Online Appendices to:

The Role of Auditor Reputation in an Emerging Audit Marketplace: Evidence from Decentralized Finance (DeFi)

W. Robert Knechel
University of Florida/University of Auckland
w.knechel@warrington.ufl.edu

Steven A. Maex
George Mason University
smaex@gmu.edu

Hyun Jong Park
Temple University
hyun.jong.park@temple.edu

Table of Contents:

Online Appendix A: Mechanics of Example DeFi Breaches

A.1. A Flash Loan Attack

One common type of exploit takes advantage of flash loans, which are uncollateralized loans taken and repaid on the same cryptographic transaction. These so-called "flash loan attacks" start with borrowing an asset (asset A) on one lending protocol (protocol A) and then exchanging that asset for another asset (asset B) on a decentralized exchange (DEX) (protocol B). Depending on the liquidity present on the exchange, this transaction can have the effect of deflating (inflating) the price of asset A (B) on that exchange. These price movements can have downstream implications for DeFi protocols that rely on the DEX for exchange rate information on the A-B asset pairing (i.e., rely on the DEX as a price "oracle").[1] Once the exchange rate for the A-B asset pairing has been manipulated, the hacker can find a lending protocol (protocol C) that relies on the exchange for pricing information. When asset B is posted as collateral to obtain asset A on this protocol (protocol C), the hacker is allowed to borrow more of asset A than would typically be allowed under normal market conditions. The hackers use a portion of the new loan obtained to pay off the original loan on protocol A and keep the remainder as profit. Inevitably, when exchange rates between assets A and B adjust back to normal market conditions, the loan taken on protocol C will be liquidated leaving the lending pools in protocol C with less total value than when the initial loan was taken. In the end, despite the technological complexity of these transactions, the fundamental manipulation is undertaken to create a classic arbitrage opportunity for specific market participants.

A.2. A Reentrancy Attack

Another common type of exploit is known as a reentrancy attack. In this case, a malicious actor calls a function in the vulnerable smart contract of the protocol (contract A) to withdraw crypto assets to a contract managed by the attacker (contract B). Contract B is designed to, upon receipt of the crypto assets, "re-enter" contract A with another call before the initial transaction is completed (resulting in an update of the balances of contract A). This process can be repeated multiple times so that the attacker can drain funds (greater than what were initially deposited) from contract A.

For more information on these and other exploits, refer to Crypto.com's whitepaper entitled, "Attacks and Exploits in DeFi" (Crypto.com 2021).

[1] Price oracles are third-party services allowing smart contracts to obtain external price data from outside of their ecosystem. Protocols instruct the smart contract to query the oracle which returns the current value of a token quoted in a specified currency (e.g., USD or ETH). If protocols only rely on a single DEX, changes in that DEX's price information are assumed to be true and accurate by the protocols' smart contracts. As such, if hackers can manipulate the asset price (so-called price oracle manipulation) on that single DEX, all protocols relying on that DEX as a price oracle become vulnerable to exploitation (Crypto.com 2021).

Online Appendix B: Audit Report Examples

B.1. Peckshield Audit Report for AAVE-V3

B.1.1. Introduction and Assessment Methodology

Number of Auditors / Reviewers = 5

Auditor Effort (Days) Not Disclosed

Number of Pages = 31

Document Properties

Client	Aave
Title	Smart Contract Audit Report
Target	Aave V3
Version	1.0
Author	Xuxian Jiang
Auditors	Patrick Liu, Stephen Bie, Jing Wang, Xuxian Jiang
Reviewed by	Yiqun Chen
Approved by	Xuxian Jiang
Classification	Public

Version Info

Version	Date	Author(s)	Description
1.0	January 10, 2022	Xuxian Jiang	Final Release
1.0-rc	November 20, 2021	Xuxian Jiang	Release Candidate #1

Contact

For more information about this document and its contents, please contact PeckShield Inc.

Name	Yiqun Chen
Phone	+86 183 5897 7782
Email	contact@peckshield.com

1 Introduction

Given the opportunity to review the design document and related smart contract source code of the Aave V3 protocol, we outline in the report our systematic approach to evaluate potential security issues in the smart contract implementation, expose possible semantic inconsistencies between smart contract code and design document, and provide additional suggestions or recommendations for improvement. Our results show that the given version of smart contracts can be further improved due to the presence of several issues related to either security or performance. This document outlines our audit results.

1.1 About Aave V3

Aave is a popular decentralized non-custodial money market protocol where users can participate as depositors or borrowers. Depositors provide liquidity to the market to earn a passive income, while borrowers are able to borrow in an over-collateralized (perpetually) or under-collateralized (one-block liquidity) fashion. The audited Aave V3 protocol includes new features for new usage scenarios and improves earlier versions on capital efficiency, security, decentralization, UX while at the same time providing new functionalities to leverage the capabilities of rollups and the growing ecosystem of competing L1s. The basic information of Aave V3 is as follows:

Table 1.1: Basic Information of Aave V3

Item	Description
Name	Aave
Website	https://aave.com/
Type	Ethereum Smart Contract
Platform	Solidity
Audit Method	Whitebox
Latest Audit Report	January 10, 2022

In the following, we show the Git repository of reviewed files and the commit hash value used in

B.1. Peckshield Audit Report for AAVE-V3 (Continued)

B.1.1. Introduction and Assessment Methodology (Continued)

this audit.

- https://github.com/aave/aave-v3-core.git (14f6148)

And this is the commit ID after all fixes for the issues found in the audit have been checked in:

- https://github.com/aave/aave-v3-core.git (c71d57d)

1.2 About PeckShield

PeckShield Inc. [13] is a leading blockchain security company with the goal of elevating the security, privacy, and usability of current blockchain ecosystems by offering top-notch, industry-leading services and products (including the service of smart contract auditing). We are reachable at Telegram (https://t.me/peckshield), Twitter (http://twitter.com/peckshield), or Email (contact@peckshield.com).

Table 1.2: Vulnerability Severity Classification

Impact \ Likelihood	High	Medium	Low
High	Critical	High	Medium
Medium	High	Medium	Low
Low	Medium	Low	Low

1.3 Methodology

To standardize the evaluation, we define the following terminology based on the OWASP Risk Rating Methodology [12]:

- Likelihood represents how likely a particular vulnerability is to be uncovered and exploited in the wild;
- Impact measures the technical loss and business damage of a successful attack;
- Severity demonstrates the overall criticality of the risk.

Table 1.3: The Full Audit Checklist

Category	Checklist Items
Basic Coding Bugs	Constructor Mismatch
	Ownership Takeover
	Redundant Fallback Function
	Overflows & Underflows
	Reentrancy
	Money-Giving Bug
	Blackhole
	Unauthorized Self-Destruct
	Revert DoS
	Unchecked External Call
	Gasless Send
	Send Instead Of Transfer
	Costly Loop
	(Unsafe) Use Of Untrusted Libraries
	(Unsafe) Use Of Predictable Variables
	Transaction Ordering Dependence
	Deprecated Uses
Semantic Consistency Checks	Semantic Consistency Checks
Advanced DeFi Scrutiny	Business Logics Review
	Functionality Checks
	Authentication Management
	Access Control & Authorization
	Oracle Security
	Digital Asset Escrow
	Kill-Switch Mechanism
	Operation Trails & Event Generation
	ERC20 Idiosyncrasies Handling
	Frontend-Contract Integration
	Deployment Consistency
	Holistic Risk Management
Additional Recommendations	Avoiding Use of Variadic Byte Array
	Using Fixed Compiler Version
	Making Visibility Level Explicit
	Making Type Inference Explicit
	Adhering To Function Declaration Strictly
	Following Other Best Practices

Number of Items Examined = 36

B.1. Peckshield Audit Report for AAVE-V3 (Continued)

B.1.2. Findings

2 | Findings

2.1 Summary

Here is a summary of our findings after analyzing the implementation of the Aave V3 protocol. During the first phase of our audit, we study the smart contract source code and run our in-house static code analyzer through the codebase. The purpose here is to statically identify known coding bugs, and then manually verify (reject or confirm) issues reported by our tool. We further manually review business logic, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.

Severity	# of Findings
Critical	0
High	2
Medium	3
Low	4
Informational	2
Total	11

We have so far identified a list of potential issues: some of them involve subtle corner cases that might not be previously thought of, while others refer to unusual interactions among multiple contracts. For each uncovered issue, we have therefore developed test cases for reasoning, reproduction, and/or verification. After further analysis and internal discussion, we determined a few issues of varying severities need to be brought up and paid more attention to, which are categorized in the above table. More information can be found in the next subsection, and the detailed discussions of each of them are in Section 3.

Number of High or Greater Findings = 2

Number of Total Findings = 9

# of Info. Items / Recommendations = 2

2.2 Key Findings

Overall, these smart contracts are well-designed and engineered, though the implementation can be improved by resolving the identified issues (shown in Table 2.1), including 2 high-severity vulnerabilities, 3 medium-severity vulnerabilities, 4 low-severity vulnerabilities, and 2 informational recommendations.

Table 2.1: Key Aave V3 Audit Findings

ID	Severity	Title	Category	Status
PVE-001	Informational	Suggested immutable For Gas Efficiency	Coding Practice	Fixed
PVE-002	Medium	Improved Logic of Pool::_addReserveToList()	Business Logic	Fixed
PVE-003	Medium	Proper And Consistent Collateral Enabling	Business Logic	Fixed
PVE-004	Low	Improvement on UserConfiguration::_getFirstAssetAsCollateralId()	Coding Practice	Fixed
PVE-005	Informational	Redundant State/Code Removal	Coding Practice	Fixed
PVE-006	High	Proper Asset Price in GenericLogic::calculateUserAccountData()	Business Logic	Fixed
PVE-007	Medium	Proper EMode Category Use in Pool::borrow()	Business Logic	Fixed
PVE-008	Low	Possible Underflow Avoidance in BorrowLogic And UserConfiguration	Coding Practices	Confirmed
PVE-009	Low	Consistent Reserve Cache Use in rebalanceStableBorrowRate()	Coding Practice	Fixed
PVE-010	Low	Health Validation in EModeLogic::executeSetUserEMode()	Business Logic	Fixed
PVE-011	High	Potential Reentrancy Risk in flashLoanSimple()	Time and State	Fixed

Beside the identified issues, we emphasize that for any user-facing applications and services, it is always important to develop necessary risk-control mechanisms and make contingency plans, which may need to be exercised before the mainnet deployment. The risk-control mechanisms should kick in at the very moment when the contracts are being deployed on mainnet. Please refer to Section 3 for details.

B.2. Example of Audit Report with Audit Effort Reported (Trail of Bits Audit for Frax)

Number of Auditors / Reviewers = 3

Auditor Effort (Days) = 4 person-weeks = 20 person-days

Executive Summary

Overview

Frax Finance engaged Trail of Bits to review the security of its smart contracts. From December 6 to December 21, 2021, a team of three consultants conducted a security review of the client-provided source code, with four person-weeks of effort. Details of the project's timeline, test targets, and coverage are provided in subsequent sections of this report.

Online Appendix C: Identification of Breaches and Definition of Breach Spillover Analysis Sample

We hand-collect a list of DeFi breaches from two sources. First, we identify breaches on CryptoSec's DeFi Hacks list, which has aggregated a list of over 100 breaches (and the associated loss from each breach) dating back to the beginning of 2020. Second, we review the list of breaches reported by DeFiYield's REKT database. We identify 267 breaches since January 2020.[2] After removing duplicates, limiting to those related to protocols covered by DefiLlama or DefiSafety, limiting to those having the breach size $1 million or more, and limiting to those for which TVL change around the breach date can be calculated, we have data on 28 breaches for our breach spillover analysis (H2). Table C.1 below outlines our sample selection procedure for breaches.

Table C.1: Breaches Sample

Filter	**N**
Breaches reported on either DefiYield's REKT Database or CryptoSec's Defi Exploits list from January 2020 to June 2022 (i.e., the final month covered by our sample of SCAs)	267
Less: Those related to protocols not covered by DefiLlama or DefiSafety	(208)
Less: Breaches of less than $1 million in value	(22)
Breaches Used to Define *Breach* Variable in Protocol-Month Sample (H1)	**37**
Less: Breaches for which DefiLlama TVL data is not available to calculate TVL change around the breach for the breached protocol	(9)
Breach Events for Breach Spillover Analysis (H2)	**28**
Distinct Breach Dates for Spillover Analysis (H2)	**25**

[2] Within DeFiYield's REKT database, we focus on breaches coded as exploits, flash loan attacks, access control issues, and oracle issues. Other scams tracked by DeFiYield (e.g., exit scams, phishing scams, etc.) do not involve vulnerabilities in smart contracts, and therefore, are not considered for analysis. We also identify and remove two cases from consideration, which upon further inspection, reflected liquidations primarily driven by coin price volatility.

Table C.2 below provides some descriptive statistics along with the amount of the breach and the change in TVL in the seven days following the breach. As specific breach dates and amounts are sometimes inconsistently reported in the two lists of breaches from which our sample is drawn, we validate and augment information in these lists with web searches and online articles related to each breach to identify (to the best of our ability) the breach date and estimated breach size for each breach.

Table C.2: Breaches in Breach Spillover Analysis

Breached Protocol	Date	Breach Amount	TVL Change (Day -1 to 7)	TVL % Change (Day -1 to 7)
Uniswap	4/18/2020	1.1M	5.2M	10.7%
Pickle	11/21/2020	19.7M	-80.4M	-73.1%
Cover Protocol	12/28/2020	9.4M	-20.7M	-74.4%
Yearn Finance	2/4/2021	11.0M	85.8M	9.4%
Growth DeFi	2/8/2021	1.3M	-29.1M	-77.9%
CREAM Finance	2/13/2021	37.0M	-613.8M	-72.6%
Homora			-12.2M	-1.4%
DODO	3/8/2021	3.8M	1.4M	2.7%
ValueDefi	5/5/2021	10.0M	-909.0M	-93.1%
Rari Capital	5/8/2021	15.0M	-12.8M	-19.7%
ValueDefi	5/8/2021	11.0M	-806.1M	-93.3%
Pancake Bunny	5/19/2021	200.0M	-2,526.5M	-63.2%
Belt Finance	5/29/2021	50.0M	-755.6M	-50.2%
Pancake Bunny	7/16/2021	2.4M	-39.4M	-5.9%
Popsicle Finance	8/3/2021	20.7M	-10.9M	-62.9%
CREAM Finance	8/30/2021	18.8M	98.0M	5.9%
SushiSwap	9/16/2021	3.0M	-687.9M	-11.3%
Compound	9/29/2021	80.0M	-2,744.9M	-15.2%
Hunny Finance	10/20/2021	2.0M	-21.2M	-61.7%
CREAM Finance	10/27/2021	130.0M	-226.0M	-13.2%
Vesper	11/2/2021	3.0M	38.0M	13.1%
Badger DAO	12/1/2021	120.0M	-251.7M	-20.2%
Grim Finance	12/19/2021	30.0M	-93.6M	-94.6%
Inverse Finance	4/2/2022	15.6M	-43.5M	-64.3%
Rari Capital	4/30/2022	80.0M	130.0M	14.8%
Saddle Finance	4/30/2022	10.0M	-134.6M	-48.1%
Venus	5/13/2022	1.4M	-650.6M	-35.3%
Inverse Finance	6/16/2022	1.2M	-0.8M	-6.4%
Total	**28 Breaches**	**887.4M**	**-10,313.1M**	**-23.8%**

Online Appendix D: Twitter Activity for Auditors Around Sampled Breaches

We model activity for the Auditor Twitter handles in our sample around the 28 breaches identified above to understand whether there is evidence of increased market attention on an auditor connected to a breached protocol in the days following the breach. Specifically, we estimate the following model in an auditor-date sample over the period covered by our main analyses (January 1, 2020, to June 30, 2022).

$$ln(1+Posts)_{a,d} \text{ or } ln(1+Mentions)_{a,d} = \beta_1 * Br\_Auditor_{a,d} + Auditor\ F.E. + Date\ F.E. + \varepsilon \quad \text{(D1)}$$

In the above model, the subscript *a* denotes the auditor and the subscript *d* denotes the date. The dependent variables (*ln(1+Posts)* and *ln(1+Mentions)*) reflect the logged counts of the daily posts by the auditor's Twitter handle and daily mentions of the auditor's Twitter handle, respectively. *Br_Auditor* is coded one if a protocol recently audited by the auditor was breached in the previous 7 days. With the inclusion of auditor and date fixed effects, the coefficient on *Br_Auditor* identifies the average increase in Twitter activity for the auditor in days *d+1* through *d+7* following a breach of one of its recently audited protocols.

We present the results of estimating this model in Table D.1 below. We find that mentions of the auditors recently conducting audits for breached protocols are significantly higher in the seven days following a given breach (column 2) while the posting activity of the auditor remains unchanged (column 1).

Table D.1: Auditor Twitter Activity Around DeFi Breaches

	(1)	**(2)**
Dependent Variable	***ln(1+Posts)***	***ln(1+Mentions)***
Br_Auditor	-0.019	0.260*
	(0.053)	(0.133)
Observations	53,505	53,505
Controls	Included	Included
Date F.E.	Yes	Yes
Auditor F.E.	Yes	Yes
(Within Auditor) Adjusted R^2	0.046	0.069

Note: Standard errors are reported based on protocol-clustered standard errors. Significance levels are presented as follows based on two-tailed tests: *** p<0.01, ** p<0.05, * p<0.10.

Online Appendix E: Effects on Native Coin Prices

We report the results of our primary models when using native token price as the proxy for protocol value in place of TVL.

Table E.1: Audits and Native Token Prices Baseline (Compare to Table 4, Panel A)

	(1)	(2)	(3)	(4)
Dependent Variable	***ln(Token_Price)$_{(t)}$***	***ln(Token_Price)$_{(t)}$***	***ln(Token_Price)$_{(t)}$***	***ln(Token_Price)$_{(t)}$***
Audit$_{(t-3\ to\ t-1)}$	0.177***		0.183***	
	(0.053)		(0.052)	
Audit$_{(t-1)}$		0.161***		0.162***
		(0.045)		(0.045)
Audit$_{(t-2)}$		0.164***		0.165***
		(0.042)		(0.041)
Audit$_{(t-3)}$		0.095**		0.098***
		(0.039)		(0.037)
Audit$_{(t+1\ to\ t+3)}$			0.039	0.026
			(0.068)	(0.069)
Q_Rating$_{(t-1)}$	0.014	0.013	0.013	0.013
	(0.011)	(0.011)	(0.011)	(0.011)
Breach$_{(t-1)}$	0.093	0.050	0.088	0.047
	(0.158)	(0.161)	(0.158)	(0.160)
ln(TVL_Month)$_{(t-1)}$	0.225***	0.222***	0.224***	0.222***
	(0.039)	(0.039)	(0.039)	(0.039)
Observations	1,646	1,646	1,646	1,646
Year-Month F.E.	Yes	Yes	Yes	Yes
Protocol F.E.	Yes	Yes	Yes	Yes
(Within Protocol) Adj. R^2	0.495	0.499	0.495	0.498

Note: Standard errors are reported based on protocol-clustered standard errors. Significance levels are presented as follows based on two-tailed tests: *** p<0.01, ** p<0.05, * p<0.10.

Table E.2: Individual Effects of Advertising and Audit Quality (Compare to Table 5, Panel A)

	(1)	(2)	(3)	(4)
Dependent Variable	$ln(Token\_Price)_{(t)}$	$ln(Token\_Price)_{(t)}$	$ln(Token\_Price)_{(t)}$	$ln(Token\_Price)_{(t)}$
Twitter Engagement Proxy	**Twitter Posts**	**Twitter Mentions**	**--**	**--**
Audit Quality Proxy	**--**	**--**	**Page Length**	**Items Examined**
$Aud\_HighAdv_{(t-3\ to\ t-1)}$	0.195***	0.261***		
	(0.060)	(0.063)		
$Aud\_LowAdv_{(t-3\ to\ t-1)}$	0.136*	0.040		
	(0.079)	(0.080)		
$Aud\_HighQ_{(t-3\ to\ t-1)}$			0.254***	0.262***
			(0.063)	(0.082)
$Aud\_LowQ_{(t-3\ to\ t-1)}$			0.106	0.122**
			(0.066)	(0.061)
$Q\_Rating_{(t-1)}$	0.014	0.014	0.014	0.013
	(0.011)	(0.011)	(0.011)	(0.011)
$Breach_{(t-1)}$	0.096	0.121	0.078	0.093
	(0.158)	(0.162)	(0.157)	(0.160)
$ln(TVL\_Month)_{(t-1)}$	0.223***	0.221***	0.224***	0.223***
	(0.039)	(0.039)	(0.039)	(0.039)
Observations	1,646	1,646	1,646	1,646
Year-Month F.E.	Yes	Yes	Yes	Yes
Protocol F.E.	Yes	Yes	Yes	Yes
(Within Protocol) Adj. R^2	0.495	0.500	0.497	0.496
$Aud\_HighAdv_{(t-3\ to\ t-1)} - Aud\_LowAdv_{(t-3\ to\ t-1)}$	0.059	0.222**		
	(0.088)	(0.095)		
$Aud\_HighQ_{(t-3\ to\ t-1)} - Aud\_LowQ_{(t-3\ to\ t-1)}$			0.148**	0.140
			(0.074)	(0.094)

Note: Standard errors are reported based on protocol-clustered standard errors. Significance levels are presented as follows based on two-tailed tests: *** p<0.01, ** p<0.05, * p<0.10.

Table E.3: Event Study Model around Breach Events (Compare to Table 7, Panel B)

Dependent Variable	(1) *TokenPrice_Chg*	(2) *TokenPrice_Chg*	(3) *TokenPrice_Chg*	(4) *TokenPrice_Chg*
Breach	-0.204***	-0.212***	-0.202***	-0.210***
	(0.052)	(0.051)	(0.052)	(0.051)
Br_Auditor	-0.013	-0.021	-0.013	-0.021
	(0.018)	(0.020)	(0.018)	(0.020)
NonBr_Auditor		-0.015		-0.014
		(0.013)		(0.013)
Q_Rating	-0.000	-0.000	-0.000	-0.000
	(0.000)	(0.000)	(0.000)	(0.000)
ln(Age)	0.003	0.001	0.003	0.001
	(0.007)	(0.008)	(0.007)	(0.008)
ln(TVL_tm1)	-0.001	-0.000	-0.001	-0.000
	(0.002)	(0.002)	(0.002)	(0.002)
Crypto_Chg	0.242***	0.244***	0.245***	0.247***
	(0.064)	(0.064)	(0.065)	(0.065)
Br_Category			0.013	0.013
			(0.015)	(0.015)
Observations	2,381	2,381	2,381	2,381
Adjusted R^2	0.016	0.016	0.016	0.016
Breach – Br_Auditor	-0.191***	-0.191***	-0.188***	-0.188***
	(0.054)	(0.054)	(0.054)	(0.054)
Br_Auditor – NonBr_Auditor		-0.007		-0.007
		(0.019)		(0.019)

Note: Standard errors are reported based on protocol-clustered standard errors. Significance levels are presented as follows based on two-tailed tests: *** p<0.01, ** p<0.05, * p<0.10.

Online Appendix F: Sensitivity Tests for Model (2)

We report the results of additional tests to evaluate the robustness of our results reported in Table 7, Panel B. We report the test coefficients (columns 1 and 2) and differences in test coefficients (columns 3 and 4) after making each adjustment to our model. These results are equivalent to column 2 in Table 7, Panel B. First, we use 360 days, rather than 180 days, as the window for which to identify protocols that shared an auditor with the breached protocol. Second, we alter the window for which we identify changes in TVL (running models in which we end the period at either *t+5* or *t+9*). Third, as alternative approaches to handle outliers in our TVL change variable, we run both a robust regression and a quantile regression with the unwinsorized TVL change as our dependent variable and find consistent results. Lastly, we entropy balance our sample between *Br_Auditor* = 1 and *Br_Auditor* = 0 protocol-breach date observations on the first three moments of the protocol attributes used as controls: *Q_Rating*, *ln(Age)*, and *ln(TVL_tm1)*. We re-estimate model (2) using the weights generated from this balancing approach. In all cases, our results are consistent with our primary analysis.

Sensitivity Test	**(1)** ***Breach***	**(2)** ***Br_Auditor***	**(3)** ***Breach – Br_Auditor***	**(4)** ***Br_Auditor – NonBr_Auditor***
Changing the Lookback Period				
Audit lookback period	-0.291***	-0.090***	-0.201***	-0.038**
set to 360 days	(0.056)	(0.020)	(0.053)	(0.019)
Changing the *TVL_Chg* Interval				
TVL_Chg defined from	-0.229***	-0.054***	-0.175***	-0.035**
t-1 to *t+5*	(0.044)	(0.015)	(0.043)	(0.014)
TVL_Chg defined from	-0.284***	-0.084***	-0.200***	-0.060**
t-1 to *t+9*	(0.067)	(0.024)	(0.068)	(0.028)
Alternative Approaches to Handle Outliers in *TVL_Chg*				
Use robust regression	-0.345***	-0.058***	-0.287**	-0.035**
	(0.116)	(0.014)	(0.116)	(0.014)
Use quantile regression	-0.336***	-0.044***	-0.292***	-0.026**
	(0.053)	(0.013)	(0.055)	(0.013)
Balancing Sample				
Entropy balancing on	-0.315***	-0.075***	-0.241***	-0.048**
Br_Auditor	(0.047)	(0.020)	(0.046)	(0.019)

Note: Standard errors are reported based on protocol-clustered standard errors. Significance levels are presented as follows based on two-tailed tests: *** p<0.01, ** p<0.05, * p<0.10.

References for Online Appendices:

Crypto.com (2021) Attacks and exploits in DeFi. https://assets.ctfassets.net/hfgyig42jimx/2l90zr21hmUG2aL7eK02Cl/ccb091f0c5247abaed4984bc3c286782/Attacks_and_Exploits_in_DeFi.pdf.