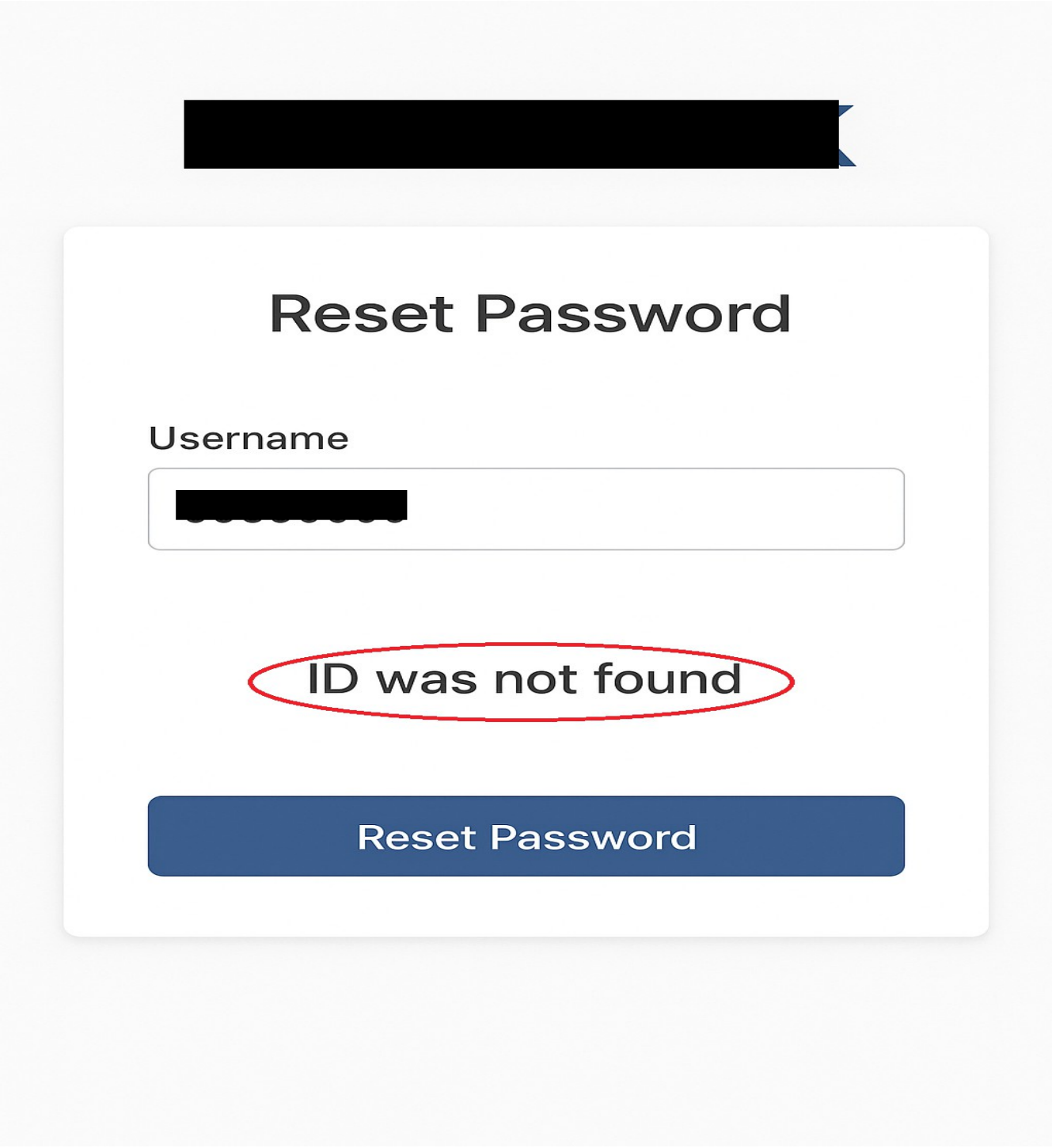
O problema, no entanto, é real — e recorrente. Em muitos sistemas da indústria financeira, os nomes ou IDs de usuários seguem padrões sequenciais como: 10000001, 10000002, 10000003... Já em aplicativos de carteiras móveis comuns em África, o ID de usuário costuma ser o próprio número de telemóvel. Mesmo quando o identificador não é sequencial, os aplicativos frequentemente revelam — de forma implícita — se um usuário é válido ou não. Um exemplo clássico: ao inserir um ID inexistente, o sistema retorna uma mensagem genérica de erro; mas ao inserir um ID válido com senha incorreta, a resposta muda para "senha inválida", confirmando a existência do usuário. Na figura 1 abaixo, esse comportamento é ilustrado com clareza.



Outra forma comum pela qual aplicativos revelam IDs de usuários ocorre durante o processo de redefinição de senha. Ao solicitar a recuperação de acesso, o sistema geralmente pede que o usuário insira seu identificador — seja nome de usuário, número de telefone ou outro ID. Em seguida, com base na resposta exibida, é possível inferir se o usuário existe ou não.

Esse comportamento é ilustrado nas figuras 2 e 3: no primeiro caso, o sistema confirma que o ID é válido ao informar que instruções para a redefinição foram enviadas; no segundo, retorna uma mensagem genérica indicando que o usuário não foi encontrado. Embora pareça inofensivo, esse tipo de resposta explícita permite a enumeração de contas válidas, o que pode ser explorado por agentes maliciosos em ataques automatizados.





À primeira vista, no par utilizado para autenticação, a parte considerada sensível e digna de proteção é a senha. A lógica predominante é que um agente externo, mesmo conhecendo o nome ou ID do usuário, estaria limitado sem acesso à senha correspondente. Afinal, após três a cinco tentativas de inserção incorreta, a conta é bloqueada — embora, como será demonstrado adiante, o próprio bloqueio possa representar uma vulnerabilidade.

Com esse cenário em mente, surge uma questão fundamental:

O Que Pode Correr Mal?

Cenário 1: Aquilo Que Todos Parecem Ignorar Sobre IDs e Senhas: Um agente malicioso consegue enumerar os IDs de dezenas, centenas, ou milhares de usuários em um aplicativo voltado para transações financeiras, utilizando qualquer uma das três técnicas abordadas anteriormente. Em vez de testar *múltiplas senhas para cada ID* — o que resultaria no bloqueio das contas — ele adota uma abordagem mais eficiente: utiliza *uma ou duas senhas* fracas e comuns, como P@ssw0rd ou 1234567, e as *testa em todos os IDs* descobertos, por meio de uma ferramenta automatizada. Tão certo quanto o nascer do sol, ele encontrará diversas contas que utilizam senhas desse tipo.

A situação torna-se ainda mais crítica em aplicativos de carteiras móveis, onde os IDs são os próprios números de telefone — o que simplifica enormemente a fase de enumeração. As senhas, por sua vez, costumam ter entre quatro e seis dígitos numéricos. Quantos usuários, você imagina, usam combinações como 0000, 1111, 1234, ou até mesmo o ano de nascimento como senha? (Sim, talvez o leitor argumente que é para isso que existem tokens de autenticação multifator ou códigos OTP. Falaremos sobre isso em breve.)

Qual o Impacto?

Uma vez que uma ou mais contas são comprometidas, o agente malicioso passa a ter liberdade para realizar ataques direcionados — como transferências indevidas, obtenção de dados de cartões de débito ou crédito, acesso a informações pessoais ou corporativas (no caso de contas empresariais), e até mesmo a subversão de camadas adicionais do sistema. Isso ocorre porque muitas das proteções existentes antes da autenticação deixam de atuar após o *login*. Em outras palavras, os desenvolvedores frequentemente assumem que todo usuário autenticado é legítimo e utilizará o sistema conforme o propósito original.

Falemos agora sobre por que o bloqueio de conta pode ser uma espada de dois gumes — e, em certos casos, agir contra o próprio objetivo de proteger os usuários.

Cenário 2: Bloqueio Em Massa como Vetor de Ataque: Neste caso, o objetivo do agente malicioso é conduzir um ataque de recusa distribuída de serviço (DDoS) — não contra a infraestrutura técnica, mas contra os próprios usuários. A intenção é privá-los do acesso às suas contas e, por consequência, impedir que a instituição financeira receba as comissões geradas pelas transações realizadas na plataforma.

De posse dos IDs de usuários, o atacante utiliza uma ferramenta automatizada para inserir senhas incorretas de forma deliberada, repetindo o processo três a cinco vezes por conta. Com isso, consegue bloquear simultaneamente milhares ou até centenas de milhares de contas legítimas ao mesmo tempo, gerando um impacto operacional significativo e potencialmente devastador.

Qual o Impacto?

Para recuperar suas senhas, os usuários legítimos geralmente precisam contatar um balcão ou loja da instituição financeira. Imagine, então, um cenário em que esse tipo de ataque ocorra fora do horário de expediente — na véspera de um fim de semana prolongado, como o Natal ou o Réveillon. Sem acesso às suas contas bancárias ou carteiras móveis, ora bloqueadas, muitos usuários teriam o feriado arruinado. O comércio e a indústria da hospitalidade perderiam receita justamente em um período de pico, e as próprias instituições financeiras sofreriam prejuízos operacionais e reputacionais. E considere o volume de trabalho administrativo necessário para desbloquear milhares de contas no primeiro dia útil após o feriado.

Um exemplo ainda mais insidioso de ataque direcionado ocorre em ambientes de competição, como leilões online. Sabendo o nome de usuário do licitador com a oferta mais alta — informação que, em muitos aplicativos, é revelada junto ao valor da licitação — o agente malicioso aguarda os minutos finais da disputa para inserir repetidamente senhas incorretas associadas aos IDs dos principais concorrentes. Com isso, bloqueia suas contas e impede que reajam a tempo. Em seguida, realiza seu próprio lance e garante a vitória, explorando uma falha de autenticação que nada tem a ver com tecnologia avançada — apenas com lógica e timing.

Mitigação:

1. **Evite IDs previsíveis:** Não utilize identificadores sequenciais, e-mails corporativos, números de telefone ou outros padrões facilmente dedutíveis. IDs previsíveis facilitam a enumeração de usuários e aumentam a superfície de ataque.
2. **Mensagens de erro ambíguas:** As respostas do sistema a tentativas de autenticação mal-sucedidas ou de recuperação de senha devem ser genéricas e não revelar se o usuário existe ou não. Exemplos recomendados:
    - “Usuário ou senha incorreta.”
    - “Se o usuário inserido existir no sistema, ele receberá uma mensagem por SMS ou e-mail.” Essa abordagem reduz a possibilidade de enumeração automatizada.
3. **CAPTCHA com entropia e limitação de tráfego:** Implemente CAPTCHA com alta entropia após múltiplas tentativas falhadas, associando-o à limitação de tráfego por cliente (navegador, dispositivo e IP) ou por conta.

- Após 5 tentativas falhadas, exija CAPTCHA e imponha um tempo de espera de 3 minutos antes que o usuário possa voltar a tentar autenticar-se.
  Se a tentativa seguinte falhar, reaplique o CAPTCHA e aumente o tempo de espera para 5 minutos.

- Continue progressivamente até um máximo de 30 minutos. Durante o período de espera, novas tentativas de autenticação devem ser tratadas como exceções silenciosas pelo servidor — filtradas e sem retorno de erro explícito.

2. O Falso Senso de Segurança Criado Pelas Senhas Fracas

Toda organização — seja grande ou pequena — deveria adotar uma política rigorosa para a criação de senhas robustas. Essa política deve estar claramente documentada, implementada pelos programadores ou fornecedores dos sistemas adquiridos, e seu cumprimento deve ser obrigatório tanto na geração inicial de senhas quanto na sua alteração pelos usuários.

Constatei, com preocupação, o quão distante essa prática está da realidade. Mesmo instituições financeiras multimilionárias, com milhões de clientes em sua base, bem como fintechs emergentes e plataformas de criptomoedas, ainda permitem o uso de senhas frágeis e previsíveis. No caso das carteiras móveis amplamente utilizadas em África, essa prática é praticamente institucionalizada.

Curiosamente, muitos aplicativos demonstram uma intenção de orientar os usuários na escolha de senhas fortes — geralmente por meio de validação em JavaScript no navegador, que verifica critérios como:

- Mínimo de 8 carateres
- Uma letra maiúscula
- Um caratere especial
- Um número

Se a senha escolhida cumpre os critérios mínimos — como ter oito caracteres, incluir uma letra maiúscula, um número e um caractere especial — o sistema geralmente indica que se trata de uma “senha forte”. Essa validação, no entanto, pode criar um falso senso de segurança. Por quê?

Porque senhas como P@ssw0rd são amplamente utilizadas e estão entre as mais comuns encontradas em vazamentos de dados após acessos indevidos a bases de dados de organizações. Apesar de atenderem aos critérios técnicos, essas senhas são previsíveis e fazem parte de listas usadas em ataques de dicionário — uma técnica que explora padrões humanos de criação de senhas e ignora a complexidade superficial.

Olhe novamente para P@ssw0rd:

- Tem oito caracteres
- Inclui uma letra maiúscula
- Contém um número
- Possui um caractere especial

Cumpre todos os requisitos formais de uma senha “forte” — mas continua sendo vulnerável, justamente por sua popularidade e previsibilidade.

O Que Pode Correr Mal?

Cenário: Quando Uma Conta Bancária Vira Trampolim para o Roubo de Identidade: Já abordamos anteriormente a questão da impersonificação de usuários e o acesso indevido às funcionalidades legítimas de uma conta. Para evitar repetição, concentro-me aqui em um risco ainda mais insidioso: o roubo de identidade.

Considere o seguinte: se um usuário escolhe uma senha de dicionário para seu aplicativo bancário — justamente onde deveria ser mais cauteloso — é altamente provável que repita esse comportamento em outros contextos. Na verdade, muitos usuários reutilizam a mesma senha em múltiplos sistemas e serviços que exigem autenticação.

Nesse cenário, o aplicativo bancário deixa de ser apenas um alvo isolado e passa a funcionar como trampolim para uma operação mais ampla. Um agente malicioso, ao comprometer essa conta, pode escalar o ataque e conduzir uma operação bem-sucedida de roubo de identidade, explorando credenciais replicadas para fins fraudulentos em plataformas diversas.

Qual o Impacto?

Servir como porta de entrada para uma fraude pode gerar consequências severas: desde danos reputacionais e perda de confiança por parte do público, até penalizações regulatórias e ações civis. Cada uma dessas repercussões representa não apenas um risco jurídico, mas também um custo financeiro significativo para a instituição — com potencial de comprometer sua imagem, sua base de clientes e sua sustentabilidade operacional.

Mitigação:

1. **Exija um comprimento mínimo de 12 caracteres**

   O aplicativo deve obrigatoriamente exigir senhas com pelo menos 12 caracteres. Não deve gerar nem permitir que os usuários criem senhas abaixo desse limite, conforme recomendado pela NIST.

2. **Implemente uma lista de senhas proibidas**

   O sistema deve rejeitar senhas fracas e previsíveis como P@ssw0rd, Password!123, Qwerty123@, entre outras. Essa lista pode ser baseada em bases de dados públicas de senhas comprometidas, como a do serviço Have I Been Pwned, integradas via consulta automática ou armazenadas localmente.

3. **Hashing e Salting**

   Nunca armazene senhas em texto simples. Utilize algoritmos modernos e robustos como **bcrypt**, **scrypt** ou **Argon2** para realizar o hashing. Cada senha deve receber um *salt* único e aleatório antes do processo de hashing, neutralizando ataques com tabelas arco-íris pré-computadas.

4. **Evite mudanças periódicas obrigatórias**

   Não force os usuários a alterar suas senhas a cada 90 dias, a menos que haja evidência de comprometimento. Essa prática, além de gerar frustração, incentiva padrões previsíveis como P@sswordVerao2025! tornar-se P@sswordInverno2025!, o que compromete a segurança. [6]

5. **Eduque os usuários sobre boas práticas de senha**

   Incentive o uso de frases de acesso ou acrónimos com significado pessoal, que são mais fáceis de lembrar e mais difíceis de decifrar. Exemplos:

- Frase de acesso: #CháVerdeÀs16hComBoloDeSura
- Acrónimo: OqDpddmjqoHns@Mt19:6 (de "O que Deus pôs debaixo do mesmo jugo, que o homem não separe – Mateus 19:6")

3. Porquê o Uso de Multifatores Pode Não Protegê-lo

A chamada autenticação de dois fatores (2FA) — ou autenticação multifator (MFA) — começou a ser testada em aplicativos web da indústria financeira no início dos anos 2000, utilizando dispositivos físicos dedicados. A adoção ganhou ritmo entre 2010 e 2012, período marcado pela prevalência de ataques de *phishing* como o **Cross-Site Request Forgery (CSRF)**. Desde então, o uso de MFA passou a ser incorporado em regulamentos como o **GDPR**, **HIPAA** e **PCI-DSS**. [4], [5]

O processo de autenticação multifator — também conhecido como verificação em dois passos (2SV) — baseia-se no princípio de que, para acessar um sistema, o usuário deve apresentar mais do que uma senha. Esse segundo fator pode ser classificado conforme sua natureza, como ilustrado na tabela abaixo:

Fator	Exemplos
Algo Que Você Sabe	Senhas, PINs, perguntas de segurança
Algo Que Você Tem	Dispositivos de autenticaçao, certificados, email, SMS, chamada
Algo Que Você É	Digitais, reconhecimento facial, verificaçao da íris
Localização	Endereços de IP e geolocalização

O princípio básico do MFA é claro: mesmo que um atacante consiga obter a senha de um determinado usuário, é altamente improvável que também tenha acesso ao seu telemóvel, ao dispositivo de autenticação ou, evidentemente, aos seus dados biométricos.

No entanto, é preciso cautela ao considerar o e-mail como segundo fator de verificação. Em muitos casos, o e-mail não deveria ser tratado como um canal seguro de autenticação — a menos que ele próprio esteja protegido por MFA. Isso porque, se comprometido, o e-mail pode servir como vetor para escalar o ataque, inclusive facilitando a redefinição de senhas em outros sistemas.

Além disso, o uso do e-mail como segundo fator pode ser conceitualmente falho: ele verifica o mesmo tipo de fator duas vezes — ou seja, duas senhas. A senha de acesso ao aplicativo financeiro e a senha de acesso ao e-mail pertencem à mesma categoria ("Algo Que Você Sabe"), o que não configura autenticação multifator genuína. Por isso, em muitas circunstâncias, o uso do e-mail como segundo fator não se qualifica como 2FA.

Dada a dificuldade de implementar reconhecimento biométrico em ambientes web via navegador, o uso de fatores do tipo “Algo Que Você É” ainda não se consolidou na indústria financeira. Padrões como o WebAuthn, embora promissores, permanecem em estágio inicial de adoção e enfrentam limitações de compatibilidade e infraestrutura.

Na prática, o segundo fator mais comum utilizado pelo setor é “Algo Que Você Tem” — geralmente na forma de:

- SMS com código de verificação
- Notificações *push*
- Códigos numéricos temporários gerados por aplicativos autenticadores como Google Authenticator ou Microsoft Authenticator

Esses métodos oferecem uma camada adicional de proteção, mas também apresentam vulnerabilidades específicas, como interceptação de SMS, phishing de códigos temporários e manipulação de notificações *push* — temas que serão abordados nas seções seguintes.

Embora a autenticação de dois fatores (2FA) seja consideravelmente mais segura do que a autenticação de um fator (1FA) — ou seja, o tradicional par ID/senha — seu nível de proteção depende diretamente da qualidade da implementação. Uma 2FA mal projetada pode ser contornada ou até mesmo completamente ignorada por agentes maliciosos.

A seguir, apresento casos reais de má implementação de 2FA observados em aplicativos em produção (ou seja, já em uso por usuários finais no ambiente real), que ilustram como falhas sutis podem comprometer a eficácia do mecanismo e expor usuários a riscos significativos.

O Que Pode Correr Mal?

Cenário 1: 2FA Sem Limite de Tentativas – Um Convite ao Abuso: Um aplicativo implementa a autenticação de dois fatores (2FA) enviando um *token* ou código OTP por SMS ao telemóvel do usuário que tenta se autenticar. No entanto, não impõe qualquer limite ao número de tentativas de login que podem ser feitas dentro de um determinado intervalo de tempo.

Um agente malicioso, já de posse do ID e da senha da vítima, tenta acessar o sistema repetidamente. A cada tentativa, o sistema envia uma nova SMS não solicitada ao usuário legítimo. Sem perceber que está sendo alvo de um ataque, o usuário se irrita com o aplicativo — que aparentemente está “enchendo” sua caixa de mensagens sem motivo — e passa a desconfiar da seriedade do canal de autenticação.

Esse mesmo efeito pode ocorrer com notificações *push*: múltiplas tentativas de login geram uma enxurrada de alertas, confundindo o usuário e minando a confiança no mecanismo de segurança. O resultado é duplo: o atacante permanece ativo e invisível, enquanto o usuário perde a confiança na tecnologia que deveria protegê-lo.

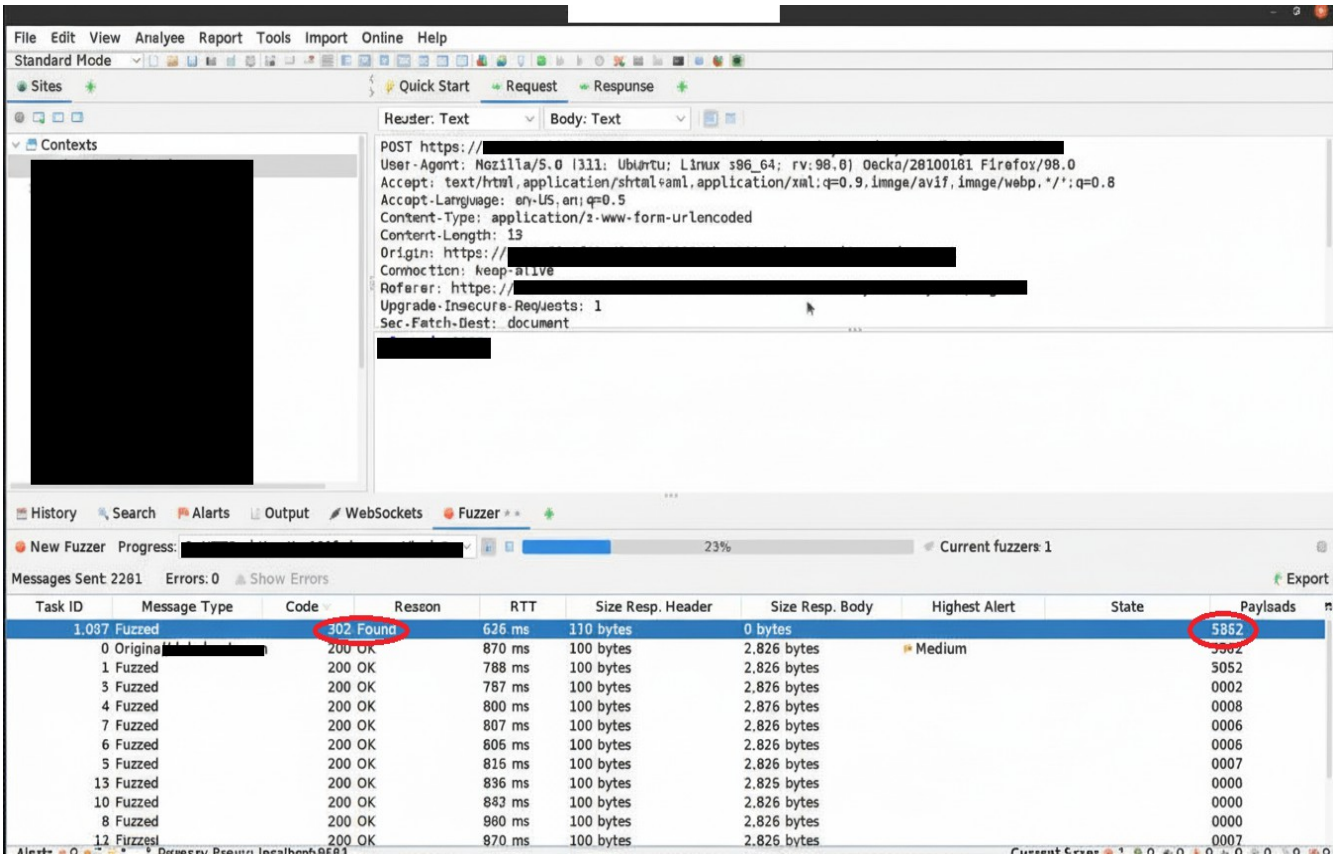
Qual o Impacto?

Ao não impor limites ao número de tentativas de autenticação com OTP dentro de um intervalo

de tempo, o aplicativo torna-se vulnerável a abusos — podendo ser explorado como uma plataforma involuntária de envio de *spam*. A vítima, ao receber uma enxurrada de mensagens não solicitadas, perde a confiança no canal de autenticação e não percebe que está sendo alvo de um ataque.

Esse tipo de comportamento pode ser interpretado como vandalismo cibernético — na melhor das hipóteses. Na pior, trata-se de uma tentativa deliberada de quebra de segurança, onde o atacante utiliza ferramentas automatizadas para descobrir o código OTP por força bruta.

Na figura 4 abaixo, observa-se um exemplo concreto: um OTP de 4 dígitos, que representa um máximo de 10,000 combinações foi decifrado em apenas 9 minutos, evidenciando como a ausência de controle de requisições pode comprometer completamente a eficácia da autenticação multifator.

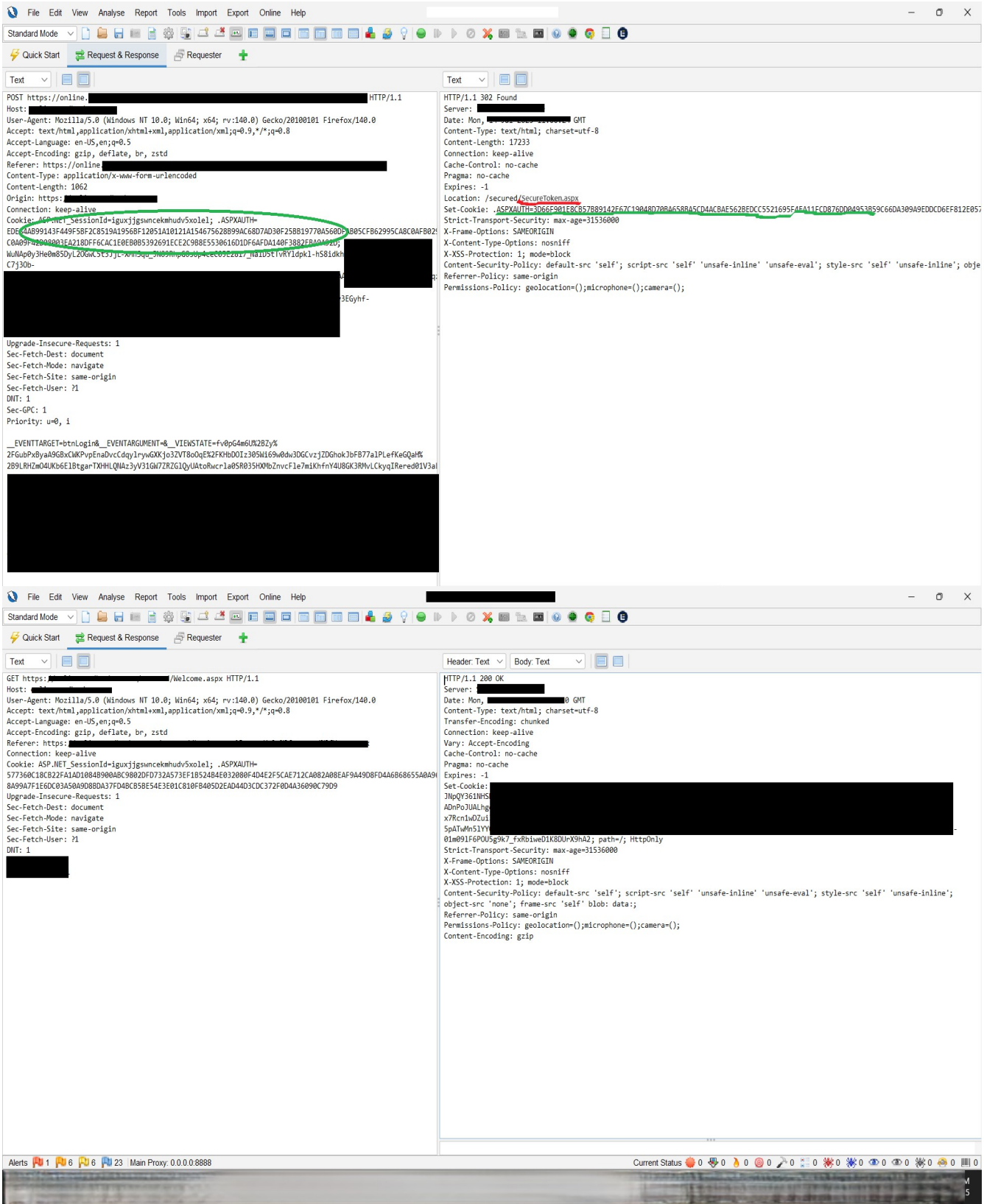


<u>*Cenário 2: Sessão Prematura — Quando o Cookie Antecede a Verificação:*</u> o usuário insere suas credenciais de acesso e realiza o *login*. Embora seja redirecionado para a página de verificação de segundo fator (OTP), o aplicativo já concede um **cookie de sessão** — o que, na prática, significa que o usuário é considerado como "autenticado".

Esse comportamento é perigoso. O agente malicioso na posse de credenciais da vítima pode ignorar a página de OTP e acessar diretamente uma área restrita — seja via navegador ou por meio de um proxy — o servidor carrega a página normalmente, sem verificar se a OTP foi inserida ou validada. Isso ocorre porque o cookie de sessão já está ativo, e o *backend* não impõe

uma verificação adicional.

Nas figuras 5 e 6, observa-se esse fluxo falho: o usuário recebe o cookie .ASPXAUTH após ser redirecionado para a página de OTP, e portanto, antes de concluir a autenticação multifator; com esse cookie e usando um proxy, o ator malicioso requisita a página *Welcome.aspx* e o sistema não valida se o segundo fator foi efetivamente cumprido antes de conceder acesso a recursos sensíveis.



Qual o Impacto?

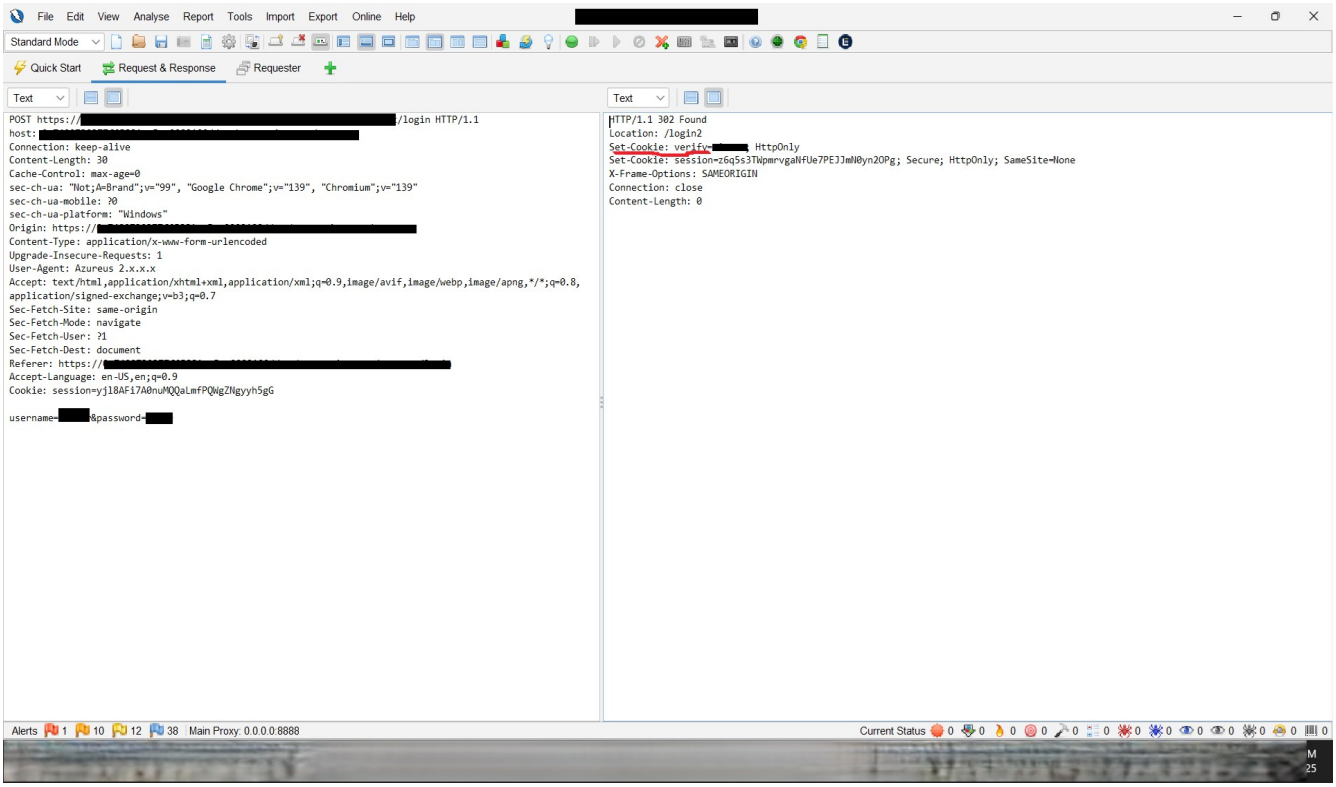
A implementação frágil de autenticação de dois fatores descrita acima compromete completamente o objetivo do mecanismo. Na prática, tudo o que um atacante precisa é descobrir o ID e a senha do usuário — o segundo fator, que deveria ser uma barreira adicional, torna-se irrelevante. O sistema retorna, literalmente, à estaca zero em termos de segurança.

Esse tipo de falha não apenas expõe dados sensíveis, mas também mina a confiança dos usuários e das partes reguladoras, que assumem que a presença de 2FA implica proteção robusta. Quando mal implementada, a autenticação multifator pode gerar uma falsa sensação de segurança — e isso, em segurança da informação, é frequentemente mais perigoso do que a ausência declarada de proteção.

Cenário 3: Parâmetros Modificáveis — Quando o Servidor Perde o Fio da Meada:
Este cenário é semelhante ao anterior, mas a falha reside em um ponto ainda mais estrutural: o servidor não estabelece uma ligação segura entre o primeiro passo da autenticação (ID/senha) e o segundo (inserção do OTP). Após o *login* inicial, o aplicativo armazena o ID do usuário em um parâmetro modificável — como um cookie de sessão ou um campo oculto no HTML da página.

Esse comportamento abre uma brecha grave. Um agente malicioso pode interceptar ou manipular esse parâmetro, substituindo o ID por outro qualquer antes de enviar o OTP. Como o servidor não valida a continuidade lógica entre os dois passos, ele aceita o OTP como válido para o ID adulterado — concedendo acesso indevido a uma conta que o atacante nunca autenticou corretamente.

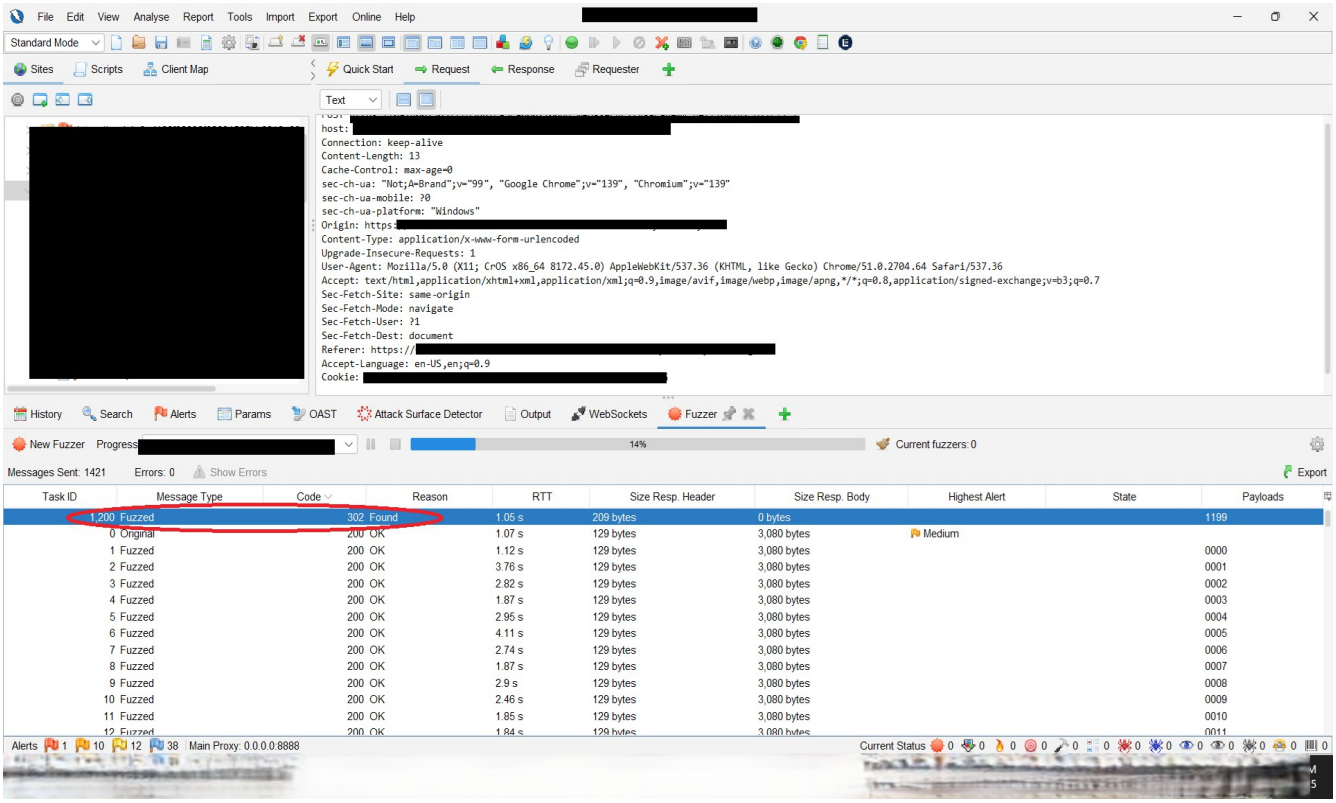
Na figura 7, observa-se esse fluxo vulnerável: o ID do usuário é armazenado de forma exposta e manipulávelno cookie, permitindo que o segundo fator seja aplicado fora de contexto, quebrando completamente o modelo de autenticação. O agente malicioso precisa apenas de modificar o parâmetro cookie: verify=xxxxx para o ID da vítima.



Mesmo que o aplicativo envie o OTP para o usuário correto — identificado pelo cookie ou parâmetro oculto original — a ausência de proteções como limite de tentativas, CAPTCHA ou tempo de expiração torna o sistema vulnerável a ataques automatizados.

Considerando que o OTP possui 4 a 6 dígitos, isso equivale a 1 milhão de combinações possíveis. Na prática, um atacante não precisa testar todas: estatisticamente, 500.000 tentativas seriam suficientes para encontrar o código correto. A uma média de 1.000 requisições por segundo, esse ataque pode ser concluído em menos de 15 minutos.

Na figura 8, observa-se esse cenário em ação: um scanner automático iterando sobre combinações até decifrar o OTP, explorando a ausência de controle de requisições e validação contextual. O resultado é claro — a autenticação multifator, que deveria proteger o usuário, é facilmente contornável por falhas de implementação.



Qual o Impacto?

O atacante consegue acessar a conta da vítima sem precisar conhecer sua senha — e muito menos o código OTP. Na prática, basta que tenha uma conta legítima no aplicativo ou que comprometa uma única vítima para escalar o ataque e comprometer potencialmente **todos os usuários registrados**.

Essa falha transforma o aplicativo em um vetor de ataque sistêmico. O que deveria ser uma barreira de segurança torna-se uma porta aberta, permitindo que um único ponto de entrada seja explorado para invadir múltiplas contas. O impacto é devastador: perda de confiança, exposição em massa, e risco regulatório elevado — tudo derivado de uma falha lógica que poderia ser evitada com validação contextual e controle de sessão adequados.

Mitigação:

1. **Implemente limites de requisições com base em IP e geolocalização**
   Sempre que um determinado número de OTPs for enviado para uma conta no mesmo processo (como autenticação), aplique um limite de requisiçõs por IP e região. Após atingir esse limite, exija CAPTCHA para cada nova tentativa não-sucedida, dificultando ataques automatizados e protegendo o canal de entrega.

2. **Defina um tempo de expiração curto para OTPs**
   As OTPs devem expirar dentro de um intervalo razoável — idealmente entre 5 e 15 minutos, conforme recomendado por práticas como TOTP (Time-Based One-Time Passwords). Isso reduz a janela de oportunidade para ataques de força bruta ou *replay*.

3. **Forneça contexto nas notificações *push* ou códigos numéricos**
   Ao utilizar autenticadores como Google ou Microsoft Authenticator, garanta que a notificação recebida pelo usuário contenha informações contextuais claras — como o nome do aplicativo, localização aproximada, horário e tipo de operação — para que o usuário possa tomar uma decisão consciente ao aprovar ou rejeitar o acesso.

4. **Audite a lógica de autenticação no servidor e no código-fonte**
   Revise cuidadosamente o fluxo de autenticação para garantir que:
   - O cookie de sessão não seja concedido antes da verificação do segundo fator.
   - O ID do usuário não seja armazenado em parâmetros manipuláveis como campos ocultos ou cookies não protegidos.
   - O segundo fator esteja logicamente vinculado ao primeiro, impedindo que o OTP seja aplicado fora de contexto.

Conclusão

Os cenários descritos ao longo deste documento estão longe de ser hipotéticos. São reais, observados diretamente pelo autor em plataformas *em produção*, pertencentes a instituições financeiras tradicionais, fintechs e sistemas de criptomoedas.

Esta breve análise evidenciou que uma vulnerabilidade aparentemente pequena nos mecanismos de autenticação pode ter implicações muito mais sérias do que o comprometimento isolado de uma conta. Em muitos casos, trata-se de uma falha sistêmica — capaz de escalar rapidamente e afetar toda a base de usuários.

Não subestime a enumeração de usuários. Embora considerada inofensiva por muitos, há situações em que **o simples fato de saber que uma pessoa ou entidade possui conta em determinado sistema já representa uma violação de segurança**.

Não delegue a responsabilidade da segurança aos usuários. Medidas robustas exigem esforço humano, sim — mas a natureza humana tende a buscar atalhos. E como demonstrado, **basta comprometer um único usuário para abrir caminho ao comprometimento de todo o sistema**.

Certifique-se de que o mecanismo de autenticação multifator (2FA) foi implementado de forma sólida, com validação contextual, controle de sessão e proteção contra automação. E embora o uso de SMS seja comum em África como segundo fator, **em um cenário ideal, a indústria financeira deveria migrar para métodos mais seguros** — como dispositivos dedicados ou aplicativos autenticadores que geram códigos.

O Autor

Bruno Ronda é pesquisador em cibersegurança e ex-executivo sénior do setor de resseguros. Originalmente formado em Ciências da Computação, e com década e meia de experiência na gestão de riscos financeiros e operacionais complexos, ele traz uma perspectiva única à cibersegurança — integrando precisão técnica com visão estratégica de negócios.

Seu trabalho abrange testes de intrusão, descoberta de vulnerabilidades, práticas de desenvolvimento seguro, revisão de código-fonte, e análise de superfície de ataque, com foco especial em sistemas baseados na *web* e dispositivos móveis, e exposição organizacional. Com base em sua vivência executiva, Bruno se especializa em traduzir achados técnicos em *insights* acionáveis para lideranças, garantindo que as iniciativas de segurança estejam alinhadas com os objetivos de resiliência e crescimento do negócio.

Sua pesquisa atual explora a resiliência em autenticação e abordagens pragmáticas para reduzir a exposição a ameaças cibernéticas emergentes, com ênfase em conectar equipes técnicas a tomadores de decisão.

Referências

[1] A. dos Santos, C. H. Oliveira, and M. Silva, *Novas tendências para autenticação na Internet*, São Paulo: FIAP, 2019.

[2] OWASP Foundation, *OWASP Top 10 – A07:2021 Identification and Authentication Failures*, OWASP, 2021. [Online]. Disponível: https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/

[3] Google Cloud, *Best practices for protection from automated threats*, reCAPTCHA Enterprise Documentation, 2023. [Online]. Disponível: https://cloud.google.com/recaptcha/docs/best-practices-oat

[4] Palo Alto Networks, *What is the Evolution of Multifactor Authentication*, Cyberpedia, 2023. [Online]. Disponível: https://www.paloaltonetworks.com/cyberpedia/what-is-the-evolution-of-multi-factor-authentication

[5] IS Decisions, *Why 2FA for Financial Services Is (Still) So Important*, 2023. [Online]. Disponível: https://www.isdecisions.com/en/blog/mfa/importance-of-2fa-for-financial-services

[6] *NIST SP 800-63B, Section 5.1.1.2 – Memorized Secrets*