

安全に任せるための セキュリティ設定

任せる範囲を、安心して広げるための、最後の1段



許可リスト・禁止リストだけでは、まだ足りない

許可・禁止・確認の3つのリスト（第6回）は、Claude Codeにお願いを聞くかどうかという、入り口の判断だった。

委任を広げ、並列に動かすほど、目を離す時間も増える。お願いの手前に、もう1段、仕組みで守る層が要る。

今日扱うのは、その「もう1段」。全部で6つある。



①サンドボックス — 実行そのものを、囲いの中に

許可・禁止リストは、あくまで入り口の判断。いったん実行を許したコマンドが、思っていたのと違うファイルまで触ってしまったら、入り口ではもう止められない。

そこで、実行そのものを囲いの中に閉じ込める。囲いの外のファイルや通信先には、そもそも手が届かない。
書き方：/sandbox コマンド、または設定ファイルの sandbox.enabled という項目。

入り口の判断と、実行中の囲い。置き換えではなく、両方そろって初めて効く。



②禁止リストは、どんなモードでも効く

パスワードが書かれたファイルは絶対に読ませない。危険な削除コマンドは絶対に打たせない。そういう最後の砦を、禁止リストに書いておく。

書き方：設定ファイルの `permissions.deny` というリストに、ルールを1行ずつ書く（例：`Read(./env)`）。

許可リストは、あとから書き換えられる。でも禁止リストだけは、誰にも上書きされない。確認を全部飛ばす危険なモードでも、例外なく効く。



③接続許可リスト — つながっていい相手だけの名簿

囲いの中からも、外の情報を調べたいときがある。あらかじめ、つながっていい相手だけをリストにしておける。

名簿にない相手には確認を挟むことも、一切つなげないという厳しい設定にすることもできる。

書き方：設定ファイルの `sandbox.network.allowedDomains` というリスト。厳しくするなら `strictAllowlist`。



④危険モードに、鍵をかける — 似ているようで、違う2つの設定

本物の鍵

`permissions.disableBypassPermissionsMode`

- ・モードに、そもそも入れなくする
- ・誰も、危険なモードに入れない
- ・本当に守りたいときは、こちら

飾りのスイッチ

`skipDangerousModePermissionPrompt`

- ・入るときの確認の一言だけを省略する
- ・モード自体には、誰でも入れてしまう
- ・安心したつもりが、安心できていない



⑤フックの検証 — 置くだけでは動かない

危険なコマンドを実行の直前で自動的に止める、見張り役をフックに任せられる（第6回）。ただし、スクリプトを置くだけでは動かない。設定ファイルに登録して、はじめて動く。

見張り役として頼るなら、本当に登録されているかを、あとで必ず確認する。

書き方：設定ファイルの hooks というトップレベルの項目に登録する。

中身を暗号で証明してくれる仕組みはない。信用できるのは、自分が設定ファイルに書いたものだけ。



⑥組織が管理する設定

1人で使っているなら、ここまでは全部自分で決めること。チームや組織で使うときは、全員に同じ禁止リスト・同じサンドボックス設定を使ってほしい。

組織が管理する設定は、個人の設定より優先される。状態を見るコマンドで、出どころを確認できる。

確認コマンド：`/status`。ファイルの場所は、OSごとに決まっている。



安全を先に固めるから、 思い切って任せられる

サンドボックスで範囲を囲い、禁止リストで最後の砦を作り、接続の名簿で相手を絞る。危険なモードには本物の鍵をかけ、フックの登録を確認し、組織の設定でチームをそろえる。

安全と、委任の広さは、対立するものではない。



お疲れさまでした

自分の現在地を測るところから始まった、全8回。
任せる範囲を、安全に広げるところまで来た。

- サンドボックスと禁止リストで、実行を封じた
- つながる相手を絞り、危険モードに鍵をかけた
- フックの登録と、組織の設定を確認した

これで、ループ／ハーネス
エンジニアリング講座は完了です。

