

DeFi on Solana • Week 5

Speaker script: every slide with its notes. Facts checked 11 to 14 Oct 2026; re-check anything marked "verify".

1

Week 5 of 6

5

Risk & security

Task Write a post-mortem of one incident, then audit one real token on Orb.

Blog post The risks of DeFi: lessons from Solana's failures.

DeFi on Solana • Week 5

Week 5 goal: mentees stop memorising hack stories and start recognising four patterns. By the end they can look at any incident and say which bucket it belongs to, and look at any token page and say whether it is safe to touch.

Timing: 30 minutes of teaching. The replays are click-driven, so you control the pace. Spend the most time on bucket 1 (the Solana-specific bug) and on Drift (the biggest loss of 2026, and no code bug at all).

Every figure on these slides comes from the course handbook, fact-checked on 11 October 2026. Loss figures are ranges because sources disagree. Say "reported" out loud.

2 Today

1 Patterns, not stories

2 Four root causes

3 Risk with no attacker

4 How analysts judge a protocol

→ Post-mortem + token audit

Today

Tell them the task first: by the end of class they will pick one incident and explain it like an analyst, and check one real token for red flags on Orb.

The framing for the whole week: hacks are not random bad luck. Almost every Solana loss falls into one of four buckets, and each bucket has one question that would have stopped it.

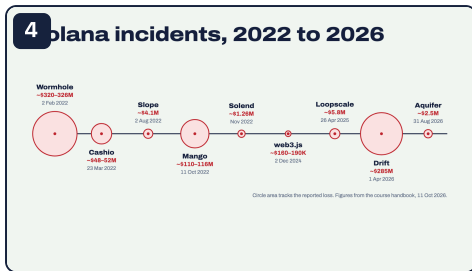
3

Patterns, not stories

Nine incidents. Four root causes.

Patterns, not stories

No notes. Let the slide speak.



Solana incidents, 2022 to 2026

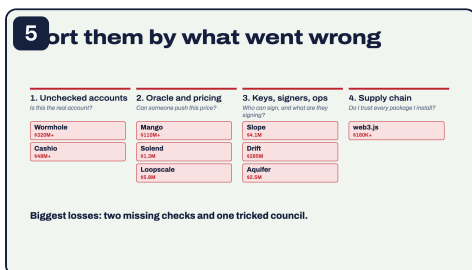
Click through one incident at a time. Circle area tracks the reported loss, so the two giants jump out: Wormhole in 2022 and Drift in 2026.

Read them in order:

- Wormhole, 2 Feb 2022, about \$320–326M. Bridge.
- Cashio, 23 Mar 2022, about \$48–52M. Stablecoin.
- Slope, 2 Aug 2022, about \$4.1M. Wallet app.
- Mango Markets, 11 Oct 2022, about \$110–116M. Trading and lending.
- Solend, Nov 2022, about \$1.26M. Lending.
- @solana/web3.js, 2 Dec 2024, about \$160–190K. A code library.
- Loopscale, 26 Apr 2025, about \$5.8M. Lending.
- Drift Protocol, 1 Apr 2026, about \$285M. Perps and lending.
- Aquifer, 31 Aug 2026, about \$2.5M.

CertiK counted 11 significant Solana attacks in 2022 alone, about \$523M in total. 2022 was the worst year; then the losses moved from code to people (Drift).

Ask: what do you notice? Usually someone says "the big ones are rare but huge". That is the right instinct for risk: size matters more than frequency.



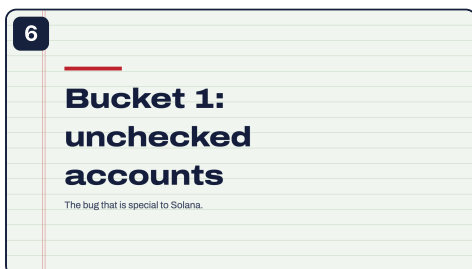
Sort them by what went wrong

Each click files one bucket. The question under each heading is the one that would have stopped every incident in that column.

1. Unchecked accounts: Wormhole, Cashio. "Is this the real account?"
2. Oracle and pricing manipulation: Mango, Solend, Loopscale. "Can someone push this price?"
3. Keys, signers and operations: Slope, Drift, Aquifer. "Who can sign, and what are they signing?"
4. Supply chain: @solana/web3.js. "Do I trust every package I install?"

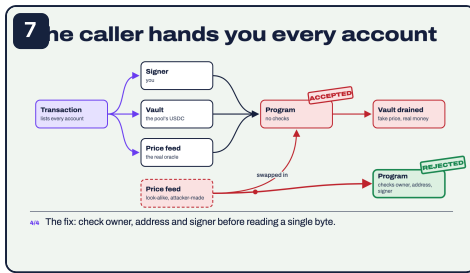
The punchline: the two biggest code losses were each one missing check. The biggest loss of all, Drift, was not code at all. That is the shape of the rest of the class.

On Ethereum the classic bug is reentrancy. On Solana it is the missing account check. That is why bucket 1 comes first.



Bucket 1: unchecked accounts

No notes. Let the slide speak.



The caller hands you every account

This is the most important Solana security idea, and finance people can get it.

A Solana program does not go and look up its data. The transaction lists every account the program will read or write, and the program receives that list as input. That is what makes Solana fast: the runtime knows in advance which transactions touch which accounts, so it can run non-overlapping ones in parallel.

The cost: anyone can build a transaction, so anyone can pass any account. If the program expects "the real price feed" and someone passes their own account with the same layout, the program has to notice. That means checking:

- owner: which program owns this account? (A real Pyth price account is owned by the Pyth program.)
- address: is it the exact account I expect?
- signer: did the right person actually sign?

Analogy for the accountants: an auditor who accepts any bank statement the client hands over, without checking it came from the bank.

Anchor does many of these checks automatically when you use typed accounts (`Account<T>`, `Signer`). Bugs appear when developers drop down to raw `AccountInfo` and forget.

10 Builders: the missing check, fixed

```
1 // Fix 2: the price feed must be the exact account, owned by the oracle program
2 #derive(Accounts)
3 pub struct Withdrawer {info: {
4     pub authority: Signer<info>,
5     if(account.info.has_one == authority)}
6     pub vault: Account<info, Vault>,
7     /// CHECK: address and owner pinned below
8     if(account.address == SOL_USD_FEED, owner == ORACLE_PROGRAM_ID)
9     pub price_feed: AccountInfo<info>,
10 }
```

Builders: the missing check, fixed

Click through the three versions.

Version 1: three holes. authority is a plain AccountInfo, so nobody checks it signed. vault is typed, so Anchor checks its owner program, but nothing ties it to this authority. price_feed is unchecked: Wormhole and Cashio in miniature.

Version 2: Signer<'info> makes Anchor require a signature. has_one = authority checks that vault.authority equals the authority account passed in. Now only the owner of this vault can withdraw.

Version 3: address = SOL_USD_FEED pins the exact account. owner = ORACLE_PROGRAM_ID checks which program owns it. In real code, prefer the oracle's typed account (for Pyth, the PriceUpdateV2 account type from the Pyth Solana receiver crate), which does the owner check for you, and also check the feed id and staleness.

Every "/// CHECK:" comment in Anchor is a place where a human promised to check something. Reviewers should read every one.

This is the builders' task this week: write version 1, write the exploit as a LiteSVM test, then fix it.

11

Bucket 2: oracles and pricing

If a price can be pushed, collateral can be faked.

Bucket 2: oracles and pricing

No notes. Let the slide speak.



Mango Markets · 11 Oct 2022

No code bug here. Every line of Mango did what it was written to do. The flaw was economic design.

1. Mango let you borrow against your account value, priced by an oracle that followed spot markets for MNGO, Mango's own thin, illiquid token.
2. The attacker funded two accounts with about \$5M USDC each (reported). Account A sold MNGO perpetual futures to account B: one short, one long, a huge position.
3. They bought MNGO on the thin spot markets the oracle watched. Reported price move: from about 3 to 4 cents to about 91 cents within minutes.
4. Account B's long position now showed an enormous unrealised profit. Mango counted that paper profit as collateral.
5. B borrowed everything Mango held, across tokens: reported ~\$110–116M.
6. MNGO fell back. The "collateral" was gone, the loans stayed: bad debt.
7. A Mango DAO vote accepted the return of \$67M as a "bounty" deal; the attacker kept the rest. Knock-on: Tulip Protocol lost about \$2.5M indirectly.

Finance link: this is collateral valuation risk. A bank would never lend against a mark-to-market value on an asset with no depth. Mango did, automatically.



Is code law?

Mentees will ask how it ended. The handbook's account:

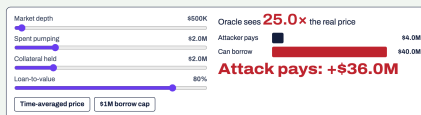
- Eisenberg publicly described it as a legal, highly profitable trading strategy.
- He was convicted in April 2024.
- On 23 May 2025 a federal judge threw out all the criminal convictions: wrong venue, and no proof that he lied to a permissionless protocol (you cannot deceive a program that never asked you a question).
- The SEC and CFTC civil cases continued.

Run a five-minute debate. Split the room. Side A: "the code allowed it, so it was fair." Side B: "manipulating a market is manipulation, whatever the venue." Then ask: would your answer change if it were a bank's lending system with the same flaw?

This is educational, not legal advice. Check the current status of the civil cases before class.

14 Does an oracle attack pay?

Solend (Nov 2022, ~\$1.26M) and Loopscale (Apr 2025, ~\$5.8M) were this pattern at smaller scales.



Does an oracle attack pay?

A toy model, not any protocol's real formula. Use it to build intuition.

How it works: buying S dollars of a token in a constant-product market with Q dollars of depth multiplies the price by $((Q + S) / Q)$ squared. The lender values the attacker's tokens at that price and lends up to the loan-to-value ratio.

Try this order live:

1. Default: \$500K depth, \$2M spent, \$2M held, 80% LTV. The oracle sees 25x. The attack pays tens of millions. That is Mango.
2. Drag market depth up to \$10M+. The pump barely moves the price. Deep markets defend themselves.
3. Reset depth, turn on "Time-averaged price". A pump that lasts minutes barely moves an hour-long average. The attack shrinks but may still pay.
4. Turn on "\$1M borrow cap". Now the attack loses money. Lesson: defences are layered. No single one is enough.

Solend, Nov 2022: an oracle price attack on a thin asset, about \$1.26M. Loopscale, 26 Apr 2025: manipulated pricing of RateX PT tokens used as collateral, about \$5.8M, around 12% of TVL; markets were paused.

15

Bucket 3: keys, signers, operations

The code was fine. The people were the way in.

Bucket 3: keys, signers, operations

No notes. Let the slide speak.

16 Why an old signature can still bite

Normal transaction

signed

Durable nonce transaction

signed

Useful for offline signing and slow multisigs. Dangerous when you do not know what you signed.

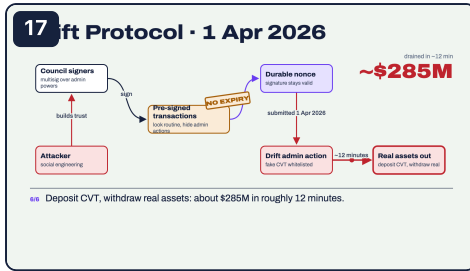
Why an old signature can still bite

You need this to understand Drift.

Every Solana transaction includes a recent blockhash. The network only accepts it while that blockhash is recent: about 150 blocks, roughly a minute. After that, the signed transaction is useless forever. That is a safety feature: a signature you gave yesterday cannot be replayed today.

A durable nonce is an opt-in alternative. Instead of a recent blockhash, the transaction uses a value stored in a nonce account. It stays valid until someone advances that nonce. That is useful: offline signing, hardware wallets in a vault, multisig members in different time zones collecting signatures over days.

The danger is the same feature: a signature you gave weeks ago can be submitted today, when you are not watching. If you did not fully understand what you signed, someone else gets to choose when it happens.



Drift Protocol · 1 Apr 2026

The biggest loss of 2026, and the strongest blog topic this week, because there was no code bug.

Per the handbook (sources: Chainalysis and TRM Labs write-ups):

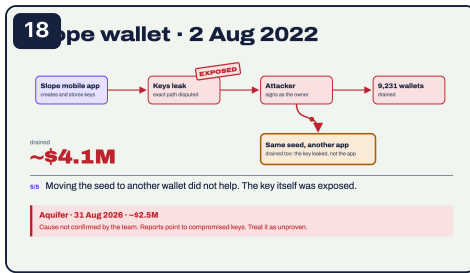
1. Drift's admin powers sat behind a council multisig: several signers must approve any admin action.
2. Attackers socially engineered council signers.
3. The signers pre-signed durable-nonce transactions that looked routine but contained hidden admin actions.
4. Because they were durable-nonce transactions, they stayed valid until used.
5. On 1 April 2026 the attackers submitted them. One admin action whitelisted a fake token, CVT, as collateral.
6. They deposited CVT and withdrew real assets: about \$285M in roughly 12 minutes.

Detail from later reporting (checked 14 Oct 2026): the Security Council was a 2-of-5 multisig with zero timelock. The fake token was "CarbonVote Token" (CVT) with an inflated oracle price, and the CVT market was created after the attacker gained admin control. The drain was 31 withdrawals in about 12 minutes. Loss estimates range from \$270M (CoinDesk) to \$286M (Elliptic). Elliptic called it suspected DPRK on 2 Apr; on 5 Apr Drift attributed it, with medium-high confidence, to UNC4736 (AppleJeus / Citrine Sleet), the group behind the Radiant hack. Sources: [elliptic.co](https://www.elliptic.co) (2026-04-02), [coindesk.com](https://www.coindesk.com) (2026-04-05).

The lessons:

- A multisig is only as strong as what each signer actually reads.
- Simulate every transaction before signing and read what it does, not what it claims.
- Admin actions like "add collateral type" need timelocks, so a bad one can be caught before it executes.

Finance link: this is operational risk and segregation of duties. Banks call it the four-eyes principle. Four eyes help only if they actually look.



Slope wallet · 2 Aug 2022

Slope: 9,231 wallets drained, about \$4.1M, per the Solana Foundation. Keys tied to the Slope mobile app had leaked. Widely reported at the time: the app sent sensitive wallet data to a monitoring service. Slope's own report disputed how much that caused. Say "the exact path is disputed".

The key lesson: the key is the account. People who had moved their Slope seed phrase into another wallet app were drained too, because the leak was the seed phrase itself, not the app.

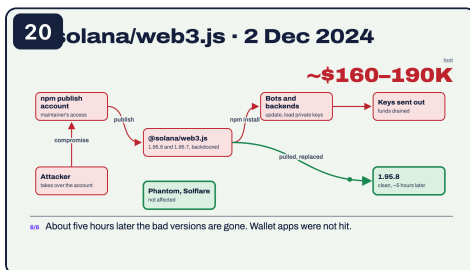
Aquifer, 31 Aug 2026, about \$2.5M (\$2.47M in some reports). The team has published no post-mortem and gave no cause. SlowMist and others suggest compromised wallets or keys; Bitquery's on-chain write-up reads more like a program flaw. It sits in this bucket provisionally: say "suspected" out loud.

Link back to rule 1 from the course map: never share a seed phrase or private key.



Bucket 4: supply chain

No notes. Let the slide speak.



@solana/web3.js · 2 Dec 2024

@solana/web3.js was the standard JavaScript library for talking to Solana. Thousands of apps install it from npm.

1. An attacker compromised an npm publish account for the package.
2. They published versions 1.95.6 and 1.95.7 containing a backdoor that sent private keys to the attacker.
3. Projects that installed or updated during the window pulled the bad version automatically.
4. Any code that loaded a private key directly (trading bots, backends holding hot keys) leaked it. Reported loss: about \$160–190K.
5. The bad versions were up for about five hours. A clean 1.95.8 followed. Wallet apps like Phantom and Solflare were not affected.

Defences: pin exact dependency versions and use a lockfile; do not auto-update in production; review what changed in an update; never keep private keys in application code if you can use a signer service or hardware key.

Finance link: third-party vendor risk. You audited your firm, but did you audit your software suppliers?

21

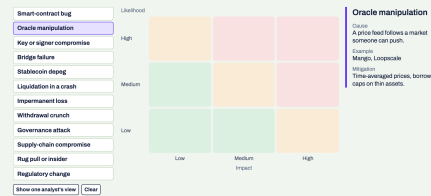
Risk with no attacker

Most money in DeFi is lost without anyone hacking anything.

Risk with no attacker

No notes. Let the slide speak.

22 Place the risks yourself



Place the risks yourself

Twelve risk categories. The curriculum asks for a taxonomy of at least ten, each with cause, consequence, example and mitigation. That is what the panel on the right shows.

Run it as a group exercise: pick a risk, ask the room "how likely, how bad?", click the cell they agree on. Then press "Show one analyst's view" and argue about the differences. There is no single right answer; the point is to justify the placement.

Notes on a few:

- Liquidation in a crash and impermanent loss are the most likely losses for ordinary users. They are not attacks. They are the product working as designed.
- Rug pulls on fresh memecoin launches are high likelihood and high impact, which is why the token audit task exists.
- Stablecoin depeg example: USDC fell to about \$0.88 in March 2023 when Circle disclosed reserves stuck at Silicon Valley Bank. It recovered once deposits were guaranteed.
- Governance attack example is from Ethereum (Beanstalk, April 2022): flash-loaned votes passed a malicious proposal.
- Regulation: cover your local rules. For Nigeria, the SEC Nigeria stance on digital assets; plus US and EU basics.

23 Read the token page first

Token page what the explorer shows

Issuer: Uniswap: Team

Mint authority: 7x2p...F35a

Freeze authority: 7x2p...F35a

Permanent delegate: None

Transfer hook: None

Transfer fee: 0%

Top 10 holders: 82%

Walk away

Mint authority active
Someone can print more and dilute you to zero.

Freeze authority active
Your token account can be frozen: you could buy but never sell.
Top 10 wallets hold 82%.
A few wallets can crash the price by selling.

Read the token page first

This is the users' task this week, so walk through it slowly.

On Orb or Solscan, every token page shows:

- Mint authority: who can create more. Revoked is what you want for most tokens.
- Freeze authority: who can freeze your token account. If someone can freeze you, they can let you buy and never let you sell.
- Token-2022 extensions. Permanent delegate can move or burn tokens from any holder's account. Transfer hook runs custom code on every transfer, which can block sales. Transfer fee takes a cut every time.
- Holders: if the top 10 wallets hold most of the supply, they can crash the price.

Context matters. Toggle "Regulated stablecoin issuer": USDC keeps both mint and freeze authority on purpose. Circle mints when you deposit dollars and must be able to freeze stolen funds. The same settings on a random memecoin are a red flag.

Live demo: open USDC's mint

(EPjFWdd5AufqSSqeM2qN1xzybapC8G4wEGGkZwyTDt1v) on Orb and read the authorities together. Then open any trending new token and compare.

24 TVL is a number, not a verdict



TVL is a number, not a verdict

TVL (total value locked) is the headline metric on DefiLlama. It is useful and easy to misread.

Click through:

0. Headline: TVL went from \$100M to \$225M in three months. Up 125%. Looks like growth.

1. Month 2: TVL rose \$50M, but no new users came. SOL went from \$100 to \$150. TVL is deposits times price.
2. Month 3: a rewards programme attracted 0.5M SOL from farmers who came for the rewards. That is rented liquidity.
3. Month 4: rewards end, farmers leave, SOL falls back. TVL is \$100M again. The real business never grew.
4. Double counting: stake SOL to get JitoSOL, deposit JitoSOL into a lender. Two protocols each report the same \$100M. Some dashboards add them. DefiLlama lets you toggle what is counted; check the settings.

The analyst habit: whenever TVL moves, ask three questions. Did prices move? Were incentives running? Is this counted elsewhere? Then look at fees and revenue, which are harder to fake.

25 Due-diligence scorecard



A due-diligence scorecard

This is the week 5 project template (curriculum chapter 5.3). Eight criteria, weighted.

The rule built into the widget: a score only counts once you mark that you have evidence. Drag the sliders first and the total stays low. That is deliberate. A protocol's website calling itself "secure" or "audited" is not evidence. Evidence is the audit report itself (who, when, what scope), an incident post-mortem, the on-chain multisig config, the fee data on DefiLlama, the commit history.

Try: set everything to 5 with no evidence. Score stays near zero, verdict "Not enough evidence to judge". Then mark evidence for six criteria and watch the radar fill in.

The weights are a suggestion. Students can argue for different weights, as long as they say why.

Separate the protocol from its token: a protocol can be healthy while its governance token falls, and the other way round.

26 Researching risk responsibly

- **Disclose** if you hold the token you write about.
- **Separate** what you saw on-chain from what you think it means.
- **Date** every number. Say what you could not verify.
- **Teach, don't advise.** Research is not "you should buy this".

Researching risk responsibly

Curriculum chapter 5.4, compressed.

Conflicts of interest are everywhere in crypto research: sponsored threads, paid "analysis", team members posting anonymously. Mentees who write publicly should disclose holdings every time. It builds trust and it is the habit of professional analysts.

Financial inclusion versus exploitation: DeFi reaches people banks ignore, and it also reaches people who cannot judge the risks. A post that makes a risky yield sound safe can hurt real readers.

The difference between education and advice: explain how something works and what the risks are; do not tell a specific person what to do with their money.

27 A program trusts a price account the caller passed in, without checking who owns it. Which bucket?

- ☐ A Oracle manipulation
- ☐ B Unchecked accounts
- ☐ C Supply chain
- ☐ D Keys and signers

Quiz: A program trusts a price account the caller passed in, without checking who owns it. Which bucket?

Trap answer: oracle manipulation. The price feed is involved, but nobody manipulated a market. They substituted an account. The question "is this the real account?" is bucket 1.

28 TVL rose 40% this month. What should you check first?

- ☐ A Whether the token price rose
- ☐ B Whether to buy the token
- ☐ C The number of tweets
- ☐ D Nothing, TVL is reliable

Quiz: TVL rose 40% this month. What should you check first?

Follow-up: what number is harder to fake than TVL? Fees paid by users, and revenue. Incentives can inflate deposits but they cannot easily make users pay fees.

29 This week's task

Users

1. Pick one incident from today.
2. Write a post-mortem: what happened, which bucket, who lost money, what would have prevented it.
3. Open one real token on **Orb**. Who holds **mint** and **freeze** authority? Any Token 2022 extensions?

Builders

1. Write a tiny **Anchor** program with a missing owner or signer check.
2. Write the **exploit** as a **LiteVM** test. It should pass (the attack works).
3. Add the check. The exploit test should now **fail**.
This is Wormhole and Cashio in 40 lines.

This week's task

Users: the post-mortem should read like an analyst's note, not a news article. Facts in one paragraph, the root cause in one sentence, prevention in bullet points. They must cite at least one primary source (the protocol's own post-mortem or a security firm's analysis).

Builders: references are the security sections of Cyfrin Updraft's free Solana course (missing signer checks, PDA validation) and Blueshift's challenges. A good exploit test asserts the attacker's balance went up. After the fix it asserts the transaction errors.

30

Blog post

"The risks of DeFi: lessons from Solana's failures."

Name the **bucket** for every incident you mention. Drift is the strongest story: a **people** problem, not a **code** problem.

Blog post

The strongest posts pick one incident and go deep, rather than listing ten. Drift is the obvious choice: the biggest loss of 2026, and no code was broken. Wormhole is the best one for builders.

Remind them: dates on every number, "reported" on every loss figure, and a link to a primary source.

31

Teacher prep

For the mentor. Skip in class.

Teacher prep

No notes. Let the slide speak.

32 You, before class

- Read **Chainalysis** or **TRM Labs** on the Drift hack. Then one write-up each for **Wormhole** and **Mango**.
- Know the **Mango legal ending**: convicted Apr 2024, convictions thrown out 23 May 2025, civil cases continuing.
- Builders' reference: the security sections of **Cyfrin Updraft's** free Solana course and **Blueshift's** challenges.
- Do the builders' task yourself: vulnerable program, **LiteVM** exploit, fix.
- Open **USDC's mint** on **Orb** and one trending new token. Screenshot both for the red-flag demo.

You, before class

Wormhole: the clearest write-ups explain the deprecated `load_instruction_at` helper and the missing `sysvar` address check. Know that one detail and you can answer most builder questions.

Mango: be ready for "so was it legal?" Point them to the debate slide. You don't have to settle it.

If you only have one hour of prep: Drift write-up (30 min), then run the TokenRedFlags demo on a real token (15 min), then rehearse the AccountSubstitution slide (15 min). Those are the three moments that carry the class.

33 The case library

Incident	Date	Loss	Root cause
Wormhole	2 Feb 2022	~\$200–320M	Fake sybil account; forged guardian signatures
Cashio	23 Mar 2022	~\$48–62M	LP collateral mint never checked; 2B CASH minted
Mango Markets	11 Oct 2022	~\$110–118M	Thin market oracle pump; 467M returned
Solend	Nov 2022	~\$1.26M	Oracle attack on a thin asset
Loopscale	26 Apr 2025	~\$5.8M	RataK PT collateral mispriced; ~12% of TVL
Slope wallet	2 Aug 2022	~\$4.1M	Keys tied to the app leaked; 9,231 wallets
Drift	1 Apr 2023	~\$25M	Signers tricked into disabling remote admin fee
Aqualite	31 Aug 2026	~\$2.5M	Course unconfirmed; reports suggest compromised keys
@leclerc/web3.js	2 Dec 2024	~\$180–190K	npm account hijack; 1.95.6 and 1.95.7 backdoored

The case library

Figures from the course handbook, checked 11 Oct 2026. Sources cited there: CertiK (2022 Solana exploits), Pine Analytics (Solana incident history), Messari (Loopscale), Chainalysis and TRM Labs (Drift), Solana's own Slope incident report, Cyfrin (web3.js compromise), Cointelegraph (Mango convictions overturned).

Ranges, not single numbers, because sources disagree and token prices at the time are counted differently.

34 Verify before class

Drift and Aquifer are recent

Drift (Apr 2026) and Aquifer (Aug 2026) figures come from early reports. Re-read the latest post-mortem; attribution and totals can change.

Mango civil cases

The SEC and CFTC cases were continuing as of the handbook check. Look up their current status.

Token pages change

Mint and freeze authorities, extensions and holder shares change. Screenshot the tokens you demo on the day.

Verify before class

Nothing on this list should be said in class as settled fact without a fresh check. Say the date you looked.
